CÁLCULO DA SIMILARIDADE ENTRE PLANILHAS ELETRÔNICAS

Gustavo de Oliveira Fernandes

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientador: Geraldo Bonorino Xexéo

Rio de Janeiro
Setembro de 2014

CÁLCULO DA SIMILARIDADE ENTRE PLANILHAS ELETRÔNICAS

Gustavo de Oliveira Fernandes

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

_____
Prof. Geraldo Bonorino Xexéo, D. Sc.

_____
Prof. Jano Moreira de Souza, Ph.D.

_____
Prof. Márcio de Oliveira Barros, D. Sc.

RIO DE JANEIRO, RJ - BRASIL
SETEMBRO DE 2014

Fernandes, Gustavo de Oliveira

Cálculo Da Similaridade Entre Planilhas Eletrônicas / Gustavo de Oliveira Fernandes. – Rio de Janeiro: UFRJ/COPPE, 2014.

XIII, 89 p.: il.; 29,7 cm.

Orientador: Geraldo Bonorino Xexéo

Dissertação (mestrado) – UFRJ/ COPPE/ Programa de Engenharia de Sistemas e Computação, 2014.

Referências Bibliográficas: p. 74-79.

1. Sistemas de informação. 2. Versionamento. 3. Versionamento de planilhas. I. Xexéo, Geraldo Bonorino. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

Dedico esta dissertação à minha mãe, Márcia Regina de Oliveira Fernandes (in memoriam), que me ensinou a não desistir.

Um beijo, minha Mestre.

Agradecimentos

De uma forma geral, agradeço a todos os que sentiram minha ausência ao longo dessa jornada em busca do título de mestre. Não foi fácil. Ainda bem que não foi, pois se tivesse sido, não teria mérito. Obrigado.

Agradeço ao povo brasileiro, que inconscientemente financiou minha jornada acadêmica através do mar infinito de impostos pagos ao Estado. Se não fosse cada uma dessas gotas depositadas, eu não teria à minha disposição uma universidade pública com a qualidade da Universidade Federal do Rio de Janeiro e, muito menos, um instituto como a COPPE – ambos de peso internacional. Ainda nesse contexto, agradeço ao CNPq, por ter apoiado meu estudo e pesquisa por meio de bolsa. Obrigado.

Agradeço ao meu orientador Xexéo, que discutiu, apoiou, divergiu e advertiu nas horas que precisava. Obrigado.

Agradeço diretamente à família, com destaque aos meus pais, Alexandre e Márcia (in memoriam), à minha irmã Natália e aos meus avós, Lourdes, Nílson (in memoriam) e Heloísa (in memoriam). Meu avô Dario (in memoriam), pra não ser injusto, também merece meu agradecimento: apesar de não o ter conhecido, é uma pedra fundamental. Sem a família, não temos nosso primeiro pilar para compreender a sociedade. Minha mãe, mestre, me encorajou a continuar minha jornada mesmo em seus momentos finais. Muito obrigado.

Agradeço à Maiara, minha namorada, companheira e amiga. Sempre ao meu lado, me apoiou nos momentos pessoais mais difíceis – às vezes sem nem saber que sua presença já me apoiava. Muito obrigado.

Agradeço aos meus amigos que, como dizem por aí, são “irmãos separados do berço”. Talvez vocês tenham sido o grupo que mais sentiu minha ausência. Sabem que foi por uma boa causa, não? Prefiro não listar os nomes pois terei de ordená-los, e sei que o simples fato de ter uma ordem de nomes será motivo para um lamento. Musk obrigado.

Agradeço aos sócios-amigos da Kendoo. Apoio, broncas e estímulos me fizeram caminhar sem desistir. Muito obrigado.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

CÁLCULO DA SIMILARIDADE ENTRE PLANILHAS ELETRÔNICAS

Gustavo de Oliveira Fernandes

Setembro/2014

Orientador: Geraldo Bonorino Xexéo

Programa: Engenharia de Sistemas e Computação

A maior parte dos usuários usa planilhas eletrônicas como ferramenta para armazenamento de dados. Mas em muitas ocasiões não são seguidos nem um padrão e nem uma regra de formatação de dados bem definida. Além disso, o compartilhamento desordenado de planilhas cria o grave problema de redundância e de falta de confiabilidade dos dados nelas armazenados. Tais comportamentos podem acabar gerando conclusões equivocadas em estudos, indicando informações mal fundamentadas para decisões de negócio, ocasionando perdas financeiras às empresas ou, simplesmente, atrapalhando a produtividade das pessoas. Este trabalho lista os problemas ocasionados pela má utilização de planilhas de dados e aponta os impactos decorrentes deles, apresenta métricas de comparação das planilhas, exemplifica como essas métricas podem auxiliar no controle de erros das planilhas e, finalmente, propõe uma arquitetura de migração de planilhas para bases de dados relacionais.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

ELECTRONIC DATASHEET SIMILARITY CALCULATION

Gustavo de Oliveira Fernandes

September/2014

Advisor: Geraldo Bonorino Xexéo

Department: Systems and Computing Engineering

Most part of computer end-users use electronic spreadsheets as a data storage tool. However, most commonly, no data pattern nor formatting template are used. Besides disorderly replication of spreadsheet files leads to a serious redundancy and lack of data reliability problem. Such conducts may cause misguided conclusion in academic researches, or improper business decisions based on defective data (which may cause huge negative financials results), or simply affecting people productivity. This research lists the problems caused by spreadsheet software misuse and points out some of their impacts, presents spreadsheet comparison metrics, exemplifies how these metrics may avoid spreadsheets error based data and, finally, proposes a software architecture to transform spreadsheet data into relational databases.

Sumário

Lista de figuras

Lista de tabelas

Capítulo 1 - Introdução

Nesse capítulo serão apresentados o contexto deste trabalho, os problemas que motivaram a pesquisa, os objetivos almejados e a estrutura geral deste documento de forma a facilitar a leitura e entendimento do leitor.

1.1. Contexto

Seja para uso doméstico, acadêmico ou corporativo, usuários dos mais variados níveis e áreas de trabalho utilizam algum software baseado em planilhas eletrônicas para organização de seus dados. Suas funcionalidades principais são a facilidade de organização tabular de registros, o grande número de fórmulas embutidas (*built in formulas*) e a flexibilidade dos *scripts* que podem ser criados pelos usuários (*macros*) e que são aplicáveis aos dados armazenados.

Software editor de planilhas eletrônicas[1] é um programa que crie e gerencie planilhas, que são capazes de organizar e analisar dados em formato tabular. Este tipo de ferramenta mostra-se extremamente popular para gerenciamento de dados e é amplamente utilizada em diversos cenários, sejam eles acadêmicos, corporativos ou pessoais. São consideradas como instrumento indispensável ou primário (quando não o único instrumento) por muitos pesquisadores, cientistas, engenheiros, analistas que precisam gerenciar conjuntos de dados dos mais variados tamanhos e naturezas. Elas ganham ainda maior destaque quando se leva em conta a flexibilidade de formatação e cálculo que têm, permitindo que não-programadores desenvolvam fórmulas que atendam aos problemas do mundo real.

Em maio de 1979 o primeiro software de planilha eletrônica focado no nicho de computadores pessoais, o VisiCalc (Figura 1 - VisiCalc) foi lançado comercialmente. Também chamado simplesmente de “folha de cálculo”, este tipo de software nos anos posteriores foi alçado do posto de mero componente auxiliar de escritório para o status de ferramenta indispensável dos usuários de computadores pessoais, utilizado por dezenas de milhões de usuários em todo mundo, desde os usuários considerados leigos até os mais avançados analistas, engenheiros e cientistas (CAMPBELL-KELLY, 2007).

[1] No decorrer deste trabalho, as planilhas eletrônicas também serão referenciadas simplesmente como planilhas, sem qualquer perda de significado dado o contexto envolvido

Figura 1 - VisiCalc

Na verdade, alguns pesquisadores vinculam o crescimento da indústria dos computadores pessoais ao surgimento e adoção deste tipo de *software*, sendo alguns deles até classificados como *killer-app's*[2] (ZYNDA, 2013). Esta credibilidade só foi possível de ser atingida por causa da flexibilidade que as planilhas de cálculo oferecem, dando liberdade a seus usuários finais de criarem, com facilidade. Planilhas para controle de estoque, balanço financeiro e analisador estatístico, capazes até de substituir pequenos sistemas de informação (CERUZZI, 2007).

Editores de planilha também permitem que usuários comuns executem operações como busca, ordenação, filtros e restrições - funcionalidades que seriam limitadas a sistemas desenvolvidos por programadores especialistas. Mesmo quando comparados com um sistema de informações simples (isto é, um sistema com poucas regras de negócio e baseado em operações primárias de inserção, recuperação, edição e deleção integradas a bancos de dados), as planilhas ganham vantagem, uma vez que seu baixo-custo e rápida implantação batem o alto custo e tempo de desenvolvimento necessários de um sistema customizado e específico.

Segundo Bob Frankston, um dos inventores do VisiCalc, ao utilizar esta ferramenta as pessoas comuns (leigas em computação) tornaram-se programadoras. Segundo (GRAD, 2007) e (CAMPBELL-KELLY, 2007) esta era realmente a meta das empresas: focar em usuários comuns, para uso profissional, para que eles não dependessem mais de programadores para tarefas cotidianas. Esse fato teve o mesmo impacto social, ainda segundo Frankston, que o impacto causado pela popularização dos aparelhos telefônicos domiciliares, na década de 1950 (KELLY ET AL, 2004).

[2] Termo designado a aplicativos de grande importância para utilização em escala de uma determinada plataforma

Desde o lançamento do então inovador VisiCalc, passando pelo marcante Lotus 1-2-3 (Figura 2 - Lotus 1-2-3) e chegando até o líder de mercado Microsoft Excel (Figura 3 - Microsoft Excel), o mercado de software de planilhas continua em pleno crescimento mesmo após 35 anos. Apesar da queda recente das vendas de PC's, a Microsoft[3] tem nas vendas de seu pacote de escritório Microsoft Office (agora também disponível para dispositivos móveis[4]) uma das maiores fontes de receita de toda a empresa - chegando a incríveis $1,5 bilhão por trimestre[5] e crescendo a cada ano. Ao considerarmos que não é raro que nichos de usuários considerem vital a presença do Microsoft Excel no seu trabalho diário (PRYOR ET AL, 2006), podemos deduzir que este programa em específico tem grande importância no mercado mundial de software.

