

UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE TECNOLOGIA

João Rubens Marchete Filho

**Utilização de árvores PQR para redução de cruzamentos em grafos
acíclicos direcionados**

Limeira, 2013.

UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE TECNOLOGIA

João Rubens Marchete Filho

**Utilização de árvores PQR para redução de cruzamentos em grafos
acíclicos direcionados**

Dissertação apresentada ao Curso de Mestrado em
Tecnologia da Faculdade de Tecnologia da
Universidade Estadual de Campinas, como parte dos
requisitos exigidos para a obtenção do título de Mestre
na área de Tecnologia e Inovação.

Supervisor/Orientador: Prof. Dr. Celmar Guimarães da Silva

ESTE EXEMPLAR CORRESPONDE À VERSÃO FINAL DA
DISSERTAÇÃO DEFENDIDA PELO ALUNO
JOÃO RUBENS MARCHETE FILHO, E ORIENTADA PELO
PROF. DR. CELMAR GUIMARÃES DA SILVA

Limeira, 2013.

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca da Faculdade de Tecnologia
Vanessa Evelyn Costa - CRB 8/8295

M332u Marchete Filho, João Rubens, 1984-
Utilização de árvores PQR para redução de cruzamentos em grafos acíclicos
direcionados / João Rubens Marchete Filho. – Limeira, SP : [s.n.], 2013.

Orientador: Celmar Guimarães da Silva.
Dissertação (mestrado) – Universidade Estadual de Campinas, Faculdade de
Tecnologia.

1. Algoritmos em grafos. 2. Visualização de informação. I. Silva, Celmar
Guimarães da. II. Universidade Estadual de Campinas. Faculdade de Tecnologia.
III. Título.

Informações para Biblioteca Digital

Título em outro idioma: Using PQR trees for reducing crossings in directed acyclic graphs

Palavras-chave em inglês:

Information visualization

Algorithms on graphs

Área de concentração: Tecnologia e Inovação

Titulação: Mestre em Tecnologia

Banca examinadora:

Celmar Guimarães da Silva [Orientador]

Luis Gustavo Nonato

Marco Antônio Garcia de Carvalho

Data de defesa: 20-09-2013

Programa de Pós-Graduação: Tecnologia

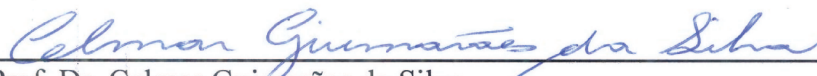
DISSERTAÇÃO DE MESTRADO EM TECNOLOGIA

ÁREA DE CONCENTRAÇÃO: TECNOLOGIA E INOVAÇÃO

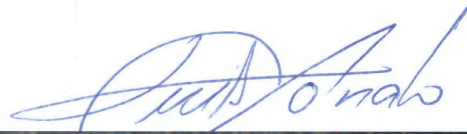
Utilização de árvores PQR para redução de cruzamentos em grafos acíclicos direcionados

João Rubens Marchete Filho

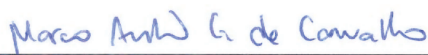
A Banca Examinadora composta pelos membros abaixo aprovou esta Dissertação:



Prof. Dr. Celmar Guimarães da Silva
Faculdade de Tecnologia
Presidente



Prof. Dr. Luis Gustavo Nonato
Universidade de São Paulo



Prof. Dr. Marco Antonio Garcia de Carvalho
Faculdade de Tecnologia

Resumo

A utilização de grafos acíclicos direcionados permite a representação gráfica de estruturas hierárquicas, o que facilita encontrar padrões e tendências durante a análise dessas estruturas. A principal abordagem de desenho automático desses grafos divide seus vértices em camadas, de tal modo que as arestas sempre apontem para uma mesma direção. Um dos principais critérios estéticos dessas estruturas visuais é evitar, sempre que possível, o cruzamento entre arestas, facilitando assim o entendimento do desenho. Os algoritmos de redução de cruzamentos devem também prover essa solução em um tempo inferior a 0,1 segundo, tornando mais apropriada sua utilização em estruturas visuais interativas. Nesta dissertação, pesquisou-se como uma estrutura de dados conhecida por árvore PQR poderia ser utilizada a fim de aperfeiçoar dois métodos de redução de cruzamentos: *BC* e *MEDIAN*. Dentre os novos métodos desenvolvidos, destacaram-se: *PQR_BC2* para o aperfeiçoamento do *BC*; *PQR_M2* para o aperfeiçoamento do *MEDIAN*. Os resultados obtidos através da aplicação dos métodos a pacotes de grafos amplamente utilizados na literatura mostram que os métodos baseados em árvores PQR superaram o método *BC* e o método *MEDIAN*, com relação à redução de cruzamentos, em 42% dos casos, empatando nesse critério em 45% dos grafos analisados. Além disso, os métodos baseados em árvores PQR também executaram em um tempo viável para aplicação em estruturas visuais interativas.

Palavras-chave: Redução de cruzamentos, árvores PQR, Visualização de Informação.

Abstract

The use of directed acyclic graphs allows the graphical representation of hierarchical structures, which eases to find patterns and trends during the analysis of these structures. The main approach to automatic design of these graphs divides its vertices in layers so that the edges always point toward the same direction. One of the major aesthetic criteria of such visual structures is to avoid, whenever possible, the crossing between the edges, thereby facilitating the understanding of the drawing. The crossing reduction algorithms must also provide this solution in a time less than 0.1 second, allowing its use in interactive visual structures. In this dissertation, was investigated as a data structure known as PQR tree could be used to improve both methods for reducing crossings: *BC* and *MEDIAN*. Among the new methods developed, stood out: *PQR_BC2* for improvement of *BC*; *PQR_M2* for improving the *MEDIAN*. The results obtained by applying the methods to packets of graphs widely used in the literature show that the methods based on PQR trees outperformed the *BC* method and the *MEDIAN* method, with respect to the crossing reduction, in 42% of cases, tying this criterion in 45% of the graph analyzed. In addition, methods based on PQR trees also performed at a feasible time for application to interactive visual structures.

Keywords: crossing reduction, PQR trees, Information Visualization.

Sumário

1	Introdução	1
1.1	Objetivos	5
1.2	Organização do texto	5
2	Desenho de grafos	7
2.1	Convenções de desenho de grafos	7
2.2	Critérios Estéticos	8
2.3	Desenho de DAGs	10
2.3.1	Modelo de Sugiyama	11
2.3.2	Outras abordagens	18
3	Algoritmos para redução de cruzamentos em DAGs	23
3.1	Redução de cruzamentos com layer-by-layer sweep	23
3.1.1	Métodos de redução em grafos bipartidos	24
3.2	Métodos alternativos para redução de cruzamentos	28
3.3	Considerações	29
4	Árvores PQR e redução de cruzamentos	31
4.1	Árvores PQR e suas aplicações	33
4.2	Método de redução de cruzamentos em DAGs utilizando Árvores PQR	35
4.2.1	Métodos <i>PQR_BC1</i> e <i>PQR_M1</i>	36
4.2.2	Métodos <i>PQR_BC2</i> e <i>PQR_M2</i>	42
4.3	Considerações	43

5	Experimentos e resultados	47
5.1	Materiais e métodos	47
5.1.1	Pacote de Grafos	48
5.1.2	Definição de camadas	50
5.1.3	Cálculo de densidade	52
5.1.4	Implementação	52
5.2	Resultados	53
5.2.1	Comparação entre os métodos <i>BC</i> e <i>PQR_BC1</i>	55
5.2.2	Comparação entre os métodos <i>BC</i> e <i>PQR_BC2</i>	63
5.2.3	Comparação entre os métodos <i>MEDIAN</i> e <i>PQR_M1</i>	71
5.2.4	Comparação entre os métodos <i>MEDIAN</i> e <i>PQR_M2</i>	78
5.3	Conclusões	86
6	Conclusão	89
6.1	Trabalhos Futuros	91

Dedicatória

Dedico este trabalho à minha esposa, companheira e amiga nos momentos mais difíceis desta jornada, a quem devo toda minha gratidão.

Agradecimentos

A Deus, que sempre esteve ao meu lado. A Ele toda honra, glória e louvor.

Ao Professor Celmar Guimarães da Silva, orientador do trabalho, por suas horas de orientação dedicadas a este trabalho.

Ao Professor João Meidanis, que gentilmente forneceu as classes para a construção de árvores PQR.

À minha esposa, que tanto me apoiou e me incentivou nas horas difíceis.

Lista de Figuras

1.1	Representação hierárquica entre empregados e empregadores	2
1.2	Exemplo de desenho de grafos com tamanhos diferentes	3
1.3	Desenho de hierarquia com padrões estéticos	3
1.4	Exemplo de redução de cruzamentos em um DAG	4
2.1	Figura de grafos que seguem a convenção de conectividade	8
2.2	Figura de um grafo que segue convenções de desenho para DAGs	9
2.3	Modificações feitas no desenho de um DAGs para atender critérios estéticos	10
2.4	Desenho de um DAG em camadas	11
2.5	Representações diversas de desenhos de DAGs	12
2.6	Representação visual de dependências de disciplinas de um curso de graduação	13
2.7	Atribuição de camadas para um DAG	15
2.8	Adição de <i>dummy nodes</i> ao DAG	15
2.9	DAG após processo de redução de cruzamentos	18
2.10	DAG após processo de atribuição de coordenadas	19
2.11	Figura representando passos da técnica	20
2.12	Figura representando técnica tridimensional de desenho de DAG	21
4.1	Exemplos de nós de uma Árvore PQR com as possíveis permutações	32
4.2	Representação de uma Árvore PQR e das suas possíveis permutações	32
4.3	Grafo bipartido com 10 vértices e 7 arestas	33
4.4	Árvores PQR com respectivas fronteiras	34
4.5	Grafo bipartido reordenado com Árvores PQR	34
4.6	DAG reordenado com algoritmo de reordenação de DAGs utilizando Árvores PQR	36

4.7	Representação gráfica de um DAG com antes e depois de submetido ao método de redução de cruzamentos com Árvores PQR	37
4.8	DAG reordenado com algoritmos <i>BC</i> e <i>PQR_BC1</i>	39
4.9	DAG reordenado com algoritmos <i>MEDIAN</i> e <i>PQR_M1</i>	41
4.10	DAG reordenado com algoritmos <i>BC</i> e <i>PQR_BC2</i>	44
4.11	DAG reordenado com algoritmos <i>MEDIAN</i> e <i>PQR_M2</i>	45
5.1	Grafos do Pacote North agrupados pelo número de vértices	49
5.2	Grafos do Pacote Rome agrupados pelo número de vértices	49
5.3	Número de vértices antes e depois da atribuição de camadas nos grafos do Pacote North	51
5.4	Número de vértices antes e depois da atribuição de camadas nos grafos do Pacote Rome	51
5.5	Diagrama de Classes da conexão OGDF (C++) e Árvore PQR (Java)	53
5.6	Grafos do Pacote North separados por densidades e percentuais de vitórias, derrotas e empates (em termos de cruzamentos) do método <i>PQR_BC1</i>	55
5.7	Grafos do pacote Rome separados por densidades e percentuais de vitórias, derrotas e empates (em termos de cruzamentos) do método <i>PQR_BC1</i>	56
5.8	Quantidade de DAGs do Pacote North, agrupadas por percentual de redução de cruzamentos provida pelos métodos <i>BC</i> e <i>PQR_BC1</i> , com relação à quantidade inicial de cruzamentos desses grafos.	57
5.9	Quantidade de DAGs do Pacote Rome, agrupadas por percentual de redução de cruzamentos provida pelos métodos <i>BC</i> e <i>PQR_BC1</i> , com relação à quantidade inicial de cruzamentos desses grafos.	57
5.10	Resultados do método <i>PQR_BC1</i> com relação à melhora dos resultados do método <i>BC</i> (vitórias) e de deterioração (derrotas) separados por porcentagem, relativos ao número de cruzamentos iniciais dos DAGs do Pacote North.	58
5.11	Resultados do método <i>PQR_BC1</i> com relação à melhora dos resultados do método <i>BC</i> (vitórias) e de deterioração (derrotas) separados por porcentagem, relativos ao número de cruzamentos iniciais dos DAGs do Pacote Rome.	58

5.12	Gráficos boxplot com a distribuição dos tempos de execução dos métodos <i>BC</i> e <i>PQR_BC1</i> para as densidades iguais ou inferiores a 0,3 para os pacotes North e Rome.	59
5.13	Gráficos boxplot com a distribuição dos tempos de execução dos métodos <i>BC</i> e <i>PQR_BC1</i> para as densidades iguais ou inferiores a 0,3 para os pacotes North e Rome.	60
5.14	Grafos do Pacote North separados por densidades e percentuais de vitórias, derrotas e empates (em termos de cruzamentos) do método <i>PQR_BC2</i> .	63
5.15	Grafos do pacote Rome separados por densidades e percentuais de vitórias, derrotas e empates (em termos de cruzamentos) do método <i>PQR_BC2</i> .	64
5.16	Quantidade de DAGs do Pacote North, agrupadas por percentual de redução de cruzamentos provida pelos métodos <i>BC</i> e <i>PQR_BC2</i> , com relação à quantidade inicial de cruzamentos desses grafos.	65
5.17	Quantidade de DAGs do Pacote Rome, agrupadas por percentual de redução de cruzamentos provida pelos métodos <i>BC</i> e <i>PQR_BC2</i> , com relação à quantidade inicial de cruzamentos desses grafos.	65
5.18	Resultados do método <i>PQR_BC2</i> com relação à melhora dos resultados do método <i>BC</i> (vitórias) e de deterioração (derrotas) separados por porcentagem, relativos ao número de cruzamentos iniciais dos DAGs do Pacote North.	66
5.19	Resultados do método <i>PQR_BC2</i> com relação à melhora dos resultados do método <i>BC</i> (vitórias) e de deterioração (derrotas) separados por porcentagem, relativos ao número de cruzamentos iniciais dos DAGs do Pacote Rome.	66
5.20	Gráficos boxplot com a distribuição dos tempos de execução dos métodos <i>BC</i> e <i>PQR_BC2</i> para as densidades iguais ou inferiores a 0,3 para os pacotes North e Rome.	67
5.21	Gráficos boxplot com a distribuição dos tempos de execução dos métodos <i>BC</i> e <i>PQR_BC2</i> para as densidades iguais ou inferiores a 0,3 para os pacotes North e Rome.	68
5.22	Grafos do Pacote North separados por densidades e percentuais de vitórias, derrotas e empates (em termos de cruzamentos) do método <i>PQR_M1</i> .	71
5.23	Grafos do pacote Rome separados por densidades e percentuais de vitórias, derrotas e empates (em termos de cruzamentos) do método <i>PQR_M1</i> .	72

5.24	Quantidade de DAGs do Pacote North, agrupadas por percentual de redução de cruzamentos provida pelos métodos <i>MEDIAN</i> e <i>PQR_M1</i> , com relação à quantidade inicial de cruzamentos desses grafos.	73
5.25	Quantidade de DAGs do Pacote Rome, agrupadas por percentual de redução de cruzamentos provida pelos métodos <i>MEDIAN</i> e <i>PQR_M1</i> , com relação à quantidade inicial de cruzamentos desses grafos.	73
5.26	Resultados do método <i>PQR_M1</i> com relação à melhora dos resultados do método <i>MEDIAN</i> (vitórias) e de deterioração (derrotas) separados por porcentagem, relativos ao número de cruzamentos iniciais dos DAGs do Pacote North.	74
5.27	Resultados do método <i>PQR_M1</i> com relação à melhora dos resultados do método <i>MEDIAN</i> (vitórias) e de deterioração (derrotas) separados por porcentagem, relativos ao número de cruzamentos iniciais dos DAGs do Pacote Rome.	74
5.28	Gráficos boxplot com a distribuição dos tempos de execução dos métodos <i>MEDIAN</i> e <i>PQR_M1</i> para as densidades iguais ou inferiores a 0,3 para os pacotes North e Rome.	75
5.29	Gráficos boxplot com a distribuição dos tempos de execução dos métodos <i>MEDIAN</i> e <i>PQR_M1</i> para as densidades iguais ou inferiores a 0,3 para os pacotes North e Rome.	76
5.30	Grafos do Pacote North separados por densidades e percentuais de vitórias, derrotas e empates (em termos de cruzamentos) do método <i>PQR_M2</i>	78
5.31	Grafos do pacote Rome separados por densidades e percentuais de vitórias, derrotas e empates (em termos de cruzamentos) do método <i>PQR_M2</i>	79
5.32	Quantidade de DAGs do Pacote North, agrupadas por percentual de redução de cruzamentos provida pelos métodos <i>MEDIAN</i> e <i>PQR_M2</i> , com relação à quantidade inicial de cruzamentos desses grafos.	79
5.33	Quantidade de DAGs do Pacote Rome, agrupadas por percentual de redução de cruzamentos provida pelos métodos <i>MEDIAN</i> e <i>PQR_M2</i> , com relação à quantidade inicial de cruzamentos desses grafos.	80

5.34	Resultados do método <i>PQR_M2</i> com relação à melhora dos resultados do método <i>MEDIAN</i> (vitórias) e de deterioração (derrotas) separados por porcentagem, relativos ao número de cruzamentos iniciais dos DAGs do Pacote North.	81
5.35	Resultados do método <i>PQR_M2</i> com relação à melhora dos resultados do método <i>MEDIAN</i> (vitórias) e de deterioração (derrotas) separados por porcentagem, relativos ao número de cruzamentos iniciais dos DAGs do Pacote Rome.	81
5.36	Gráficos boxplot com a distribuição dos tempos de execução dos métodos <i>MEDIAN</i> e <i>PQR_M2</i> para as densidades iguais ou inferiores a 0,3 para os pacotes North e Rome.	82
5.37	Gráficos boxplot com a distribuição dos tempos de execução dos métodos <i>MEDIAN</i> e <i>PQR_M2</i> para as densidades iguais ou inferiores a 0,3 para os pacotes North e Rome.	83

Lista de Tabelas

5.1	Diferença entre as médias de tempo de <i>BC</i> e <i>PQR_BC1</i>	61
5.2	Resumo dos resultados obtidos pela execução dos algoritmos <i>PQR_BC1</i> e <i>BC</i> . . .	62
5.3	Diferença entre as médias de tempo de <i>BC</i> e <i>PQR_BC2</i>	69
5.4	Resumo dos resultados obtidos pela execução dos algoritmos <i>PQR_BC2</i> e <i>BC</i> . . .	70
5.5	Diferença entre as médias de tempo de <i>MEDIAN</i> e <i>PQR_M1</i>	77
5.6	Resumo dos resultados obtidos pela execução dos algoritmos <i>PQR_M1</i> e <i>MEDIAN</i>	77
5.7	Diferença entre as médias de tempo de <i>MEDIAN</i> e <i>PQR_M2</i>	84
5.8	Resumo dos resultados obtidos pela execução dos algoritmos <i>PQR_M2</i> e <i>MEDIAN</i>	85
5.9	Resumo dos resultados obtidos da execução dos algoritmos <i>PQR_BC1</i> , <i>PQR_BC2</i> , <i>PQR_M1</i> e <i>PQR_M2</i>	86

Lista de Algoritmos

1	Algoritmo <i>longest path ranking</i>	14
2	Algoritmo <i>layer-by-layer sweep</i>	17
3	Algoritmo <i>Greedy-Switch</i>	25
4	Algoritmo <i>Split</i>	26
5	Algoritmo BC	27
6	Algoritmo MEDIAN	28
7	Algoritmo de reordenação de DAGs utilizando Árvores PQR	35
8	Algoritmo <i>PQR_BC1</i>	38
9	Algoritmo <i>PQR_M1</i>	40
10	Algoritmo <i>PQR_BC2</i>	42
11	Algoritmo <i>PQR_M2</i>	42

Capítulo 1

Introdução

Grafos são estruturas amplamente utilizadas para modelagem de problemas de natureza relacional. Esses problemas estão nas mais diversas áreas do conhecimento, tais como computação (estudo de redes de computadores), biologia (estudo de redes neurais), química (estudo de ligações atômicas) e sociologia (estudo de redes sociais). Os vértices de um grafo podem representar as entidades de um determinado problema, e suas arestas, o relacionamento entre as entidades.

Alguns problemas são mais facilmente compreendidos e interpretados quando é possível representá-los graficamente. Dentre eles, estão os problemas de natureza hierárquica, tais como: gráficos de PERT (*Program Evaluation and Review Technique*), circuitos VLSI (*Very Large Scale Integration*), árvores genealógicas, dependência de classes de software, diagramas entidade-relacionamento e representação de conhecimento (TANG; CHEN, 2008) (BATTISTA et al., 1999). Um tipo específico de grafo, conhecido como DAG (*directed acyclic graph* - grafo acíclico direcionado), é amplamente utilizado para modelar essas estruturas. DAGs são grafos cujas arestas possuem sentido e não produzem ciclos (BATTISTA et al., 1994). Essas características permitem que uma representação gráfica de uma estrutura hierárquica seja mais facilmente interpretada. Como exemplo, a Figura 1.1 mostra uma hierarquia de empregados e empregadores: em 1.1a, adota-se a abordagem de descrição narrativa da hierarquia e em 1.1b utiliza-se um desenho da mesma estrutura. É possível observar claramente em 1.1b, o relacionamento entre as entidades, enquanto que em 1.1a esta informação fica mais difícil de ser encontrada.

A utilização de representações visuais permite uma compreensão mais rápida de alguns problemas, já que essas estruturas possuem propriedades gráficas que são mais rapidamente processadas

Pessoa 8 trabalha para **Pessoa 4 e 7**. **Pessoa 7** trabalha para **Pessoa 6**;

Pessoa 6 trabalha para **Pessoas 5 e 1**;

Pessoa 4 trabalha para **Pessoa 2**. **Pessoa 2** trabalha para **Pessoa 1**;

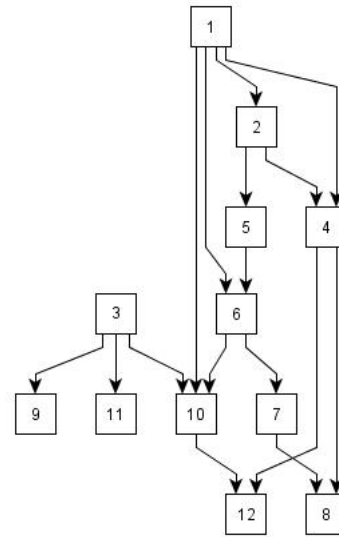
Pessoa 10 trabalha para **peessoas 6, 3 e 1**;

Pessoa 12 trabalha para **Pessoas 10 e 4**;

Pessoa 4 trabalha para **Pessoas 2 e 1**;

Pessoa 11 trabalha para **Pessoas 4 e 3**;

Pessoa 9 trabalha para **Pessoa 3**.



(a) Descrição narrativa

(b) Representação gráfica

Figura 1.1: Exemplo de hierarquia. Em 1.1a, descrição narrativa de hierarquia de empregados e empregadores. Em 1.1b, representação visual da mesma.

pela percepção visual do que a representação textual (MAZZA, 2009). Apesar de grafos auxiliarem na representação visual de problemas que envolvem estruturas relacionais, desenhá-los manualmente de forma que sirvam de auxílio para o entendimento das estruturas que modelam se torna uma tarefa difícil quando o número de vértices e arestas é elevado. Como exemplo, na Figura 1.2 é possível observar o desenho aleatório de dois grafos: em 1.2a, um grafo com cinco vértices e cinco arestas; em 1.2b, um grafo com 100 vértices e 99 arestas. Vale ressaltar que grafos com milhares de nós e arestas são facilmente encontrados, por exemplo, no estudo de redes complexas (CHEN, 2006).

O desenho automático de DAGs é alvo de diversas pesquisas no campo de *Visualização de Informação* e da *Matemática Discreta* (BATTISTA et al., 1999) (GANSNER et al., 1998). Garantir uma visualização que possibilite um fácil entendimento faz com que seja necessária a utilização de critérios estéticos de desenho (MUTZEL; EADES, 2002), os quais melhoram a percepção de padrões e ajudam na compreensão da representação visual de grafos (EADES; SUGIYAMA, 1991)(PURCHASE et al., 2002).

Nesse sentido, é imprescindível destacar a importância do uso de ferramentas que gerem um desenho automático de grafos (MUTZEL; EADES, 2002). A Figura 1.3 apresenta o grafo da Fi-

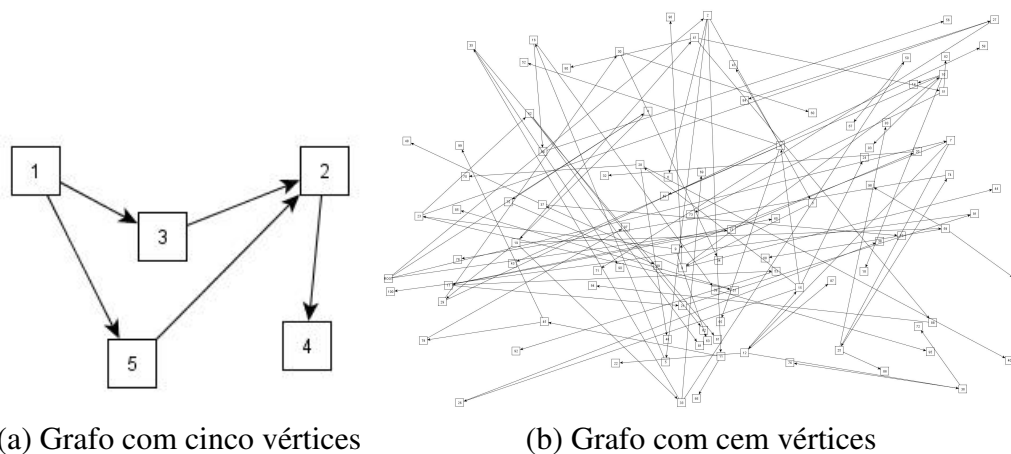


Figura 1.2: Desenho de grafos com tamanhos diferentes. Em 1.2a, grafo com cinco vértices e cinco arestas; em 1.2b, grafo com 100 vértices e 99 arestas

Figura 1.2b depois de processada por um programa que atende os critérios estéticos de desenho de hierarquias.

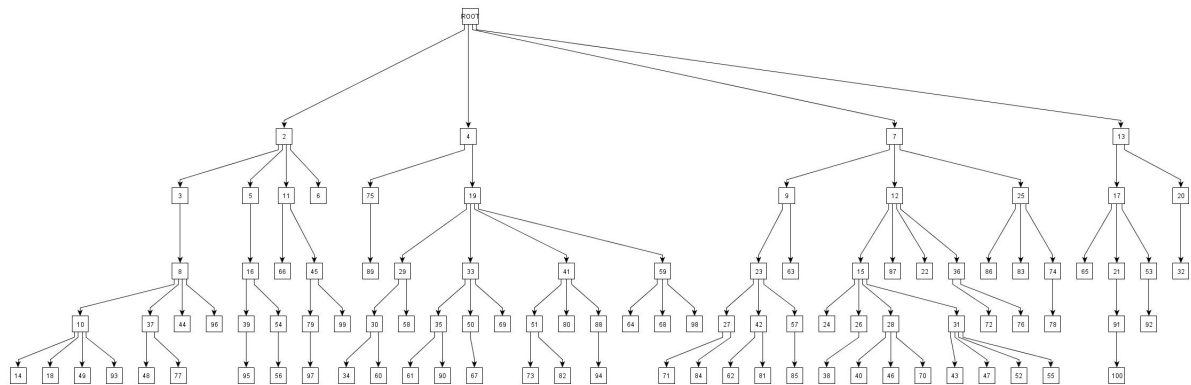


Figura 1.3: Figura 1.2b depois de processada por um programa que atende os critérios estéticos de desenho de hierarquias.

Um dos critérios mais importantes para o desenho de DAGs, como descrito por Purchase (1997), é o de minimizar o cruzamento entre arestas. A Figura 1.4 mostra o desenho de um DAG (Figura 1.4a) que teve seus vértices reposicionados para que o cruzamento entre arestas fosse reduzido (Figura 1.4b). Note que na Figura 1.4b, as relações entre os vértices ficam mais claras.

Além de seguirem critérios estéticos de desenho, uma outra característica desejável para programas de desenho automático de DAGs é a possibilidade de executarem em um tempo viável (WARFIELD, 1977) (BATTISTA et al., 1994), de tal modo que usuários possam interagir com o desenho através do uso de computadores. Sobre a importância de se prover essa interação em es-

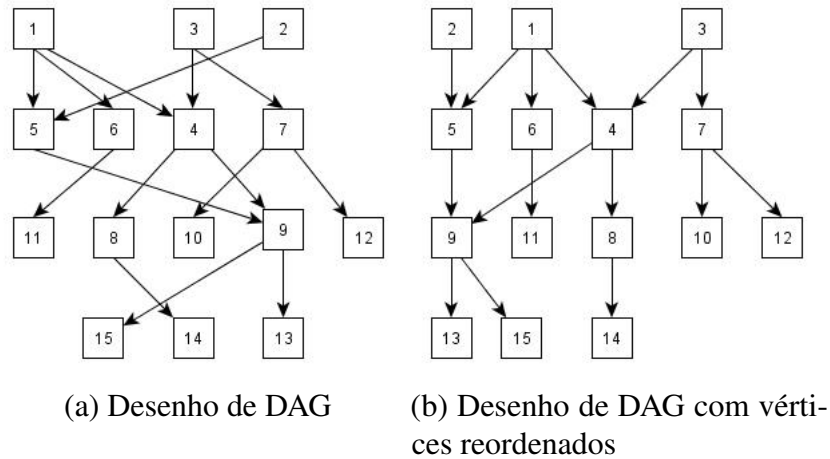


Figura 1.4: Exemplo de redução de cruzamentos em um DAG

estruturas visuais, Card et al. (1999) afirmam que representações visuais interativas podem ampliar a cognição do usuário, dando a ele a capacidade de obter novos *insights* do diagrama que modela o problema tratado.

Dentre os algoritmos que se destacam com relação aos critérios redução de cruzamentos e tempo de execução estão os métodos baricêntrico (método *BC*) (SUGIYAMA et al., 1981) e o método mediano (método *MEDIAN*) (EADES; WORMALD, 1994). Esses algoritmos se baseiam em um cálculo de posições médias e medianas, respectivamente, para determinar a ordem que os vértices aparecem no DAG, obtendo uma boa relação custo-benefício entre redução de cruzamentos e tempo gasto.

Recentemente, Melo (2012) introduziu técnicas que utilizam uma estrutura de dados conhecida como Árvore PQR (TELLES; MEIDANIS, 2005) para resolver problemas de seriação de matrizes, obtendo bons resultados quando comparados à utilização de técnicas que envolvem o método *BC*. Siirtola e Makinen (2005) relacionam o problema de seriação de matrizes com os problemas de redução de cruzamentos descritos neste trabalho. A partir dessa associação e dos resultados obtidos por Melo (2012), é possível considerar a utilização de Árvores PQR como uma estratégia promissora para redução de cruzamentos em DAGs.

1.1 Objetivos

Dado que Árvores PQR têm um grande potencial em resolver problemas de seriação de matrizes, e que estes problemas tem forte relação com o problema de redução de cruzamentos em DAGs, o objetivo desse trabalho é testar a hipótese que a utilização de Árvores PQR associadas aos métodos *BC* e *MEDIAN* melhora o desempenho destes com relação à redução do número de cruzamentos, sem deteriorar significativamente o tempo de execução com vistas à aplicação em estruturas visuais interativas.

1.2 Organização do texto

O texto que se segue está organizado da seguinte forma:

- O Capítulo 2 apresenta a revisão bibliográfica sobre desenho de grafos e DAGs especificamente, introduzindo conceitos e técnicas sobre o desenho automático dessas estruturas.
- O Capítulo 3 apresenta os principais algoritmos de redução de cruzamentos em DAGs, enfatizando os métodos *BC* e *MEDIAN*.
- O Capítulo 4 apresenta a estrutura de dados Árvore PQR, mostrando o seu funcionamento e suas áreas de aplicabilidade.
- O Capítulo 4.2 descreve os algoritmos propostos neste trabalho para aperfeiçoar os métodos *BC* e *MEDIAN*.
- O Capítulo 5 analisa os resultados obtidos do uso dos algoritmos propostos neste trabalho e os compara com os resultados de *BC* e *MEDIAN* utilizando pacote de grafos já consagrados na literatura.
- O Capítulo 6 apresenta a conclusão do trabalho.

Capítulo 2

Desenho de grafos

Conforme apresentado no Capítulo 1, o desenho automático de grafos se torna um recurso muito útil para analisar problemas determinados por objetos e suas relações, a fim de se buscar padrões e tendências. Para a criação desses desenhos, existem algoritmos que visam definir o posicionamento dos elementos do grafo de modo a aplicar critérios estéticos que facilitem sua compreensão. Na Seção 2.1, são mostradas as principais convenções de desenho de grafos encontradas na literatura. Na Seção 2.2, são descritos os principais critérios estéticos de desenho de grafos e, particularmente, de DAGs, mostrando o efeito da utilização deles na interpretação dos desenhos. Na Seção 2.3.1 é apresentada uma das abordagens mais tradicionais para o desenho de DAGs, utilizada durante os experimentos realizados neste trabalho e na Seção 2.3.2 são apresentadas abordagens alternativas a esta.

2.1 Convenções de desenho de grafos

Um dos fatores que mais contribuem para a qualidade do desenho de grafos está no conhecimento da classe ao qual o grafo pertence. Saber se o grafo é acíclico, planar ou direcionado, permite que uma imagem gerada por um programa de computador ilustre melhor as propriedades desse grafo. Uma vez descoberta essa classe, é possível estabelecer convenções (regras de desenho) a fim de se representar graficamente o grafo.

Uma das convenções mais comuns é a utilização de um desenho de linhas contínuas para representar arestas (BATTISTA et al., 1999), também chamado de conectividade (CARD et al.,

1999). A Figura 2.1 apresenta dois desenhos de grafos que seguem essa regra. Em 2.1a, é possível observar que o grafo possui arestas de linhas contínuas que se dividem durante o trajeto de suas conexões entre os vértices 1 e 4, 4 e 5. Em 2.1b, ocorre o mesmo para os vértices 6 e 7, porém as linhas se alternam entre seguimentos horizontais e verticais.