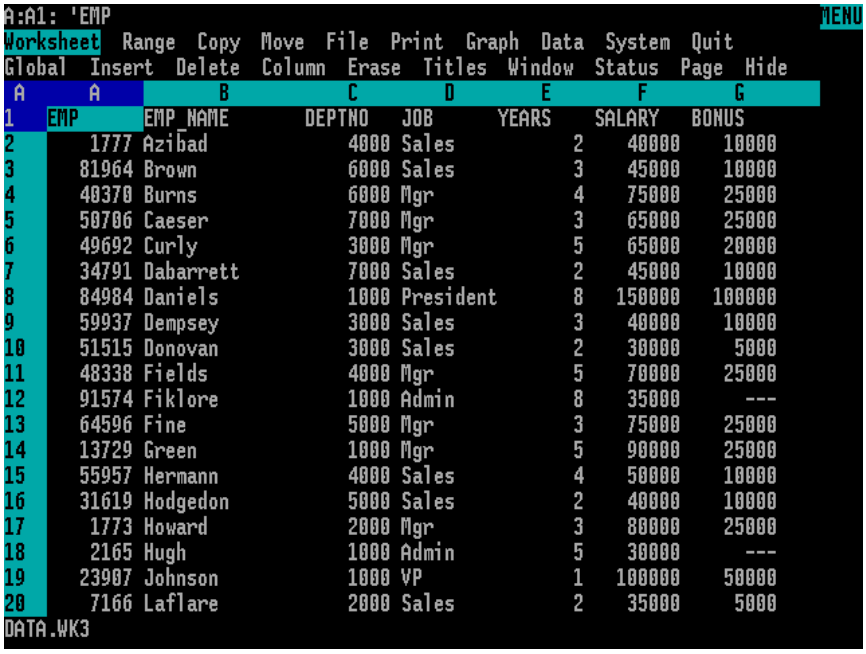


Figura 2 - Lotus 1-2-3

[3] http://www.microsoft.com, disponível em 06/09/14

[4] http://office.microsoft.com/en-001/mobile/, disponível em 06/09/14

[5] http://www.computerworld.com/s/article/9240906/Office_revenue_smells_sweet_even_as_PC_sales_stink_it_up, disponível em 06/09/14

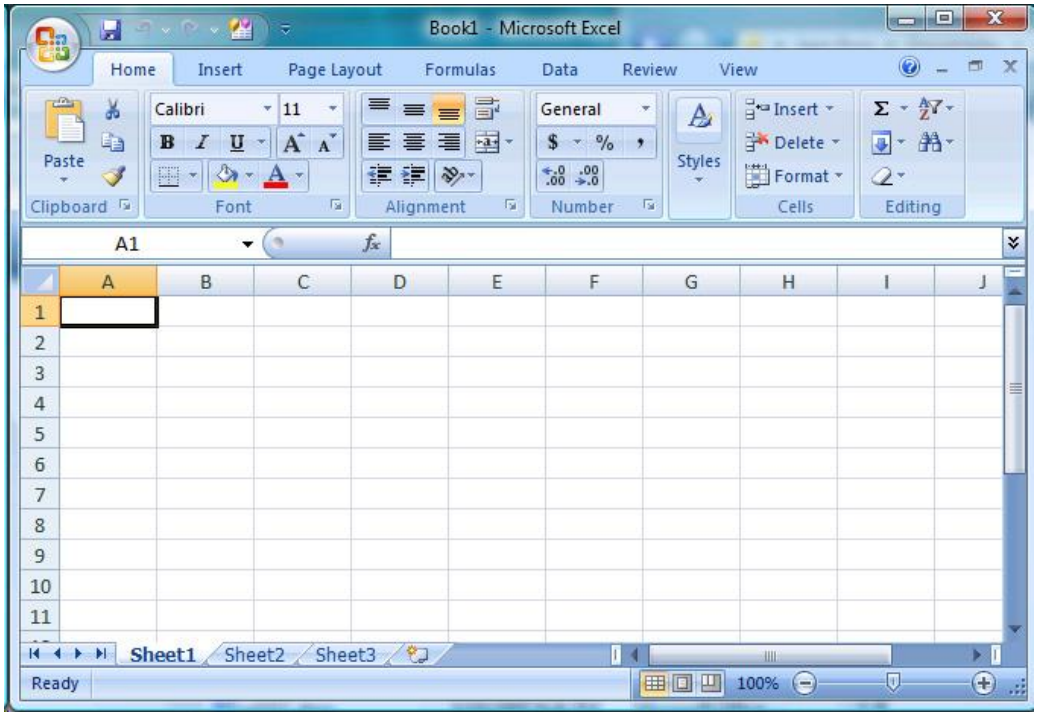


Figura 3 - Microsoft Excel

1.2. Problemas e motivação

Há algumas falhas graves que surgem quando as funcionalidades de uma planilha não são bem exploradas ou quando há falha humana na utilização das planilhas. PANKO (1998) lista alguns estudos que mostram erros que ocorreram em planilhas reais, de empresas dos mais variados portes e segmentos. Segundo o autor, todo estudo que visa checar a ocorrência de erros em planilhas consegue achá-los em uma quantidade que seria considerada inaceitável em qualquer organização. Essa forte afirmação indica que os erros são inerentes à existência de planilha.

Com a proliferação de pontos de acesso à internet e o uso massivo de intranets corporativas, a replicação descontrolada de arquivos acrescentou mais um fator crítico: planilhas podem ser modificadas por diversos usuários em diferentes ambientes com os mais distintos níveis de conhecimento e expertise. Logo, por estar sujeita a uma série de modificações, uma planilha (ou conjunto de planilhas) pode passar de um estado correto (onde não há erros) para um estado falho (onde existe pelo menos um erro) com incrível facilidade.

Levando em consideração a importância histórica e econômica destes programas de computador, seria razoável a busca de uma solução ou um método de contenção para os problemas decorrentes da utilização inadequada de planilhas.

Uma solução possível seria a comparação evolutiva de arquivos de planilhas eletrônica. Isso é, uma forma de monitorar ao longo do tempo, quais planilhas eletrônicas foram modificadas a partir de outras. Para isso, é de suma importância obter um método computacional que compare duas planilhas quaisquer e indique se elas são ou não similares. Se elas são razoavelmente similares, é possível (e até provável) que uma tenha dado origem à outra.

1.3. Objetivos

O objetivo principal dessa pesquisa é levantar o estado da arte em algoritmos de cálculo de similaridade entre duas planilhas quaisquer.

Como objetivos secundários, listam-se a proposta de uma solução capaz de comparar e monitorar a evolução de planilhas (com base em algoritmos de similaridade) e a proposta de uma arquitetura preliminar em que seja viável a migração de planilhas para um modelo relacional - onde seria mais fácil o controle da entrada e da manipulação de dados afim de evitar erros.

1.4. Organização

O trabalho está organizado da seguinte forma:

1. Este capítulo, que apresenta ao leitor o contexto da pesquisa, os problemas que a motivaram e os objetivos almejados;
2. Um capítulo que retrata o cenário atual e a importância do uso das planilhas, bem como do cuidado que se deve ter com elas e de trabalhos relacionados à essa dissertação;
3. Um capítulo para apresentar uma arquitetura preliminar de migração de dados e meta-dados para bases relacionais;
4. Um capítulo que apresenta as métricas utilizadas para estabelecer a similaridade entre planilhas, assim como uma comparação entre as métricas apresentadas;
5. Um capítulo para resumir a conclusão obtida com o desenvolvimento desta pesquisa e para apresentar possíveis trabalhos futuros;
6. As referências bibliográficas consultadas.

Capítulo 2 - Cenário atual

É necessário organizar o conhecimento já existente na literatura para se elaborar um estudo capaz de atingir os objetivos descritos no Capítulo 1.1. Para isso, são descritos neste capítulo os assuntos pertinentes a esse trabalho, bem como alguns trabalhos correlatos, resultado de pesquisa de autores com temas semelhantes.

Inicialmente será apresentada ao leitor a importância do uso de planilhas por usuários de computadores pessoais, com destaque para o ambiente corporativo. Em seguida serão apresentados os riscos de falhas de utilização no uso de planilhas. E, finalmente, são listados alguns trabalhos correlatos ao tema tratado ao longo desta dissertação.

2.1. A importância do uso de planilhas

Como citado no Capítulo 1.1, as planilhas de dados são utilizadas em escala há tanto tempo quanto os computadores pessoais. Seja para apenas compilar alguns poucos dados, para organizar e filtrar conjuntos maiores, para operar como pequenos bancos de dados ou seja para agregar informações que podem ser cruciais para tomadas de decisão, as planilhas eletrônicas são de valor indiscutível no dia-a-dia da maioria dos usuários de computadores pessoais.

BAKER (2006) fez um levantamento do uso das planilhas eletrônicas por parte de 846 ex-alunos de cursos de MBA (dos EUA e da Europa) como forma de corroborar a importância dessas ferramentas nas organizações. Apesar de segmentada pela formação acadêmica, a população entrevistada pode ser caracterizada como suficientemente heterogênea nos seguintes quesitos: gênero, idade, perfil e tamanho da empresa em que trabalham e o cargo desempenhado. Ou seja, pode-se dizer que essa amostra populacional representa bem a variedade dos usuários de planilhas. Seguem algumas informações interessantes resultantes do questionário aplicado e que foram pertinentes a esse trabalho.

Perguntados sobre a importância das planilhas no seu dia-a-dia de trabalho:

- Apenas 1% citou que planilhas “não são importantes”;
- 79% responderam que elas são “muito importantes” ou “críticas” para suas tarefas.

Sobre o tipo de uso feito pelas planilhas, o resultado é demonstrado pelo gráfico exibido na Figura 4. Pode-se concluir que o uso de planilhas é bem variado, desde operações simples como controle de nomes e de endereços até análises e projeções financeiras.

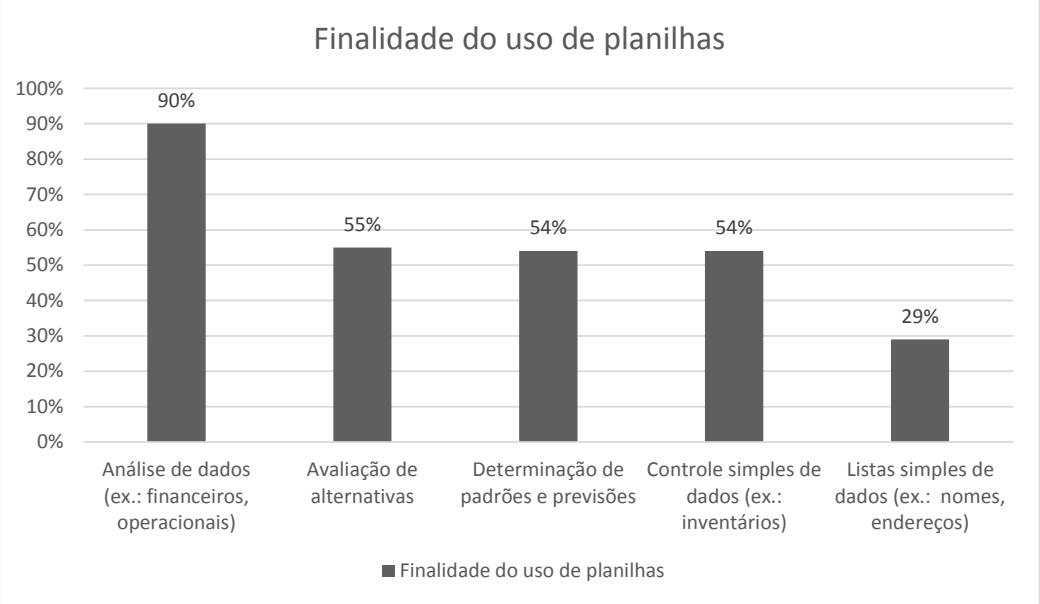


Figura 4 - Finalidade do uso de planilhas

O gráfico exibido na Figura 5 ilustra o tamanho médio das planilhas utilizadas. A partir dele é possível afirmar que a boa parte (61%) das planilhas utilizadas têm até 1.000 células e a grande maioria do conjunto total (89%) compreende planilhas com até 10.000 células. Esta é uma quantidade um tanto quanto expressiva, que direciona o atual trabalho a focar em planilhas com esse tamanho.

Prosseguindo a análise, uma informação bastante interessante é vista na Figura 6, que ilustra o uso compartilhado das planilhas. Segundo o autor do questionário, muitos usuários podem “contar uma história de vida” de suas planilhas, onde elas foram criadas para uso próprio (apenas 1 usuário) e, ao longo do tempo, mais pessoas visualizaram e/ou editaram os arquivos. O índice percentual de planilhas compartilhadas por grupos de usuários chega a 78%, e outros 10% são tão utilizadas que tornam-se ativos permanentes nas instituições.

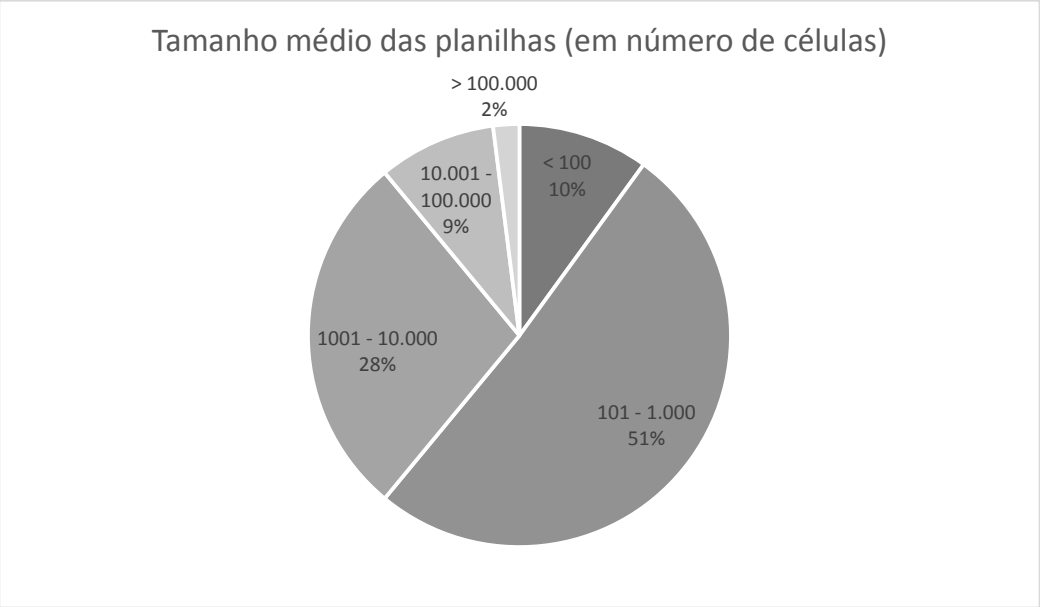


Figura 5 - Tamanho médio das planilhas

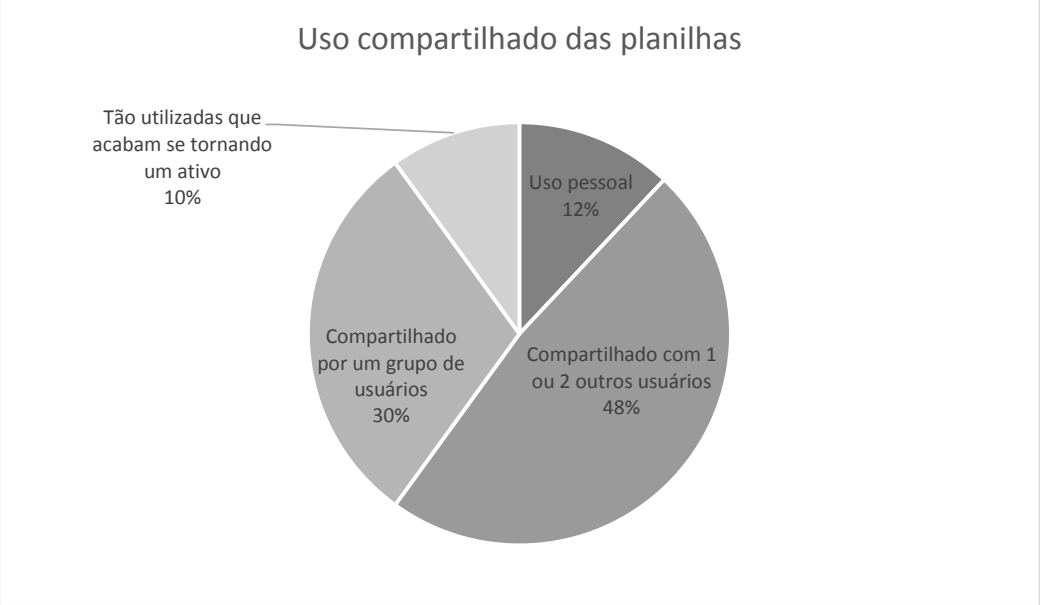


Figura 6 - Uso compartilhado das planilhas

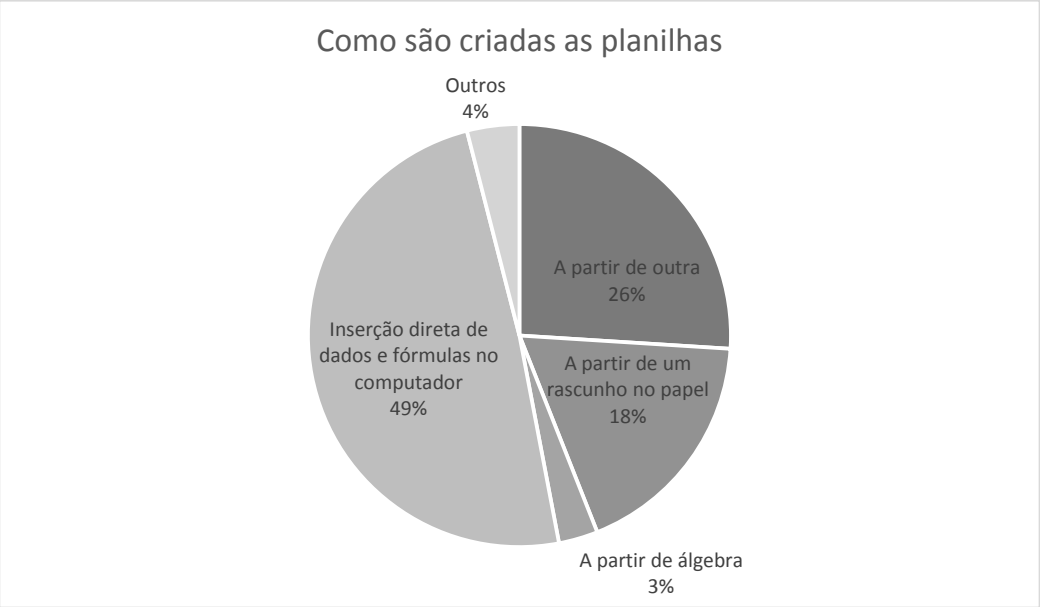


Figura 7 - Como são criadas as planilhas

A Figura 7 mostra como as planilhas surgem nas empresas. Quase metade (49%) passa por uma criação quase caótica, onde o projeto e elaboração da planilha são prescindidos e os dados são construídos conforme a necessidade – talvez isso seja fruto do ambiente dinâmico e estressante que rege as empresas. Uma parcela importante das planilhas (26%) nasce a partir de outras. É o chamado “copia-e-cola”, onde um usuário que já faz uso de uma determinada planilha apenas a duplica para evitar o retrabalho de iniciar uma nova a partir do marco zero.

Sobre a frequência de uso das planilhas, 56% das planilhas são usadas pelo menos uma vez por semana – sendo 18% diariamente.

Os percentuais mostrados nos gráficos das figuras Figura 6 e Figura 7 indicam que um número expressivo das planilhas acaba, na verdade, não tendo um chamado “ciclo de vida”. Um ciclo de vida parte da premissa que algo é criado, sofre alterações, dá origem a outros elementos e acaba morrendo (ou deixa de ser usado, no caso das planilhas). No contexto das planilhas talvez seja mais apropriado dizer que muitas têm um “ciclo eterno”, onde elas são criadas por seus usuários, evoluem (sofrem adaptações, modificações), são compartilhadas com outros (deram origem a outros elementos) e acabam se tornando eternas nas corporações, onde novos contratados fazem uso de uma planilha que começou a ser construída previamente – e pode ter sofrido diversas alterações por funcionários ou ex-funcionários.

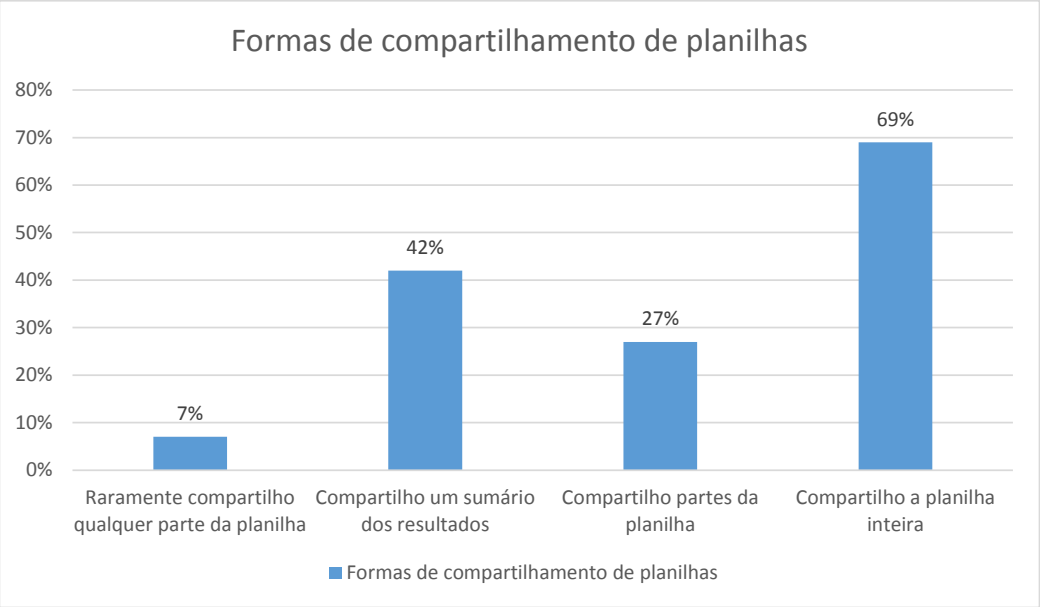


Figura 8 - Compartilhamento das planilhas

Finalmente, a Figura 8 ilustra como o compartilhamento das planilhas é feito por parte dos usuários. É claro que a maior parte dos usuários compartilham ao menos um pedaço da planilha, mas é vital observar que quase 70% deles compartilham o arquivo inteiro contendo a planilha. Isso significa que é muito comum que um modelo inteiro contendo estrutura, fórmula e dados seja utilizado e replicado de forma livre.

Concluindo essa seção, podemos já fazer algumas considerações, guiadas pelas estatísticas apresentadas. Foi apresentado que praticamente 4 em 5 usuários (79% do total) consideram as planilhas de grande importância para seu dia-a-dia e as usam para diversas operações (críticas e não críticas para o negócio). São usadas em grande escala planilhas de pequeno ou médio porte (61% têm até 1000 células) e elas são compartilhadas por diversos usuários, mas sem qualquer controle.

Somando ainda a informação do “ciclo eterno das planilhas” com a frequência do uso delas, podemos construir uma visão onde toda semana são criadas planilhas de forma caótica ou baseadas em outras que foram criadas de forma caótica e têm seu uso compartilhado entre vários usuários.

Na seção seguinte serão mostradas algumas consequências desse comportamento.

2.2. Falhas causadas por planilhas

A seção anterior mostrou o quanto as planilhas são úteis para a maior parte dos usuários, com destaque para o cenário das empresas. No entanto, é importante considerar

que seu uso, como todas as demais ferramentas, deve ser feito com atenção. Caso contrário, falhas catastróficas podem surgir. Considerando as informações apresentadas anteriormente, não é difícil imaginar que o cenário corporativo (onde planilhas são criadas e usadas frequentemente, compartilhadas e reutilizadas sem precaução e passadas de geração a geração de funcionários que entram e saem de empresas) dê origem a diversas falhas de utilização.