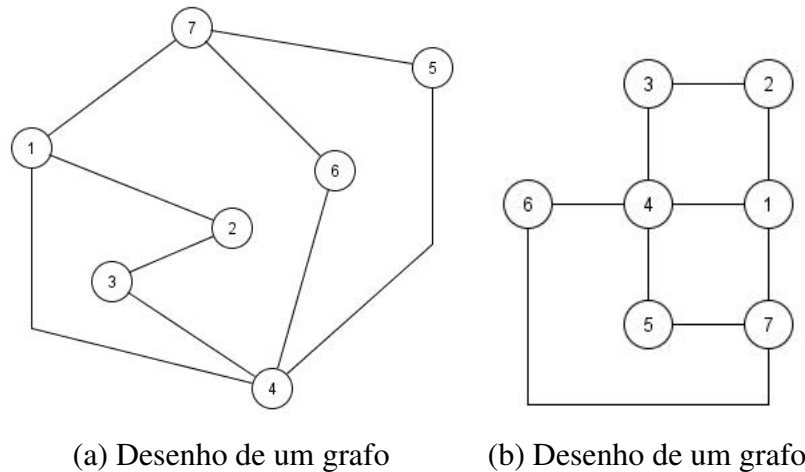


Figura 2.1: Figura de grafos que seguem a convenção de conectividade.

Essa regra é convenientemente adotada quando é necessária uma flexibilidade no desenho para se conectar vértices distantes. Porém, deve-se destacar que o uso excessivo de curvas (conexões entre arestas) em uma mesma aresta dificulta o entendimento do desenho (BATTISTA et al., 1999).

Em especial, a convenção geral para dígrafos acíclicos (ou grafos acíclicos direcionados) é adotar um desenho em que todas as arestas apontem para uma mesma direção, vertical ou horizontalmente, como mostra a Figura 2.2.

2.2 Critérios Estéticos

Outro fator de grande influência no entendimento do desenho de grafos está nas propriedades gráficas específicas chamadas de critérios estéticos (BATTISTA et al., 1999). Essas regras facilitam a leitura e interpretação dos desenhos, garantindo sua qualidade (MUTZEL; EADES, 2002), (SUGIYAMA et al., 1981). Dentre os critérios mais comuns, pode-se destacar:

- Minimizar o número total de cruzamentos entre arestas;

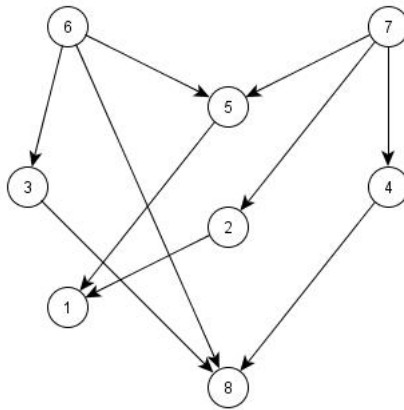


Figura 2.2: Figura de um grafo que segue convenções de desenho para DAGs.

- Minimizar a área total do desenho;
- Minimizar a soma do comprimento das arestas;
- Minimizar o número total de curvas ao longo das arestas.

Battista et al. (1999) destacam que geralmente esses critérios se conflitam e, mesmo quando isso não ocorre, é difícil encontrar um algoritmo que lide com todos os critérios necessários ao mesmo tempo. A Figura 2.3 apresenta um exemplo comum de conflito de critérios, em que é necessário aumentar o tamanho da área do desenho para minimizar o número de cruzamentos entre arestas.

A saída encontrada para essa situação é a adoção de uma ordem de precedência para cada critério estético a ser seguido (MUTZEL; EADES, 2002). Desse modo, algoritmos especialistas podem lidar com critérios estéticos específicos em uma ordem de precedência predefinida. Essa abordagem também permite determinar, em ordem, qual dos critérios é mais importante no desenho.

Mesmo tendo sido escolhida a maneira como deve ser desenhado o grafo, e tendo sido estabelecida a ordem dos critérios, é necessário mensurar quão bom é um desenho de um grafo. Nesse sentido, Purchase (1997) realizou um experimento empírico em que cinco critérios são avaliados: número de curvas nas arestas, número de cruzamentos, ângulo mínimo entre arestas, desenho ortogonal em grade e simetria do desenho. O objetivo foi encontrar qual deve ser a ordem de prioridade deles, considerando quais deles causam mais influência na leitura do desenho de grafos. Utilizando diferentes desenhos de um mesmo grafo, ele constatou que reduzir o número de cruzamentos entre arestas é o critério que mais influencia na compreensão das informações relacionais contidas no

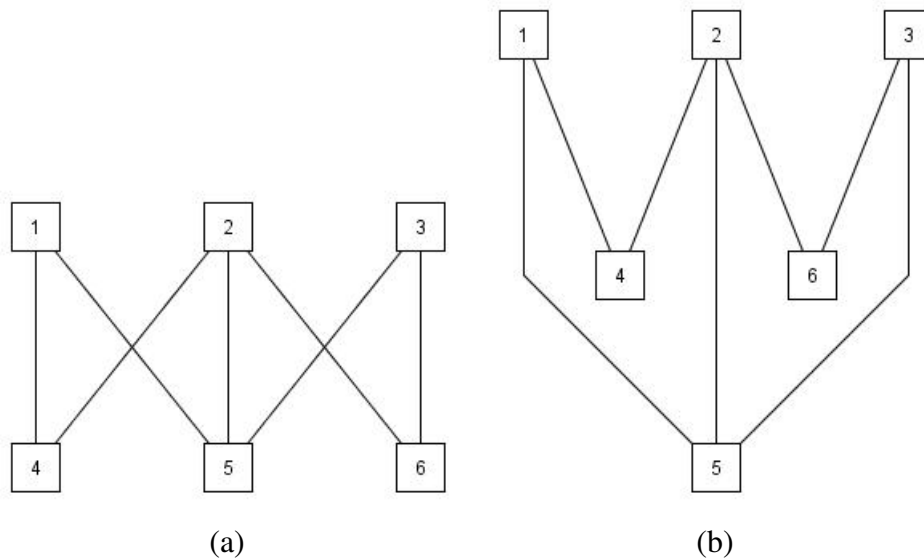


Figura 2.3: Modificações feitas no desenho de um DAGs para atender critérios estéticos. Em 2.3a, o DAG é desenhado de tal modo que ocupe a menor altura possível. Em 2.3b, o DAG é desenhado de tal modo que possua o menor número de cruzamentos de arestas possível.

desenho dos grafos e que, portanto, deve ser o critério de maior prioridade.

2.3 Desenho de DAGs

Da mesma forma que para grafos em geral, pode-se definir regras de representação visual também para DAGs. A convenção mais adotada nesses casos propõe a criação de um desenho cuja direção do fluxo possua um único sentido, os vértices do grafo sejam distribuídos igualmente sobre a área do desenho e o cruzamento entre arestas seja o menor possível (EADES; SUGIYAMA, 1991).

As técnicas mais comuns de desenhos de DAGs utilizam uma representação proposta por Sugiyama et al. (1981). Eles definem um DAG em camadas como sendo um grafo direcionado $G = (V, E)$ em que o conjunto de nós V pode ser particionado em subconjuntos L_1, L_2, \dots, L_n , chamados de camadas, e cada aresta pertencente a E é única e conecta apenas vértices de camadas diferentes, apontando sempre para uma mesma direção. Um DAG em camadas é denotado então por $G = (V, E, n)$ em que n define o número de camadas. A Figura 2.4 mostra o desenho de um DAG em camadas.

As camadas podem ser distribuídas horizontal ou verticalmente. A Figura 2.5 mostra algumas

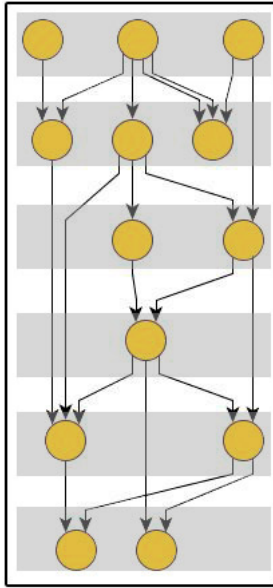


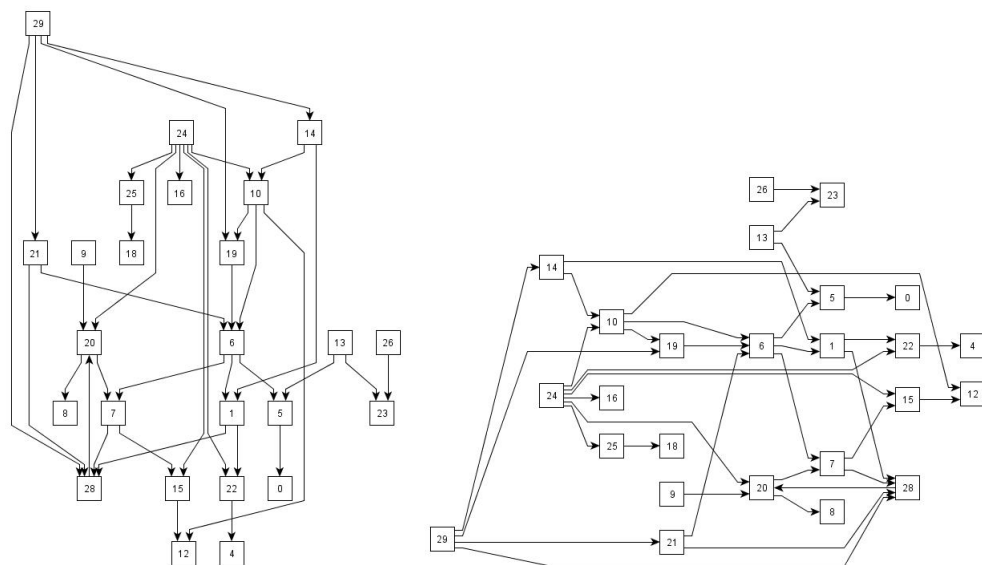
Figura 2.4: Desenho de um DAG em camadas, representadas por um retângulo cinza.

representações possíveis de DAGs. Em 2.5a, observa-se o desenho de um DAG cujas camadas são dispostas na horizontal e as arestas apontam de cima para baixo. Em 2.5b, o mesmo DAG, com as camadas dispostas na vertical e as arestas apontando da esquerda para direita. Em 2.5c, o DAG tem suas camadas dispostas na horizontal, porém, suas arestas apontam de baixo para cima. Neste documento, sem perda de generalidade, será adotado o desenho cujas camadas são distribuídas horizontalmente e as arestas apontam de uma camada superior para uma camada inferior.

Nas Seções 2.3.1 e 2.3.2 serão apresentadas as principais técnicas de desenho de DAGs.

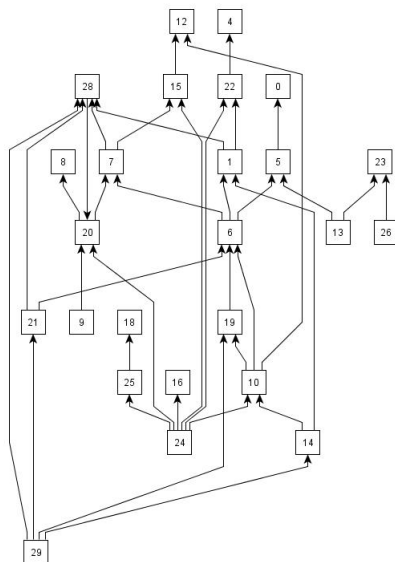
2.3.1 Modelo de Sugiyama

Uma técnica muito comum para o desenho automático de DAGs foi apresentada em 1981 por Sugiyama et al. (1981), e ficou conhecida como *modelo de Sugiyama (Sugiyama framework)*. A técnica é dividida em três passos: atribuição de camadas, na qual os vértices são escolhidos e distribuídos em camadas, de modo que cada vértice ocupe apenas uma camada; descruzamento de arestas, em que a ordem posicional dos vértices na camada é definida, visando cruzar o mínimo de arestas possível; atribuição de coordenadas, em que são definidas as coordenadas verticais e horizontais para todos os vértices pertencentes ao DAG. Para cada passo do *modelo de Sugiyama*, existem algoritmos desenvolvidos para atender os critérios estéticos do desenho. Um desafio desses



(a) DAG em camadas horizontais

(b) DAG em camadas verticais



(c) DAG em camadas horizontais

Figura 2.5: Desenho de um DAG. Em 2.5a, camadas dispostas na horizontal com arestas apontando para baixo. Em 2.5b, camadas dispostas na vertical com arestas apontando para direita. Em 2.5c, camadas dispostas na horizontal com arestas apontando para cima.

algoritmos é conciliar esses critérios com eficiência (tempo de execução) (BATTISTA et al., 1999).

Definição de camadas

Em algumas aplicações, a definição das camadas e dos vértices pertencentes a elas é feita pelo usuário, como a representação de dependências de disciplinas de um curso e o relacionamento de pré-requisitos entre elas, mostrada na Figura 2.6. Neste exemplo cada disciplina tem um semestre predefinido, que corresponde a uma camada do DAG apresentado. Em outras situações, deseja-se que um programa distribua automaticamente camadas e vértices.

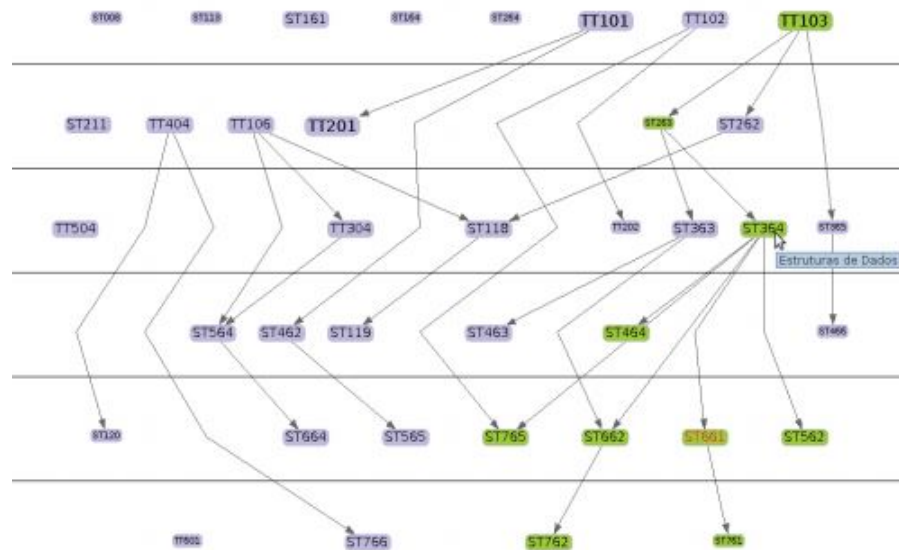


Figura 2.6: Representação visual de dependências de disciplinas de um curso de graduação. Adaptado de Silva et al. (2012)

O passo de atribuição de camadas do *modelo de Sugiyama* consiste em atribuir os vértices a camadas horizontais. Sendo $G = (V, E)$ um dígrafo acíclico, dividir G em camadas é particionar V em subconjuntos L_1, L_2, \dots, L_n , tal que se $(u, v) \in E$, $u \in L_i$ e $v \in L_j$, então $i < j$. A altura do grafo é dada pelo número de camadas e a largura do grafo é dada pelo número de vértices pertencentes à camada com maior número de vértices (BATTISTA et al., 1999).

Por exemplo, o algoritmo *longest path ranking* é um algoritmo de definição de camadas (KAUFMANN; WAGNER, 2001) que pode ser executado em tempo linear em dígrafos acíclicos definindo um número mínimo de camadas. O Algoritmo 1 é uma versão do algoritmo *longest path ranking*. Ele define dois conjuntos de vértices U e Z vazios inicialmente. Quando um vértice é selecionado, ele é adicionado à camada *camadaAtual* e adicionado ao conjunto U . Assim, U representa o conjunto de todos os vértices já adicionados às camadas. O conjunto Z representa todos os vértices adicionados

à camada abaixo da *camadaAtual*. Um novo vértice v a deve obedecer simultaneamente duas condições: v não pode estar associado a nenhuma camada ainda, e todas as suas arestas de saída devem incidir sobre vértices já associados à camadas abaixo da *camadaAtual* - ou seja, devem pertencer a Z . Na descrição do algoritmo, $N_G^+(v)$ representa os vértices em G para os quais as arestas de saída de v apontam. Para esta versão do algoritmo, as camadas são consideradas de baixo para cima, sendo a primeira camada a camada inferior.

Algoritmo 1: Algoritmo *longest path ranking*

Entrada: DAG $G = (V, E)$

Saída: DAG $G = (V, E, n)$

início

$U \leftarrow \emptyset$;

$Z \leftarrow \emptyset$;

$camadaAtual \leftarrow 1$;

enquanto $U \neq V$ **faça**

 Selecione vértice $v \in V \setminus U$ com $N_G^+(v) \subseteq Z$;

se v foi selecionado **então**

 Atribua v à camada de numero $camadaAtual$;

$U \leftarrow U \cup \{v\}$;

senão

$camadaAtual \leftarrow camadaAtual + 1$;

$Z \leftarrow Z \cup U$

fim se

fim enqto

fim

Para exemplificar este algoritmo, a Figura 2.7a mostra um DAG com posição aleatória de seus vértices; por sua vez, a Figura 2.7b apresenta o resultado da aplicação desse algoritmo sobre o mesmo grafo.

Quando uma aresta se expande por mais de uma camada, são criados *dummy nodes* (falsos vértices) nas camadas adjacentes pelas quais ela passa. A adição de *dummy nodes* é necessária devido à execução dos algoritmos de redução de cruzamentos, que são processados tomando-se as camadas do DAG duas a duas, em um processo conhecido como *layer-by-layer sweep*, descrito em detalhes adiante. Um DAG em camadas cujas arestas são definidas apenas em camadas adjacentes através da adição de *dummy nodes* é chamado de *proper hierarchy* (hierarquia própria). A Figura 2.8 mostra a criação de *dummy nodes* (representados por círculos pequenos) para o DAG apresentado na Figura 2.7b, nos casos em que as arestas se estendem por mais do que uma camada.

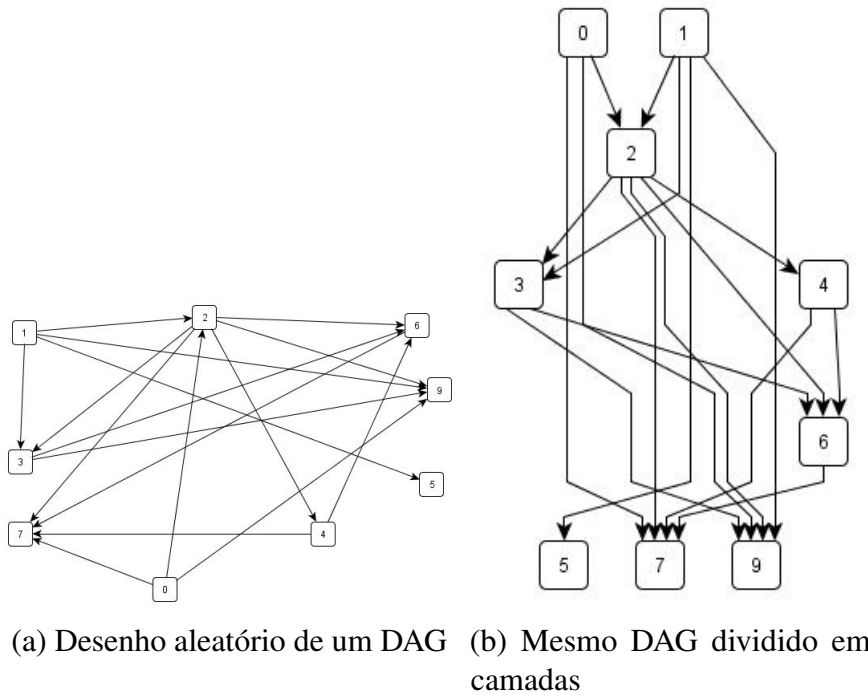


Figura 2.7: Representação da atribuição de camadas para um DAG

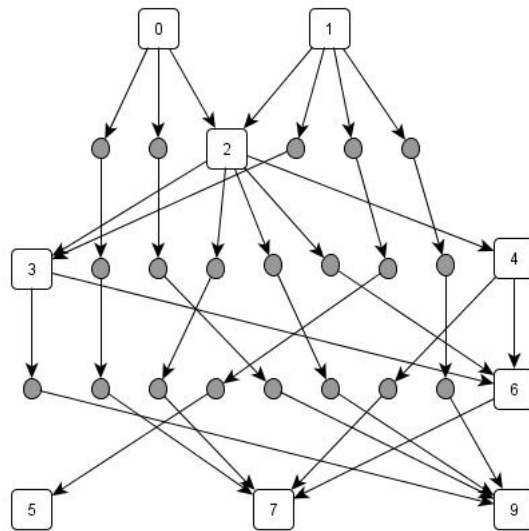


Figura 2.8: Adição de *dummy nodes* ao DAG da Figura 2.7b.

Algoritmos de atribuição de camadas utilizam técnicas para minimizar o tamanho e largura do desenho, além de criar um número mínimo de vértices falsos possível (os quais aumentam a complexidade dos algoritmos), a fim de produzir desenhos mais agradáveis (TANG; CHEN, 2008), (NIKOLOV et al., 2005) e (GANSNER et al., 1993).

Redução de cruzamentos

O segundo passo do *modelo de Sugiyama* consiste em minimizar o número de cruzamentos entre as arestas que anteriormente foram dispostas em camadas. Esse passo não se preocupa em definir coordenadas exatas no eixo x , mas apenas em encontrar uma ordem para cada vértice em cada camada, já que o número de cruzamentos não depende do desenho das arestas do DAG, mas apenas da ordem dos vértices de cada camada. Através de um processo de permutação dos vértices de uma mesma camada, esse passo visa encontrar uma ordem para os vértices na qual o número de cruzamentos entre as arestas seja o menor possível. Uma das técnicas mais comumente utilizadas, segundo Battista et al. (1999), é conhecida como *layer-by-layer sweep* e é apresentada no Algoritmo 2.

O Algoritmo 2 é uma das versões da técnica *layer-by-layer sweep*, que além de receber uma hierarquia própria como parâmetro de entrada, recebe também um número de falhas (*fails*), escolhido pelo usuário do algoritmo. Esse número determina quantas vezes consecutivas o algoritmo pode falhar em reduzir cruzamentos, em que este número de cruzamentos é calculado pela função *calculaCruzamentos*, depois de uma execução de *iteraHierarquia*. Enquanto esse número não é alcançado, um processo de redução de cruzamentos é acionado através da função *iteraHierarquia* que retorna o novo número de cruzamentos da hierarquia própria. Esse processo ocorre da seguinte forma:

- Escolhe-se uma ordem fixa para os vértices de uma camada L_i , em que i é o número da primeira camada caso *direção* seja igual a *TopDown*, ou o número da última camada caso *direção* seja igual a *DownTop*. Em seguida, os vértices pertencentes à camada L_{i+1} são reordenados quando *direção* é igual a *TopDown*, ou os vértices pertencentes à camada L_{i-1} são reordenados quando *direção* é igual a *DownTop*.
- Ao término da reordenação, i é incrementado em um, e o processo é repetido até atingir a última camada (*TopDown*) ou a primeira camada (*DownTop*) da *hierarquia própria*.

Esse processo se repete até que seja alcançado um dado número de fails, encerrando o algoritmo. Os métodos de redução de cruzamentos, detalhados no Capítulo 3, podem ser escolhidos arbitrariamente neste processo, desde que atuem em um DAG previamente organizado em camadas. A Figura 2.9 mostra o DAG apresentado na Figura 2.8, após passar pelo processo de *layer-by-layer sweep*.

Algoritmo 2: Algoritmo *layer-by-layer sweep*

Entrada: hierarquia própria H , $fails$

início

```
   $k\_antigo \leftarrow \text{calculaCruzamento}(H)$  ;  
   $hierarquia\_anterior \leftarrow H$  ;  
   $nfails \leftarrow fails + 1$  ;  
   $direo \leftarrow TopDown$  ;  
  se  $k \neq 0$  então  
    repita  
       $iteraHierarquia(direcao)$  ;  
       $k\_novo \leftarrow \text{calculaCruzamento}(H)$  ;  
      se ( $direcao = TopDown$ ) então  
        ( $direcao \leftarrow DownTop$  ;  
      senão  
        ( $direcao \leftarrow TopDown$  ;  
      fim se  
      se  $k\_novo < k\_antigo$  então  
         $hierarquia\_anterior \leftarrow H$  ;  
        se  $k\_novo = 0$  então  
          fim do algoritmo ;  
        fim se  
         $k\_antigo = k\_novo$  ;  
         $nfails \leftarrow fails + 1$   
      senão  
        ( $nfails \leftarrow nfails - 1$   
      fim se  
    até  $nfails > 0$  ;  
  fim se  
fim
```

Atribuição de coordenadas

O último passo do *modelo de Sugiyama* consiste em atribuir as coordenadas finais para os vértices do DAG, além de substituir os *dummy nodes* criados no primeiro passo por *curvas* que farão com que o desenho das arestas se torne um desenho de polígonos abertos. Para alcançar um desenho final satisfatório, os algoritmos utilizados nesse passo precisam gerar desenhos compactos, nos quais não haja sobreposição de arestas nos vértices (GANSNER et al., 1993). Como exemplo, a Figura 2.10 mostra o DAG apresentado na Figura 2.9, após a execução do último passo do *modelo de Sugiyama*.

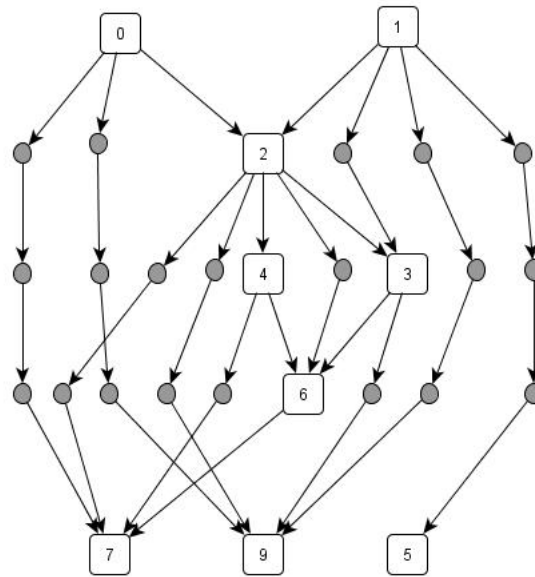


Figura 2.9: DAG da Figura 2.8 após processo de redução de cruzamentos.

2.3.2 Outras abordagens

Existem abordagens alternativas ao *modelo de Sugiyama*. Dentre elas podem-se destacar a técnica de representação planar e a representação tridimensional.

Representação planar

Uma técnica de desenho de DAGs alternativa ao *modelo de Sugiyama* baseia-se em planarização (*upward planarization*) apresentada por Chimani et al. (2010a) para minimizar o número de cruzamentos entre arestas e gerar um desenho automático do grafo. Chimani et al. (2010a) definem um grafo planar a partir de um DAG removendo arestas deste grafo (quando necessário). O subgrafo obtido desse processo (chamado de *feasible upward planar subgraph*) é disposto de modo que todas as arestas apontem para uma mesma direção monotonicamente, convenção discutida na Seção 2.1. Após essa etapa, os vértices que foram removidos são reinsertidos ao grafo, e são criados *dummy nodes* no lugar do cruzamento das arestas, caso existam, para que o subgrafo continue planar. Isso é necessário para garantir que todos os vértices sejam reinsertidos de tal maneira que o critério de desenho seja obedecido. Ao término dessa etapa, os falsos vértices são substituídos pelos respectivos cruzamentos entre as arestas.

A Figura 2.11a mostra um DAG com duas arestas selecionadas para remoção. A Figura 2.11b

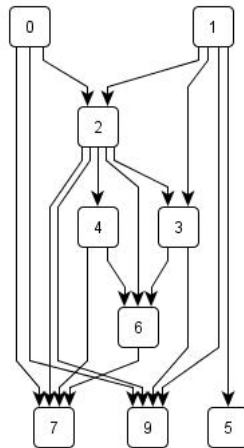


Figura 2.10: DAG da Figura 2.9 após processo de remoção de *dummy nodes* e atribuição de coordenadas.

mostra o subgrafo consequente da remoção das arestas. A Figura 2.11c mostra a adição de *dummy nodes* entre os vértices (4,3) e (6,5). A Figura 2.11d mostra o DAG após o término do método *upward planarization*.

Dos testes realizados por (CHIMANI et al., 2010a), pode-se destacar que essa técnica produziu desenhos com menor número de cruzamentos quando comparados à métodos baseados em *layer-by-layer sweep*, apesar do tempo gasto para a sua execução ser superior ao desses métodos. Além disso, o desenho é gerado de tal forma que não é possível estabelecer a separação dos vértices em camadas. Para resolver essa questão, Chimani et al. (2010b) apresentam uma técnica que utiliza o resultado da planarização do DAG para calcular as camadas para cada vértice, tal como definido no *modelo de Sugiyama* (SUGIYAMA et al., 1981). Ambas as técnicas de planarização não permitem separar o passo de atribuição de vértices e camadas do passo de redução de cruzamentos, dificultando sua utilização em sistemas que permitem a interação com usuários, como exemplo o CourseViewer (SILVA et al., 2012).

Representação tridimensional

Hong e Nikolov (2005) propõem utilizar o *modelo de Sugiyama* e estendê-lo para uma representação tridimensional. Nessa proposta, os passos do *modelo de Sugiyama* são mantidos, e é adicionada uma nova etapa, entre os passos de atribuição de camadas e ordenação de vértices, que divide as camadas do DAG em dois planos paralelos. Assim, os vértices de cada camada, ficam dispostos em

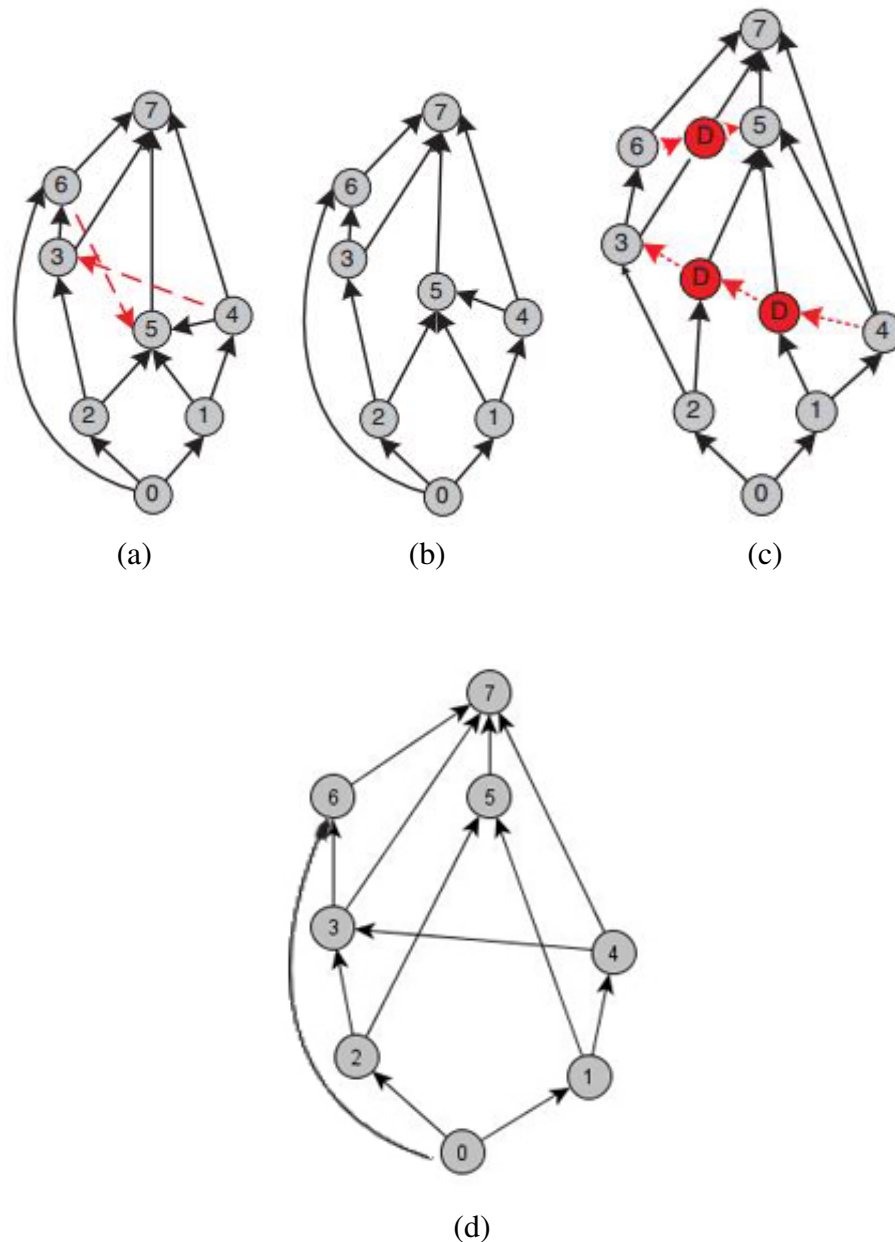
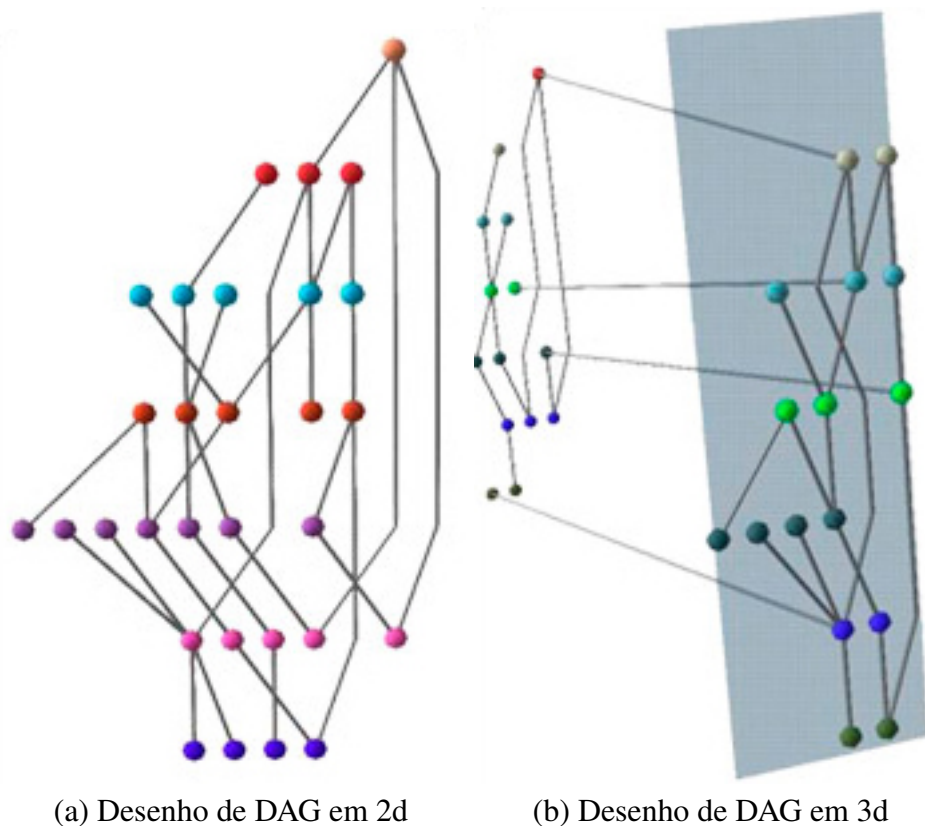


Figura 2.11: Passos da técnica *upward planarization*. Adaptado de Chimani et al. (2010a).

duas paredes imaginárias, paralelas, como mostra a Figura 2.12.