Na verdade, tais falhas são tão comuns e tão antigas quanto a própria utilização das planilhas. A ocorrência de erros em planilhas ou de erros decorrentes da má utilização de planilhas é um problema tão crítico nas empresas e ambientes acadêmicos, que iniciativas independentes foram tomadas buscando reduzir a ocorrência destes problemas ou visando criar técnicas para, ao menos, reduzir os impactos causados. Destacam-se neste universo o EUSPRIG[6] e o DataUp[7]

O EUSPRIG (European Spreadsheet Risks Interest Group) oferece a diretores, gerentes e profissionais em geral uma série de trabalhos relacionados à gerência de risco associado ao uso de planilhas. Eles se colocam como a maior fonte mundial de informações acerca de métodos e processos para testar, corrigir, documentar, arquivar, comparar e controlar as incontáveis planilhas utilizadas para o meio corporativo. Seu maior propósito é diminuir as consequências desastrosas decorrentes de informações imprecisas armazenadas nas planilhas. Sua principal contribuição é organizar uma conferência anual que estimula o debate de novos problemas, soluções e ferramentas entre pesquisadores, vendedores, consultores, reguladores, auditores e outros profissionais que trabalham direta ou indiretamente com a utilização de planilhas. Citam ainda que as consequências incluem multas financeiras, oscilações indesejadas de índices de ações e prisões. Pelo fato das informações financeiras de grandes corporações estarem armazenadas em sistemas de informação, os gestores precisam ter garantias que as informações nestes sistemas são confiáveis. Este assunto tomou maior importância após os vazamentos de dados em 2001[8], o congresso americano aprovou a lei conhecida como Sarbanes-Oxley (também conhecida como SOX), onde os executivos de empresas com ações na bolsa de Nova York são responsabilizados criminalmente por desvios nas demonstrações financeiras.

[6] http://www.eusprig.org/, disponível em 06/09/14

[7] http://dataup.cdlib.org/, disponível em 06/09/14

[8] http://www.governancadeti.com/2010/07/o-que-e-governanca-de-ti-e-para-que-existe/, disponível em 06/09/14

Grupos de pesquisadores dedicam-se há cerca de 30 anos (quase o mesmo tempo de vida que os softwares editores de planilhas) à busca de soluções de contorno para os problemas ocorridos na utilização de planilhas. Foram listados em (PANKO, 1998) e (PANKO, 2008) alguns estudos que mostram erros que ocorreram em planilhas reais, de empresas dos mais variados portes e segmentos e com variadas consequências. Neste trabalho, são citados casos como o relatado em (DAVIES & IKIN, 1987), onde 10 escritórios diferentes tiveram suas planilhas analisadas, e em quase 74% delas ocorreu pelo menos um tipo de erro.

Para ter-se noção do tamanho de impacto causado, um dos erros envolvia a transferência de $7 milhões. Em (PANKO, 2008), foram auditadas quase 4.000 células de uma grande planilha orçamentária, onde foram encontradas brechas que teriam o impacto de mais de $1 bilhão se não tivessem sido descobertas a tempo. Há relatos[9] como o da empresa Fidelity´s Magellan, em que um auditor esqueceu o sinal de subtração para indicar uma quantia negativa em um campo que representava uma despesa de $1,3 bilhão. A ausência do sinal transformou a despesa em receita, e os dividendos foram calculados com uma diferença total de $2,3 bilhões. Em outro caso, a empresa canadense TransAlta efetuou uma negociação de compra de energia elétrica do governo dos EUA por $24 milhões acima da quantia correta – falha de cálculo originada por um erro de cópia-e-cola em uma planilha. A Universidade de Toledo certa vez superestimou o valor de uma receita em $2,4 milhões e quase criou um rombo financeiro na instituição. O erro, causado por uma falha de digitação numa fórmula de planilha, foi descoberto a tempo e impediu um prejuízo maior.

Enfim, há diversos relatos na literatura em que falhas de planilhas são a causa direta de prejuízos enormes. Se não financeiros, elas podem acabar gerando conclusões equivocadas em estudos, indicando informações mal fundamentadas para decisões de negócio, manchando a imagem de instituições ou, simplesmente, atrapalhando a produtividade das pessoas (GANSEL, 2008).

Os tipos de erros que podem ocorrer em planilhas têm uma certa variedade em relação à sua natureza e são globalmente agrupados em dois segmentos: os erros quantitativos e os erros qualitativos (TEO & TAN, 1999). O objetivo de se distinguir os tipos de erros encontrados é de mapear métodos distintos de correção e/ou de prevenção de erros. Em outras palavras, para cada tipo diferente de erro, uma abordagem diferente deve ser levada em consideração para preveni-lo ou remediá-lo.

[9] http://www.cio.com/article/2438188/enterprise-softwareight-of-the-worst-spreadsheet-b/enterprise-software/eight-of-the-worst-spreadsheet-blunders.html, disponível em 06/09/14

Os quantitativos são, na maior parte das vezes, mais simples de se identificar e de se corrigir. São representados por cálculos incorretos e o exemplo mais comum é representado por uma soma de células, onde uma célula é esquecida fora da série de células a serem consideradas e a soma efetuada é diferente do valor real.

Os erros qualitativos, também chamados de erros latentes, são criados quando há falha na elaboração da planilha. Isso é, quando a planilha é projetada para armazenar determinados dados e cálculos que não representem bem o domínio ou que tenha sua estrutura inadequada para eventuais mudanças do cenário envolvido. Por exemplo, uma planilha que tem fórmulas baseadas em datas e que está limitada a um conjunto muito pequeno de datas perto do intervalo de datas que é utilizado na prática está fadada ao retrabalho e manutenção. Outra situação bastante comum é quando a planilha não está totalmente preparada para os testes condicionais: uma célula pode ter um valor numérico qualquer, mas só tem seus valores tratados quando são menores do que zero e maiores do que zero - e quando o valor da célula é zero há falha de cálculo – pois o caso não foi planejado adequadamente. Não é raro que erros qualitativos gerem indiretamente erros quantitativos (PANKO & HALVERSON, 1996).

Alguns autores vão além e expandem os erros quantitativos em outros subtipos, com destaque para os que são listados a seguir:

- Erros mecânicos - falhas simples, como erro de digitação, ou vinculação a uma célula errada.
- Erros lógicos - falhas na elaboração de fórmulas utilizadas para computar os valores. São mais comuns do que os erros mecânicos, porém mais difíceis de se detectar e de se corrigir.
- Erros de omissão - ocorrem quando um valor, fórmula ou mapeamento do domínio é ignorado e não é inserido na planilha. São considerados mais perigosos e podem ser extremamente difíceis de se detectar, uma vez que não se sabe que está faltando a informação representada pelos dados ausentes.

Enquanto muitos podem pensar somente em ocorrências triviais dos tipos de erros listados acima, como falhas de digitação e trocas de sinais de positivo por negativo, ou vice-versa, PANKO (1998) apresentou uma taxonomia de erros, revisada por PANKO & AURIGEMMA (2010), que mostra que eles podem ser bem mais complexos.

Para a maior parte dos problemas classificados pela taxonomia de PANKO, muitas ferramentas e processos de auditoria foram criados. No entanto, esses erros dizem respeito à manipulação de apenas uma instância de planilha. Quando são manipulados vários

arquivos que derivam de uma mesma planilha, criando várias instâncias de uma mesma planilha original, novos problemas devem ser enfrentados.

Num mundo perfeito, um arquivo de planilhas eletrônicas deveria ter um único curador: um responsável por controlar quem acessa o arquivo, quem pode consultar seus dados, quem pode filtrar um subconjunto de atributos, quem pode editar seus dados e estrutura e assim por diante. Obviamente o mundo real é bem diferente do descrito anteriormente, e algumas situações desastrosas vem à tona. Algumas são listadas e explicadas abaixo:

Entrada desregrada de dados

A primeira é relacionada à grande flexibilidade proporcionada pelas planilhas. Desde o momento que uma planilha é criada, alguns cuidados devem ser tomados para que sua utilização seja bem feita. Restrições devem ser aplicadas às células, colunas e/ou linhas que a compõem de forma a evitar que dados inconsistentes sejam inseridos por usuários desavisados, mal orientados ou mal-intencionados. Infelizmente tais regras nem sempre são aplicadas, o que acaba implicando em planilhas instáveis: planilhas com dados ausentes, com dados inválidos (um valor de texto em um campo numérico ou uma data em um campo de hora, por exemplo) ou, pior ainda, planilhas com dados válidos, porém semanticamente errados (por exemplo uma nota 11,0 de um aluno, onde o domínio do campo seria definido entre 0 e 10). Esta última situação seria facilmente evitada através de uma restrição adequada à célula, evitando que o valor semanticamente incorreto fosse inserido.

Ausência de controle de acesso ou controle de acesso mal aplicado

Outra falha ocorre em ambientes multiusuário, geralmente em redes sem controle total de dados, onde um conjunto de usuários não deveria ter acesso a um arquivo de planilha ou deveria ter o acesso restrito a ele. Um exemplo trivial é o típico escritório corporativo onde usuários compartilham estações de trabalho ou acessam pastas compartilhadas na rede interna. Basta que um usuário não-autorizado acesse uma planilha restrita (seja porque a pasta compartilhada não é protegida, seja porque uma estação de trabalho não foi bloqueada adequadamente pelo usuário anterior) para que os dados sejam comprometidos. Vazamentos de dados têm sido uma notícia bastante comum (LAYLAND, 2007) e alguns vazamentos podem comprometer empresas inteiras - e dezenas ou centenas de milhões de dólares. No caso de usuários que devem ter o acesso restrito (quando só podem ver uma determinada coluna, uma determinada linha, mas não podem ver outra, por exemplo), a melhor solução é criar réplicas da planilha original, cada uma com o conjunto de dados que cada usuário tem permissão de acessar e editar, semelhante ao conceito de visões (*views*) em um SGBD. Porém, esta medida pode acabar criando diversos transtornos, alguns explicados a seguir.

Múltiplas versões de um mesmo arquivo

Quando múltiplos usuários compartilham cópias distintas de um único arquivo, uma terceira falha ocorre: a da não-garantia da concorrência. A partir da primeira alteração feita em um dos arquivos, todos os outros tornam-se obsoletos, apesar de poderem ser ainda válidos de acordo com a estrutura da planilha. Quando a segunda das cópias sofre uma modificação, diferente da modificação aplicada à primeira cópia, três versões de arquivo são então formadas: a original, a que sofreu a primeira alteração e a que sofreu a segunda alteração. Se forem alterações limitadas aos dados inseridos, pode ser trivial reuni-los novamente em um arquivo centralizado no momento apropriado. Mas se as alterações não forem bem coordenadas (novamente o papel do curador da planilha se faz necessário) ou se as alterações forem estruturais (uma coluna é modificada, adicionada ou removida, por exemplo), pode ser bem difícil executar a reunião dos arquivos dispersos. É fácil notar que bastam alguns usuários utilizando cópias de uma mesma planilha para estabelecer-se um ambiente quase caótico, onde na maioria das vezes apenas um sistemas de controle de versões concorrentes poderia realizar a operação conhecida como *merge* para reagrupar todas as versões em apenas uma fonte.