Uma versão adaptada dos métodos utilizados na abordagem clássica foi desenvolvida para o passo de reordenação dos vértices nas camadas. O desenho final consiste em dois planos paralelos, contendo desenhos em 2D, com algumas arestas conectando as camadas entre os planos. Para evitar problemas de visualização, como oclusão, Hong e Nikolov (2005) propõem a utilização de técnicas de navegação em que haja a possibilidade de movimento de câmera e visualização separada de cada



(a) Desenho de DAG em 2d

(b) Desenho de DAG em 3d

Figura 2.12: Técnica tridimensional de desenho de DAG. Adaptado de (HONG; NIKOLOV, 2005)

parede.

Apesar da utilização de dois planos para representar um DAG gerar um desenho com menos cruzamentos, na falta de recursos gráficos tridimensionais, a utilização da representação de DAGs em planos paralelos pode maximizar a complexidade da visualização, possivelmente dificultando a utilização desta técnica.

Capítulo 3

Algoritmos para redução de cruzamentos em DAGs

Minimizar o cruzamento entre arestas de um DAG é um dos critérios estéticos que mais influencia em seu entendimento, como visto na Seção 2.2. O problema de redução de cruzamentos em DAGs é conhecido como MLCM (*multi-layer crossing minimization*). Encontrar uma solução para um MLCM equivale a encontrar uma ordem para os vértices de cada camada de seu DAG de tal modo que o número de cruzamentos de arestas seja o menor possível.

Esse capítulo apresenta os principais algoritmos de redução de cruzamentos em DAGs. Na Seção 3.1, serão apresentados algoritmos heurísticos que utilizam a técnica *layer-by-layer sweep* como parte de sua estratégia para reduzir cruzamentos em DAGs. Na Seção 3.2, serão apresentados métodos alternativos de redução de cruzamentos em DAGs, baseados em meta-heurísticas e programação linear.

3.1 Redução de cruzamentos com *layer-by-layer sweep*

A técnica conhecida como *layer-by-layer sweep* permite que as camadas do DAG sejam tomadas duas a duas para a redução de cruzamentos (vide Seção 2.3.1). Nesse processo, são utilizados algoritmos que se propõem a reduzir cruzamentos em DAGs de duas camadas, tidos como grafos bipartidos (cuja orientação das arestas não é importante), nos quais uma das camadas é fixada enquanto se encontra uma ordem para os vértices da outra camada. O problema de redução de

cruzamentos em grafos bipartidos com uma camada fixada é conhecido como OLCM (*one-layer crossing minimization*)(JUNGER; MUTZEL, 1997).

Um grafo bipartido é um grafo $G = (L_1, L_2, E)$, em que L_1 e L_2 são conjuntos de vértices disjuntos e o conjunto de arestas $E \subseteq L_1 \times L_2$. O número de cruzamentos em um grafo bipartido pode ser denotado como $cross(G, \pi_1, \pi_2)$, calculado pela Equação 3.1, em que π_i representa uma ordem dos vértices de L_i e $\pi_i(v)$ representa a posição do vértice v em L_i . Se u e v são vértices em L_2 , o número de cruzamentos c_{uv} entre arestas incidentes em u e arestas incidentes em v depende apenas da posição relativa de u e v (BATTISTA et al., 1999, p. 281). Assim, para $u \neq v \in L_2$, c_{uv} é o número de pares de arestas $\{(u, w), (v, z)\}$, onde $\pi_1(z) < \pi_1(w)$.

$$cross(G, \pi_1, \pi_2) = \sum_{\pi_2(u) < \pi_2(v)} c_{uv}. \quad (3.1)$$

Desse modo, dados G e π_1 , resolver um OLCM significa encontrar um π_2 que minimize $cross(G, \pi_1, \pi_2)$.

3.1.1 Métodos de redução em grafos bipartidos

Nas seções seguintes, serão apresentados quatro métodos de redução de cruzamentos em grafos bipartidos com uma camada fixada. Esses métodos podem ser adotados como algoritmos da função *iteraHierarquia*, apresentada no Algoritmo 2, em que se resolve um OLCM.

Método Greedy-Switch

O método *Greedy-Switch*, apresentado por Eades e Kelly (1986), tem sua estrutura baseada no algoritmo bubble-sort, trocando pares adjacentes de vértices, como descreve o Algoritmo 3.

Algoritmo 3: Algoritmo *Greedy-Switch*

Entrada: $G(L_1, L_2, E)$; uma ordem π_1 para L_1

Saída: uma ordem π_2 para L_2

início

Escolha um π_2 para L_2

repita

para $u \leftarrow 1$ **até** $|L_2| - 1$ **faça**

se $c_{u(u+1)} > c_{(u+1)u}$ **então**

 permuta os vértices das posições u e $u + 1$;

fim se

fim para

até *que o número de cruzamentos não diminua;*

fim

O método *Greedy-Switch* itera por todos os pares consecutivos da camada, permutando-os caso haja redução de cruzamentos. Esse processo se repete até que nenhuma permutação possa ser feita para que o número de cruzamentos seja reduzido.

Método Split

O método *Split*, apresentado por Eades e Kelly (1986), executa de maneira análoga ao algoritmo quick-sort. Um vértice $v \in L_2$ é escolhido como pivô p , colocando-se cada vértice $u \in L_2$, $u \neq p$ à esquerda de p se $c_{up} < c_{pu}$, e à direita de p caso contrário. O Algoritmo 4 descreve o funcionamento do método *Split*.

Algoritmo 4: Algoritmo *Split*

Entrada: $G(L_1, L_2, E)$; uma ordem π_1 para L_1

Saída: uma ordem π_2 para L_2

início

se $L_2 \neq \emptyset$ **então**

 Escolha um vértice pivô $p \in L_2$;

$V_e, V_d \leftarrow \emptyset$

para cada $u \in L_2$ *em que* $u \neq p$ **faça**

se $c_{up} \leq c_{pu}$ **então**

$V_e \leftarrow V_e + u$

senão

$V_d \leftarrow V_d + u$

fim se

fim para cada

 aplique recursivamente *Split* para V_e e V_d concatenando suas saídas

fim se

fim

Método BC

O método baricêntrico (método *BC*), apresentado por Sugiyama et al. (1981), calcula a posição de cada vértice $u \in L_2$ como sendo a média das posições de cada um de seus vértices vizinhos $v \in L_1$. Os valores dessas médias são ordenados de forma crescente, determinando assim a posição dos vértices da camada L_2 . O Algoritmo 5 descreve o funcionamento do método *BC*. No algoritmo, $deg(u)$ recebe a contagem de vizinhos de u " $|Nu|$ ", e $avg(u)$ armazena a média aritmética das posições dos vizinhos de u .

Algoritmo 5: Algoritmo BC

Entrada: $G(L_1, L_2, E)$; uma ordem π_1 para L_1

Saída: uma ordem π_2 para L_2

início

```
para  $u \leftarrow 1$  até  $|L_2|$  faça
   $N_u \leftarrow$  vizinhos de  $L_2(u)$  ;
   $deg(u) \leftarrow |N_u|$  ;
  para todo  $v \in N_u$  faça
    |  $soma \leftarrow soma + \pi_1(v)$ 
  fim para todo
  se  $deg(u) = 0$  então
    |  $avg(u) \leftarrow 0$ 
  senão
    |  $avg(u) \leftarrow soma/deg(u)$ 
  fim se
fim para
selecione  $\pi_2(u)$  como  $avg(u)$  para todo  $u \in L_2$ 
```

fim

Uma importante propriedade do método *BC* é sua capacidade de, dados $G(L_1, L_2, E)$ e π_1 para L_1 , encontrar π_2 para L_2 que produza zero cruzamentos sempre que π_2 existe (BATTISTA et al., 1999).

Método MEDIAN

Semelhante ao método *BC*, o método *MEDIAN*, apresentado por Eades e Wormald (1994), atribui a posição de cada vértice $u \in L_2$ a partir do cálculo da mediana dos vizinhos de u . O Algoritmo 6 descreve o funcionamento do método *MEDIAN*.

Algoritmo 6: Algoritmo MEDIAN

Entrada: $G(L_1, L_2, E)$; uma ordem π_1 para L_1

Saída: uma ordem π_2 para L_2

início

```
para  $u \leftarrow 1$  até  $|L_2|$  faça
     $N_u \leftarrow$  vizinhos de  $L_2(u)$  ;
     $j \leftarrow |N_u|$  ;
    se  $j = 0$  então
         $med(u) \leftarrow 0$ ;
    senão se  $(j \% 2) \neq 0$  então
         $med(u) \leftarrow \pi_1(v_{j/2})$  ;
    senão
         $med(u) \leftarrow (\pi_1(v_{j/2}) + \pi_1(v_{1+j/2}))/2$  ;
    fim se
fim para
selecione  $\pi_2(u)$  como  $med(u)$  para todo  $u \in L_2$ 
```

fim

O método *MEDIAN* possui duas propriedades importantes: dados $G(L_1, L_2, E)$ e π_1 para L_1 , é encontrado π_2 para L_2 que produza zero cruzamentos sempre que π_2 existe; dados $G(L_1, L_2, E)$ e π_1 para L_1 , $med(G, \pi_1) \leq 3opt(G, \pi_1)$, em que $opt(G, \pi_1)$ é o número mínimo de cruzamentos dada uma ordem π_1 para L_1 (BATTISTA et al., 1999).

3.2 Métodos alternativos para redução de cruzamentos

Além da utilização de métodos heurísticos, existem alguns métodos alternativos que objetivam resolver MLCMs, OLCMs e TLCMs (*two-layer crossing problem*), tais como técnicas de programação de inteiros (ZHENG; BUCHHEIM, 2007), programação semidefinida (CHIMANI et al., 2012) e algoritmos genéticos (KUNTZ et al., 2006)(WEI-XIANG; JING-WEI, 2001).

Técnicas de programação de inteiros têm sido aplicadas para solucionar o problema de redução de cruzamentos para grafos bipartidos com uma das camadas fixas (OLCM) e com duas camadas li-

vres (TLCM)(JUNGER; MUTZEL, 1997). Esses métodos permitem encontrar soluções ótimas para um OLCM ou um TLCM, que podem ser utilizadas para avaliar algoritmos heurísticos. Apesar disso, a utilização dessas técnicas não é comum em sistemas interativos, pois não é possível estabelecer um tempo polinomial para a execução desses métodos (KAUFMANN; WAGNER, 2001).

Junger e Mutzel (1997) desenvolveram um algoritmo utilizando técnicas de programação linear e um método *branch and cut*, que retorna uma solução ótima para os problemas de OLCM. Eles mostraram que uma solução ótima pode ser encontrada em curtos tempos de execução quando uma camada é fixada. Porém, para o problema com duas camadas livres, o algoritmo é combinado com o uso de heurísticas, e só é eficaz para instâncias que possuem até 15 vértices na menor camada.

Outras técnicas de programação exata são utilizadas para resolver OLCMs e TLCMs. Zheng e Buchheim (2007) desenvolveram um algoritmo baseado em programação quadrática binária, podendo ser transformado em um modelo de programação linear de inteiros que, diferentemente da técnica proposta por Junger e Mutzel (1997), não necessita de heurísticas para solucionar um TLCM.

Para atacar o problema de *MLCM*, Chimani et al. (2012) propõem o uso de uma técnica baseada em programação semidefinida (*SDP*). Através de uma heurística baseada em *SDP*, eles conseguiram calcular soluções ótimas para grafos nunca calculadas antes. Uma observação importante é dada ao fato que a densidade do grafo não é relevante na utilização dessa técnica, ao contrário dos métodos baseados em programação linear de inteiros.

A utilização de meta-heurísticas tem sido outra maneira de se atacar o problema de cruzamentos em desenho de DAGs. Kuntz et al. (2006) apresentam um algoritmo genético adaptado para resolver um *MLCM*. O algoritmo combina operações de *crossover* de algoritmos genéticos com o método *BC*. O uso desse método permite um refinamento das operações genéticas que, em algumas partes do algoritmo, podem ser custosas quando comparadas a essa heurística.

3.3 Considerações

Muitas técnicas procuram aperfeiçoar o método *BC*, que produz resultados muito bons quando levados em consideração o número de cruzamentos final e o tempo de execução. Marti e Laguna (MARTIN; LAGUNA, 2003) apresentam um experimento comparando diferentes métodos que

visam resolver um *TCLM*. Nesses experimentos, pode-se destacar o fato de que uma versão do método *BC* combinado com o método *Split* (GANSNER et al., 1998) mostra-se a melhor escolha dentre outras doze, quando considerados os critérios número de cruzamentos e tempo de execução como críticos.

Junger e Mutzel (1997), em seus experimentos com programação linear, comparam várias heurísticas para o problema de *TLCM* com o algoritmo que haviam proposto, concluindo que o método *BC* provê as melhores soluções para grafos esparsos, considerando-se o custo benefício entre número de cruzamentos e tempo de execução.

Essas considerações sugerem que a utilização de métodos associados para resolver o problema de redução de cruzamentos em DAGs ou em grafos bipartidos pode trazer bons resultados levando-se em consideração o número de cruzamentos e o tempo de execução dos métodos.

Capítulo 4

Árvores PQR e redução de cruzamentos

Uma Árvore PQR é uma estrutura de dados apresentada por Meidanis et al. (1998) capaz de representar todas as permutações válidas de um domínio de objetos através de restrições que definem a maneira como esses objetos podem se agrupar. Como exemplo, dado o domínio de objetos $D = \{1,2,3\}$ e a restrição $r = \{1,2\}$, a Árvore PQR gerada com D e r fornece as seguintes permutações: $\{1,2,3\}$, $\{3,1,2\}$, $\{2,1,3\}$ e $\{3,2,1\}$.

Para que isso seja possível, além de ser uma árvore enraizada, ela possui as seguintes propriedades: nós P têm no mínimo dois filhos; nós R têm no mínimo três filhos; os filhos de nó P e nó R podem ser permutados entre si; nós Q têm no mínimo três filhos; as permutações permitidas para os filhos de nó Q são somente duas: a ordem original da lista de filhos, e a ordem inversa. A Figura 4.1 representa as permutações válidas de acordo com essas propriedades.

Para obter a representação do conjunto de permutações possíveis, é necessário que sejam fornecidos como objetos do algoritmo de construção da árvore um conjunto de elementos $U = \{x_1, x_2, \dots, x_n\}$ e um conjunto de restrições $R = \{r_i, r_{i+1}, \dots, r_n\}$ em que r_i é um subconjunto de U de tamanho maior ou igual a 2. A fronteira da árvore representa uma das permutações que respeita as restrições impostas quando possível. Quando não se é possível satisfazer todas as restrições, a Árvore PQR indica as restrições que não foram satisfeitas através de um nó chamado nó R, e a partir de sua fronteira é possível obter uma das permutações que represente as restrições válidas e conflitantes.

Considere-se, como exemplo, o conjunto $U = \{0,1,2,3,4\}$ e as restrições $R_1 = \{0,1\}$, $R_2 = \{3,4\}$, $R_3 = \{2,3\}$. A Figura 4.2 apresenta um desenho da Árvore PQR gerada com base em U e em $R = \{R_1, R_2, R_3\}$, bem como as permutações possíveis, de acordo com as permutações permitidas

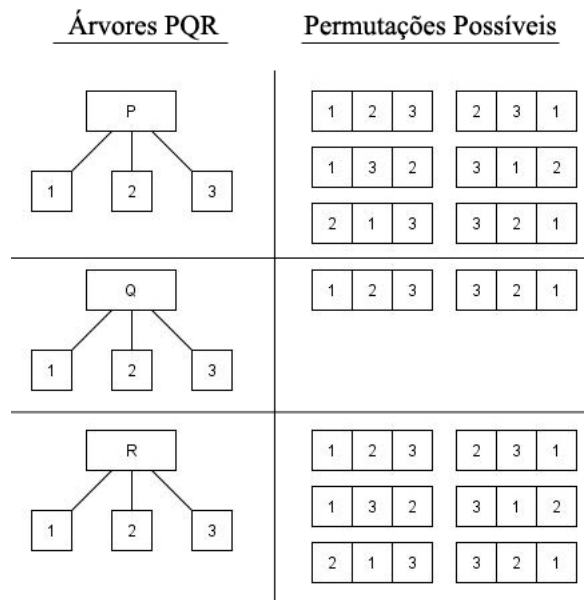


Figura 4.1: Exemplos de nós de uma Árvore PQR com as possíveis permutações.

pelos nós da árvore. Observa-se que para essa árvore, todas as permutações são válidas, pois como não há nó R, todas as restrições são obedecidas nessas permutações.

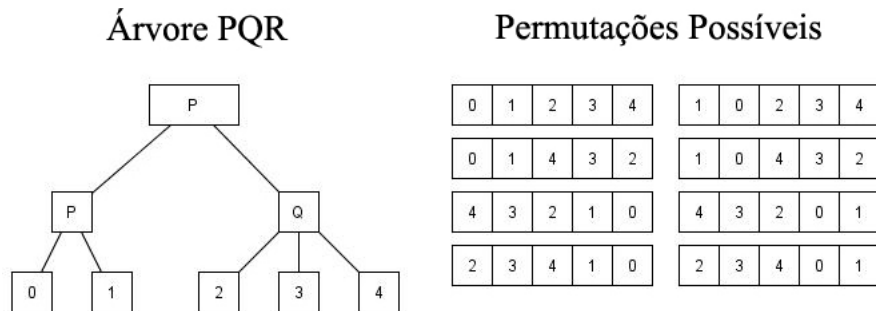


Figura 4.2: Representação de uma Árvore PQR e das suas possíveis permutações.

Caso exista uma restrição conflitante, os elementos que geram conflito são colocados em um nó do tipo R. Dado como exemplo o conjunto $U = \{0,1,2,3,4\}$ e as restrições $R_1 = \{0,1\}$, $R_2 = \{3,4\}$, $R_3 = \{2,3\}$, $R_4 = \{2,4\}$. As restrições R_2 , R_3 e R_4 não podem ser obedecidas simultaneamente, o que dá origem a um nó do tipo R. Neste caso, o resultado obtido é o par $\{0,1\}$ ou $\{1,0\}$ variando com quaisquer ordens de $\{2,3,4\}$.

4.1 Árvores PQR e suas aplicações

Árvores PQR podem ser utilizadas para resolver diversos problemas de permutação. Meidanis et al. (1998) propuseram a utilização de Árvores PQR para a resolução do problema dos uns consecutivos (PIC), que é um problema de reordenação de matrizes binárias. Recentemente, Melo (2012) propôs o uso de Árvores PQR para o problema de seriação de matrizes binárias (processo de reordenação automática de linhas e colunas dessas matrizes).

Siirtola e Makinen (2005) relacionam o problema de agrupamento de valores em uma matriz (PIC, por exemplo) com o problema de redução de cruzamentos em um grafo bipartido. Partindo desse pressuposto, Marchete (2010) propôs a utilização de Árvores PQR como parte de um novo método para redução de cruzamentos em grafos bipartidos. Marchete (2010) definiu U como todos os vértices da camada L_1 do grafo. Para cada vértice da camada L_2 , definiu uma restrição composta pelos vizinhos desse vértice em L_1 , e adicionou-a ao conjunto R . Feito isto, criou uma Árvore PQR com base em U e R . Novamente, a fronteira da Árvore PQR indicou uma nova ordem para os vértices de L_1 . O mesmo processo pode ser feito trocando os papéis de L_1 e L_2 . Seguindo este algoritmo no exemplo da Figura 4.3, é possível definir o conjunto de elementos $U = \{a,b,c,d,e\}$ e o conjunto de restrições $R = \{r_1,r_2\}$ em que $r_1 = \{c,e\}$ e $r_2 = \{a,b\}$ para a Árvore PQR que representa a camada superior; o conjunto de elementos $U = \{f,g,h,i,j\}$ e o conjunto de restrições $R = r_3$ em que $r_3 = \{h,i,j\}$ para a Árvore PQR que representa a camada inferior. A Figura 4.4 representa as Árvores PQR geradas para ambas as camadas.

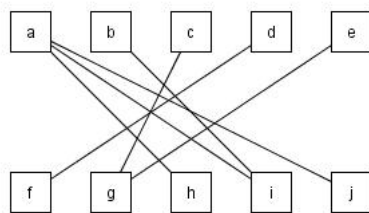


Figura 4.3: Grafo bipartido com 10 vértices e 7 arestas.

A partir da fronteira das Árvores PQR (representada pelas folhas da esquerda para a direita), é possível redefinir as posições dos vértices no grafo bipartido, como mostra a Figura 4.5.

Marchete (2010) mostrou que, apesar da característica de agrupamento das Árvores PQR, os resultados obtidos para redução de cruzamentos em grafos bipartidos nem sempre são bons, pois

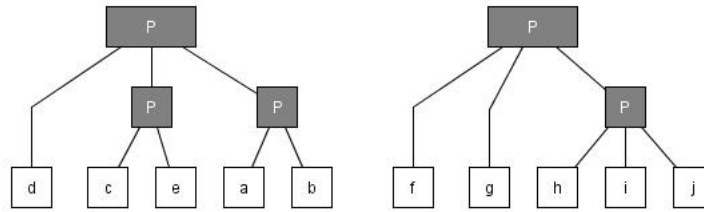


Figura 4.4: Árvores PQR gerada a partir do grafo da Figura 4.3 com respectivas fronteiras.

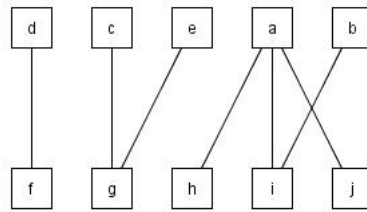


Figura 4.5: Grafo bipartido da Figura 4.3 reordenado com Árvores PQR.

o resultado do processamento dessa estrutura de dados é influenciado pela ordem de entrada das restrições. Para melhorar os resultados obtidos, Marchete propôs o uso do método *BC* em uma fase de refinamento, o que trouxe melhores resultados em uma comparação com a execução do próprio método *BC* para grafos bipartidos com densidades inferiores a 0,2. Outro resultado importante do trabalho de Marchete é a indicação do uso de Árvores PQR como parte da estratégia de redução de cruzamentos em estruturas visuais interativas, pois o tempo de execução do método que envolve Árvores PQR é adequado para sua utilização nessas estruturas, uma vez que executam em menos de 0,1 segundo.

4.2 Método de redução de cruzamentos em DAGs utilizando Árvores PQR

Para utilizar Árvores PQR como parte da estratégia proposta pelo modelo de Sugiyama para reduzir cruzamentos de um DAG, considere um DAG $G(L,E,n)$ em que $L = L_1, \dots, L_n$ represente o conjunto de camadas onde estão dispostos os vértices do DAG, E represente o conjunto de arestas e n o número de camadas. Para cada camada L_i , define-se uma Árvore PQR $T(U,R)$ cujo conjunto universo U seja composto pelos vértices pertencentes à camada. Adicionalmente, cada conjunto de elementos de L_i que tenha um mesmo vizinho em L_{i-1} ou L_{i+1} é adicionado como uma restrição do conjunto de restrições R . Isso é feito com o objetivo de tornar consecutivos esses vértices na reordenação a ser calculada. Após criar a árvore $T(U,R)$, toma-se sua fronteira como nova ordem dos vértices da camada L_i . O Algoritmo 7 descreve esse processo.

Algoritmo 7: Algoritmo de reordenação de DAGs utilizando Árvores PQR

Entrada: DAG $G(L,E,n)$

Saída: Árvore PQR t

início

para $i \leftarrow 1$ **até** n **faça**

$U \leftarrow$ vértices de L_i ;

$V \leftarrow L_{i-1} \cup L_{i+1}$;

para cada $v \in V$ **faça**

$r \leftarrow$ vizinhos de v em U ;

$R \leftarrow R \cup \{r\}$

fim para cada

$t \leftarrow$ *FronteiraArvorePQR*(U,R) ;

 reordene L_i de acordo com t

fim para

fim

Como exemplo, a Figura 4.6a apresenta o desenho de um DAG com três camadas, nove vértices e 4 cruzamentos. Após submetido ao algoritmo de reordenação de DAGs utilizando Árvores PQR (Algoritmo 7), o DAG passa a possuir zero cruzamentos, como mostra a Figura 4.6b.

Apesar da característica de agrupar elementos com uma determinada regra de similaridade ser importante para o descruzamento de arestas (como visto, por exemplo, nos métodos *BC* e *MEDIAN* (BATTISTA et al., 1999), testes preliminares indicaram que a utilização das fronteiras das Árvores

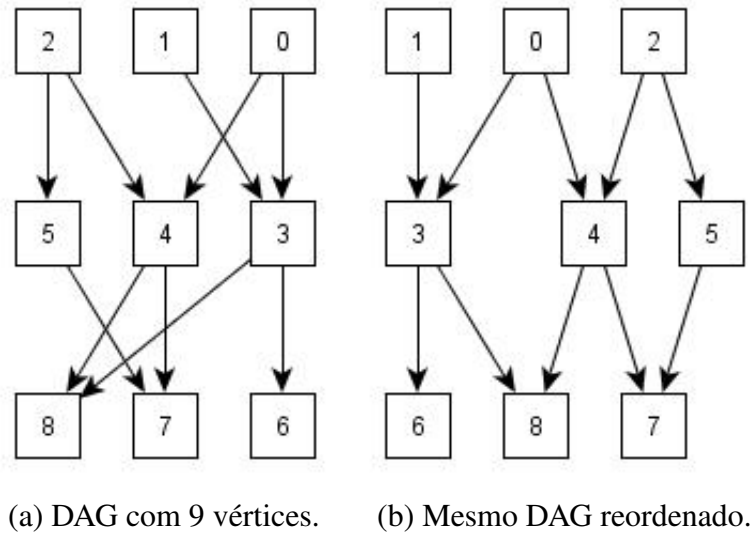


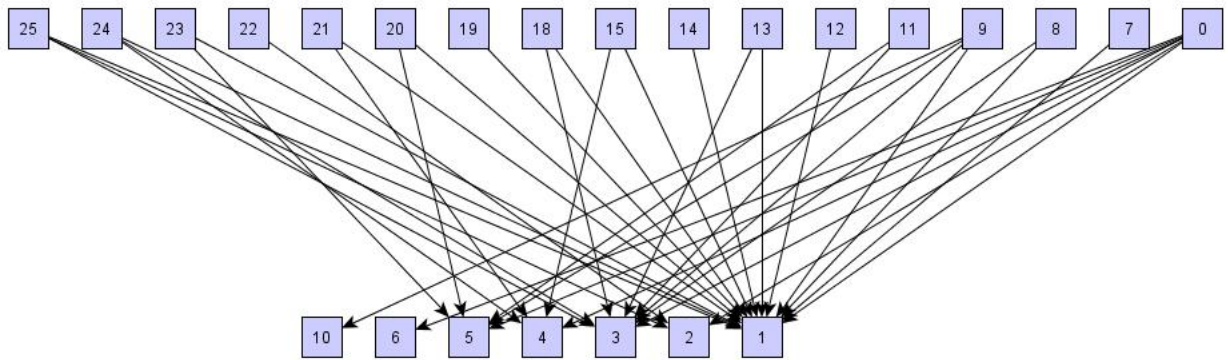
Figura 4.6: DAG reordenado com algoritmo de reordenação de DAGs utilizando Árvores PQR.

PQR como nova ordem dos vértices não é suficiente para obter bons resultados de redução de cruzamentos em DAGs, como exemplificado na Figura 4.7.

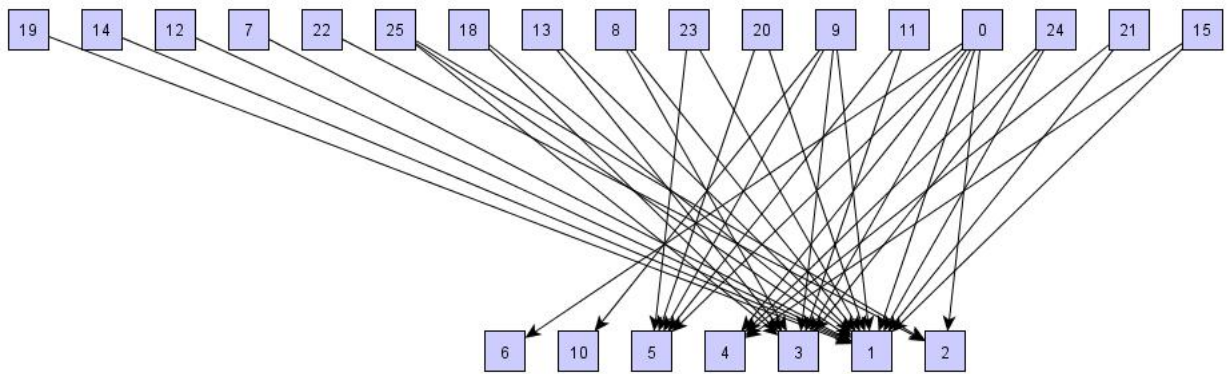
A Figura 4.7a mostra o desenho de um DAG com 276 cruzamentos. Na Figura 4.7b é possível observar que o número de cruzamentos do DAG aumentou para 320. Testes preliminares mostraram que esse é um problema recorrente, e por isso optou-se por associar o uso de Árvores PQR com outros métodos de redução de cruzamentos. Assim, foram desenvolvidos dois novos algoritmos que utilizam Árvores PQR como parte de suas estratégias de redução de cruzamentos. A esses, deu-se o nome de *PQR_BC1* e *PQR_M1*.

4.2.1 Métodos *PQR_BC1* e *PQR_M1*

O método *PQR_BC1* é uma versão aprimorada do Algoritmo 7, no qual as fronteiras das Árvores PQR obtidas não determinam diretamente a ordem dos vértices do DAG. Elas passam por um processo de reordenação que envolve o uso do método *BC*. Dessa maneira, executa-se o método *BC* após a ordenação obtida pelas fronteiras das Árvores PQR em um processo de *layer-by-layer sweep*, seguindo o *modelo de Sugiyama*, como descreve o Algoritmo 8. A pré-ordenação dos vértices é feita com as fronteiras da Árvore PQR no intuito de agrupar vértices de tal forma que o método *BC* encontre melhores soluções, como visto na Figura 4.8. O número de *fails* é definido por *fr* durante



(a) DAG com 276 cruzamentos.



(b) DAG com 320 cruzamentos.

Figura 4.7: DAG com 24 vértices distribuídos em duas camadas. Em 4.7a, DAG com a posição original dos vértices (276 cruzamentos). Em 4.7b, DAG após submetido à redução de cruzamentos com Árvores PQR (320 cruzamentos).

a execução de *layer-by-layer sweep*. Os experimentos do Capítulo 5 comparam os resultados do uso desse novo método com os resultados do método *BC*.