Desorganização de arquivos

A última falha listada diz respeito à desorganização dos arquivos que contém planilhas. Na verdade, ela não diz respeito a uma planilha ou um conjunto de instâncias de uma mesma planilha, mas sim a um conjunto de planilhas distintas. Esse conjunto pode estar em um localizado em um sistema de arquivos compartilhado (disco compartilhado numa rede, por exemplo). Mais genericamente, esse problema pode afetar qualquer tipo de arquivo e pode ocorrer quando um usuário ou grupo de usuários guarda(m) diversos arquivos em um diretório ou árvore de diretórios sem se preocupar(em) com a organização deles. A consequência desta desorganização é a possibilidade de ocorrência de cópias desnecessárias de um mesmo arquivo, gerando redundância de dados e podendo incorrer em inconsistências (quando, por exemplo, um usuário faz uso das cópias dos arquivos sem perceber que se trata de arquivos distintos).

2.3. Ciclo de vida de uma planilha

A partir das observações feitas acerca do uso de planilhas em ambientes corporativos, Baker (2006) apresenta um ciclo de vida de planilha, que foi adaptado e reproduzido na Figura 9. Segundo o autor, cada etapa do ciclo pode anteceder outra desde que indicado pelas precedências – exibidas sob a forma de setas na ilustração. A etapa 1, por exemplo, antecede a 2. Mas é também antecedida da etapa 2 e da etapa 7 – exatamente por ser um ciclo, o conceito permite caminhos cíclicos entre as etapas.

Neste ciclo de vida, são descritas 7 etapas principais, enumeradas e descritas a seguir.

1. **Projetar**: listada como a fase mais crítica para se adequar as melhores práticas de uso de uma planilha, pois pode influenciar todas as outras etapas. Alguns autores (TEO AND TAN, 1997; TEO AND LEE-PARTRIDGE, 2001) demonstraram que muitos erros originados nessa fase difíceis de se detectar.
2. **Testar**: a etapa de teste é reconhecidamente importante, pois é onde são validados os conceitos pensados na etapa de projeto. No entanto, muitas empresas não executam essa etapa, e sequer têm uma política clara para isso (CALE, 1994).
3. **Documentar**: Menos executada que a etapa de testes, a documentação é raramente utilizada pelas organizações. Cale (1994) afirma que 90% das empresas listadas por ele confirmaram que a falta de documentação é de fato um problema sério, mas mesmo assim a maioria das empresas abria mão dessa etapa.
4. **Utilizar**: O uso efetivo de uma planilha. Visualização dos dados, entrada, modificação, cálculo dos dados, enfim, exatamente a etapa de utilização das planilhas.
5. **Compartilhar**: O compartilhamento das planilhas tornou-se quase uma regra. Seja por e-mail, cópia através da rede interna, duplicação via mídia externa, enfim, praticamente todos compartilham uma planilha no dia-a-dia das empresas. E esse comportamento não tende a mudar. Pelo contrário, com a estabilidade e qualidade das redes (tanto intranets quando a própria internet), é provável que o compartilhamento de arquivos só aumente.
6. **Modificar**: Baker (2006) cita que a modificação de planilhas é frequente e que o ciclo de vida é curto pois as regras de negócio mudam e requerem a criação de uma nova planilha. Mais de 85% das planilhas estudadas sofreram alterações e, em média, cada uma foi alterada sete vezes.
7. **Arquivar**: Último passo do ciclo listado, a etapa de arquivamento é considerado importante para manter o conhecimento dentro da organização e para prover uma base de planilhas disponíveis para uso. Pode ser tão simples como a separação de um conjunto de planilhas numa área de backup, mas muitas empresas não formalizam esse procedimento.

As últimas três etapas (compartilhamento, modificação e arquivamento) dizem respeito exatamente ao problema que a arquitetura proposta nesta pesquisa busca resolver.

Mais especificamente, o item 6 diz respeito ao problema de melhor controle de alterações sobre o qual esse capítulo trata.

O arquivamento (etapa 7) é alcançado naturalmente ao somar os conceitos de versionamento às medidas de similaridade que serão apresentadas no decorrer das próximas seções.

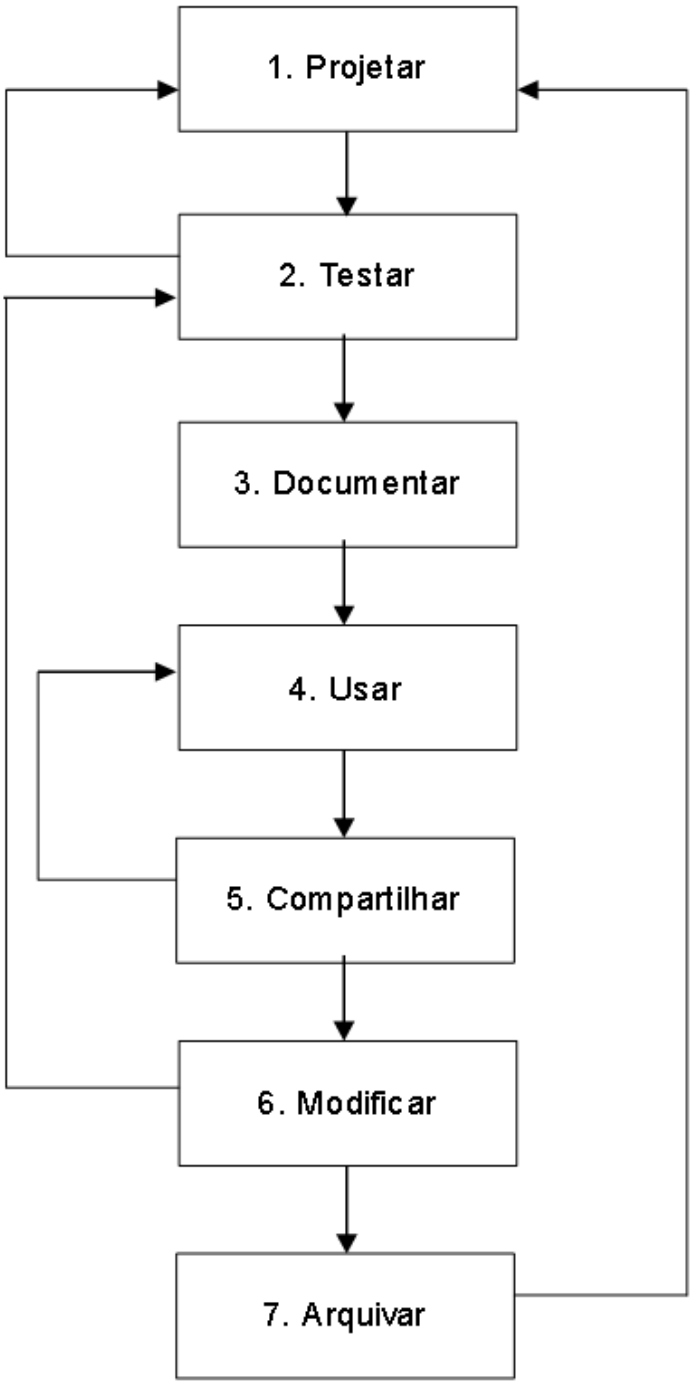


Figura 9 - Ciclo de vida de planilhas

2.4. Versionamento e evolução de planilhas

Sistemas de versionamento de arquivos (SVA) são conjuntos de programas que mantêm armazenadas versões anteriores de um arquivo, permitindo que os usuários visualizem, comparem e recuperem um arquivo a partir de um estado anterior (SOULES, 2003).

Por que então simplesmente não usar um mecanismo de versionamento de arquivos já existente para controle das planilhas? Seria possível, uma vez que os arquivos de planilhas são, na verdade, arquivos que seguem o padrão do seu respectivo sistema de gerenciador de planilhas.

No entanto, os principais objetivos de se utilizar um SVA são executar backups de forma eficiente e facilitar operações de recuperação de desastre (MUNISWAMY-REDDY, 2004) e partem dos princípios que os arquivos versionados são criados sequencialmente e são mantidos organizadamente no sistema. Essas premissas nem sempre são verdadeiras quando tratamos do ambiente com múltiplos usuários.

Além disso, o histórico de alterações guardadas por um tradicional SVA é voltado para arquivos de texto. Isso é, ao comparar-se o estado atual de um arquivo com seu estado anterior, serão vistas as diferenças textuais em função das linhas (que são os elementos formadores de um arquivo de texto). Um exemplo pode ser visto na Figura 10 onde o arquivo "DiffContext.cpp" é comparado com uma de suas versões anteriores através do programa WinMerge[10]. As linhas alteradas entre os dois arquivos são destacadas para facilitar a visualização pelo usuário.

Seria razoável então utilizar, no contexto das planilhas, um SVA com mecanismo comparador que exibisse as alterações em função dos elementos formadores das planilhas: células, linhas e colunas.

[10] http://winmerge.org/, disponível em 06/09/2014

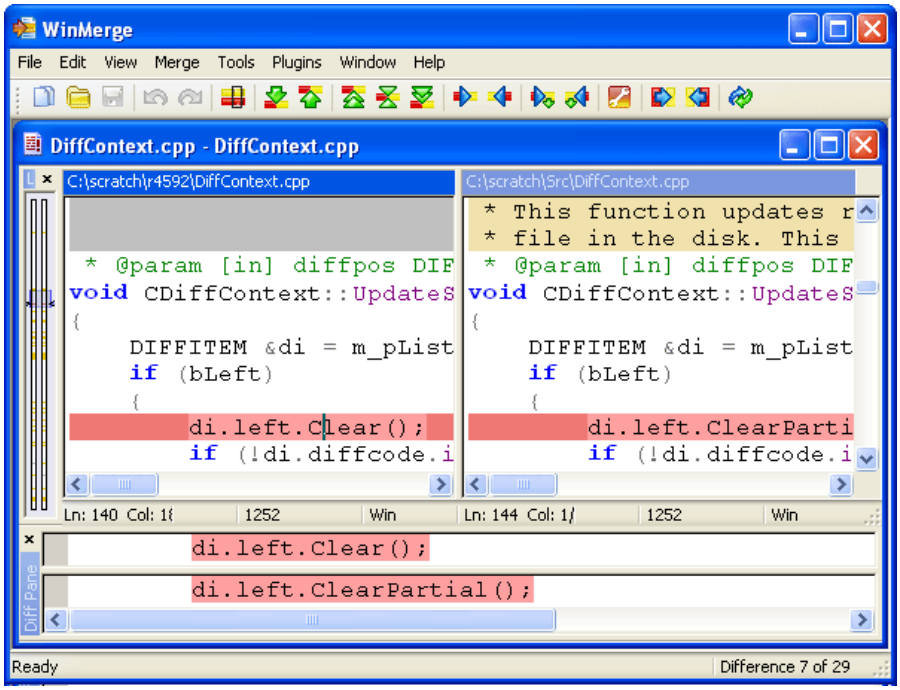


Figura 10 - Comparação de arquivo textual

O Microsoft Excel já conta com um comparador destes. Um exemplo[11] de sua utilização pode ser visto na Figura 11.

No entanto, essa funcionalidade opera apenas com dois arquivos por vez e busca alinhamento máximo entre os dois arquivos. Em outras palavras, é uma boa escolha quando se tem certeza que os dois arquivos a serem comparados têm, de fato, a mesma origem.

Mas o que fazer quando não se sabe de onde dois arquivos vieram ou, pior ainda, quando se está analisando 3 ou mais planilhas? E como mapear a evolução delas ao longo do tempo?

[11] Fonte: http://i.stack.imgur.com/Nwnb1.gif, disponível em 06/09/2014

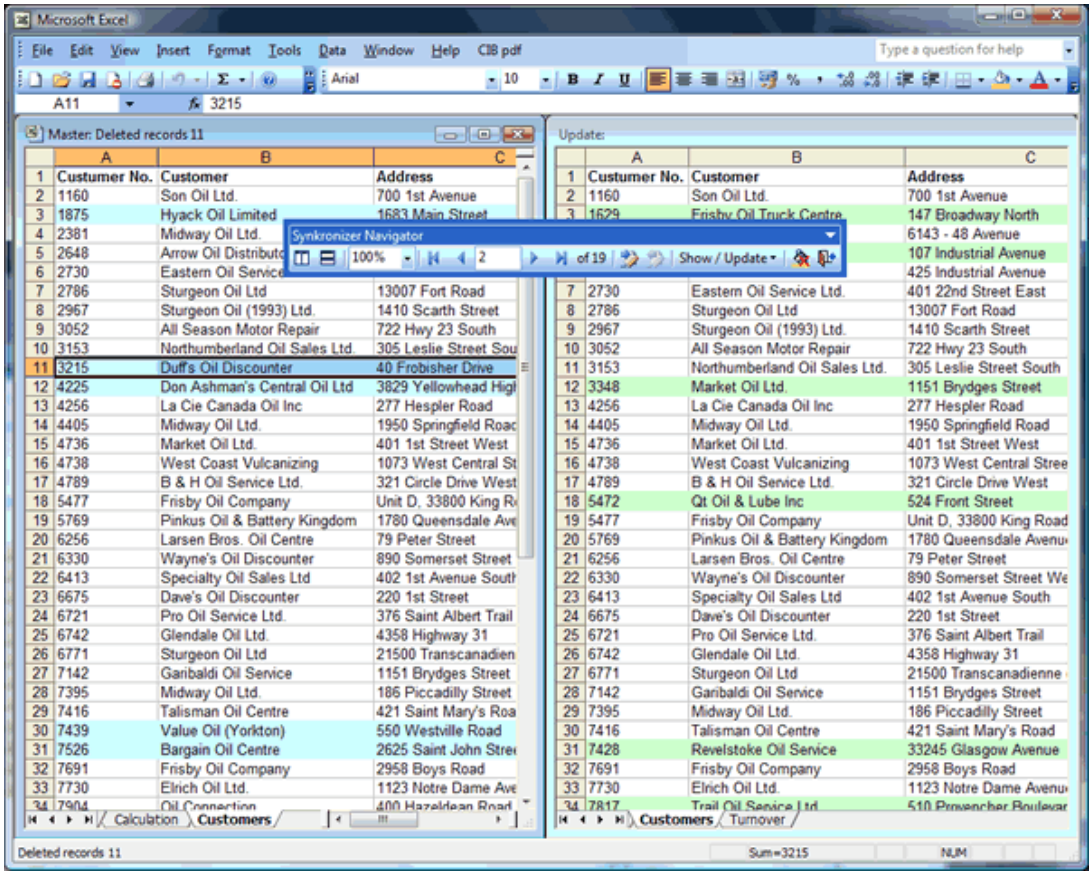


Figura 11 - Comparador do Microsoft Excel

O conceito de evolução de planilhas é abordado neste trabalho como um conjunto de modificações feitas ao longo do tempo a uma planilha. Mais formalmente, se considerarmos uma planilha ***P***, tal que

$$P = \{a_{ij} \in P \mid 0 \leq i \leq C, 0 \leq j \leq L\},$$

onde ***a*** é uma célula que pertence à planilha ***P***, ***C*** é o número de colunas de ***P*** e ***L*** é o número de linhas de ***P***, e um conjunto de operações **Ω**, que representa todas as modificações possíveis a planilhas, tal que

$$\Omega = \{m_n(a) \in \Omega \mid a \in P, n > 0\},$$

onde $m_n(a)$ é a n-ésima modificação aplicada a uma célula ***a*** qualquer de S. É possível então afirmar que há um conjunto Ω', subconjunto de Ω, que transforma P_1 em P_2, onde P_2 é uma instância modificada de P_1. Como existe um conjunto infinito de operações Ω que podem ser aplicadas a uma planilha, é possível transformar qualquer P_1 em P_2.

Entra em cena então o fator similaridade, que pode ser lido como "número de modificações necessárias para transformar uma planilha em outra˜. Naturalmente, quanto

menos modificações forem necessárias aplicar em uma planilha para transformá-la em outra, mais similares entre si são as duas planilhas.

Também deve-se levar em consideração o fator tempo. Ele serve para indicar qual planilha é anterior ou posterior a outra e será vital para a criação de uma "linha do tempo" de planilhas, como exibido na Figura 18. Arquivos de planilhas são documentos digitais e, como tais, possuem diversos meta-dados, que auxiliam a catalogação e ordenação dos documentos. Um dos meta-dados mais comuns é exatamente o campo que indica quando um documento foi criado e/ou alterado pela última vez (esse campo é chamado muitas vezes de timestamp) e ele pode ser utilizado para determinar se uma planilha é anterior ou posterior à outra similar.

O fator similaridade é crucial para a análise feita das duas planilhas uma vez que é ele que indicará o quão custoso é transformar uma planilha noutra. Nas próximas seções serão tratadas métricas de similaridade aplicados a alguns contextos e, em seguida, às planilhas. Essas métricas de similaridade são a base para se comparar as planilhas e aferir se elas têm a mesma origem ou não.

2.5. Comparações de documentos, grafos e planilhas

2.5.1. Comparação de documentos

São conhecidas na literatura algumas formas distintas de se comparar dois documentos quaisquer, seja pelo conteúdo (BAEZA & YATES, 1999), pela estrutura (NIERMAN, 2002), pelo esquema do documento (RAHM & BERNSTEIN, 2004) ou seja por esquemas com ontologias (AUMUELLER, 2005). Com foco em documentos textuais, tais abordagens analisam lexicamente os documentos, enquanto outras fazem uso de análise semântica (MAEDCHE, 2002).

A ideia geral das abordagens de comparação por conteúdo consiste em computar valores de métricas de similaridades entre os documentos e realizar a comparação destes valores inferidos. Caso os valores obtidos pela computação das métricas sejam próximos, os documentos avaliados são considerados razoavelmente similares.

Uma das principais métricas léxicas de similaridade é a distância de Levenshtein, proposta por LEVENSHTEIN (1966). Ela consiste em avaliar pares de cadeias de caracteres ordenados e calcular a quantidade mínima de mutações necessárias para transformar uma

cadeia na outra. Uma mutação pode ser a adição, substituição ou remoção de um dos caracteres que formam a cadeia.

Um exemplo simples é a comparação entre as palavras (exemplos simples de cadeias) *gato* e *rato*, que têm distância de edição igual a 1, uma vez que basta substituir o caractere *g* pelo caractere *r* para transformar a primeira na segunda - e vice-versa. Se considerarmos que uma planilha é uma cadeia ordenada de células, podemos aplicar o mesmo conceito de distância de edição computando o número de mutações necessárias para transformar uma planilha em outra. Em outras palavras, a distância de Levenshtein aplicada a duas planilhas quaisquer computaria o número de células distintas entre elas. Para fins práticos, somente seria considerado o valor da célula, e não sua formatação e nem a fórmula que a compõe. Logo, as mutações possíveis seriam de inserção, substituição e deleção de células.

Uma outra abordagem bastante comum, mais genérica, é o cálculo do *coeficiente de* Jaccard (JACCARD, 1901), em que se calcula a similaridade entre dois conjuntos. Os dois conjuntos são similares proporcionalmente à razão entre a interseção e a união dos elementos que compõem os conjuntos. Mais formalmente, considerando coeficiente de Jaccard entre os conjuntos *A* e *B* como J(A,B), temos:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

A título de exemplo, podemos computar J(gato, rato) como sendo

$$\frac{|a,t,o|}{|g,r,a,t,o|} = \frac{3}{5} = 0{,}6.$$

Como se pode notar, o coeficiente de Jaccard é representado por um número real compreendido no intervalo (0, 1), diferente da comparação à distância de Levenshtein, uma vez que o domínio de valores possíveis do coeficiente de Jaccard encontra-se num intervalo limitado enquanto a distância de Levenshtein pode assumir qualquer valor inteiro entre 0 (planilhas totalmente similares) e ∞. Na prática, o valor máximo da distância de Levenshtein é o número máximo de células que uma planilha pode comportar. Aplicando o conceito do coeficiente de Jaccard às planilhas, pode-se computar o número de células em comum entre duas planilhas quaisquer e o número total de células distintas entre elas para calcular-se o coeficiente.

Uma outra métrica de similaridade bastante utilizada é a distância de cosseno, onde os documentos avaliados são transformados para um modelo vetorial e cada documento é representado por um vetor cujas dimensões representam a quantidade de vezes que um

termo ocorre naquele documento. É calculado então o cosseno do ângulo formado pelos dois vetores, e quanto mais próximo de 1 mais similares são os documentos. Utilizando novamente o exemplo entre *gato* e *rato*, a distância de cosseno entre essas duas palavras pode ser representada por:

Representação vetorial de *gato*: (1, 0, 1, 1, 1)

Representação vetorial de *rato*: (0, 1, 1, 1, 1)

$$cos(\theta)=\frac{(1*0)+(0*1)+(1*1)+(1*1)+(1*1)}{\sqrt{(1^2+0^2+1^2+1^2+1^2)*(0^2+1^2+1^2+1^2+1^2)}}=\frac{3}{\sqrt{4*4}}=\frac{3}{4}=0{,}750$$

Uma forma de aplicar a distância de cosseno à similaridade de planilhas é usar o mesmo conceito utilizado acima, mas considerar a contagem de células para a transformação para o modelo vetorial.

Muitas outras métricas de similaridades de cadeias de caracteres foram criadas e sua utilização depende do problema enfrentado e da abordagem escolhida. Uma comparação simples e eficiente entre as principais[12] pode dar ao leitor uma noção das peculiaridades entre elas e das abordagens mais apropriadas para cada uma delas.

Diferentemente da comparação de conteúdo, a comparação estrutural tem como conceito a comparação entre documentos estruturados, como por exemplo documentos no formato XML. Novamente é necessário estabelecer uma métrica de comparação entre as estruturas, como a distância de edição entre árvores proposta em (NIERMAN, 2002), que leva em consideração o mesmo conceito da distância de Levenshtein: a similaridade é inversamente proporcional ao número de operações necessárias para transformar uma árvore em outra. Se considerarmos uma planilha como uma árvore, onde células adjacentes são nós filhos de outra célula, podemos aplicar este tipo de comparação.

2.5.2. Comparação de grafos

A área de comparação de grafos distintos tem tido foco significativo na última década dentro do meio acadêmico e uma importante pesquisa realizada por (GAO ET AL, 2010) resume o surgimento, a evolução e o estado da arte dela. Uma das abordagens mais importantes, e que tem sido amplamente utilizada em análises e reconhecimento de padrões, é a distância de edição de grafos (*graph edit distance* - GED), que pode ser caracterizada como uma medida de similaridade entre pares de grafos distintos e é tolerante a erros.

[12] http://asecuritysite.com/forensics/simstring, disponível em 06/09/14

Uma das primeiras contribuições para o estudo de algoritmos para GED foi feita por (SANFELIU & FU, 1983), que basicamente contava o número de renomeações, inserções e deleções feitas a nós e arestas de um grafo para transformá-lo em outro. (MESSMER & BUNKE, 1994; MESSMER & BUNKE, 1998) expandiram esse conceito ao considerar a edição feita a sub-grafos. Por contar com algumas diferenças em relação à tradicional distância de edição de cadeias de caracteres (*string edit-distance*) outras abordagens foram propostas, como a consideração do tamanho máximo de sub-grafo (BUNKE, 1997).