Algoritmo 8: Algoritmo *PQR_BCI*

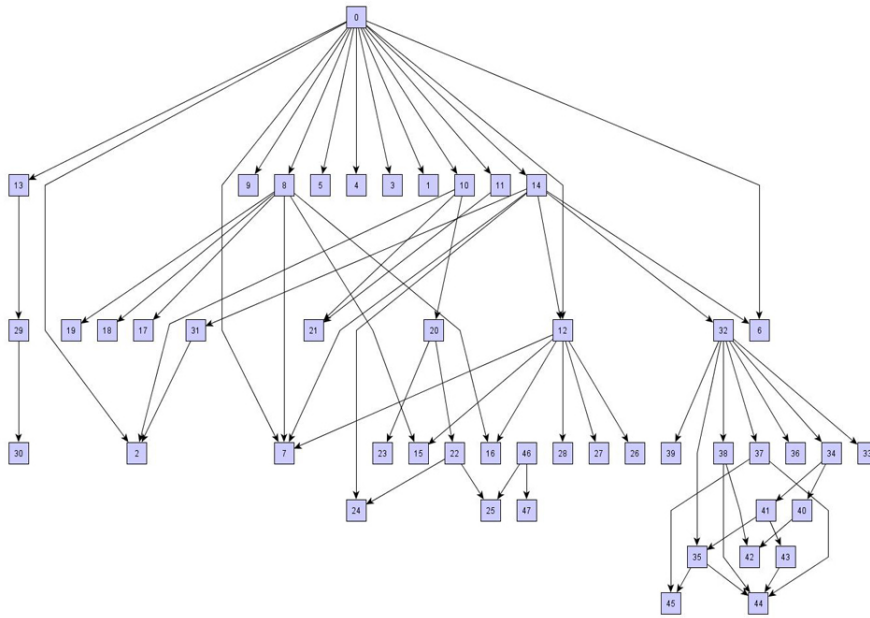
Entrada: DAG $G(L, E, n), ff$

Saída: Árvore PQR t

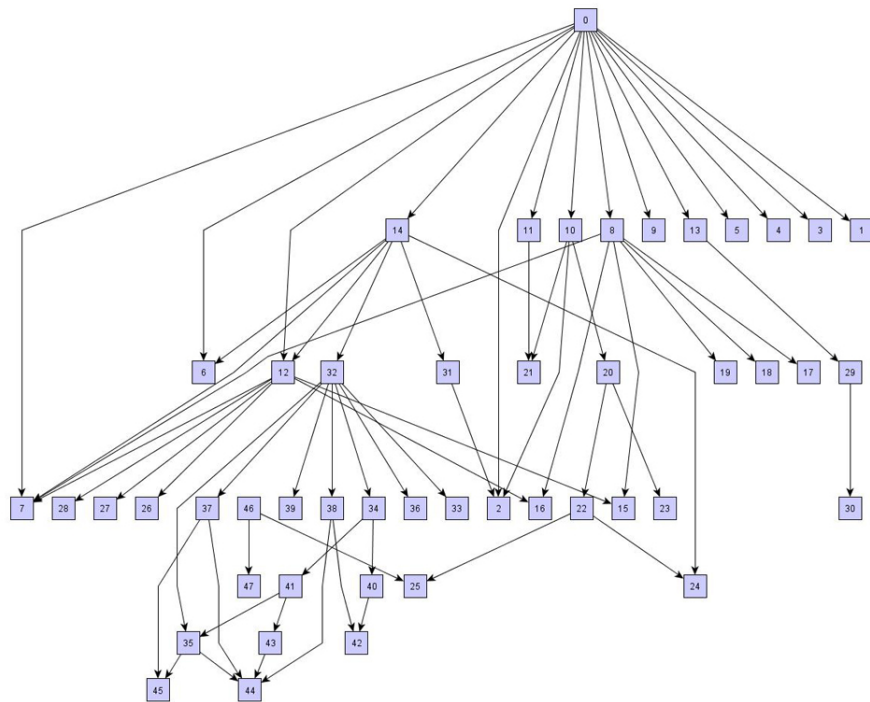
início

 | execute Algoritmo 7 ;
 | execute *layer-by-layer sweep* com método *BC* usando $fails = ff$;

fim



(a) DAG com 7 camadas e 48 vértices processado pelo método *BC*.



(b) Mesmo DAG processado pelo método *PQR_BC1*.

Figura 4.8: DAG reordenado. Em 4.8a, DAG reordenado com algoritmo *BC* (51 cruzamentos). Em 4.8b, mesmo DAG reordenado com algoritmo *PQR_BC1* (44 cruzamentos).

O método *PQR_M1* foi desenvolvido seguindo-se os moldes de *PQR_BC1*, porém utilizando-se como parte de sua estratégia o método *MEDIAN* ao invés de *BC*. O Algoritmo 9 descreve o método *PQR_M1*. A Figura 4.9 mostra um mesmo DAG processado pelos métodos *MEDIAN* e *PQR_M1*. Os experimentos do Capítulo 5 comparam os resultados do uso desse método com os resultados do método *MEDIAN*.

Algoritmo 9: Algoritmo *PQR_M1*

Entrada: DAG $G(L,E,n), ff$

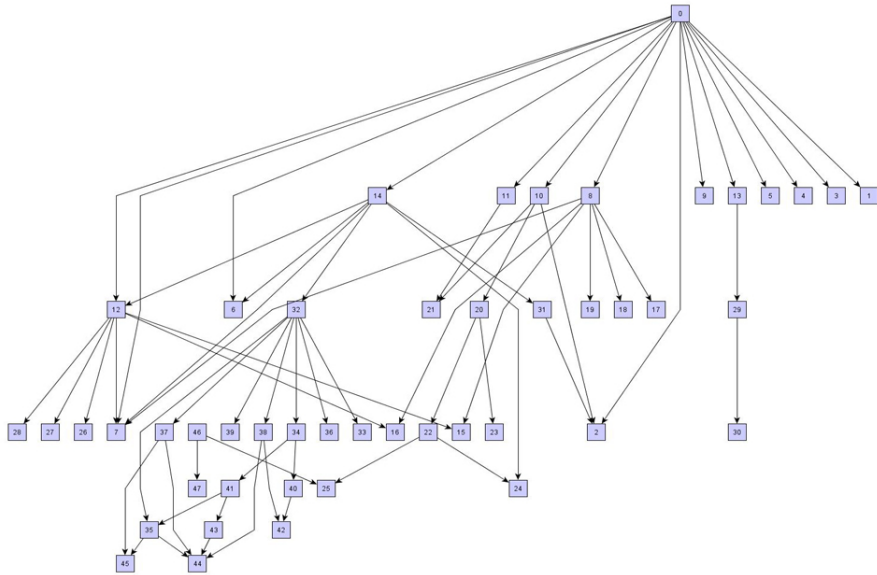
Saída: Árvore PQR t

início

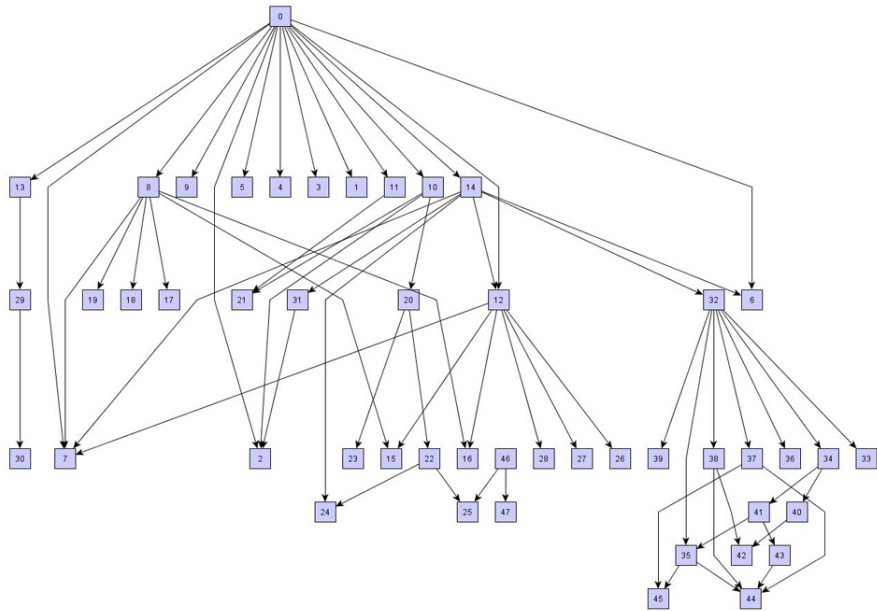
 | execute Algoritmo 7 ;

 | execute *layer-by-layer sweep* com método *MEDIAN* usando $fails = ff$;

fim



(a) DAG com 7 camadas e 48 vértices processado pelo método *MEDIAN*.



(b) Mesmo DAG processado pelo método *PQR_MI*.

Figura 4.9: DAG reordenado. Em 4.9a, DAG reordenado com algoritmo *MEDIAN* (53 cruzamentos). Em 4.9b, mesmo DAG reordenado com algoritmo *PQR_MI* (43 cruzamentos).

4.2.2 Métodos PQR_BC2 e PQR_M2

A partir das observações realizadas com os métodos PQR_BC1 e PQR_M1 , notou-se que a ordem dos elementos dentro de uma restrição passada para uma Árvore PQR, bem como a ordem com que estas restrições são definidas, influencia diretamente na produção da fronteira da árvore, que por sua vez, é utilizada para o processamento final das posições dos vértices de cada camada. Melo (2012) observou que um pré-processamento na fase de formação dessas restrições pode trazer resultados melhores no uso de Árvores PQR para problemas de seriação. Partindo dessa observação, buscou-se obter estratégias de pré-processamento das restrições, a fim de produzir melhores resultados de ordenação de vértices.

Com isso, desenvolveu-se os métodos PQR_BC2 e PQR_M2 , que utilizam respectivamente os métodos BC e $MEDIAN$ na fase de formação de restrições das Árvores PQR, além de serem executados também na fase de processamento das fronteiras obtidas.

Os Algoritmos 10 e 11 descrevem respectivamente os métodos PQR_BC2 e PQR_M2 , nos quais fr representa o número de *fails* na fase de pré-processamento das restrições e ff o número de *fails* relativo ao processamento da fronteira.

Algoritmo 10: Algoritmo PQR_BC2

Entrada: DAG $G(L,E,n)$, fr , ff

Saída: Árvore PQR t

início

 execute *layer-by-layer sweep* com método BC usando $fails = fr$;

 execute Algoritmo 7 ;

 execute *layer-by-layer sweep* com método BC usando $fails = ff$;

fim

Algoritmo 11: Algoritmo PQR_M2

Entrada: DAG $G(L,E,n)$, fr , ff

Saída: Árvore PQR t

início

 execute *layer-by-layer sweep* com método $MEDIAN$ usando $fails = fr$;

 execute Algoritmo 7 ;

 execute *layer-by-layer sweep* com método $MEDIAN$ usando $fails = ff$;

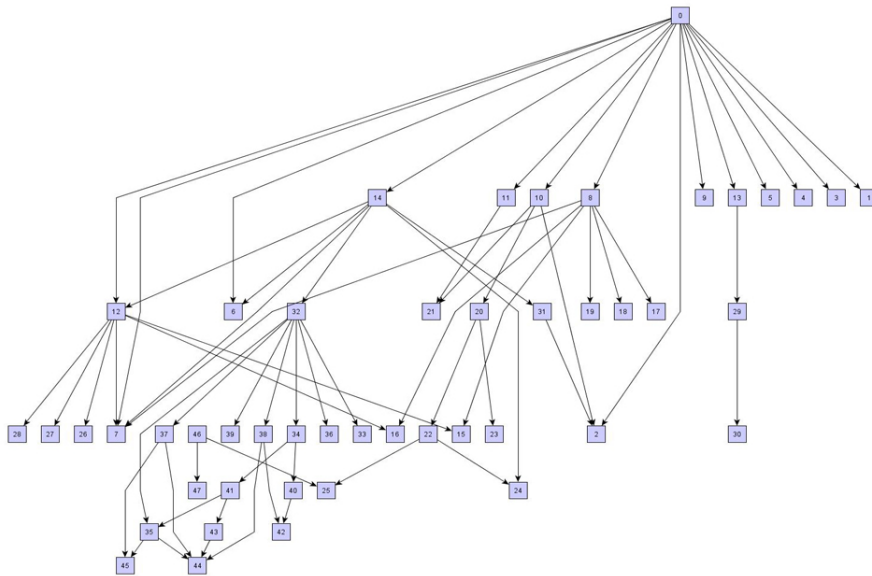
fim

A Figura 4.10 mostra o DAG da Figura 4.9 sendo reordenado com os métodos *BC* e *PQR_BC2*, e a Figura 4.11 mostra o mesmo DAG sendo reordenado com os métodos *MEDIAN* e *PQR_M2*. Os experimentos do Capítulo 5 comparam os resultados do uso do método *BC* com os resultados do método *PQR_BC2*, e os resultados do método *MEDIAN* com os resultados do método *PQR_M2*.

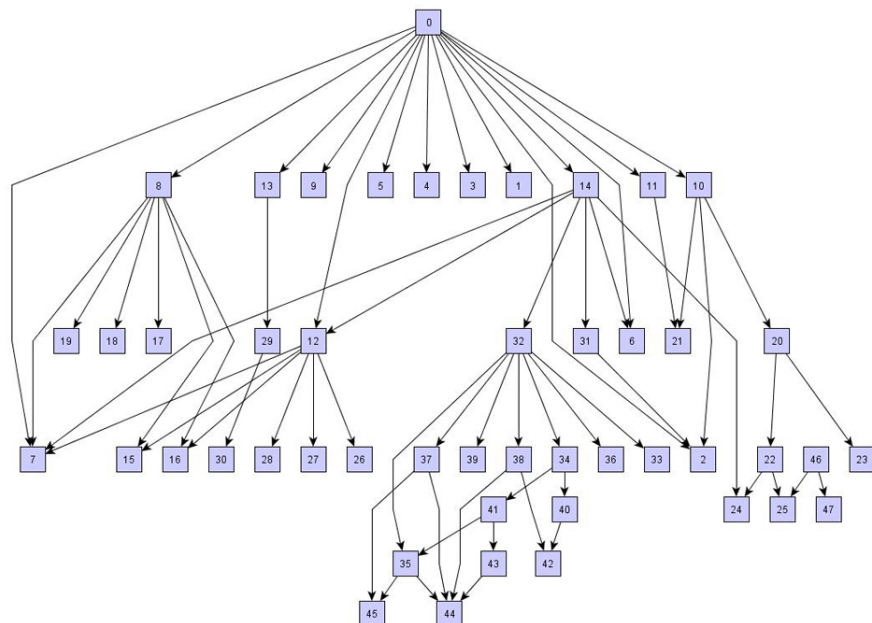
4.3 Considerações

Este capítulo descreveu os algoritmos propostos nesse trabalho. Foram propostos ao todo cinco algoritmos dos quais os quatro últimos são híbridos entre o primeiro algoritmo proposto e *BC* ou *MEDIAN*, consagrados pela literatura.

No Capítulo 5, os resultados dos algoritmos *PQR_BC1* e *PQR_BC2* serão comparados com os resultados de *BC*, e os algoritmos *PQR_M1* e *PQR_M2* serão comparados com os resultados do método *MEDIAN*.

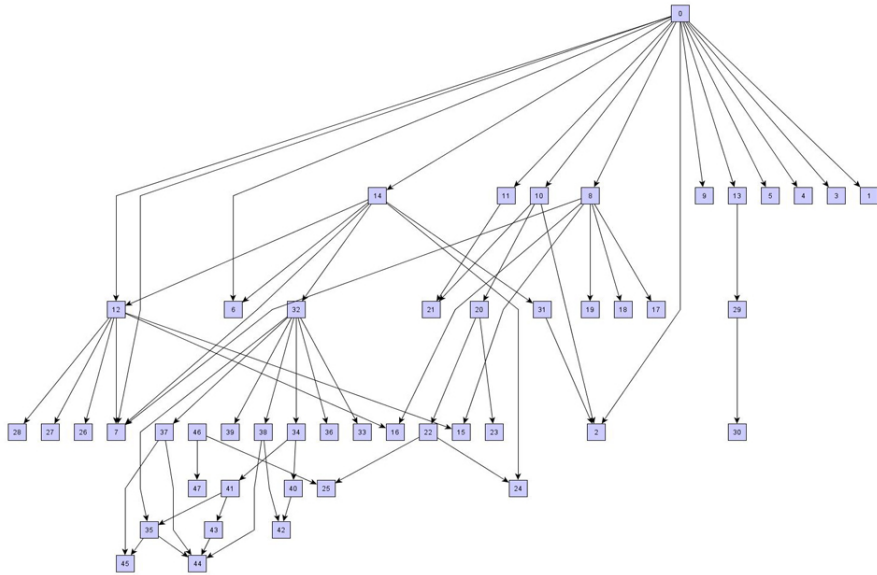


(a) DAG com 7 camadas e 48 vértices processado pelo método *BC*.

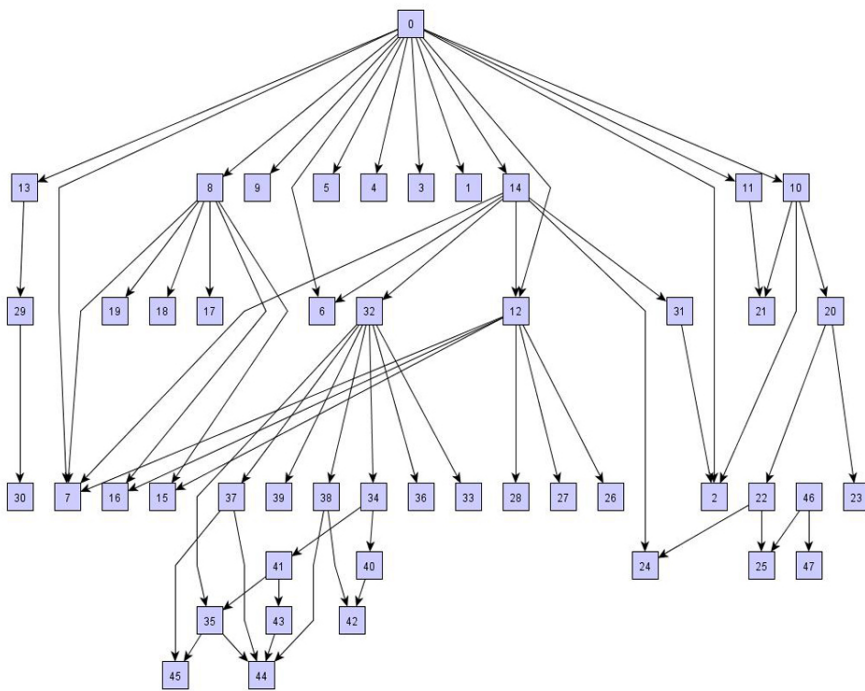


(b) Mesmo DAG processado pelo método *PQR_BC2*.

Figura 4.10: DAG reordenado. Em 4.10a, DAG reordenado com algoritmo *BC* (51 cruzamentos). Em 4.10b, mesmo DAG reordenado com algoritmo *PQR_BC2* (21 cruzamentos).



(a) DAG com 7 camadas e 48 vértices processado pelo método *MEDIAN*.



(b) Mesmo DAG processado pelo método *PQR_M2*.

Figura 4.11: DAG reordenado. Em 4.11a, DAG reordenado com algoritmo *MEDIAN* (53 cruzamentos). Em 4.11b, mesmo DAG reordenado com o algoritmo *PQR_M2* (33 cruzamentos).

Capítulo 5

Experimentos e resultados

Este capítulo apresenta os resultados obtidos com a utilização dos métodos *PQR_BC1*, *PQR_BC2*, *PQR_M1* e *PQR_M2* no processo de redução de cruzamentos em DAGs. Esses resultados foram adquiridos através de experimentos que objetivam comparar os métodos *BC* e *MEDIAN* com as novas técnicas propostas neste trabalho. A Seção 5.1 apresenta os métodos e materiais utilizados durante a fase de experimentos. A Seção 5.2 apresenta os resultados das comparações entre os métodos propostos neste trabalho e os métodos *BC* e *MEDIAN*.

5.1 Materiais e métodos

Nos experimentos a seguir, serão utilizados dois critérios para comparação entre a técnica proposta neste trabalho e os algoritmos *BC* e *MEDIAN*: número de cruzamentos total do DAG e tempo de execução. Ambos os critérios são importantes no que concerne ao entendimento do desenho de um DAG e sua utilização em estruturas visuais interativas, como discutido na Seção 2.3. A qualidade de um DAG é mensurada neste trabalho pela contagem do número de cruzamentos; quanto menor esse valor, melhor a qualidade do DAG (PURCHASE, 1997).

Durante o processo de criação do algoritmo, foi adotada a técnica *layer-by-layer sweep*, uma vez que, ainda antes da etapa de descruzamento de arestas, ela permite a atribuição de camadas a vértices, diferentemente das técnicas de planarização. Isso se deu novamente por conta da utilização de DAGs em sistemas de visualização interativa, já que em alguns cenários a definição de camadas é predefinida pelo contexto ou definida pelos usuários sob demanda.

A Seção 5.1.1 apresenta os pacotes de grafos e suas características, utilizados durante os experimentos realizados neste trabalho. A Seção 5.1.2 descreve a maneira como os DAGs foram separados em camadas. A Seção 5.1.3 apresenta o cálculo de densidade de DAGs utilizado durante os experimentos e a Seção 5.1.4 apresenta detalhes da implementação dos algoritmos desenvolvidos e comparados neste trabalho.

5.1.1 Pacote de Grafos

Para a execução dos experimentos, foram utilizados dois pacotes distintos de grafos: Rome Graphs (BATTISTA et al., 1997b)(aqui denominado Pacote Rome) e North DAGs (BATTISTA et al., 1997a)(aqui denominado Pacote North). Ambos os pacotes são conhecidos na literatura por uma forte relação com exemplos reais de uso (CHIMANI et al., 2012).

O Pacote North foi introduzido em uma comparação entre algoritmos de desenho de DAGs. O pacote possui 1277 dígrafos originados de um conjunto de 5114 dígrafos coletados por Stephen North nos laboratórios da AT&T Bell. Battista et al. (1997a) selecionaram apenas os grafos que seguiam características específicas, como número de vértices entre 10 e 100 e apenas um grafo de cada classe isomórfica.

Por sua vez, o Pacote Rome possui 11582 grafos com número de vértices entre 10 e 100, criados a partir de um conjunto inicial de 112 grafos, obtidos através de companhias de software e organizações governamentais italianas, livros de referência em engenharia de software e desenho de banco de dados, artigos que tratam de visualização de software e teses de alunos das áreas de visualização de software e banco de dados da Universidade de Roma *La Sapienza*.

Apesar de ser um pacote de grafos não direcionados, o Pacote Rome tem sido adotado para testes em DAGs, mudando-se artificialmente a direção das arestas de acordo com a ordem dos vértices nos arquivos de descrição de cada grafo (EIGLSPERGER; KAUFMANN, 2001). Os grafos desconexos não serão utilizados nesse trabalho. Por esse motivo, serão adotados apenas 11531 grafos do pacote. As Figuras 5.1 e 5.2 apresentam a distribuição dos grafos do Pacote North e Pacote Rome, respectivamente, agrupados pelo número de vértices.

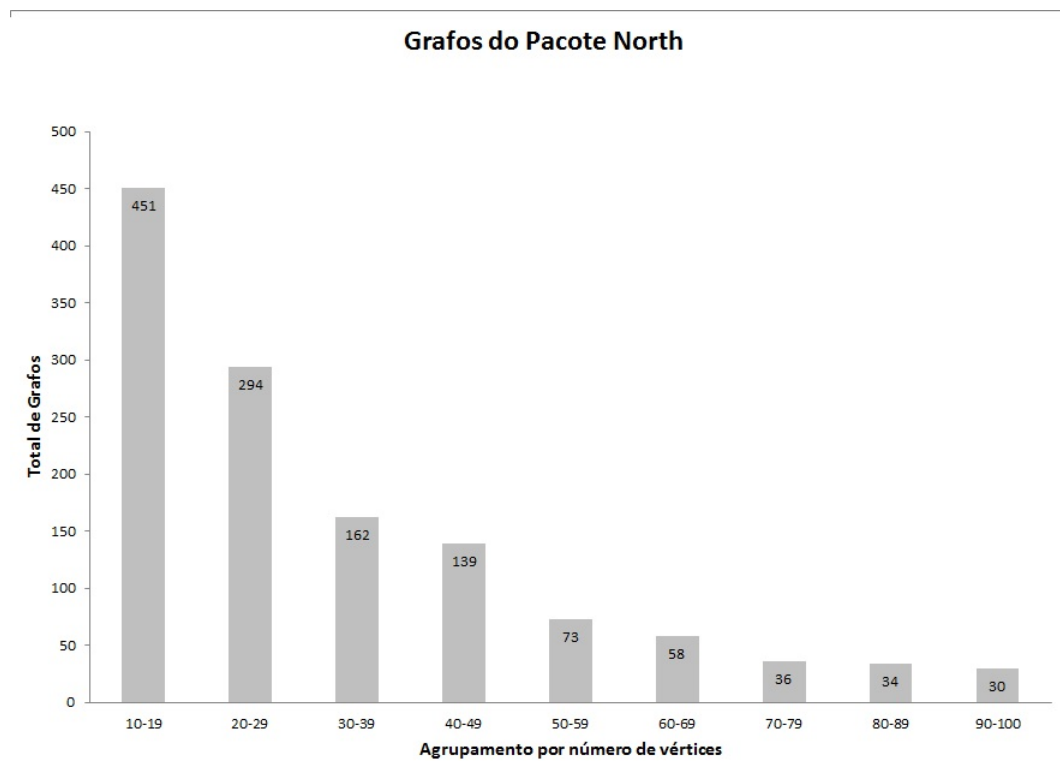


Figura 5.1: Grafos do Pacote North agrupados pelo número de vértices.

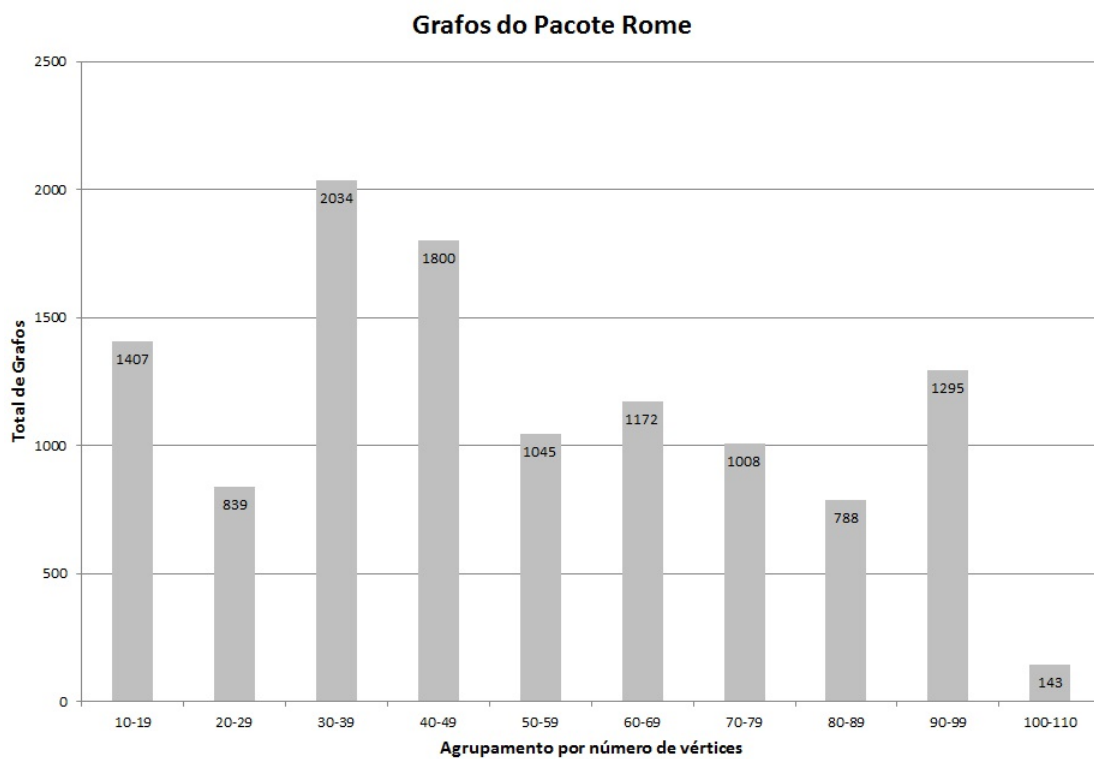


Figura 5.2: Grafos do Pacote Rome agrupados pelo número de vértices.

5.1.2 Definição de camadas

Para a execução dos experimentos deste trabalho, uma vez que os vértices dos grafos não estavam previamente atribuídos a camadas, foi necessário fazer essa atribuição. Para tanto, adotou-se o algoritmo *longest path ranking*, apresentado na Seção 2.3.1, a fim de automatizar este processo. Uma vez executado esse procedimento, é necessário efetuar a adição de *dummy nodes* ao grafo para que ele se torne uma *hierarquia própria*, e possa ser executado pelos métodos de redução de cruzamentos. Por esse motivo, perdeu-se a referência inicial do número de vértices de cada instância (grafos) dos pacotes de grafos, exigindo um novo tipo de agrupamento para efeito de análises e medições. Neste trabalho, pelo número variado de vértices e camadas consequentes da execução do algoritmo *longest path ranking* e da adição de *dummy nodes*, optou-se pelo agrupamento por densidades. Os gráficos das Figuras 5.3 e 5.4 mostram o crescimento do número de vértices das instâncias pertencentes aos pacotes North e Rome respectivamente.

Observa-se que o Pacote Rome apresenta um padrão mais uniformizado com relação ao número de *dummy nodes* adicionados aos grafos quando comparado ao Pacote North que, em alguns casos, alcança mais de mil vértices por grafo. A configuração dos grafos adotada neste trabalho considera a quantidade total de vértices de um grafo sendo a quantidade inicial mais a quantidade de *dummy nodes* adicionados, visto que a execução dos algoritmos de redução de cruzamentos de arestas está diretamente relacionada ao número total de vértices do grafo.

Os grafos de cada pacote foram distribuídos em seis grupos, representados pelos intervalos de suas densidades d , sendo $d = \{[0;0,1[; [0,1;0,2[; [0,2;0,3[; [0,3;0,4[; [0,4;0,5[; [0,5;0,6[]\}$. Os grafos com densidade superior a 0,6 não foram considerados nesse trabalho e equivalem a 0,2% do total de grafos do Pacote Rome e 3% do total de grafos do Pacote North. Esse pequeno número de instâncias dificultou a obtenção de resultados comparativos conclusivos nessas densidades, motivo pelo qual foram desconsideradas.

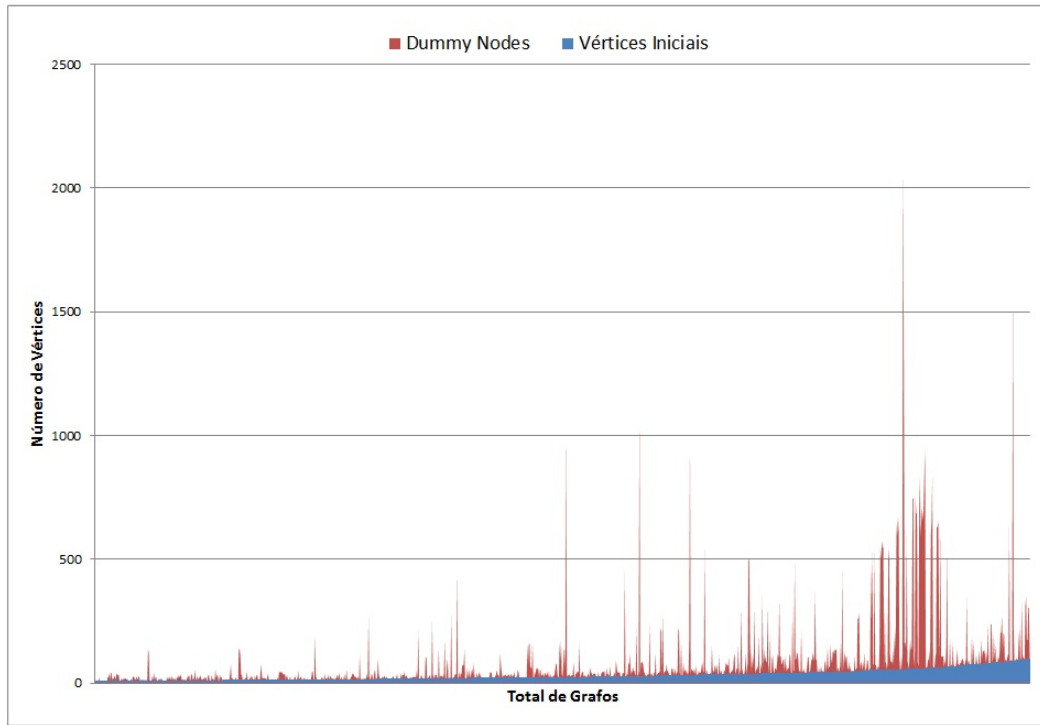


Figura 5.3: Número de vértices antes e depois da atribuição de camadas nos grafos do Pacote North.

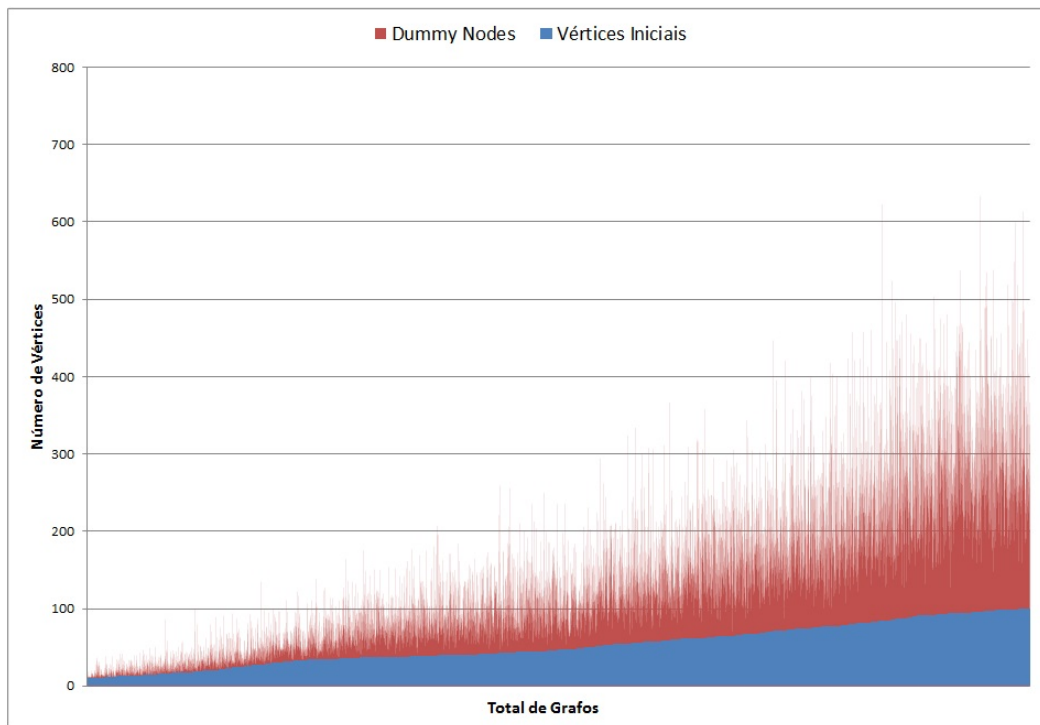


Figura 5.4: Número de vértices antes e depois da atribuição de camads nos grafos do Pacote Rome.