GED's podem ser computadas tanto para grafos com atributos quanto para grafos sem atributos. Grafos com atributos consideram, como a própria classificação indica, atributos (nome, valor etc.) para os nós e arestas do grafo, ao passo que grafos sem atributos consideram apenas as informações de conectividade da estrutura (GAO ET AL, 2010). Por considerar apenas a informação de conectividade, os grafos sem atributos geralmente têm suas estruturas convertidas para cadeias de caracteres, onde é aplicada a tradicional distância de edição de cadeia de caracteres. Uma das formas de realizar esta conversão e executar a comparação de grafos e árvores é através de programação dinâmica (ZHANG & SHASHA, 1989; ZHANG, 1996). Também foram tentadas abordagens como distância de Hamming (WILSON & HANCOCK, 1997), distância de Levenshtein (MYERS ET AL, 2000) e cadeias de Markov (WEI, 2004).

2.5.3. Comparação de planilhas

Feito o levantamento sobre abordagens diferentes de comparação de documentos de texto e grafos, nesta seção será abordado o problema de se comparar planilhas.

A distância de edição é um problema que pode ser informalmente reescrito como "dados dois elementos quaisquer, quão diferentes entre si eles são?". Intuitivamente, podemos afirmar que se os dois elementos têm um valor baixo de distância de edição, eles são razoavelmente similares (ou semelhantes). Esta é uma conclusão trivial, uma vez que um baixo valor de edição indica poucas modificações a serem feitas para transformar um elemento no outro.

De forma geral, o problema de computar a distância de edição é bem explorado quando o problema trata de cadeias de caracteres - *strings*. Em diversas obras, como (CORMEN, 2001) e (DASGUPTA, 2006), é apresentada uma solução para computar a distância de edição de duas *strings* baseada em técnica de programação dinâmica.

A programação dinâmica pode ser formulada resumidamente como "uma forma computacional de resolver problemas grandes baseando-se na solução de diversos

problemas menores, cujos resultados ajudam a solucionar o problema maior, principal". Ao contrário de outras técnicas de "dividir para conquistar", a programação dinâmica não faz uso de recorrências. O raciocínio utilizado é dividir corretamente o problema principal em uma série de subproblemas menores e sequenciais, armazenando o resultado destes subproblemas em uma estrutura de dados (geralmente uma tabela) que seja preenchida gradativamente até que o resultado principal seja facilmente obtido em função dos resultados anteriores. Este tipo de abordagem é particularmente útil para situações onde um pequeno grupo de subproblemas é repetido exponencialmente. Faz sentido então computar cada um dos subproblemas uma única vez e armazenar suas respectivas soluções em uma estrutura cujo acesso seja eficiente para uso posterior. (PARBERRY, 1995).

É razoável pensar então que o problema de computar a similaridade entre duas planilhas quaisquer pode ser visto, na verdade, como uma distância de edição de planilhas. Ele pode ser reescrito então como "qual a melhor maneira de aplicar o algoritmo de distância de edição a estruturas de planilhas?". Esta é uma pergunta interessante, uma vez que a distância de edição tradicional é aplicada a strings, cadeias unidimensionais de caracteres. O novo desafio é então encontrar uma solução computacionalmente eficiente para a distância de edição bidimensional.

De fato, há na literatura (BAEZA-YATTES, 1998; ARSLAN, 2007) algumas propostas de se implementar este raciocínio, inclusive de forma mais genérica utilizando um número N qualquer de dimensões.

Chambers (2010) propôs o algoritmo *SheetDiff*, voltado para planilhas de dados. Mais recentemente, o *RowColAlign,* uma evolução do *SheetDiff,* foi divulgada (HARUTYUNYAN, 2012), apresentando sensíveis melhorias em relação ao seu antecessor. Estes dois últimos trabalhos, *SheetDiff* e *RowColAlign*, mostraram-se bastante similares à ideia base proposta aqui e, por isso, serão utilizados como base de comparação. Para isso, é necessária uma visão mais próxima dos conceitos abordados em ambos algoritmos, a ser feita a seguir.

O SheetDiff é um algoritmo guloso, iterativo cuja ideia base é buscar modificações entre duas planilhas A e B de forma a maximizar, em cada iteração, o valor de uma métrica de comparação. Dentre as modificações possíveis (inserção/remoção de linha/coluna), é aplicada a modificação cujo resultado faz as planilhas ficarem mais próximas entre si. Este processo é repetido até que a similaridade entre A e B alcance um determinado valor de *threshold*. Alcançado o valor, são computadas as modificações célula-a-célula de forma a igualar as planilhas examinadas. Em algumas situações, o algoritmo pode entrar em *loop* infinito, pois ele continua a calcular as modificações possíveis sem nunca atingir o *threshold* estabelecido. Obviamente pode-se diminuir o *threshold* de forma a evitar o loop, mas esta

ação faz com que as modificações de linhas/colunas parem de ser efetuadas após poucas iterações, e aumenta o número necessário de modificações célula-a-célula para igualar A e B.

O RowColAlign analisa duas planilhas A e B e calcula um *score* de alinhamento entre A e B (análogo à *longest common subsequence*). Esse alinhamento é caracterizado uma outra planilha T(A,B) formadas pelas linhas e colunas que A tem em comum com B - tolerando algumas alterações unitárias de células (isto é: as linhas e colunas incluídas em T(A,B) podem não ser 100% iguais às linhas e colunas que foram originalmente A ou B). Da mesma forma, pode ser construída uma planilha T(B,A), que considera o alinhamento de B em relação a A. T(A,B) e T(B,A) têm as mesmas dimensões (mesmo número de linhas e colunas), mas suas células são diferentes. A diferença entre as células explica-se pelo fato de que as alterações necessárias para transformar A em B são diferentes das que transformam B em A.

A implementação atual do RowColAlign não leva em consideração o fato de uma matriz ser maior que outra: ele considera o alinhamento delas, as linhas/colunas extras não representam um fator de comparação. Isto é: se compararmos as planilhas A e B, dado que A está contido em B (ex.: B é composto de A acrescido de uma linha), a similaridade é a máxima possível. Isso não é um resultado bom, uma vez que este valor indicaria que A é igual a B, o que não é verdade. Também é notado como falha o fato de planilhas pequenas terem baixa precisão (são indicadas como sendo iguais, quando na verdade não o são).

Além disso, o RowColAlign retorna apenas o valor de similaridade entre dois arquivos *.csv, contendo os dados das planilhas que se quer comparar. Ele não retorna as edições necessárias para se transformar A em B ou B em A, nem diz quantas edições são necessárias.

Retornar quais e/ou quantas edições são necessárias para transformar uma planilha em outra é uma informação importante, pois é a essência de se traçar uma linha do tempo entre várias planilhas. Por exemplo, dadas as matrizes A e B abaixo (Figura 12 e Figura 13), a comparação célula-a-célula resultaria em uma distância de edição 6 - pois 6 células tiveram seus respectivos valores alterados para que uma matriz fosse transformada na outra.

1	2	3
1	2	3
1	2	3

Figura 12 - Matriz A

1	**3**	**2**

1	**3**	**2**
1	**3**	**2**

Figura 13 - Matriz B

Tomando outro exemplo, analisando as matrizes A e D (Figura 14) observamos que a distância de edição entre A e D também é 6, tal qual entre A e B.

6	**4**	**9**
1	**5**	**7**
10	2	3

Figura 14 - Matriz D

O fato das duas distâncias serem iguais não reflete a diferença de informação disposta nas matrizes. Tomando como base a matriz A, em B foi permutado um par de colunas, enquanto em D foram modificados alguns valores de todas as linhas e colunas em relação a A, sem um padrão.

Considerando E(X,Y) a distância de edição entre duas matrizes X e Y quaisquer, poderia ser mais interessante representar E(A,B) = 2, que é o número de colunas substituídas, ou até mesmo E(A,B) = 1, número de permutações feitas entre A e B.

Chegamos então a um ponto interessante, onde concluímos que a distância de edição E(A,B) entre duas matrizes A e B quaisquer pode ter valores distintos dependendo da abordagem utilizada. Essa situação pode ser representada por um grafo de transformações entre A e B, exibido na Figura 15.

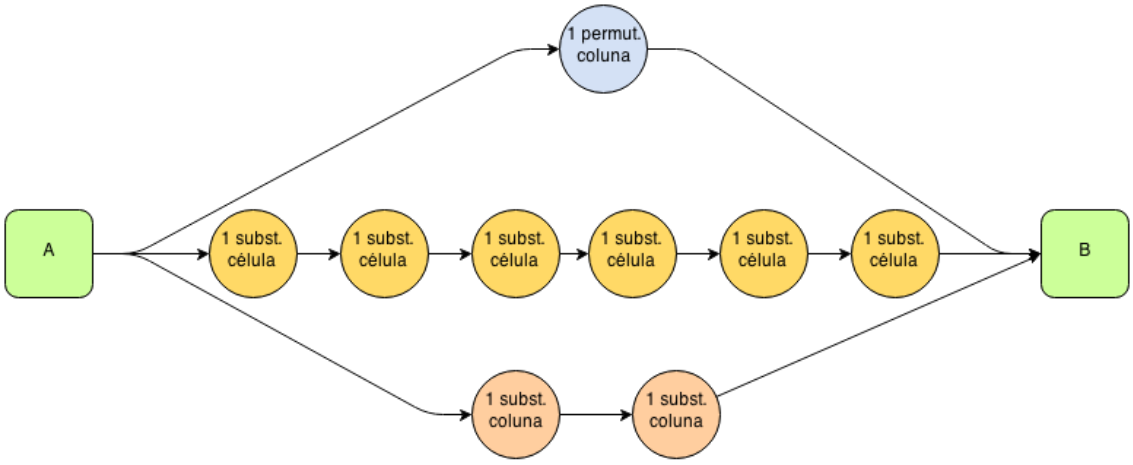


Figura 15 – Grafo de comparações entre duas planilhas

Como o objetivo final é estabelecer um valor que represente o número mínimo de operações para transformar A em B, o valor de E(A,B) passa a ser o resultado de cálculo de menor caminho no grafo de transformações entre A e B. Com esta abordagem é possível ainda estabelecer pesos para cada uma das operações, onde cada nó representa uma transformação e cada aresta entre dois nós representa a precedência entre eles. O maior desafio então é identificar quais os nós e arestas possíveis de um gráfico de transformações e estabelecer os pesos para as arestas. Ou seja, como identificar eficientemente quais operações devem ser listadas neste grafo.

O termo "eficientemente" foi empregado para destacar que poderia listar-se N transformações possíveis, inclusive com combinações e permutações das transformações mais básicas, gerando um grafo cíclico ou elevando exponencialmente o número de casos a serem avaliados. Naturalmente este não seria um raciocínio eficiente e considera-se somente o número mínimo de transformações em cada caminho, de forma a gerar um grafo acíclico.