5.1.3 Cálculo de densidade

O cálculo da densidade de um DAG em camadas adotado nesse trabalho se diferencia do cálculo comumente utilizado para grafos, que é a razão do número de arestas presentes no grafo pelo número de arestas de um grafo completo de mesmo tamanho. Visto que em uma *proper hierarchy* as arestas dos vértices pertencentes a uma determinada camada não podem se ligar a um vértice que não seja de suas camadas adjacentes, o cálculo da densidade de um DAG será o mesmo utilizado por Kuntz et al. (2006), dado por:

$$d = \frac{m}{m_{max}} \quad (5.1)$$

sendo m o número total de arestas do DAG, em que

$$m_{max} = \sum_{k=2}^h |L_{k-1}| |L_k| \quad (5.2)$$

sendo h o total de camadas e L_i o conjunto de nós atribuídos à camada i .

5.1.4 Implementação

A execução dos experimentos foi feita em um notebook com processador Intel Core i5 de 2,5 GHz. Foi utilizado um programa escrito em C++ chamado OGDF (*Open Graph Drawing Framework*) (CHIMANI et al., 2007) que traz uma implementação do *modelo de Sugiyama* com os métodos *BC* e *MEDIAN*. A implementação do algoritmo de Árvores PQR foi feita por Dilly (2006) em Java. Foram utilizadas APIs da linguagem Java conhecidas como JNI (*Java Native Interface*) para fazer a conexão entre o OGDF e a implementação das Árvores PQR. A Figura 5.5 ilustra o esquema de conexão entre as classes do OGDF (C++) e as classes da Árvore PQR (Java).

No programa OGDF foi criada uma classe principal em que são estabelecidos os parâmetros do grafo (nós, vértices, distância entre os nós no desenho) a ser transformado em um DAG em camadas. A partir do atributo *sl* (instância da classe *SugiyamaLayout*), é possível transmitir os parâmetros do grafo para a classe responsável por executar os passos do *modelo de Sugiyama*. Durante o passo de redução de cruzamentos, o método *reduzCruzamentos()*, além de conter a implementação de *layer-by-layer sweep*, chama o método *getOrdemPQR()*, quando necessário, para receber a ordem

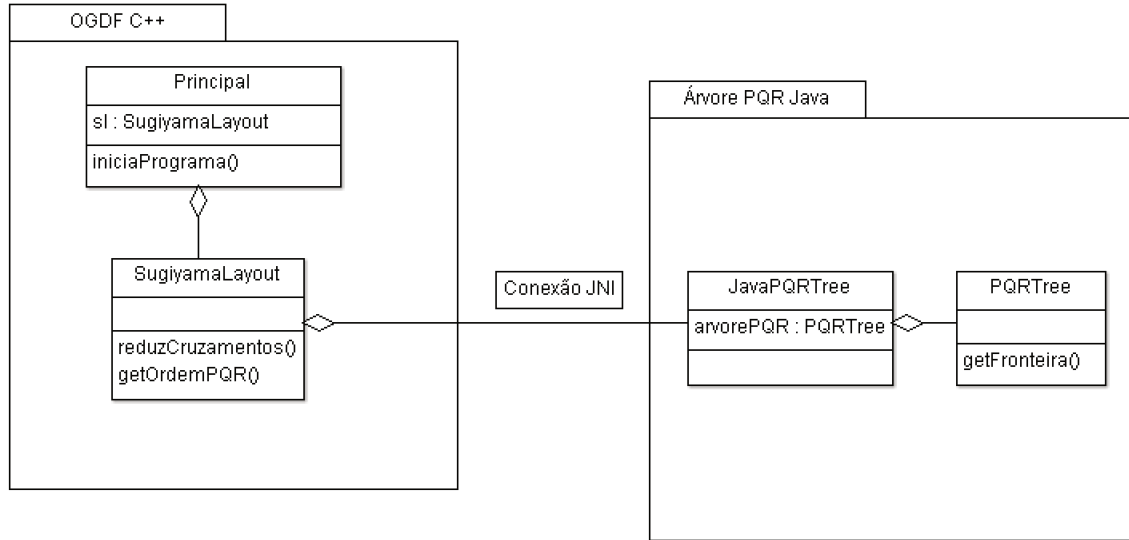


Figura 5.5: Diagrama de Classes da conexão OGDF (C++) e Árvore PQR (Java).

dos vértices do grafo dada pelas fronteiras das Árvores PQR. O método *getOrdemPQR()* utiliza uma conexão JNI que faz a ligação com as classes escritas em Java, responsáveis pela execução do algoritmo de Árvores PQR. Nessa conexão são transmitidos os conjuntos U e R de objetos, necessários para a execução desse algoritmo, e são recebidas as novas ordens dos vértices, dadas pelo método *getFronteira()*. O programa OGDF então se encarrega de reordenar o grafo de acordo com as informações vindas do programa de execução de Árvores PQR.

5.2 Resultados

Essa seção apresenta os resultados obtidos da comparação entre os algoritmos desenvolvidos neste trabalho e os algoritmos *BC* e *MEDIAN*. Serão utilizados os grafos dos *Pacotes North e Rome*, em que os critérios avaliados são o número total de cruzamentos e o tempo de execução.

O número de *fails* para o algoritmo *layer-by-layer sweep* foi escolhido empiricamente, utilizando-se como critério o tempo médio máximo de 0,1 segundo para os grupos de grafos separados por pacotes e densidades. Observou-se que após 10 *ff* (*fails* na fase de processamento da fronteira) e 2 *fr* (*fails* na fase de pré-processamento das restrições), o número de vitórias e derrotas permanece quase inalterado para todos os algoritmos comparados em ambos os pacotes, porém o tempo médio

continua crescendo, excedendo o limite de 0,1 segundo. Logo, o número de *fr* foi definido em 2 e o número de *ff* foi definido em 10.

Para a representação dos resultados relativos ao número de cruzamentos, optou-se pela utilização de gráficos de barras empilhadas, representando o número de vezes que cada método se saiu melhor na redução de cruzamentos, separando os grafos por densidades . Além desses resultados, pode-se observar a partir de histogramas em quanto cada método melhorou a qualidade dos DAGs percentualmente.

Para obter-se resultados relativos às comparações propostas anteriormente, optou-se por verificar a diferença da qualidade dos DAGs produzidos nos casos em que os métodos propostos perdem e vencem dos métodos comparados. Isso se deu através da diferença dos resultados, em percentual, em que os métodos desenvolvidos ficam distantes dos seus respectivos métodos de comparação, levando-se em consideração o número de cruzamentos dos DAGs antes da execução dos métodos de redução de cruzamentos.

Para as comparações relacionadas ao tempo de execução, optou-se pela análise de gráficos boxplot, nos quais é possível observar não apenas valores centrais, tais como média e mediana, mas também obter uma visão geral sobre a simetria dos dados e de sua distribuição.

As próximas subseções apresentam e discutem os resultados da comparação entre os métodos propostos e os métodos *BC* ou *MEDIAN*.

5.2.1 Comparação entre os métodos *BC* e *PQR_BC1*

As Figuras 5.6 e 5.7 mostram respectivamente sobre os grafos do Pacote North e Rome, o total de vitórias, derrotas e empates (em termos de cruzamentos) do método *PQR_BC1*, expresso em percentagens.

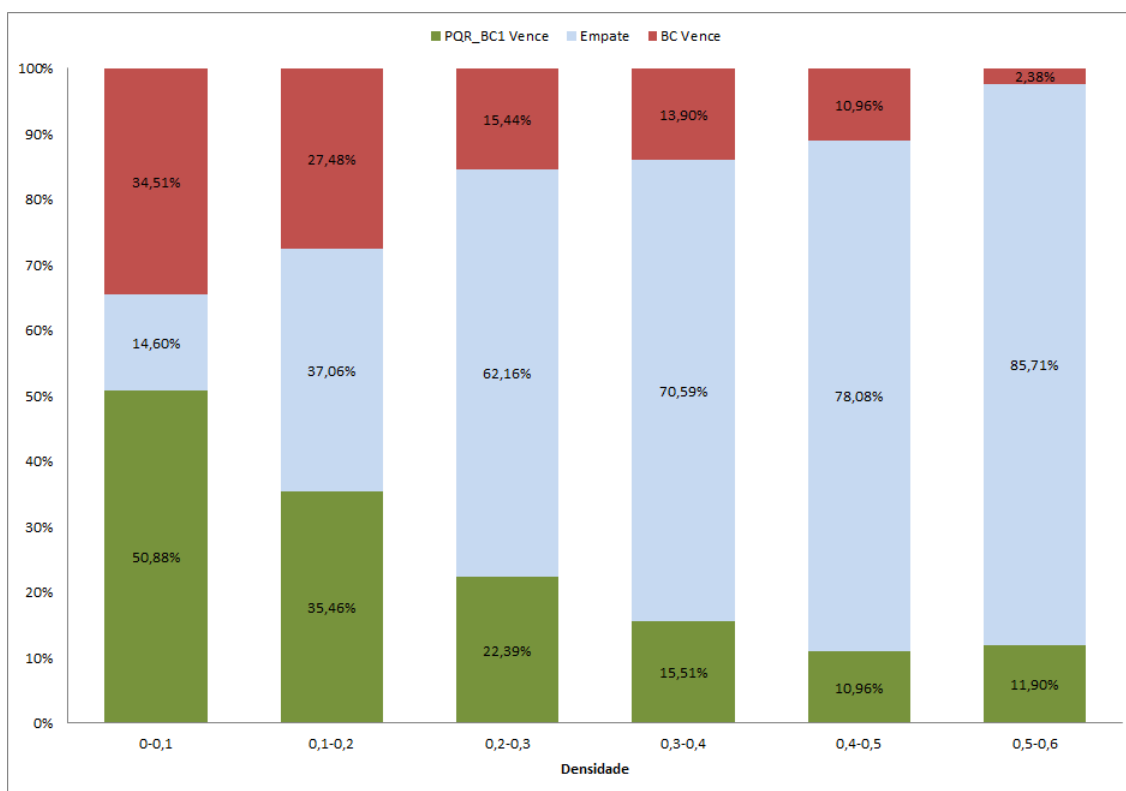


Figura 5.6: Grafos do Pacote North separados por densidades e percentuais de vitórias, derrotas e empates (em termos de cruzamentos) do método *PQR_BC1*.

Observa-se em ambos os pacotes que, à medida que a densidade dos grafos aumenta, os métodos empatam mais com relação à redução de cruzamentos. É possível afirmar também que o método *PQR_BC1* leva uma pequena vantagem no número de vezes que obtém melhores resultados do que o método *BC*, observando-se o seu percentual de vitórias em todos os grupos de densidades, em ambos os pacotes de grafos.

Além de observado o número de vezes que o método *PQR_BC1* supera o *BC*, é essencial analisar a qualidade dos resultados obtidos por ambos os métodos. Os histogramas das Figuras 5.8 (Pacote North) e 5.9 (Pacote Rome) apresentam o número total de DAGs (eixo y) agrupados pela percentagem de redução de cruzamentos (eixo x) que esses sofreram com relação aos seus números

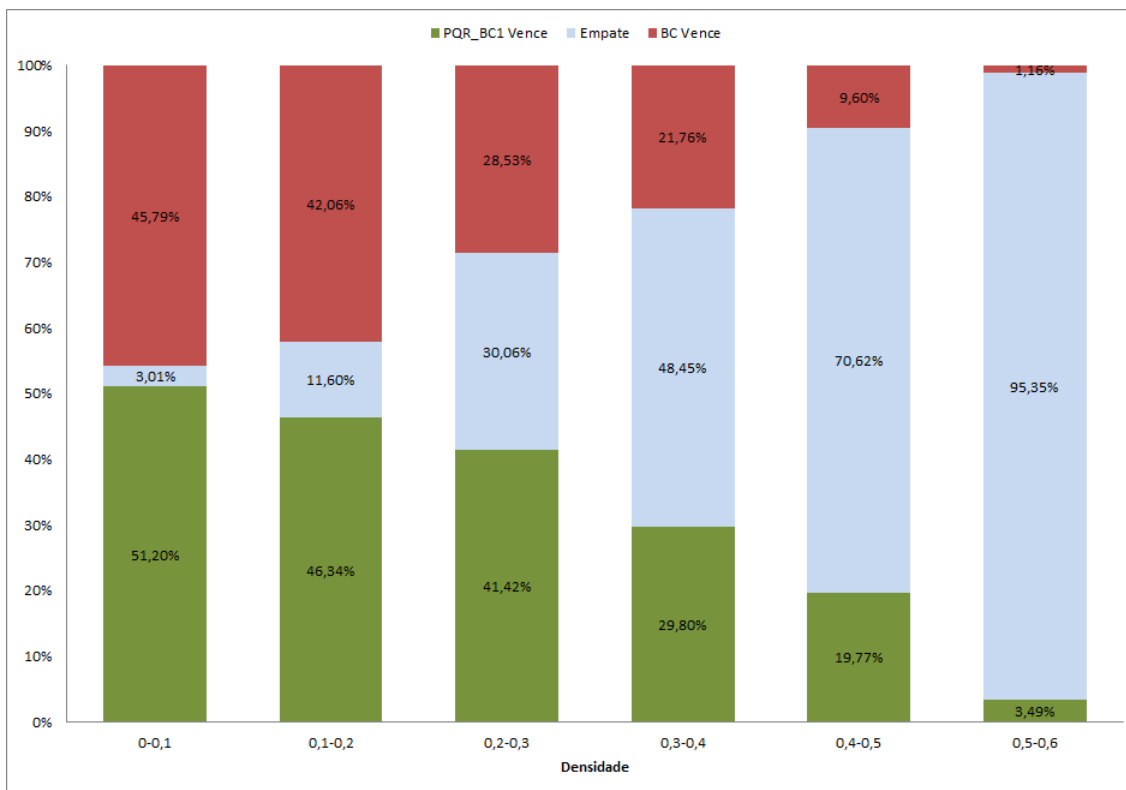


Figura 5.7: Grafos do pacote Rome separados por densidades e percentuais de vitórias, derrotas e empates (em termos de cruzamentos) do método *PQR_BC1*.

de cruzamentos iniciais.

Nota-se que existe um número maior de grafos que obtiveram seus cruzamentos reduzidos entre 80 e 100% dos cruzamentos iniciais quando utilizado o método *PQR_BC1* em ambos os pacotes de grafos (314 grafos para o pacote Rome e 12 para o Pacote North). Através dessa análise é possível observar que o número considerável de derrotas do método *PQR_BC1* (Figuras 5.7 e 5.8) não é o suficiente para afirmar que não se deve utilizar este método para redução de cruzamentos já que quando executado, ele pode alcançar menor número de cruzamentos do que o próprio *BC*.

Os gráficos das Figuras 5.10 e 5.11 apresentam dados sobre a qualidade do método *PQR_BC1* nos casos em que é vitorioso e derrotado, agrupando os DAGs dos Pacotes North e Rome, respectivamente, pela porcentagem de redução de cruzamentos de cada um deles com relação ao seu número inicial de cruzamentos.

Para o Pacote North, observa-se que o método *PQR_BC1* melhora de 0 a 5% os resultados que foram obtidos executando-se *BC* sobre os mesmos DAGs. Na maioria dos casos em que *PQR_BC1* é derrotado, os resultados obtidos ficam de 10 a 20% inferiores aos obtidos pelo *BC*. Apesar disso, o

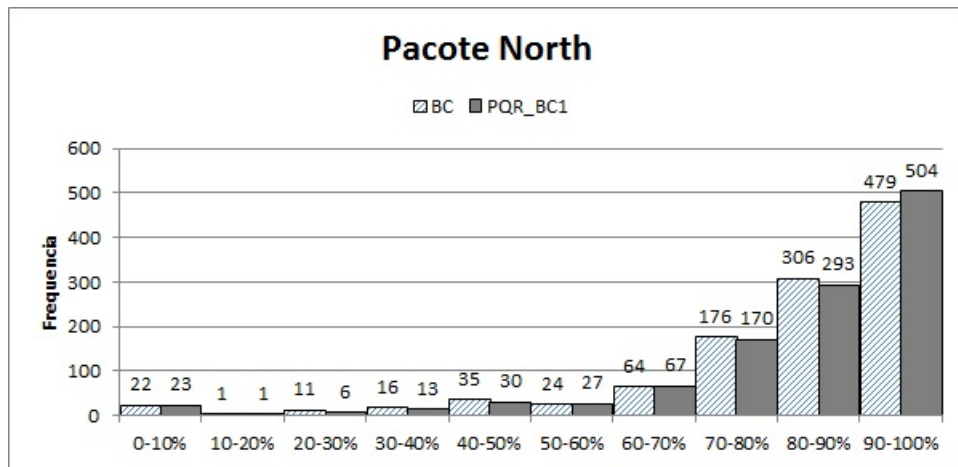


Figura 5.8: Quantidade de DAGs do Pacote North, agrupadas por percentual de redução de cruzamentos provida pelos métodos *BC* e *PQR_BC1*, com relação à quantidade inicial de cruzamentos desses grafos.

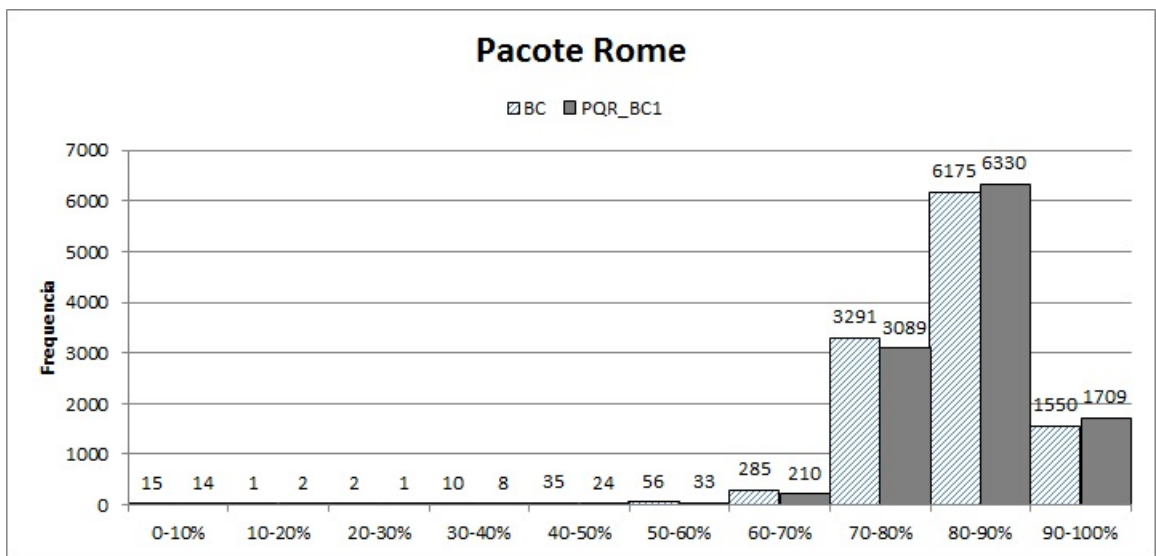


Figura 5.9: Quantidade de DAGs do Pacote Rome, agrupadas por percentual de redução de cruzamentos provida pelos métodos *BC* e *PQR_BC1*, com relação à quantidade inicial de cruzamentos desses grafos.

método *PQR_BC1* obtém maior número de vezes resultados que ficam mais do que 20% acima dos resultados obtidos por *BC* do que resultados que ficam 20% abaixo. Para o Pacote Rome, observa-se uma grande semelhança proporcional com relação ao quanto *PQR_BC1* se sobressai nas vitórias e o quanto deteriora os resultados de *BC* nas derrotas.

Outro fator importante na análise dos resultados comparando ambos os métodos é o tempo de execução. As Figuras 5.12 e 5.13 exibem um comparativo entre a distribuição dos tempos

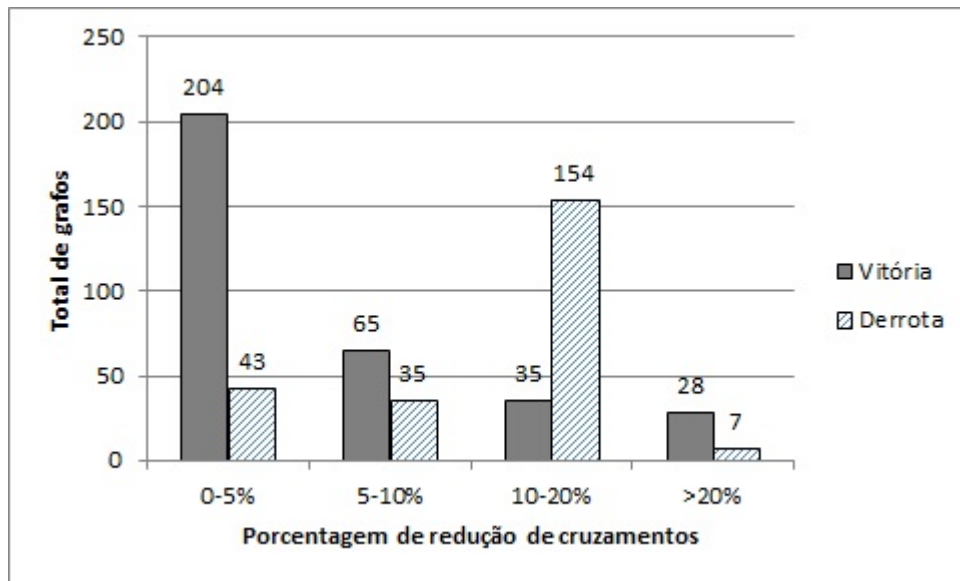


Figura 5.10: Resultados do método *PQR_BC1* com relação à melhora dos resultados do método *BC* (vitórias) e de deterioração (derrotas) separados por porcentagem, relativos ao número de cruzamentos iniciais dos DAGs do Pacote North.

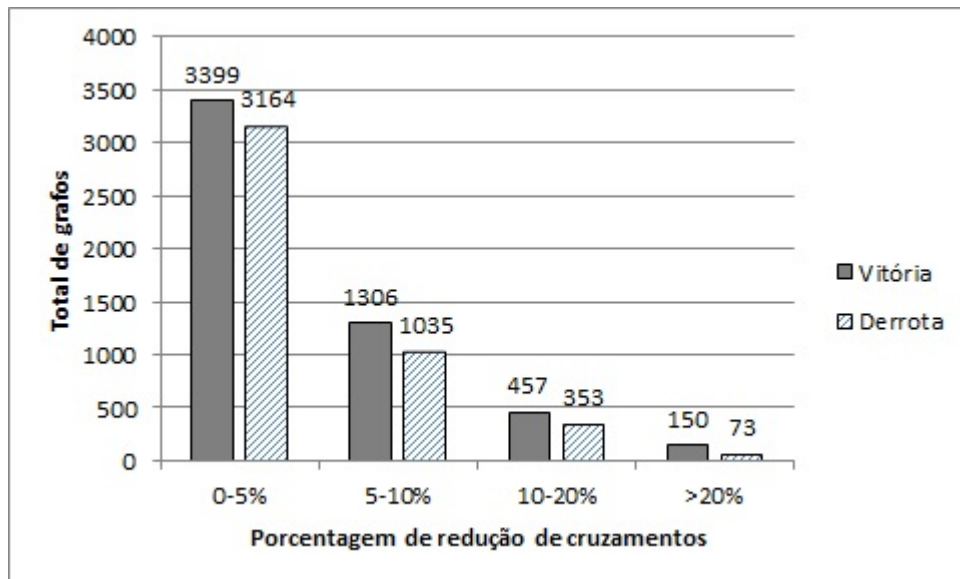


Figura 5.11: Resultados do método *PQR_BC1* com relação à melhora dos resultados do método *BC* (vitórias) e de deterioração (derrotas) separados por porcentagem, relativos ao número de cruzamentos iniciais dos DAGs do Pacote Rome.

dos métodos *BC* e *PQR_BC1*, separados por densidades, nas quais os resultados do Pacote North aparecem na coluna à esquerda e os do Rome à direita.

Os pontos vermelhos em cada boxplot indicam a média dos resultados e a faixa preta indica a mediana. Os valores discrepantes são dados por $disc < Q1 - 1,5AIQ$ ou $disc > Q3 + 1,5AIQ$ em que

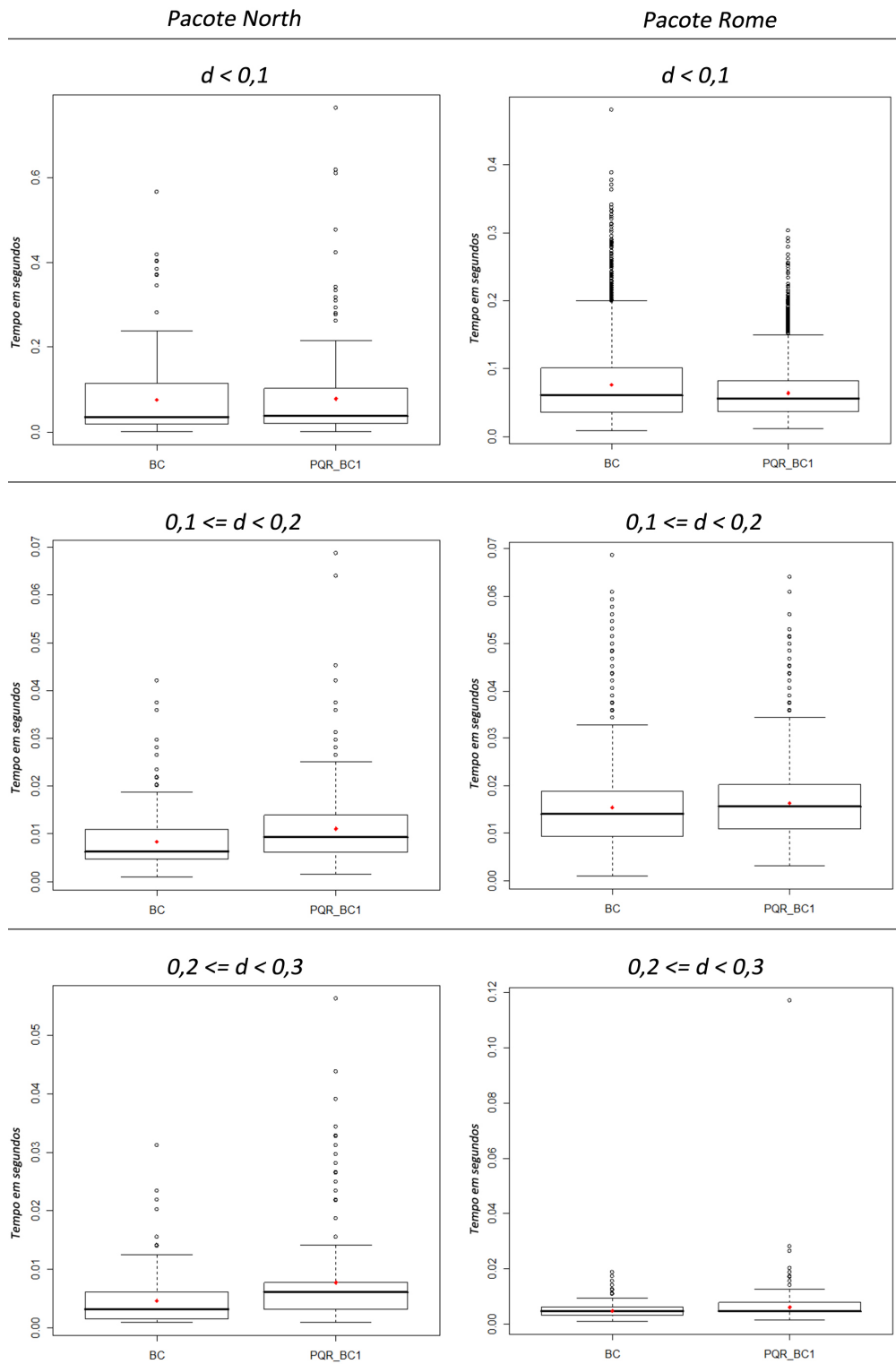


Figura 5.12: Gráficos boxplot com a distribuição dos tempos de execução dos métodos *BC* e *PQR_BC1* para as densidades iguais ou inferiores a 0,3 para os pacotes North e Rome.

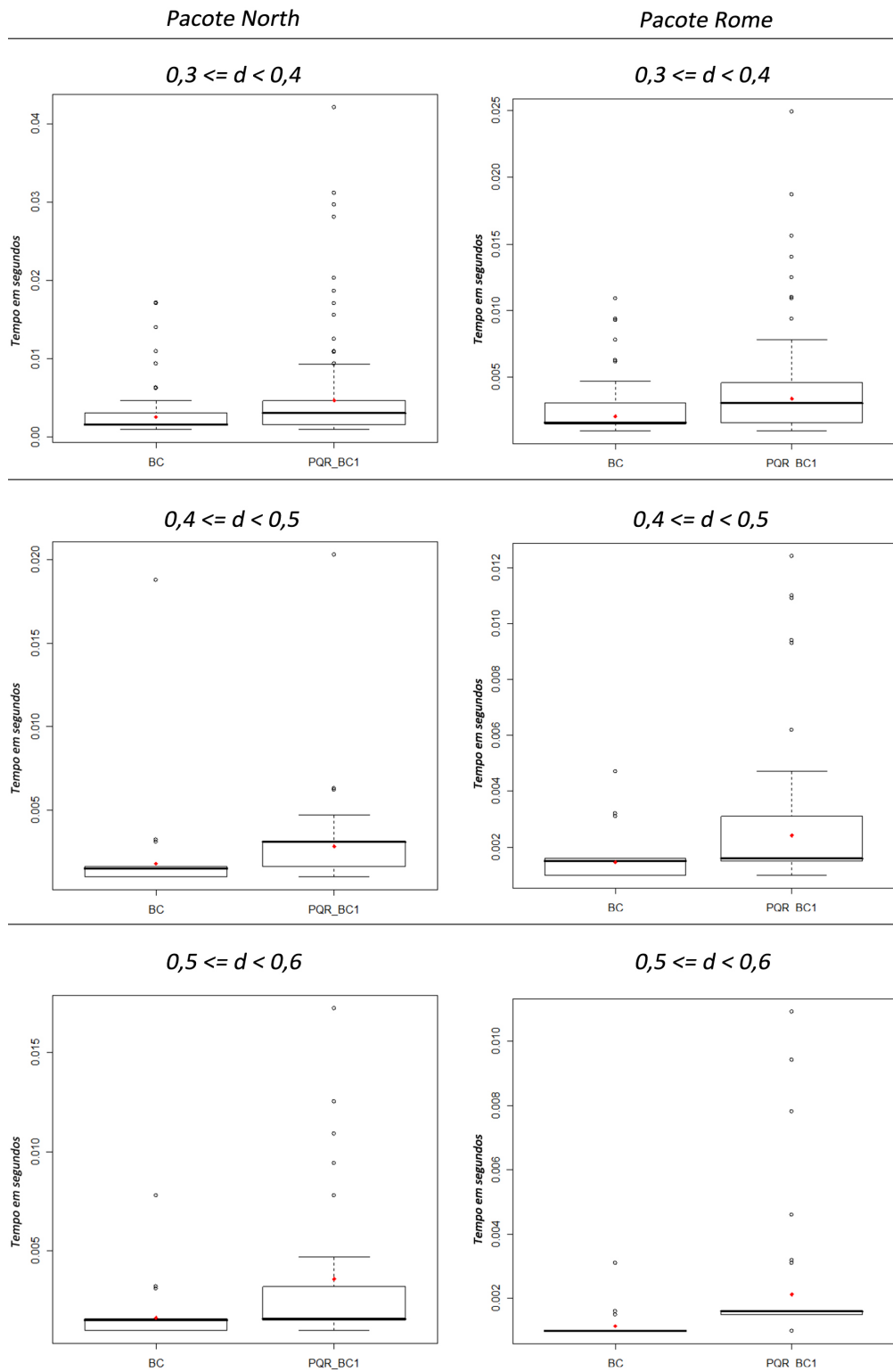


Figura 5.13: Gráficos boxplot com a distribuição dos tempos de execução dos métodos *BC* e *PQR_BC1* para as densidades iguais ou superiores a 0,3 para os pacotes North e Rome.

AIQ representa a amplitude interquartil $Q3 - Q1$, $Q1$ representa o primeiro quartil e $Q3$ representa o último quartil.

Observa-se pelas Figuras 5.12 e 5.13 que existe, na execução de ambos os métodos, uma quantidade significativa de valores discrepantes, além do comportamento assimétrico dos dados. O gráfico de boxplot indica que os grafos de maior tempo de execução são aqueles pertencentes ao grupo com densidade igual ou inferior a 0,1. Uma análise mais precisa do boxplot que representa esse grupo mostra que a distribuição dos dados é assimétrica à esquerda, indicando que a maior concentração dos dados está entre 0,001 e 0,05. Além disso, a maioria dos grafos de ambos os pacotes com estas densidades é executada em um tempo inferior a 0,1 segundo (75% dos casos), tanto para a execução do método BC como para a do método PQR_BC1 . Com relação à variação dos dados, pouco se pode afirmar em um comparativo entre os métodos BC e PQR_BC1 , para os casos de densidade inferior a 0,3, pois ambas possuem comportamento similar. Para densidades superiores a 0,3, nota-se uma maior variabilidade dos dados obtidos pelo uso do método PQR_BC1 .

As médias em cada gráfico mostram que, em geral, o método PQR_BC1 tem uma média de execução relativamente superior à do método BC . Como complemento, a Tabela 5.1 mostra a diferença das médias de BC e PQR_BC1 em segundos, destacando a relação entre o tempo de execução dos métodos e a densidade dos grafos. Em ambos os pacotes de grafos, o tempo de execução dos métodos não influencia na escolha de um dos métodos, dada a diferença menor do que 0,1 segundo.

Densidades	Pacote North	Pacote Rome
[0;0,1[0,003	-0,012
[0,1;0,2[0,003	0,001
[0,2;0,3[0,003	0,001
[0,3;0,4[0,002	0,001
[0,4;0,5[0,001	0,001
[0,5;0,6[0,002	0,001

Tabela 5.1: Diferença entre as médias de tempo de BC e PQR_BC1 . Os valores positivos indicam o quanto, em segundos, a média de PQR_BC1 está acima da média de BC

A Tabela 5.1 mostra que, no pior dos casos, a diferença entre as médias do tempo de execução de BC e PQR_BC1 é de três milissegundos, o que permite afirmar que ambos os métodos executam com tempos equivalentes. Além disso, é possível observar que o método PQR_BC1 teve uma média

de execução inferior à de *BC* nos grafos com densidade de até 0,1 no Pacote Rome.

A Tabela 5.2 mostra um resumo das comparações dos resultados obtidos nesta seção. Observa-se uma diferença de 6 pontos percentuais entre o número de vitórias e derrotas do método *PQR_BCI*. Tanto nos casos em que *PQR_BI* vence quanto nos casos em que perde, a diferença média da qualidade alcançada é de 5 pontos percentuais. Com relação ao tempo médio de execução, o algoritmo *PQR_BCI* é 0,004 segundos mais rápido que o algoritmo *BC*.

	<i>PQR_BCI</i>	<i>BC</i>
Vitórias (percentual)	45%	39%
Média de redução de cruzamentos com relação aos grafos originais nos casos de vitória de <i>PQR_BCI</i> (percentual)	85%	80%
Média de redução de cruzamentos com relação aos grafos originais nos casos de derrota de <i>PQR_BCI</i> (percentual)	80%	85%
Tempo médio de execução (segundos)	0,036	0,04

Tabela 5.2: Resumo dos resultados obtidos pela execução dos algoritmos *PQR_BCI* e *BC*

5.2.2 Comparação entre os métodos *BC* e *PQR_BC2*

As Figuras 5.14 e 5.15 mostram o total de vitórias, derrotas e empates (em termos de cruzamentos) do método *PQR_BC2* em uma comparação com o método *BC*, expresso em porcentagens, utilizando-se os pacotes de grafos North e Rome respectivamente.

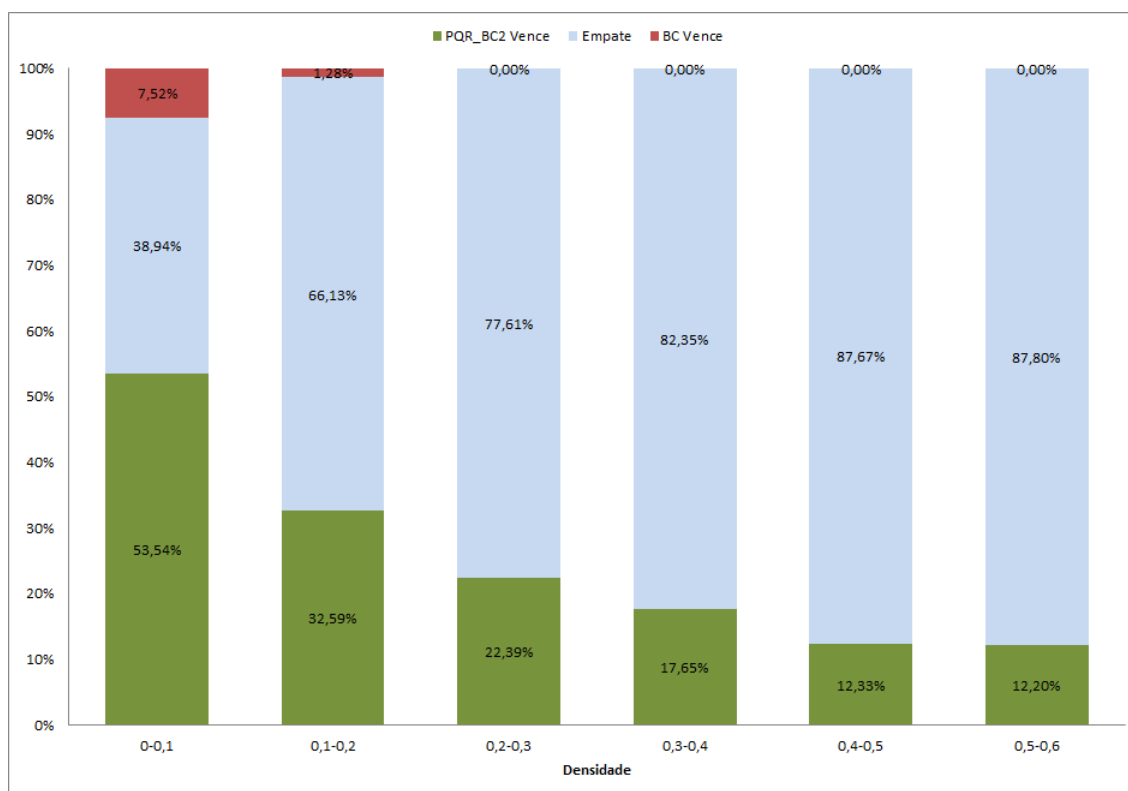


Figura 5.14: Grafos do Pacote North separados por densidades e percentuais de vitórias, derrotas e empates (em termos de cruzamentos) do método *PQR_BC2*.

Assim como acontece com o método *PQR_BC1*, à medida que a densidade dos grafos aumenta, os métodos empatam mais com relação à redução de cruzamentos. Porém, é evidente que o método *PQR_BC2* leva maior vantagem no número de vezes que obtém melhores resultados do que o método *BC* em todos os grupos de densidades, em ambos os pacotes de grafos.

Os histogramas das Figuras 5.16 e 5.17 trazem os resultados da qualidade de ambos os métodos para os grafos dos Pacotes North e Rome respectivamente. Para os grafos do Pacote North (Figura 5.16), é possível notar que o método *PQR_BC2* tem uma maior concentração de casos em que os DAGs têm o número de cruzamentos reduzidos entre 90 a 100%: 49%, comparados a 42% do *BC*. Isso indica que o método *PQR_BC2* se sobressai ao método *BC* com relação à qualidade dos DAGs

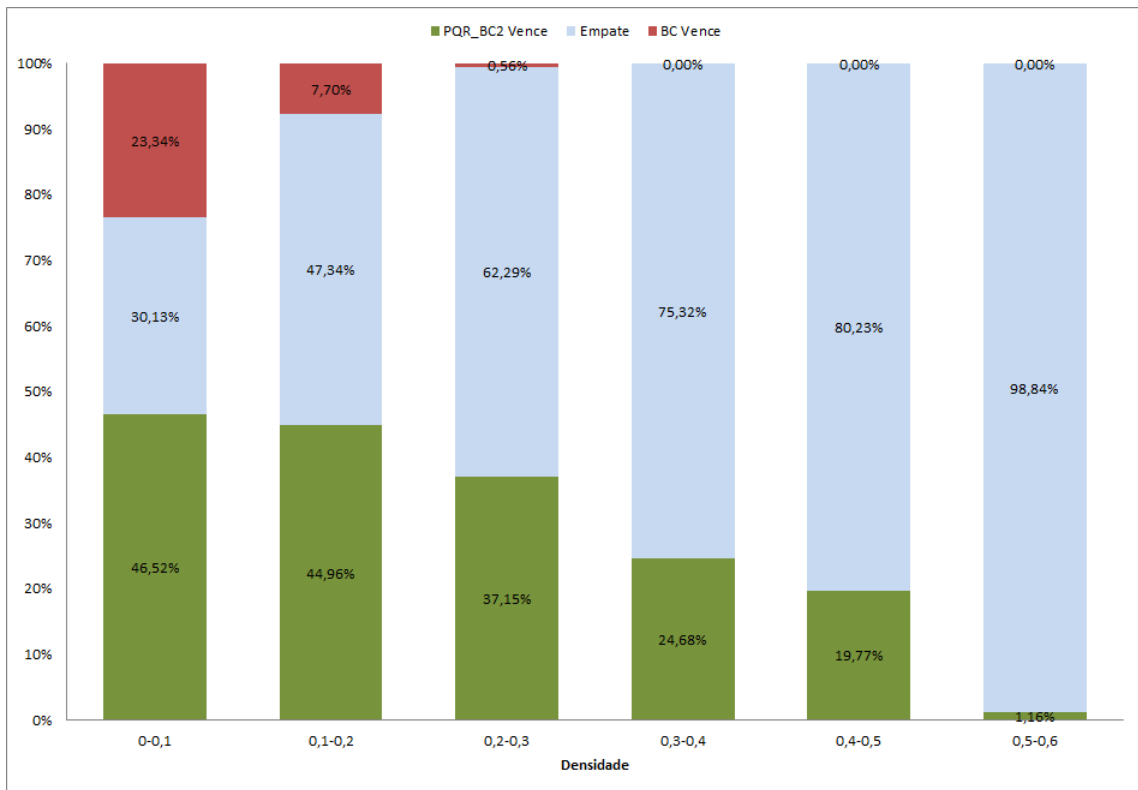


Figura 5.15: Grafos do pacote Rome separados por densidades e percentuais de vitórias, derrotas e empates (em termos de cruzamentos) do método *PQR_BC2*.

obtidos com sua execução, observando-se a característica de menor número de cruzamentos.

Nos casos do pacote Rome (Figura 5.17), o método *PQR_BC2* obtém resultados similares aos do Pacote North, alcançando uma redução de cruzamentos entre 80 e 100% em 78% dos DAGs contra 68% do método *BC*.

Os gráficos das Figuras 5.18 e 5.19 apresentam os dados da qualidade de *PQR_BC2* nos casos de vitória e derrota para os Pacotes North e Rome respectivamente.

É possível observar que em 100% dos casos em que *PQR_BC2* perde para *BC* no Pacote North, os resultados são até 5% piores que os resultados obtidos com a execução do método *BC*. Por outro lado, nos casos em que *PQR_BC2* supera o método *BC* no Pacote North, aproximadamente 10% desses superam *BC* em uma diferença acima de 20%. Somando-se os casos em que *PQR_BC2* supera *BC* com uma diferença superior a 5%, o número de DAGs é de aproximadamente 40% do total de vitórias.

Nos casos em que *PQR_BC2* supera o método *BC* no pacote Rome, aproximadamente 3% desses superam *BC* em uma diferença acima de 20%. Somando-se os casos em que *PQR_BC2* supera *BC*

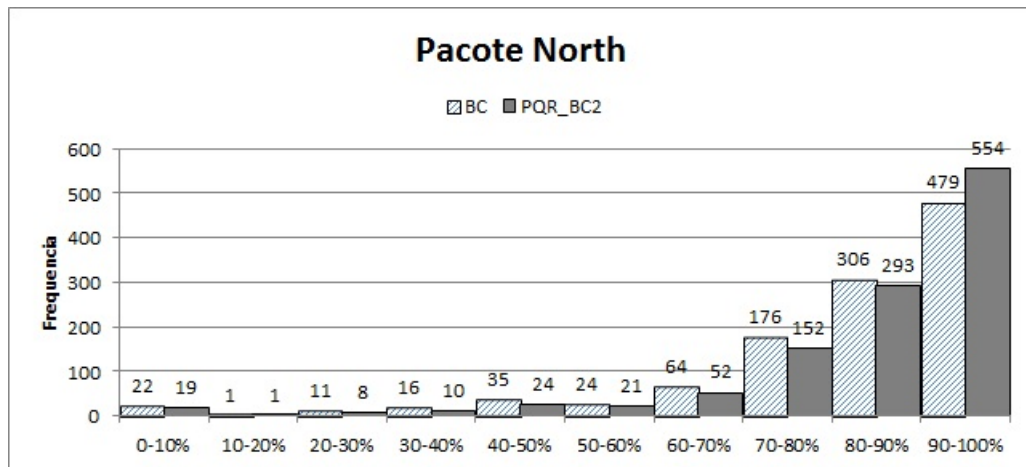


Figura 5.16: Quantidade de DAGs do Pacote North, agrupadas por percentual de redução de cruzamentos provida pelos métodos *BC* e *PQR_BC2*, com relação à quantidade inicial de cruzamentos desses grafos.

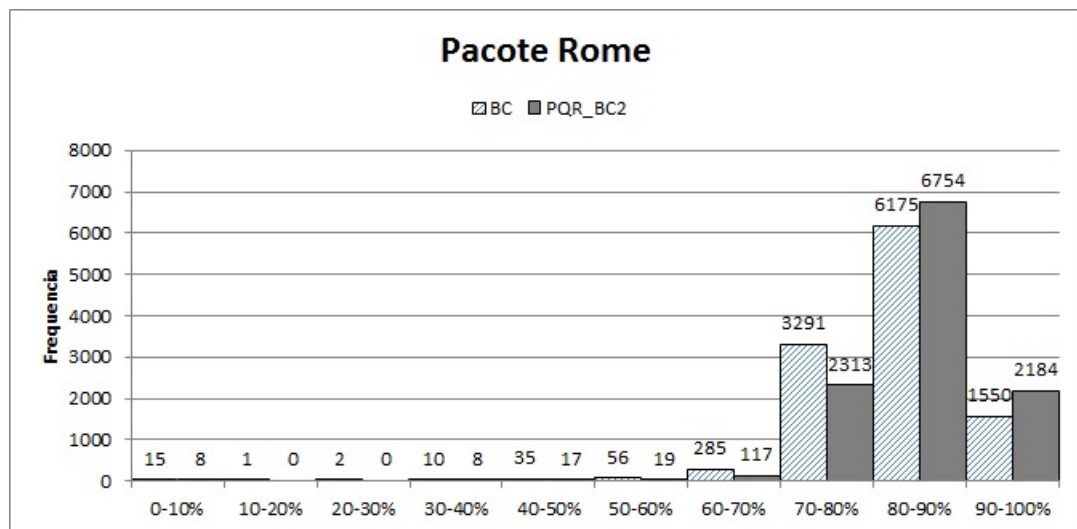


Figura 5.17: Quantidade de DAGs do Pacote Rome, agrupadas por percentual de redução de cruzamentos provida pelos métodos *BC* e *PQR_BC2*, com relação à quantidade inicial de cruzamentos desses grafos.

com uma diferença superior a 5%, o número de DAGs é de aproximadamente 35% do total de vitórias.

As Figuras 5.20 e 5.21 apresentam um comparativo entre a distribuição dos tempos dos métodos *BC* e *PQR_BC2*, separados por densidades, nas quais os resultados do Pacote North aparecem na coluna à esquerda, e os do Rome, à direita.

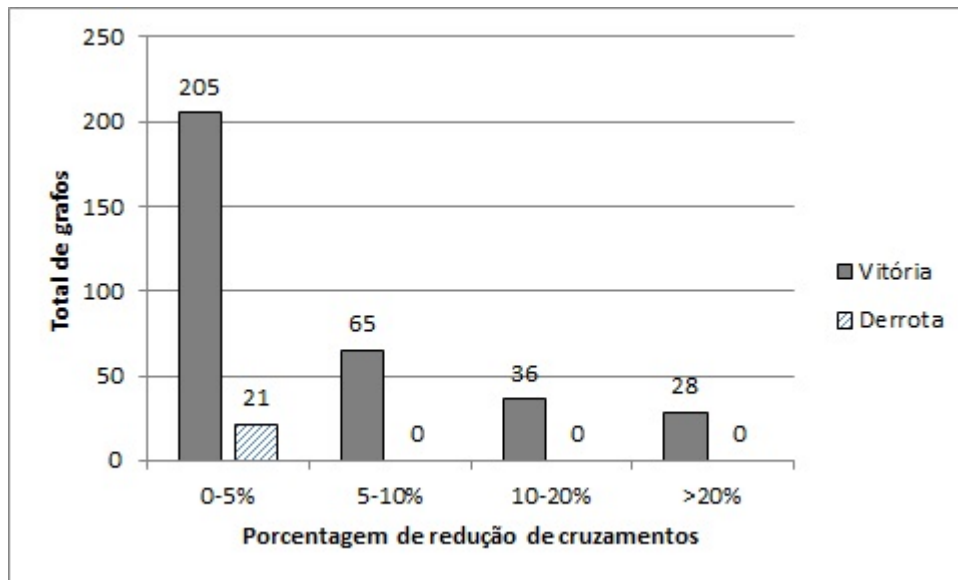


Figura 5.18: Resultados do método *PQR_BC2* com relação à melhora dos resultados do método *BC* (vitórias) e de deterioração (derrotas) separados por porcentagem, relativos ao número de cruzamentos iniciais dos DAGs do Pacote North.

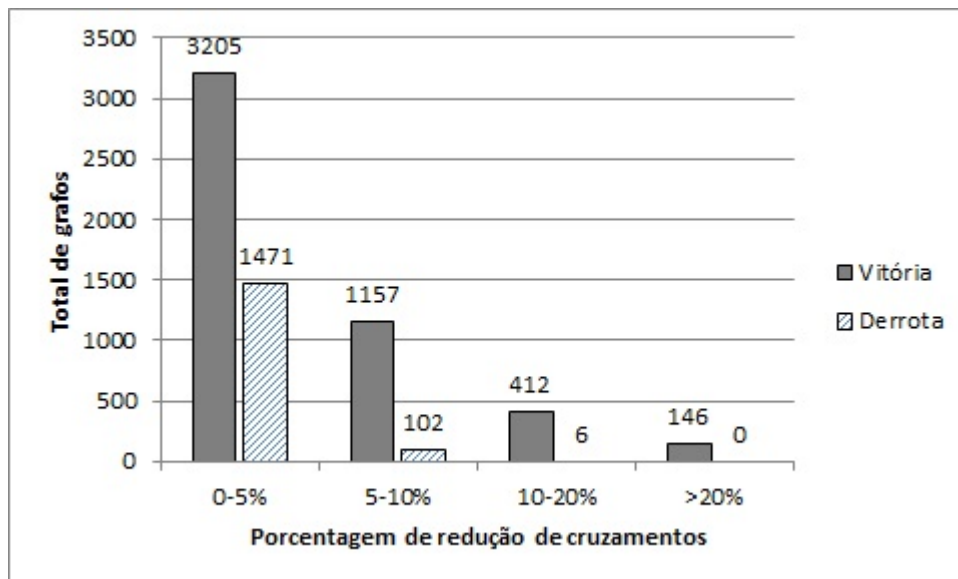


Figura 5.19: Resultados do método *PQR_BC2* com relação à melhora dos resultados do método *BC* (vitórias) e de deterioração (derrotas) separados por porcentagem, relativos ao número de cruzamentos iniciais dos DAGs do Pacote Rome.

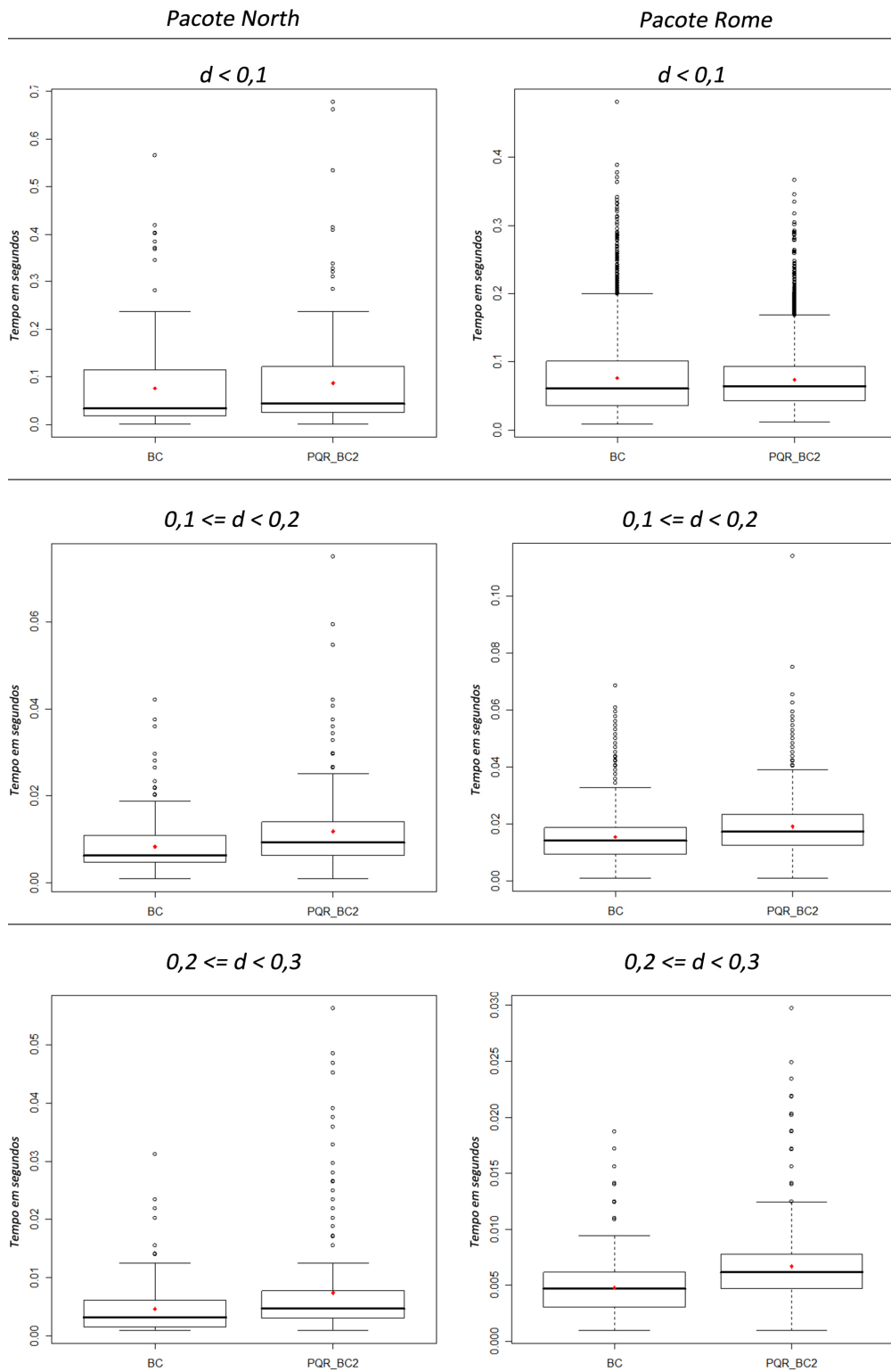


Figura 5.20: Gráficos boxplot com a distribuição dos tempos de execução dos métodos *BC* e *PQR_BC2* para as densidades iguais ou inferiores a 0,3 para os pacotes North e Rome.

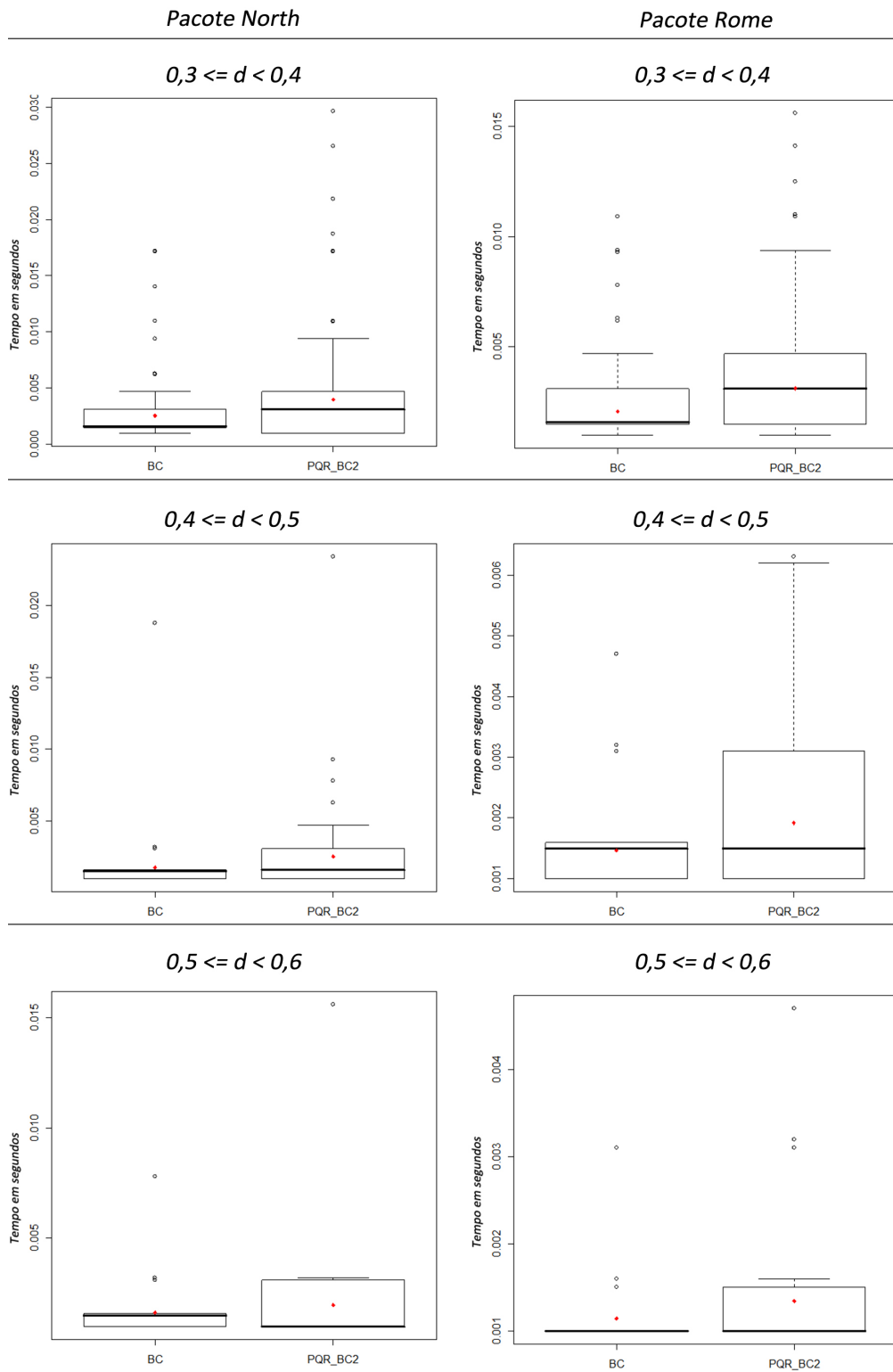


Figura 5.21: Gráficos boxplot com a distribuição dos tempos de execução dos métodos *BC* e *PQR_BC2* para as densidades iguais ou superiores a 0,3 para os pacotes North e Rome.

Observa-se nas Figuras 5.20 e 5.21 que o método *PQR_BC2* segue um comportamento similar ao de *BC* com relação à distribuição de dados. Para os casos com densidade igual ou inferior a 0,1, ambos os métodos levam muito mais tempo para serem executados do que nos casos das demais densidades. As análises de simetria e variação feitas para o método *PQR_BC1* continuam válidas para o método *PQR_BC2*.

Com relação à variação dos dados, para grafos com densidade superiores a 0,3, nota-se que a variabilidade aumentou no uso do método *PQR_BC2* com relação ao *PQR_BC1*, que já tinha uma variabilidade maior com relação ao *BC*.

Outro ponto importante a se destacar está na relação entre o tempo de execução dos métodos e a densidade dos grafos. Em ambos os pacotes de grafos, o tempo de execução dos métodos não influencia na escolha de um deles, dada a diferença menor do que 0,1 segundo.

A média dos tempos de execução indica que o método *PQR_BC2* não sofreu diferença significativa com relação ao método *PQR_BC1*, sendo relativamente superior à do método *BC*. A Tabela 5.3 mostra a diferença das médias de *BC* e *PQR_BC2* em segundos.

Densidades	Pacote North	Pacote Rome
[0;0,1[0,011	-0,002
[0,1;0,2[0,004	0,004
[0,2;0,3[0,003	0,002
[0,3;0,4[0,001	0,001
[0,4;0,5[0,001	0,000
[0,5;0,6[0,000	0,000

Tabela 5.3: Diferença entre as médias de tempo de *BC* e *PQR_BC2*. Os valores positivos indicam o quanto, em segundos, a média de *PQR_BC2* está acima da média de *BC*

A Tabela 5.3 mostra que, no pior dos casos, a diferença entre as médias de execução de *PQR_BC2* e *BC* é de 0,011 segundos, o que equivale a um décimo aproximadamente do tempo limite de execução estabelecido neste trabalho de um décimo de segundo. Além disso, é possível observar que o método *PQR_BC2* teve uma média de execução inferior à de *BC* nos grafos com densidade de até 0,1 no Pacote Rome.

A Tabela 5.4 mostra um resumo das comparações dos resultados obtidos nesta seção. Existe uma diferença de 29 pontos percentuais do número de vezes que o método *PQR_BC2* supera o método *BC* com relação à qualidade. Nos casos em que vence, o método *PQR_BC2*, em média, supera o

método *BC* em 5 pontos percentuais no número de cruzamentos do DAG original, alcançando 85% de redução de cruzamentos. Nos casos em que perde, o método *PQR_BC2* fica em média 2 pontos percentuais abaixo dos resultados obtidos com *BC*, alcançando 81% de redução de cruzamentos. A diferença média de tempo entre os métodos é de 0,001 segundo a mais do método *PQR_BC2*.

	<i>PQR_BC2</i>	<i>BC</i>
Vitórias (percentual)	42%	13%
Média de redução de cruzamentos com relação aos grafos originais nos casos de vitória de <i>PQR_BC2</i> (percentual)	85%	80%
Média de redução de cruzamentos com relação aos grafos originais nos casos de derrota de <i>PQR_BC2</i> (percentual)	81%	83%
Tempo médio de execução (segundos)	0,041	0,04

Tabela 5.4: Resumo dos resultados obtidos pela execução dos algoritmos *PQR_BC2* e *BC*

5.2.3 Comparação entre os métodos *MEDIAN* e *PQR_M1*

As Figuras 5.22 e 5.23 mostram respectivamente sobre os grafos do Pacote North e Rome, o total de vitórias, derrotas e empates (em termos de cruzamentos) do método *PQR_M1*, expresso em percentagens.

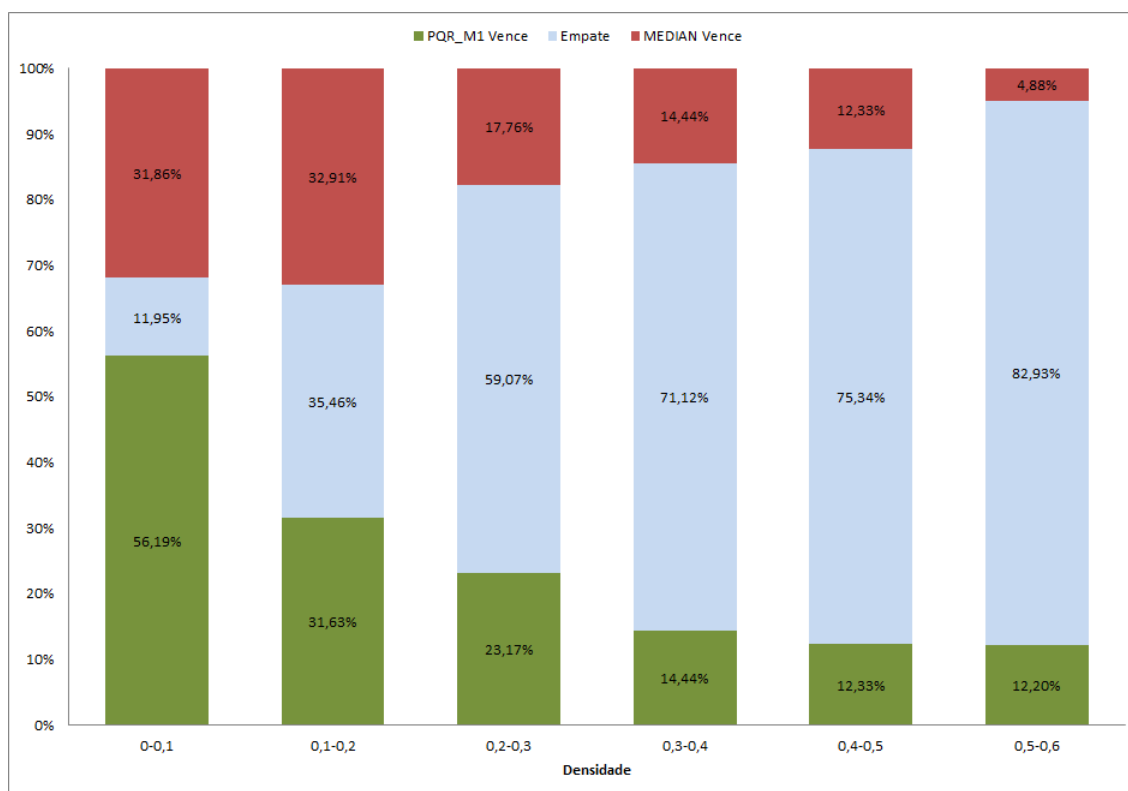


Figura 5.22: Grafos do Pacote North separados por densidades e percentuais de vitórias, derrotas e empates (em termos de cruzamentos) do método *PQR_M1*.

À medida que a densidade dos grafos aumenta, em ambos os pacotes, os métodos empatam mais com relação à redução de cruzamentos. A partir de densidades superiores a 0,2, o método *PQR_M1* passa a obter uma percentagem de vitórias mais significativa do que de derrotas no Pacote Rome. É importante observar também que o método *PQR_M1* obtém melhores resultados do que o método *MEDIAN* observando-se o seu percentual de vitórias em todos os grupos de densidades, em ambos os pacotes de grafos, exceto para o grupo de densidade $0,1 < d < 0,2$ do Pacote North (1,28% de diferença).

Os histogramas das Figuras 5.24 e 5.25 indicam a qualidade de ambos os métodos para os grafos dos Pacotes North e Rome respectivamente. Nota-se que existe um número maior de grafos que

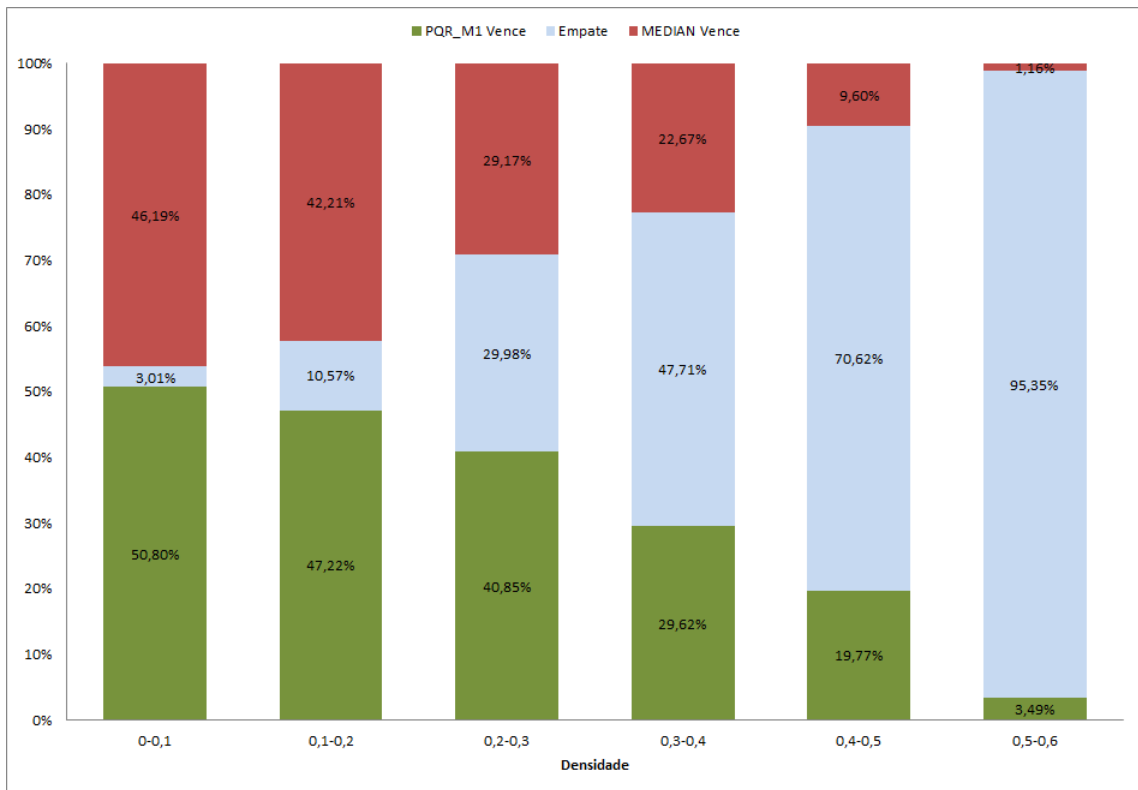


Figura 5.23: Grafos do pacote Rome separados por densidades e percentuais de vitórias, derrotas e empates (em termos de cruzamentos) do método *PQR_M1*.

obtiveram seus cruzamentos reduzidos entre 80 e 100% dos cruzamentos iniciais quando utilizado o método *PQR_M1* do que com o *MEDIAN* em ambos os pacotes de grafos (275 grafos para o pacote Rome e 12 para o Pacote North). Assim, pode-se afirmar que o método *PQR_M1* consegue alcançar resultados melhores em uma quantidade de vezes maior do que o método *MEDIAN*.

Os gráficos das Figuras 5.26 e 5.27 apresentam os dados da qualidade de *PQR_M1* nos casos de vitória e derrota para os Pacotes North e Rome respectivamente.

Para ambos os pacotes, observa-se uma semelhança proporcional com relação ao quanto *PQR_M1* se sobressai nas vitórias e o quanto deteriora os resultados de *MEDIAN* nas derrotas.

As Figuras 5.28 e 5.29 apresentam um comparativo entre a distribuição dos tempos dos métodos *MEDIAN* e *PQR_M1*, separados por densidades, nas quais os resultados do Pacote North aparecem na coluna à esquerda e os do Rome à direita.

Na Figura 5.28, nota-se que os grafos com densidade inferior a 0,1 levam um período de tempo muito maior do que os demais grafos para serem executados. Porém, a distribuição dos dados nessa densidade é assimétrica à esquerda, indicando que a maior concentração dos dados está entre 0,001

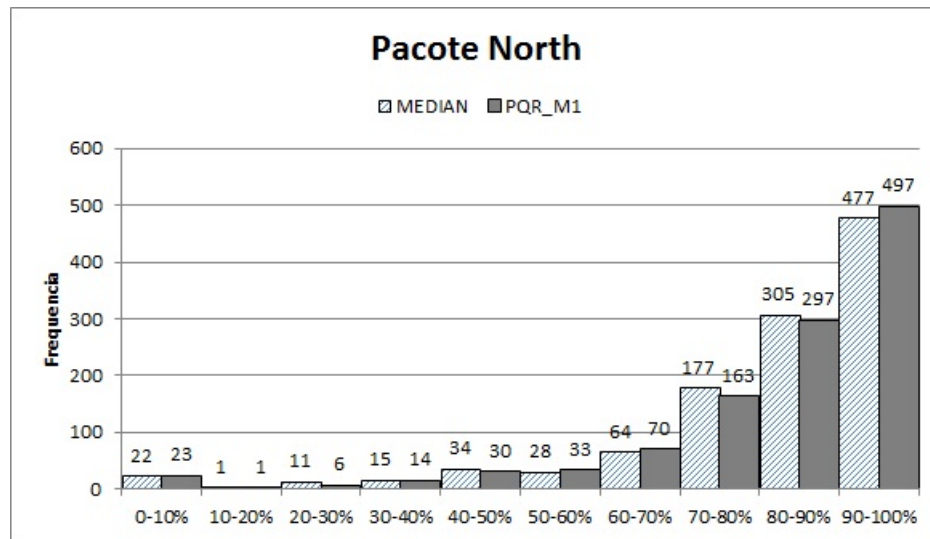


Figura 5.24: Quantidade de DAGs do Pacote North, agrupadas por percentual de redução de cruzamentos provida pelos métodos *MEDIAN* e *PQR_M1*, com relação à quantidade inicial de cruzamentos desses grafos.

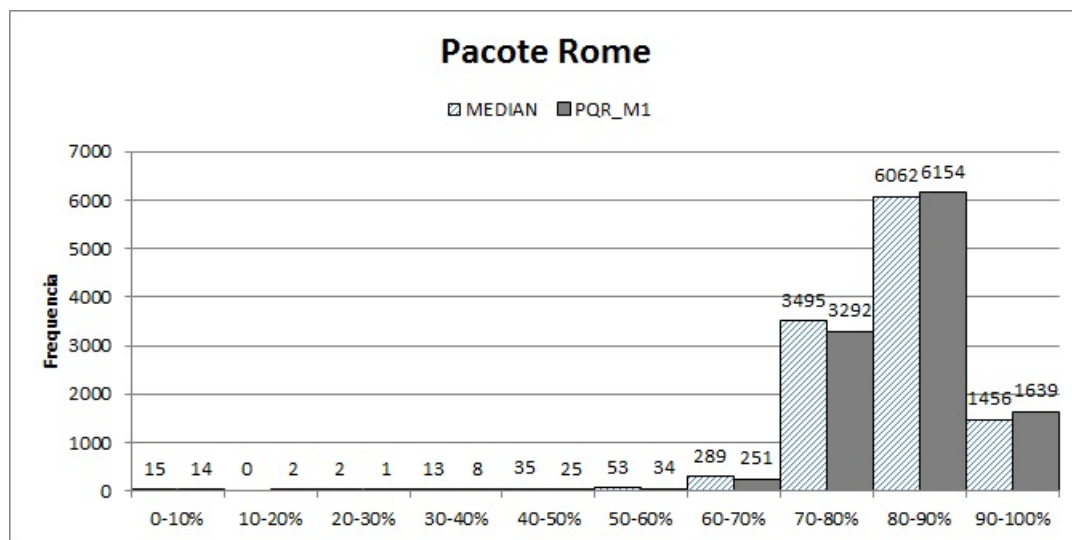


Figura 5.25: Quantidade de DAGs do Pacote Rome, agrupadas por percentual de redução de cruzamentos provida pelos métodos *MEDIAN* e *PQR_M1*, com relação à quantidade inicial de cruzamentos desses grafos.

e 0,05. A maioria dos grafos de ambos os pacotes com densidade inferior a 0,1 é executada em um tempo inferior a 0,1 segundo (75% dos casos), tanto se utilizando o método *MEDIAN* como o método *PQR_M1*.

O método *PQR_M1* tem uma média de tempo de execução relativamente superior ao do método *MEDIAN*. A Tabela 5.5 mostra a diferença das médias de *MEDIAN* e *PQR_M1* em segundos.

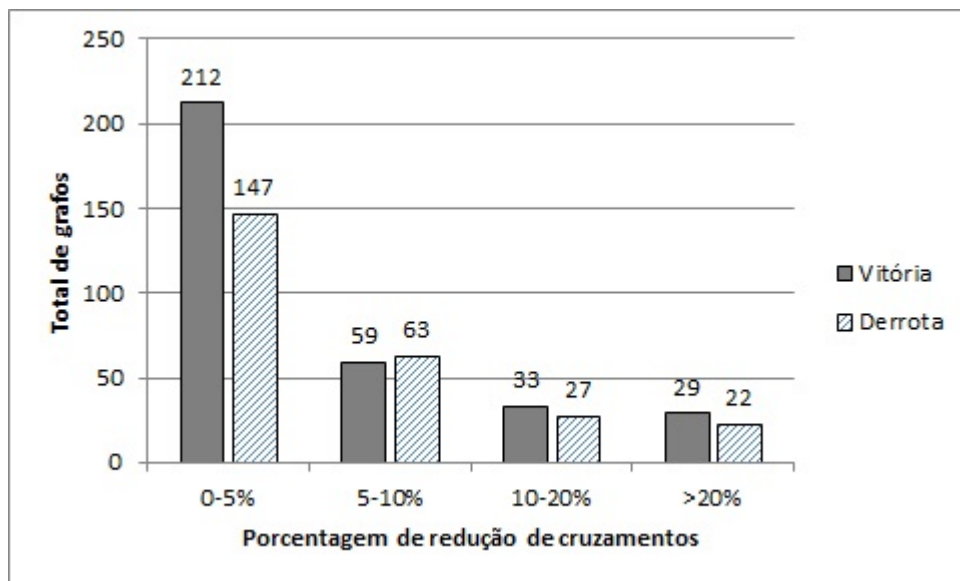


Figura 5.26: Resultados do método *PQR_M1* com relação à melhora dos resultados do método *MEDIAN* (vitórias) e de deterioração (derrotas) separados por porcentagem, relativos ao número de cruzamentos iniciais dos DAGs do Pacote North.

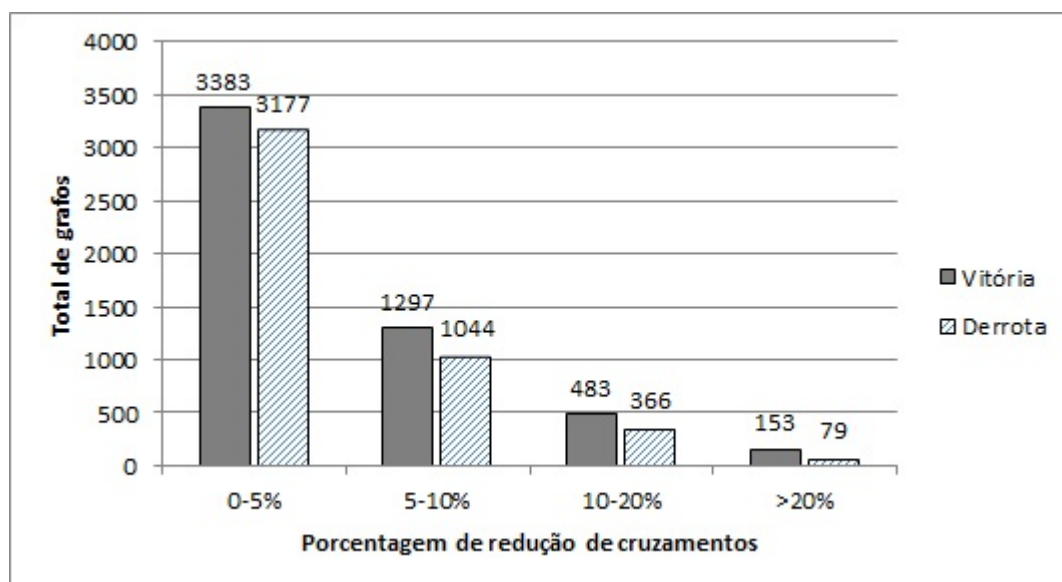


Figura 5.27: Resultados do método *PQR_M1* com relação à melhora dos resultados do método *MEDIAN* (vitórias) e de deterioração (derrotas) separados por porcentagem, relativos ao número de cruzamentos iniciais dos DAGs do Pacote Rome.

A Tabela 5.5 mostra que, no pior dos casos, a diferença entre as médias do tempo de execução de *MEDIAN* e *PQR_M1* é de 0,004 segundos, o que permite afirmar que ambos os métodos executam com tempos equivalentes.

A Tabela 5.6 mostra um resumo das comparações dos resultados obtidos nesta seção. Observa-se

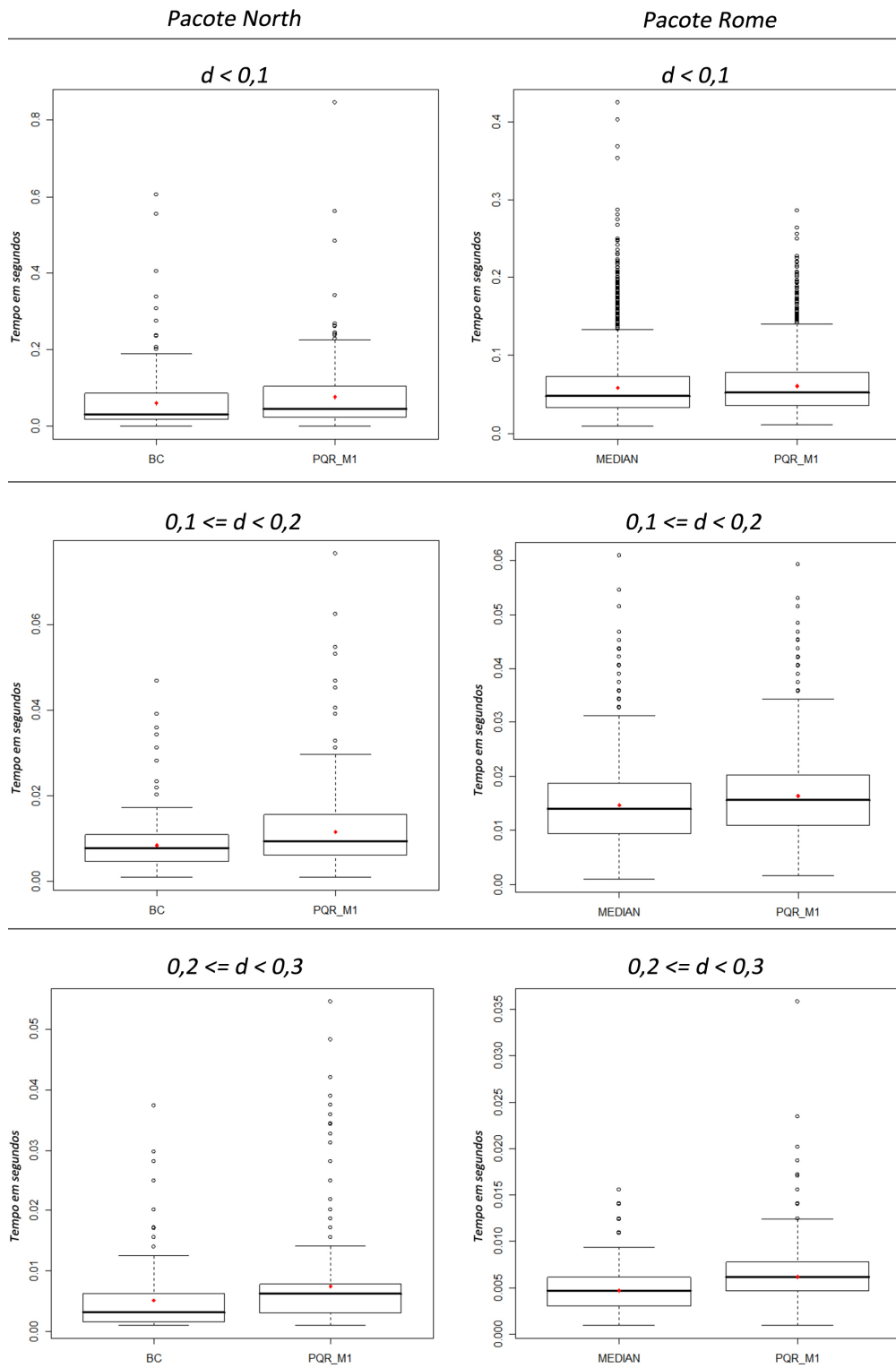


Figura 5.28: Gráficos boxplot com a distribuição dos tempos de execução dos métodos *MEDIAN* e *PQR_M1* para as densidades iguais ou inferiores a 0,3 para os pacotes North e Rome.

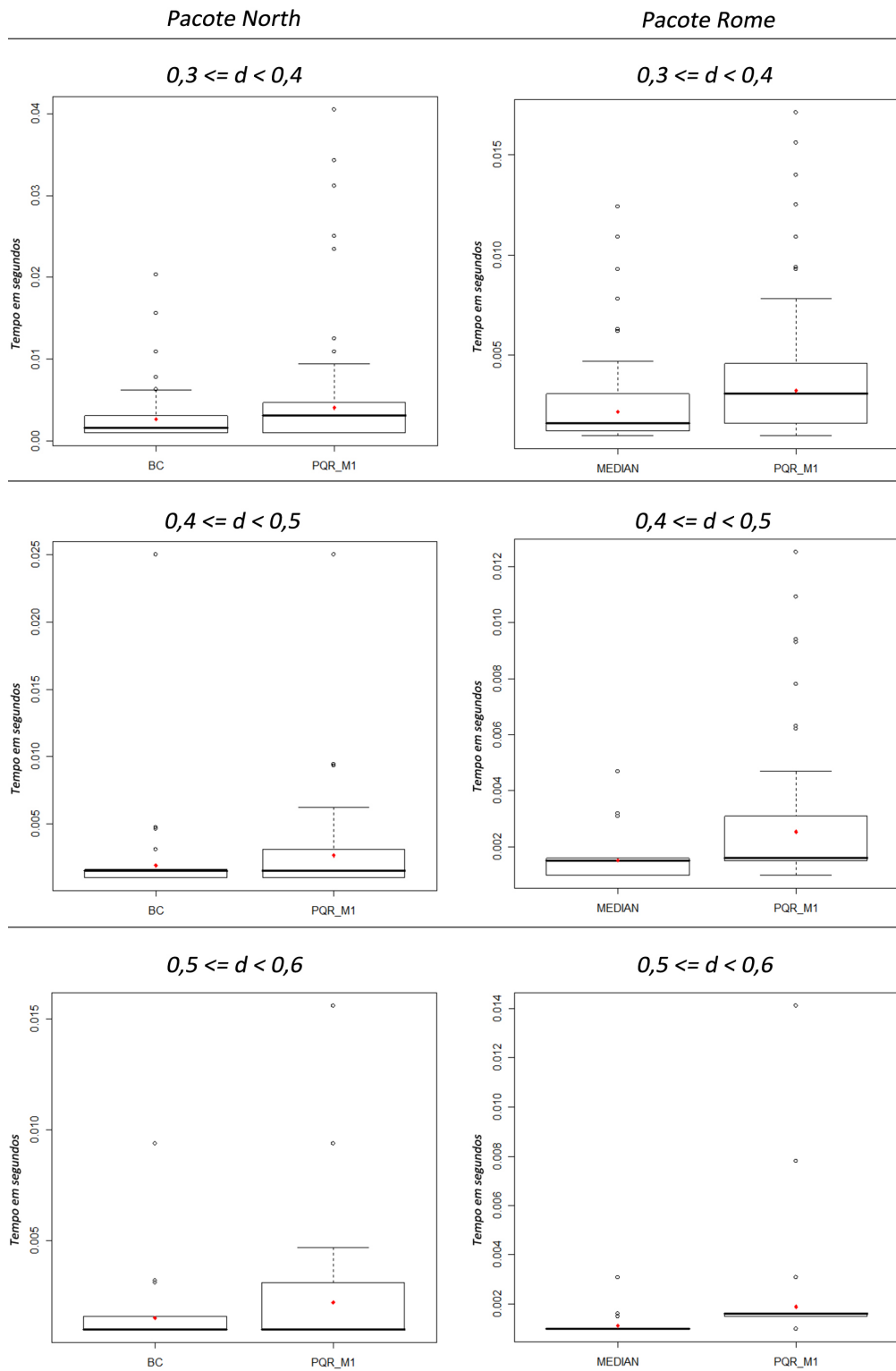


Figura 5.29: Gráficos boxplot com a distribuição dos tempos de execução dos métodos *MEDIAN* e *PQR_M1* para as densidades iguais ou superiores a 0,3 para os pacotes North e Rome.

Densidades	Pacote North	Pacote Rome
[0;0,1[0,004	0,002
[0,1;0,2[0,002	0,002
[0,2;0,3[0,002	0,001
[0,3;0,4[0,002	0,001
[0,4;0,5[0,001	0,001
[0,5;0,6[0,002	0,001

Tabela 5.5: Diferença entre as médias de tempo de *MEDIAN* e *PQR_MI*. Os valores positivos indicam o quanto, em segundos, a média de *PQR_MI* está acima da média de *MEDIAN*

uma diferença de 6 pontos percentuais entre o número de vitórias e derrotas do método *PQR_MI*. Para os casos em que *PQR_MI* vence, a diferença média da qualidade alcançada é superior em 12 pontos percentuais ao número de cruzamentos. Nos casos em que *PQR_MI* perde, a diferença média da qualidade alcançada é de 11 pontos percentuais. Com relação ao tempo médio de execução, o algoritmo *PQR_MI* é 0,001 segundo mais lento que o algoritmo *MEDIAN*.

	<i>PQR_MI</i>	<i>MEDIAN</i>
Vitórias (percentual)	45%	39%
Média de redução de cruzamentos com relação aos grafos originais nos casos de vitória de <i>PQR_MI</i> (percentual)	88%	76%
Média de redução de cruzamentos com relação aos grafos originais nos casos de derrota de <i>PQR_MI</i> (percentual)	76%	87%
Tempo médio de execução (segundos)	0,033	0,032

Tabela 5.6: Resumo dos resultados obtidos pela execução dos algoritmos *PQR_MI* e *MEDIAN*

5.2.4 Comparação entre os métodos *MEDIAN* e *PQR_M2*

As Figuras 5.30 e 5.31 mostram o total de vitórias, derrotas e empates (em termos de cruzamentos) do método *PQR_M2* em uma comparação com o método *MEDIAN*, expresso em porcentagens, utilizando-se os pacotes de grafos North e Rome respectivamente.

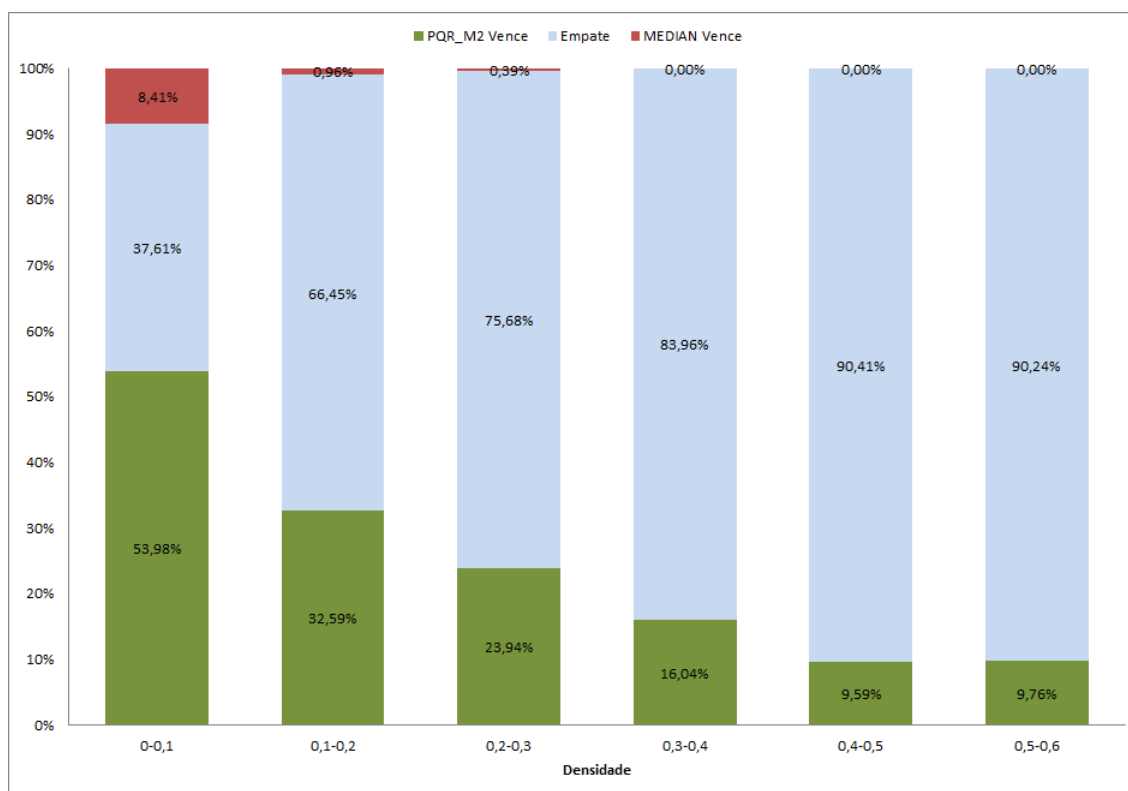


Figura 5.30: Grafos do Pacote North separados por densidades e percentuais de vitórias, derrotas e empates (em termos de cruzamentos) do método *PQR_M2*.

Em ambos os pacotes observa-se que *PQR_M2* obtém um percentual de vitórias muito superior ao *MEDIAN*. Destaca-se que para densidades muito altas (acima de 0,5), ambos são equivalentes, já que o número de empates ultrapassa os 90%.

Os histogramas das Figuras 5.32 e 5.33 trazem os resultados da qualidade de ambos os métodos para os grafos dos Pacotes North e Rome respectivamente. Para os grafos do Pacote North (Figura 5.32), é possível notar que o método *PQR_M2* tem uma maior concentração de casos em que os DAGs têm o número de cruzamentos reduzidos entre 90 a 100%: 48%, comparados a 42% do *MEDIAN*. Isso indica que o método *PQR_M2* se sobressai ao método *MEDIAN* com relação à qualidade dos DAGs obtidos com sua execução, observando-se a característica de menor número

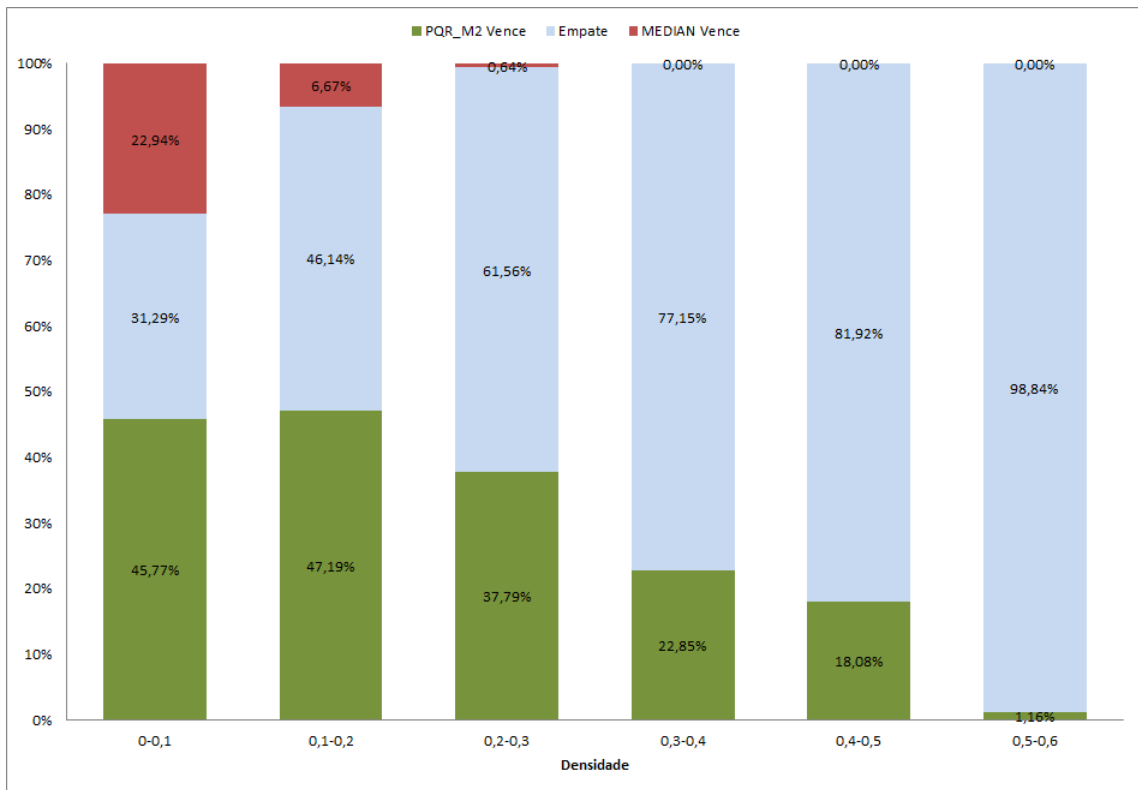


Figura 5.31: Grafos do pacote Rome separados por densidades e percentuais de vitórias, derrotas e empates (em termos de cruzamentos) do método *PQR_M2*.

de cruzamentos.

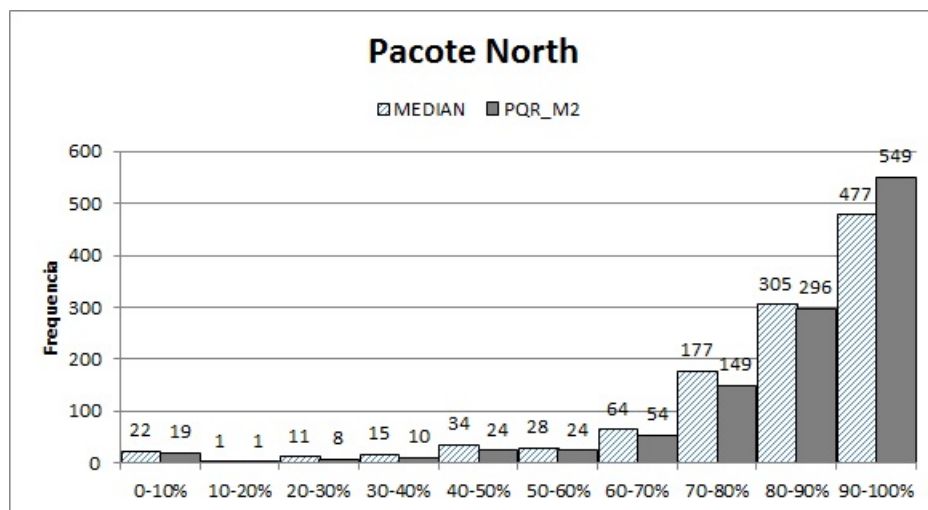


Figura 5.32: Quantidade de DAGs do Pacote North, agrupadas por percentual de redução de cruzamentos provida pelos métodos *MEDIAN* e *PQR_M2*, com relação à quantidade inicial de cruzamentos desses grafos.

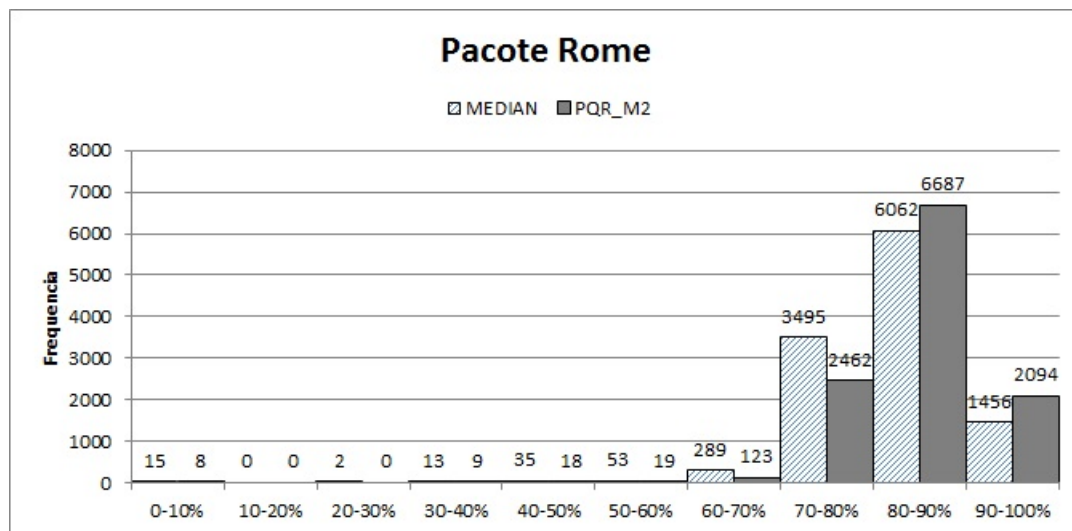


Figura 5.33: Quantidade de DAGs do Pacote Rome, agrupadas por percentual de redução de cruzamentos provida pelos métodos *MEDIAN* e *PQR_M2*, com relação à quantidade inicial de cruzamentos desses grafos.

Nos casos do pacote Rome (Figura 5.33), o método *PQR_M2* obtém resultados similares aos obtidos com o Pacote North, alcançando uma redução de cruzamentos entre 80 e 100% em 77% dos DAGs contra 66% do método *MEDIAN*.

Os gráficos das Figuras 5.34 e 5.35 apresentam os dados da qualidade de *PQR_M2* nos casos de vitória e derrota. É possível observar que em 100% dos casos em que *PQR_M2* perde para *MEDIAN* no Pacote North, os resultados são no máximo 5% piores que os resultados obtidos com a execução do método *MEDIAN*. Nos casos em que *PQR_M2* supera o método *MEDIAN* no Pacote North, aproximadamente 8% desses superam *MEDIAN* em uma diferença acima de 20%. Somando-se os casos em que *PQR_M2* supera *MEDIAN* com uma diferença superior a 5%, o número de DAGs é de aproximadamente 40% do total de vitórias.

Nos casos em que *PQR_M2* supera o método *MEDIAN* no pacote Rome, aproximadamente 3% desses superam *MEDIAN* em uma diferença acima de 20%. Somando-se os casos em que *PQR_M2* supera *MEDIAN* com uma diferença superior a 5%, o número de DAGs é de aproximadamente 36% do total de vitórias.

As Figuras 5.36 e 5.37 apresentam um comparativo entre a distribuição dos tempos dos métodos *MEDIAN* e *PQR_M2*, nas quais os resultados do Pacote North aparecem na coluna à esquerda, e os do Rome, à direita, separados por densidades.

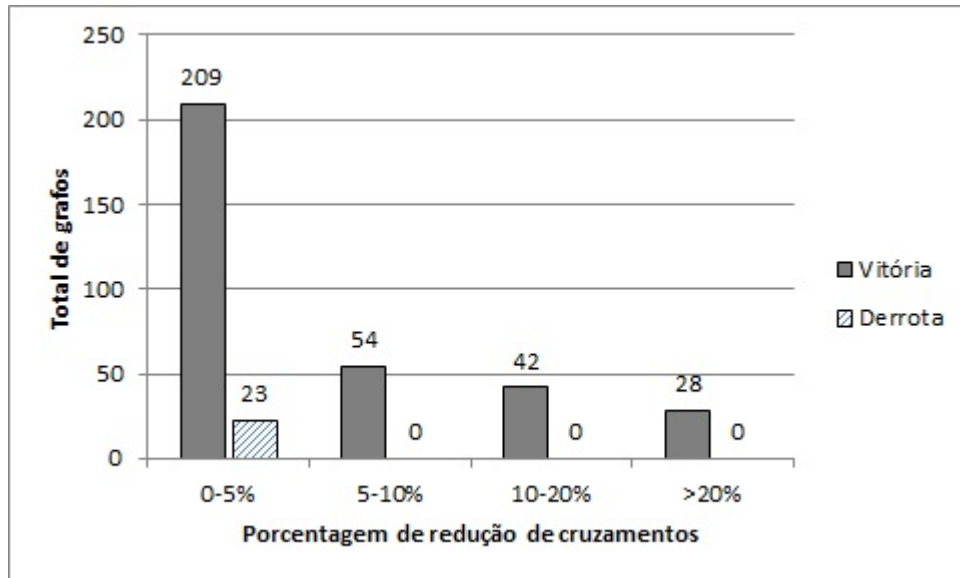


Figura 5.34: Resultados do método *PQR_M2* com relação à melhora dos resultados do método *MEDIAN* (vitórias) e de deterioração (derrotas) separados por porcentagem, relativos ao número de cruzamentos iniciais dos DAGs do Pacote North.

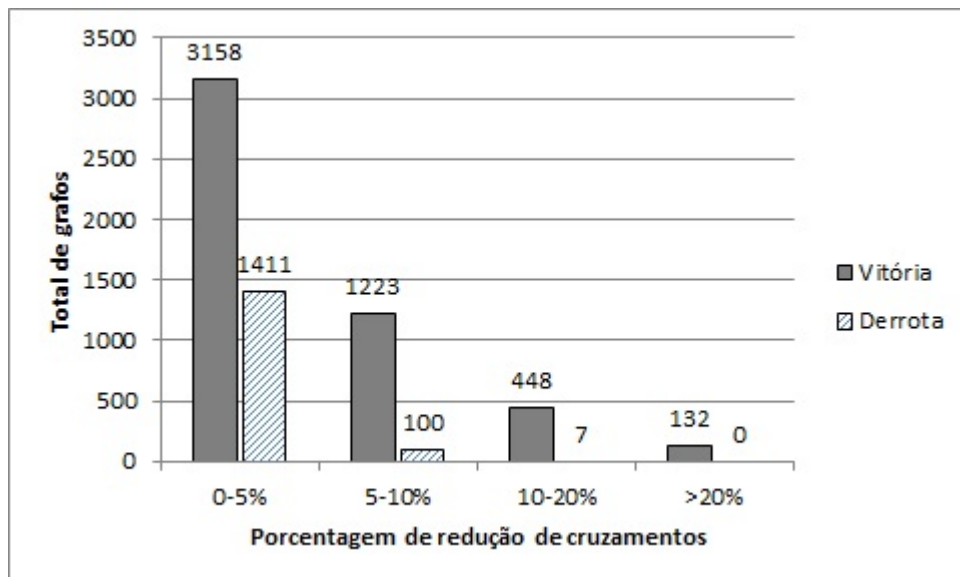


Figura 5.35: Resultados do método *PQR_M2* com relação à melhora dos resultados do método *MEDIAN* (vitórias) e de deterioração (derrotas) separados por porcentagem, relativos ao número de cruzamentos iniciais dos DAGs do Pacote Rome.

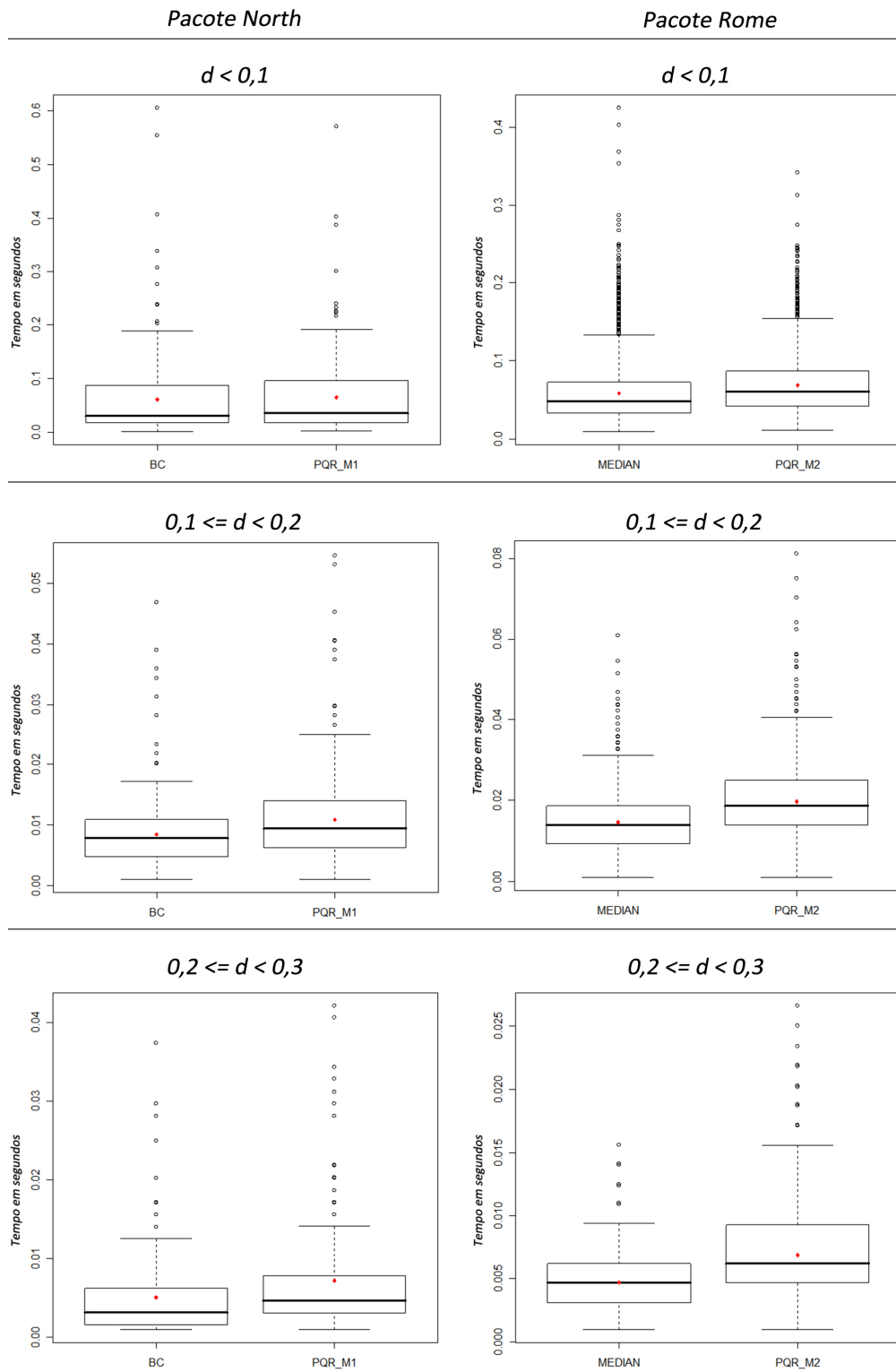


Figura 5.36: Gráficos boxplot com a distribuição dos tempos de execução dos métodos *MEDIAN* e *PQR_M2* para as densidades iguais ou inferiores a 0,3 para os pacotes North e Rome.

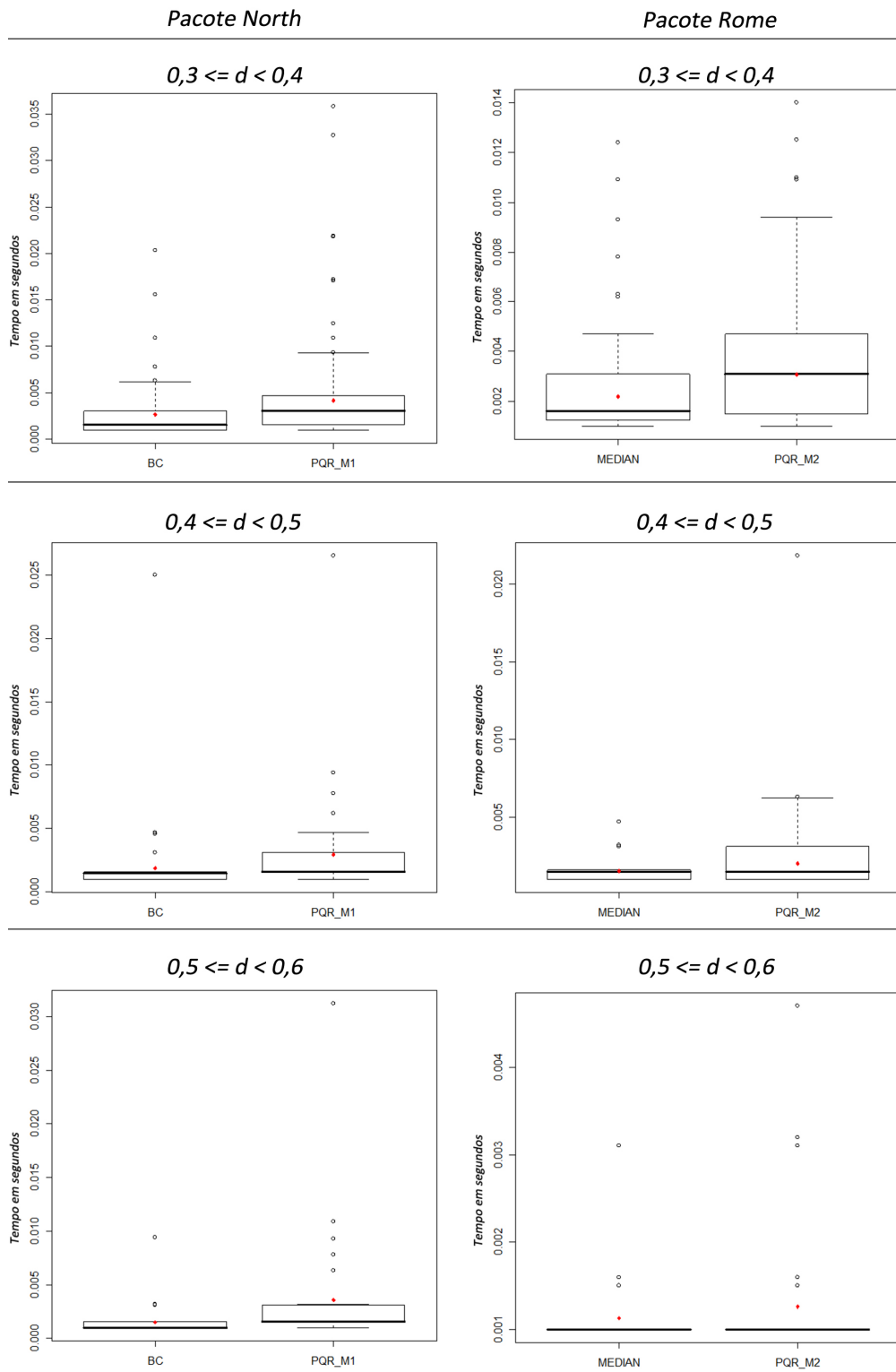


Figura 5.37: Gráficos boxplot com a distribuição dos tempos de execução dos métodos *MEDIAN* e *PQR_M2* para as densidades iguais ou superiores a 0,3 para os pacotes North e Rome.

Observa-se pelas Figuras 5.36 e 5.37 que o método *PQR_M2* segue um comportamento semelhante ao *MEDIAN* com relação à distribuição de dados. Para os casos com densidade igual ou inferior a 0,1, o método leva muito mais tempo para ser executado do que nos casos das demais densidades. As análises de simetria e variação feitas para o método *PQR_M1* têm o mesmo resultado para o método *PQR_M2*. Pode-se também apontar para uma maior variabilidade dos dados na utilização do método *PQR_M2* nos tempos dos grafos com densidade superior a 0,3, quando comparada à variabilidade de *MEDIAN* e *PQR_M1*.

A média dos tempos de execução indica que o método *PQR_M2* não sofreu diferença significativa com relação ao método *PQR_M1*, tendo uma média relativamente superior à do método *MEDIAN*. A Tabela 5.7 mostra a diferença das médias de *MEDIAN* e *PQR_M2* em segundos.

Densidades	Pacote North	Pacote Rome
[0;0,1[0,016	0,010
[0,1;0,2[0,003	0,005
[0,2;0,3[0,002	0,002
[0,3;0,4[0,001	0,001
[0,4;0,5[0,001	0,000
[0,5;0,6[0,001	0,000

Tabela 5.7: Diferença entre as médias de tempo de *MEDIAN* e *PQR_M2*. Os valores positivos indicam o quanto, em segundos, a média de *PQR_M2* está acima da média de *MEDIAN*

A Tabela 5.7 mostra que, no pior dos casos, a diferença entre as médias do tempo de execução de *MEDIAN* e *PQR_M2* é de 0,016 segundos.

A Tabela 5.8 mostra um resumo das comparações dos resultados obtidos nesta seção. Existe uma diferença de 30 pontos percentuais do número de vezes que o método *PQR_M2* supera o método *MEDIAN* com relação à qualidade. Nos casos em que vence, o método *PQR_M2*, em média, supera o método *MEDIAN* em 3 pontos percentuais no número de cruzamentos do DAG original, alcançando 85% de redução de cruzamentos. Nos casos em que perde, o método *PQR_M2* fica em média 1 ponto percentual abaixo dos resultados obtidos com *MEDIAN*, alcançando 80% de redução de cruzamentos. A diferença média de tempo entre os métodos é de 0,006 segundos a mais para o método *PQR_M2*, comparada ao tempo de *MEDIAN*.

	<i>PQR_M2</i>	<i>MEDIAN</i>
Vitórias (percentual)	42%	12%
Média de redução de cruzamentos com relação aos grafos originais nos casos de vitória de <i>PQR_M2</i> (percentual)	85%	82%
Média de redução de cruzamentos com relação aos grafos originais nos casos de derrota de <i>PQR_M2</i> (percentual)	80%	81%
Tempo médio de execução (segundos)	0,038	0,032

Tabela 5.8: Resumo dos resultados obtidos pela execução dos algoritmos *PQR_M2* e *MEDIAN*

5.3 Conclusões

A Tabela 5.9 resume os resultados obtidos pelas comparações entre os métodos desenvolvidos neste trabalho e os métodos *BC* e *MEDIAN*. Os resultados (expressos em pontos percentuais) mostram a diferença entre as médias obtidas das medições relacionadas ao número de vitórias e derrotas de cada método e do total de redução de cruzamentos dos DAGs originais.

	Comparação com método <i>BC</i>		Comparação com método <i>MEDIAN</i>	
	<i>PQR_BC1</i>	<i>PQR_BC2</i>	<i>PQR_M1</i>	<i>PQR_M2</i>
Diferença do número de vitórias	6	29	6	30
Diferença das médias de redução de cruzamentos nos casos de vitória	5	5	12	5
Diferença das médias de redução de cruzamentos nos casos de derrota	5	2	11	1
Diferença das médias do tempo de execução	-0,004	0,001	0,001	0,006

Tabela 5.9: Resumo dos resultados obtidos da execução dos algoritmos *PQR_BC1*, *PQR_BC2*, *PQR_M1* e *PQR_M2*.

A partir da Tabela 5.9 é possível observar que, tanto o método *PQR_BC1* quanto o método *PQR_M1* alcançam resultados parecidos com os dos métodos aos quais foram comparados com relação ao quanto reduzem de cruzamentos nos DAGs. Apesar disso, ambos os métodos levam uma pequena vantagem com relação ao número de vezes que alcançam resultados melhores (6 pontos percentuais de diferença).

Com relação ao tempo de execução, o método *PQR_BC1*, em média, foi o único que se mostrou mais eficiente do que o método ao qual foi comparado (método *BC*), sendo 0,004 segundos mais rápido. O método *MEDIAN* se mostrou, em média, mais rápido do que todos os métodos analisados neste trabalho (0,032 segundos, como visto na Tabela 5.6).

O método *PQR_M1* foi o que alcançou melhores resultados médios de redução de cruzamentos nos casos em que vence do método *MEDIAN*, superando este em 12 pontos percentuais com relação ao número de cruzamentos originais (88% de redução de cruzamentos, como visto na Tabela 5.6). Porém, foi o que obteve piores resultados nos casos em que perdeu (76% de redução de cruzamentos, como visto na Tabela 5.6), tendo uma diferença média do método *MEDIAN* de 11 pontos

percentuais.

O método *PQR_BC2* obteve resultados melhores com 5 pontos percentuais de diferença do método *BC* nos casos em que vence (85% de redução de cruzamentos, como visto na Tabela 5.4). Nos casos em que perde, ele obteve resultados muito aproximados aos do método *BC*, o que caracterizou uma diferença de apenas 2 pontos percentuais entre as médias (81% de redução de cruzamentos, como visto na Tabela 5.4). Além disso, o método *PQR_BC2* apresenta resultados piores em apenas 13% dos grafos, como visto na Tabela 5.4, o que caracteriza uma diferença de 29 pontos percentuais entre o número de vitórias e derrotas. Com relação ao tempo médio de execução, o método *PQR_BC2* fica 0,001 segundo acima do tempo de execução de *BC*.

O método *PQR_M2*, quando comparado ao método *MEDIAN*, obteve uma diferença do número de vitórias de 30 pontos percentuais, reduzindo em média 85% de cruzamentos contra 80% do método *MEDIAN* nos casos em que vence (Tabela 5.8). Nos casos em que perde, o método *PQR_M2* tem uma diferença de 1 ponto percentual dos resultados obtidos pelo método *MEDIAN*. Com relação ao tempo médio de execução, o método *PQR_M2* fica 0,006 segundos acima do tempo de execução de *MEDIAN*.

Capítulo 6

Conclusão

Nesse trabalho foram apresentadas as principais técnicas de redução de cruzamentos em grafos acíclicos direcionados. Partindo do princípio que DAGs são estruturas utilizadas para representação visual de hierarquias, e que existe a necessidade de interação com essas estruturas por parte do usuário que as analisa, procurou-se explorar mais especificamente os métodos que visam reduzir o maior número de cruzamentos possível em um baixo período de tempo, pois existe a preocupação de proporcionar interações que possam responder no tempo máximo de 0,1 segundo.

Outra característica importante para o uso de DAGs como estruturas visuais é a opção, por parte do usuário, de definir quais serão as camadas do DAG e quais vértices devem pertencer a essas camadas. Por essa necessidade, adotou-se o *modelo de Sugiyama*, que faz a separação do passo de redução de cruzamentos com o passo de definição de camadas e coordenadas para os vértices, diferentemente das técnicas de planarização.

Ao buscar na literatura métodos de redução de cruzamentos de arestas com as características acima descritas, foram encontrados os métodos *BC* e *MEDIAN* como sendo aqueles que têm o melhor desempenho na relação entre tempo e número de cruzamentos. Apesar disso, observou-se que ambos os métodos podem ser aprimorados a fim de encontrarem melhores resultados.

Na tentativa de aperfeiçoar o método *BC* e o método *MEDIAN* utilizando-se o *modelo de Sugiyama*, desenvolveu-se quatro novos algoritmos que utilizam em suas estratégias uma estrutura de dados conhecida como Árvore PQR: *PQR_BC1*, *PQR_M1*, *PQR_BC2* e *PQR_M2*. Uma Árvore PQR tem características de agrupamento de conjuntos que se mostrou eficiente para o problema de seriação de matrizes (intimamente ligado ao problema de redução de cruzamentos em DAGs).

Com os resultados obtidos com esse trabalho, concluiu-se que o uso de Árvores PQR pôde aperfeiçoar ambos os métodos *BC* e *MEDIAN* no que diz respeito à qualidade dos DAGs (número de cruzamentos reduzidos) sem deteriorar significativamente o tempo de execução dos métodos.

Também é possível verificar pelos resultados do método *PQR_BC1* e *PQR_M1* que não há dados que comprovem um desempenho melhor ou pior de ambos os métodos quando comparados aos respectivos *BC* e *MEDIAN*. Pode-se apenas afirmar que eles encontram soluções melhores para aproximadamente um terço dos grafos analisados, ao passo que são superados pelos métodos comparados em aproximadamente um terço dos grafos também.

É possível afirmar que ambos os métodos *PQR_BC2* e *PQR_M2* superaram os métodos aos quais foram comparados (*BC* e *MEDIAN* respectivamente) com relação à redução de cruzamentos, melhorando os resultados de 42% dos grafos comparados (5254 instâncias para o *PQR_BC2* e 5294 para o *PQR_M2*). Em aproximadamente 10% dos resultados, os métodos desenvolvidos não são vitoriosos, encontrando porém soluções bem aproximadas dos métodos comparados (diferença de no máximo 2 pontos percentuais). Para os demais casos, os métodos *PQR_BC2* e *PQR_M2* encontram a mesma solução. Com relação ao tempo de execução, a diferença se encontra na ordem de milissegundos, o que indica que o tempo de execução dos métodos desenvolvidos é semelhante ao tempo de execução dos métodos comparados.

É importante destacar que os resultados obtidos nesse trabalho se baseiam em dois pacotes de grafos comumente utilizados na literatura (North DAGs e Rome Graphs) para a realização de testes como os propostos nessa dissertação. Apesar disso, não existe uma predefinição de número de arestas, número de vértices e número de camadas dos grafos desses pacotes, sendo necessário transformar cada instância em uma *proper hierarchy* antes da execução dos experimentos. Isso dificultou o agrupamento desses grafos para uma análise mais precisa com relação aos testes de tempo de execução e qualidade, pois a utilização de algoritmos para a criação de uma *proper hierarchy* fez com que os conjuntos de grafos perdessem algumas características iniciais como número de vértices e arestas. Apesar disso, ressalta-se que o agrupamento por densidades se mostrou uma técnica importante e satisfatória para a análise dos resultados, caracterizando-se como uma contribuição apresentada nessa dissertação para trabalhos futuros relacionados a testes com ambos os pacotes de grafos.

6.1 Trabalhos Futuros

Como uma extensão desse trabalho, podem-se destacar os seguintes pontos:

- Comparar os algoritmos obtidos nesse trabalho com algoritmos genéticos de redução de cruzamentos. Observou-se recentemente que algoritmos genéticos produzem bons resultados de redução de cruzamentos, além de serem executados com tempos satisfatórios.
- Submeter os algoritmos desenvolvidos nesse trabalho a testes com o programa DAGmar (BACHMAIER et al., 2012). Esse programa permite a criação de instâncias de grafos padronizadas com relação ao número de vértices por camada, quantidade de camadas e densidade, fato que pode auxiliar em uma análise estatística mais precisa.
- Utilizar os algoritmos propostos como parte de sistemas de visualização interativa de DAGs, como por exemplo, o CourseViewer (SILVA et al., 2012), que é um programa que usa gráficos interativos para representar históricos e catálogos de cursos.
- Desenvolver algoritmos que envolvam métodos heurísticos de redução de cruzamentos na fase de criação das Árvore PQR, e não apenas na fase de formação de restrições e processamento de permutações, criando assim algoritmos híbridos e não apenas associativos.
- Fazer testes específicos com grafos bipartidos, visando confirmar resultados obtidos em trabalho anterior (MARCHETE, 2010).

Referências Bibliográficas

BACHMAIER, C.; GLEIBNER, A.; HOFMEIER, A. DAGmar: Library for DAGs. Department of Informatics and Mathematics University of Passau, Germany. 2012.

BATTISTA, G. D.; EADES, P.; TAMASSIA, R.; TOLLIS, I. G. Algorithms for drawing graphs: an annotated bibliography. *Computational Geometry*, v. 4, p. 235–283, 1994.

BATTISTA, G. D.; EADES, P.; TAMASSIA, R.; TOLLIS, I. G. *Graph Drawing: Algorithms for the Visualization of Graphs*. New Jersey, US: Prentice Hall, 1999.

BATTISTA, G. D.; GARG, A.; LIOTTA, G.; PARISE, A.; TAMASSIA, R.; TASSINARI, E.; VARGIU, F.; VISMARA, L. Drawing directed acyclic graphs: An experimental study. In: *Proceedings of the Symposium on Graph Drawing*. London, UK: Springer-Verlag, 1997. (GD '96), p. 76–91.

BATTISTA, G. D.; GARG, A.; LIOTTA, G.; TAMASSIA, R.; TASSINARI, E.; VARGIU, F. An experimental comparison of four graph drawing algorithms. *Computational Geometry*, v. 7, n. 5-6, p. 303 – 325, 1997.

CARD, S. K.; MACKINLAY, J. D.; SHNEIDERMAN, B. (Ed.). *Readings in information visualization: using vision to think*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999.

CHEN, C. *Information Visualization: Beyond the Horizon*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

CHIMANI, M.; GUTWENGER, C.; JUNGER, M.; KLEIN, K.; SCHULZ, P. M. M. The open graph drawing framework. In: *15th International Symposium on Graph Drawing 2007*. Sydney, UK: Springer-Verlag, 2007. (GD '07).

CHIMANI, M.; GUTWENGER, C.; MUTZEL, P.; WONG, H.-M. Layer-free upward crossing minimization. *J. Exp. Algorithmics*, ACM, New York, NY, USA, v. 15, p. 2.2:2.1–2.2:2.27, 2010.

CHIMANI, M.; GUTWENGER, C.; MUTZEL, P.; WONG, H.-M. Upward planarization layout. In: *Proceedings of the 17th international conference on Graph Drawing*. Berlin, Heidelberg: Springer-Verlag, 2010. (GD'09), p. 94–106.

CHIMANI, M.; HUNGERLANDER, P.; JUNGER, M.; MUTZEL, P. An SDP approach to multi-level crossing minimization. *J. Exp. Algorithmics*, ACM, New York, NY, USA, v. 17, p. 3.3:3.1–3.3:3.26, 2012.

- DILLY, B. L. G. Implementação de árvore PQR. Monografia (conclusão de curso) - Instituto de Computação, Universidade Estadual de Campinas, Campinas. 2006.
- EADES, P.; KELLY, D. Heuristics for drawing 2-layered networks. *Ars Combin*, v. 21, p. 89–98, 1986.
- EADES, P.; SUGIYAMA, K. How to draw a directed graph. *J. Inf. Process.*, Information Processing Society of Japan, Tokyo, Japan, Japan, v. 13, n. 4, p. 424–437, 1991.
- EADES, P.; WORMALD, N. Edge crossing in drawing of bipartite graphs. *Algorithmica*, v. 11, p. 379–403, 1994.
- EIGLSPERGER, M.; KAUFMANN, M. An approach for mixed upward planarization. In: *Proceedings of the 7th International Workshop on Algorithms and Data Structures*. London, UK, UK: Springer-Verlag, 2001. (WADS '01), p. 352–364.
- GANSNER, E. R.; KOUTSOFIOS, E.; VO, S. C. N. K.-P. A technique for drawing directed graphs. *IEEE Transactions on Software Engineering*, v. 19, n. 3, p. 214–230, 1993.
- GANSNER, E. R.; NORTH, S. C.; VO, K.-P. DAG - a program that draws directed graphs. *Software Practice Experience*, v. 18, n. 11, p. 1047–1062, 1998.
- HONG, S.-H.; NIKOLOV, N. S. Layered drawings of directed graphs in three dimensions. In: *Proceedings of the 2005 Asia-Pacific Symposium on Information Visualisation*. Berlin, Heidelberg: Australian Computer Society, Inc., 2005. (GD'05, v. 4), p. 69–74.
- JUNGER, M.; MUTZEL, P. 2-layer straightline crossing minimization: performance of exact and heuristic algorithms. *Journal of Graph Algorithms and Applications*, v. 1, p. 1–25, 1997.
- KAUFMANN, M.; WAGNER, D. (Ed.). *Drawing graphs: methods and models*. London, UK, UK: Springer-Verlag, 2001.
- KUNTZ, P.; PINAUD, B.; LEHN, R. Minimizing crossings in hierarchical digraphs with a hybridized genetic algorithm. *Journal of Heuristics*, Springer Netherlands, v. 12, p. 23–36, 2006.
- MARCHETE, J. R. F. Utilização de árvores PQR na reorganização de grafos bipartidos. Trabalho de graduação interdisciplinar - Faculdade de Tecnologia, Universidade Estadual de Campinas, Campinas. 2010.
- MARTIN, R.; LAGUNA, M. Heuristics and meta-heuristics for 2-layer straightline crossing minimization. *Discrete Applied Mathematics*, v. 127, p. 665–678, 2003.
- MAZZA, R. *Introduction to Information Visualization*. 1. ed. [S.l.]: Springer-Verlag, 2009.
- MEIDANIS, J.; PORTO, O.; TELLES, G. P. On the consecutive ones property. *Discrete Applied Mathematics*, v. 8, p. 325–354, 1998.
- MELO, M. F. de. Aprimoramento do algoritmo PQR-sort para a reordenação de matrizes binárias. Dissertação (Mestrado) - Faculdade de Tecnologia, Universidade Estadual de Campinas, Campinas. 2012.

- MUTZEL, P.; EADES, P. Graphs in software visualization. In: *Revised Lectures on Software Visualization, International Seminar*. London, UK, UK: Springer-Verlag, 2002. p. 285–294.
- NIKOLOV, N. S.; TARASSOV, A.; BRANKE, J. In search for efficient heuristics for minimum-width graph layering with consideration of dummy nodes. *Journal of Experimental Algorithmics (JEA)*, ACM, New York, NY, USA, v. 10, 2005.
- PURCHASE, H. C. Which aesthetic has the greatest effect on human understanding? In: *Proceedings of the 5th International Symposium on Graph Drawing*. London, UK, UK: Springer-Verlag, 1997. (GD '97), p. 248–261.
- PURCHASE, H. C.; CARRINGTON, D.; ALLDER, J.-A. Empirical evaluation of aesthetics-based graph layout. *Empirical Software Engineering*, Springer Netherlands, v. 7, p. 233–255, 2002.
- SIIRTOLA, H.; MAKINEN, E. Constructing and reconstructing the reorderable matrix. *Information Visualization*, Palgrave Macmillan, v. 4, n. 1, p. 32–48, 2005.
- SILVA, C. G.; INOUE, M. T.; MENDONÇA, P. J. C. Courseviewer - ferramenta para visualização de catálogos de cursos universitários e históricos escolares. In: *Anais do VIII Simpósio Brasileiro de Sistemas de Informação*. São Paulo, SP: [s.n.], 2012.
- SUGIYAMA, K.; TAGAWA, S.; TODA, M. Methods for visual understanding of hierarchical system structures. *IEEE Transactions on Systems, Man and Cybernetics*, v. 11, n. 2, p. 109–125, 1981.
- TANG, H.; CHEN, S. Research on layering algorithm of DAG. In: *Proceedings of the 2008 International Conference on Computer Science and Software Engineering - Volume 02*. Washington, DC, USA: IEEE Computer Society, 2008. (CSSE '08), p. 271–274.
- TELLES, G. P.; MEIDANIS, J. Building PQR trees in almost-linear time. *Electronic Notes in Discrete Mathematics*, v. 19, p. 33–39, 2005.
- WARFIELD, J. N. Crossing theory and hierarchy mapping. *IEEE Transactions on Systems, Man and Cybernetics*, v. 7, n. 7, p. 505–523, 1977.
- WEI-XIANG, S.; JING-WEI, H. Edge crossing minimization algorithm for hierarchical graphs based on genetic algorithms. *Wuhan University Journal of Natural Sciences*, Wuhan University, co-published with Springer, v. 6, p. 555–559, 2001.
- ZHENG, L.; BUCHHEIM, C. A new exact algorithm for the two-sided crossing minimization problem. In: *Proceedings of the 1st international conference on Combinatorial optimization and applications*. [S.l.]: Springer-Verlag, 2007. (COCOA'07), p. 301–310.