2.6. Edição de planilhas na web

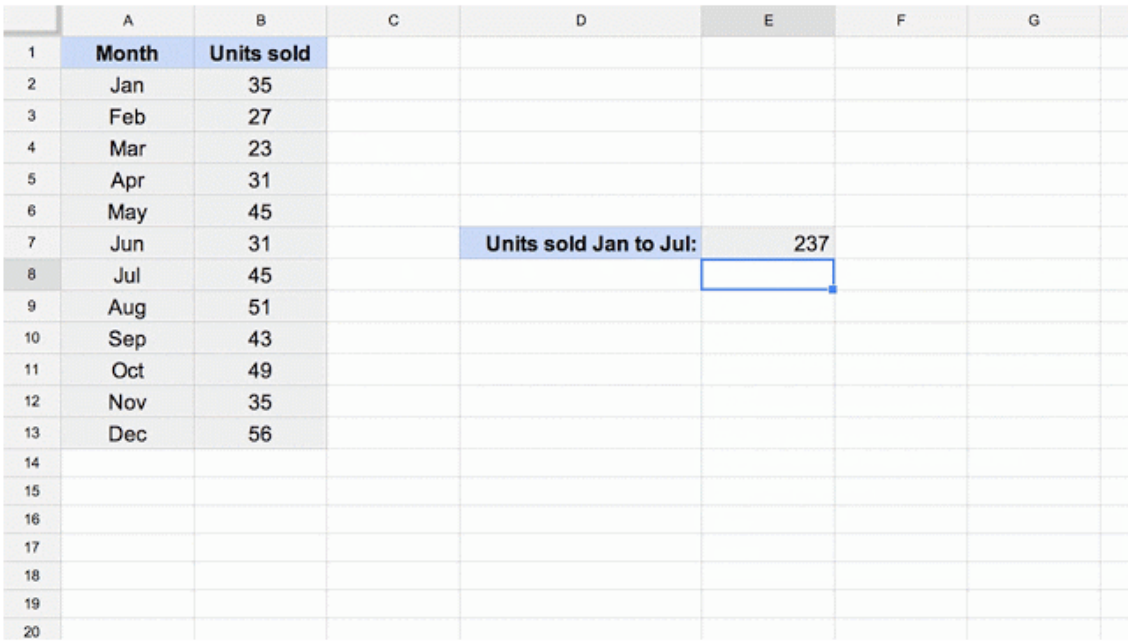
Como explicado acima (Figura 4), a maior parte dos usuários corporativos faz uso das planilhas eletrônicas para diversas finalidades. Mais da metade (54%) as usa para controle de dados e/ou para determinação de padrões e previsões, enquanto 29% usam para operações triviais de listagens de dados. Além disso, 78% das planilhas são compartilhadas (são utilizadas por mais de uma pessoa).

Com base no uso compartilhado, é de se supor que um ambiente colaborativo seja a melhor alternativa para atender essa demanda. Não por acaso, grandes empresas estão convergindo para um cenário colaborativo na web para o uso de diversas ferramentas de escritório, inclusive as planilhas eletrônicas.

O Google Drive[13], por exemplo, oferece ambiente web com armazenamento de arquivos compartilhados (com restrição de acesso por usuário ou grupo de usuários) e edição colaborativa entre múltiplos usuários. O Google Sheets[14] é o editor de planilhas acoplado ao Google Drive. A Figura 16 mostra um exemplo de tela do Google Sheets, que tem visualização muito semelhante ao líder de mercado Microsoft Excel.

[13] http://www.google.com/intx/pt-BR/enterprise/apps/business/products/drive/, disponível em 06/09/2014

[14] http://www.google.com/intx/pt-BR/enterprise/apps/business/products/sheets/, disponível em 06/09/2014



	A	B	C	D	E	F	G
1	**Month**	**Units sold**					
2	Jan	35					
3	Feb	27					
4	Mar	23					
5	Apr	31					
6	May	45					
7	Jun	31		**Units sold Jan to Jul:**	237		
8	Jul	45					
9	Aug	51					
10	Sep	43					
11	Oct	49					
12	Nov	35					
13	Dec	56					
14							
15							
16							
17							
18							
19							
20							

Figura 16 - Google Sheets

A nova versão do Microsoft Office[15], chamada de *Office 365*, também já faz uso de um ambiente web com uso compartilhado e colaborativo entre os usuários. Além deles, o Zoho Spreadsheet[16] também oferece esses recursos, comprovando a tendência do mercado de migrar as aplicações para ambientes de colaboração web.

No entanto, nenhuma dessas soluções resolve totalmente o problema de entrada desregrada dos dados. Chen (2013) indica que a melhor solução para esse entrave pode ser a migração de dados para um sistema de informação baseado em bancos de dados relacionais. Considerando que o tamanho médio da maior parte das planilhas é inferior a 10.000 células, é razoável afirmar que um banco de dados relacional tem capacidade de comportar esse volume de informação.

Entra em cena então outro problema: como migrar planilhas e mantê-las sincronizadas com um modelo de dados relacional? Esta pergunta será melhor abordada na próxima seção, que trata da extração de dados e meta-dados a partir de planilhas.

2.7. Trabalhos similares

Os problemas listados anteriormente já são conhecidos na literatura que trata sobre o cenário das planilhas eletrônicas. Como forma de limitar as falhas apresentadas, geralmente

[15] http://office.microsoft.com/en-us/business/what-is-office-365-for-business-FX102997580.aspx, disponível em 06/09/2014

[16] https://www.zoho.com/docs/online-spreadsheet.html, disponível em 06/09/2014

são propostas diretrizes internas (governança de TI) ou migrações das informações das planilhas para sistemas de informação. Essa migração geralmente é baseada na transformação das planilhas para modelos relacionais de dados com a posterior aplicação de regras de negócio na entrada de dados. Em outras palavras, é restrita boa parte da flexibilidade oferecida pelas planilhas.

2.7.1. Extração de dados e meta-dados

A extração de dados e meta-dados a partir de planilhas é um tema em pesquisa há alguns anos. Como uma das principais utilizações de planilhas é a organização de dados no formato tabular, a principal linha de pesquisa é a manipulação de dados tabulares armazenados em planilhas e em outros documentos. A manipulação compreende detecção, extração e interpretação de domínio das tabelas analisadas. As fontes de dados podem ser as próprias planilhas, documentos de texto, páginas no formato HTML e, até mesmo, imagens de micro-filmes.

Documentos no formato HTML representam uma importante fonte de dados para manipulação de dados tabulares, uma vez que cerca de 52% das páginas web apresentam uma ou mais tabelas (LIM, 1999). O reconhecimento deste tipo de tabela pode ser feito pela análise espacial (identificação dos limites da tabela) (LOPRESTI, 1999), por técnicas de clustering hierárquico somado a critérios léxicos para classificação de elementos da tabela (HU, 2001), pela interpretação simples das tags HTML (HURST, 2001). Em (TAO, 2003) são citadas ainda outras técnicas para interpretação da estrutura de tabelas HTML.

Outra fonte de dados interessante é formada por documentos em microfilme, que contêm muitas informações, mas cuja extração e organização manual é uma tarefa lenta e suscetível a erros (TUBS, 2002). Como um passo inicial em direção à automação do acesso a tais informações, foi proposto um algoritmo para automaticamente identificar padrões nos registros encontrados em tabelas de microfilmes para domínios pré-especificados. O algoritmo recebe como entrada um arquivo no formato XML que descreve individualmente as células de uma tabela retirada de um documento em microfilme e busca, para cada registro, as células que juntas representam a informação do registro. Duas características fundamentam o algoritmo: o layout que tradicionalmente forma as tabelas e a classificação por rótulos (“label matching”) referentes a uma ontologia específica de um domínio. O algoritmo alcançou uma acurácia de 92% no corpus de teste composto por tabelas de microfilmes genealógicos. O reconhecimento da estrutura da tabela é feito através da separação das células em células de rótulo, células de valor ou células vazias. Com base nestes dados e em características intrínsecas ao domínio (chamadas de *domain features*), é

possível criar regras de correlação, que buscam relacionar e ponderar os valores encontrados para cada uma das features analisadas anteriormente. Cada uma das regras avalia se o valor extraído da célula é razoável para o contexto analisado e para a estrutura de tabela inferida. É interessante notar que, da mesma forma que nos documentos HTML, a extração de tabelas a partir de imagens digitalizadas de micro-filmes depende fortemente da criação de ontologias de domínio e, portanto, tendem a perder eficácia quando utilizadas em domínios variados. A solução indicada é que para um problema que envolva diversos domínios, talvez seja interessante selecionar apenas features genéricas, comuns a vários domínios.

Voltando para o cerne desta dissertação, a extração de tabelas a partir de planilhas nem sempre é trivial, mesmo considerando que a estrutura de arquivos de planilhas sugere a criação de dados puramente tabulares. Não é raro que usuários deixem de definir explicitamente as tabelas em uma planilha. Na verdade, é muito comum que isso ocorra. Os usuários acostumaram-se a usar tabelas indefinidas (tabelas sem delimitação de colunas e cabeçalhos) nas planilhas e simplesmente não tiveram um motivo imposto para se adaptarem a tal funcionalidade. Logo, muitos continuam a usar o método antigo de simplesmente catalogar dados organizados tabularmente, porém sem definir explicitamente a estrutura da tabela utilizada.

Engels & Erwig (2005) e Abraham (2005) citam um modelo de alto nível para representação de tabelas chamado ClassSheet. Ele permite que o desenvolvedor modele objetos de negócio dentro de uma planilha, fazendo uso dos conceitos de UML (BOOCH, 2006) mas sem perder as facilidades de se trabalhar com uma estrutura bidimensional de representação de dados. No entanto, é necessário que um usuário tenha esse conhecimento prévio para realizar essa modelagem simples dentro da própria planilha, o que pode ser um obstáculo para a maior parte dos usuários leigos. Wang (1996) apresentou o conceito de “tabela abstrata”, que é uma representação de tabelas independente do layout utilizado para a representação das mesmas. Através dela, é possível representar planilhas como tabelas, e convertê-las para entidades de um modelo relacional.

Uma vez que a tabela não esteja explicitamente definida em uma planilha, faz-se necessário um mecanismo para detectar sua existência e extrair os dados e meta-dados relativos a ela para que seja feita a transformação dos dados de planilhas para modelos relacionais. (CHEN, 2013) lista alguns estudos recentes que visam fazer esta operação, possibilitando a integração entre planilhas e sistemas gerenciadores de bancos de dados. Cita também que alguns sistemas de extração (AHMAD, 2003; HUNG, 2011) requerem intervenções diretas de usuários por cada tipo de planilha, o que é inviável para conjuntos muito grandes de planilhas.

Em resumo, a maior parte dos trabalhos relacionados à identificação de tabelas (tanto sua estrutura como seus dados) cita o grave obstáculo da heterogeneidade dos domínios envolvidos e a complexa estrutura de algumas tabelas, como as tabelas aninhadas. O uso de ontologias pode ser uma saída, mas cada domínio possui um conjunto distinto de palavras-chave que podem auxiliar a identificação de células, rótulos e atributos de uma tabela. Então é necessário identificá-lo antes de realizar o processamento de uma tabela - o que pode ser inviável, caso não existam informações a priori sobre um determinado domínio. Há a possibilidade de usuários indicarem qual domínio, mas para conjuntos grandes de dados essa não é uma opção prática. Logo, não há ainda um mecanismo no "estado da arte" para realizar esse processo.

De uma forma abstrata, podemos descrever o problema de transformação de planilhas em modelos relacionais como a seguir: seja $\mathbf{R}$ o universo de todos os esquemas relacionais possíveis e seja $\boldsymbol{\rho}$ o universo de todas as estruturas de planilhas possíveis. Definimos como $\boldsymbol{\Omega}$ o conjunto de todas operações possíveis de tal forma que um subconjunto de operações $\{\Omega \mid \Omega \in \boldsymbol{\Omega}\}$ que, quando feitas em certa ordem, levam um elemento $\{P \mid P \in \boldsymbol{\rho}\}$ em um elemento $\{R \mid R \in \mathbf{R}\}$. Podemos dizer então que duas planilhas P_1 e P_2 são relacionalmente equivalentes se existe um conjunto Ω_1 que transforma P_1 em R_1 e um conjunto Ω_2 que transforma P_2 também em R_1. Ou seja, se:

$$\begin{cases} \Omega_1(P_1) \rightarrow R_1 \\ \Omega_2(P_2) \rightarrow R_1 \end{cases} \text{E}$$

Podemos ainda considerar que duas planilhas são relacionalmente similares se existe um elemento $\{\Omega_1 \mid \Omega_1 \in \boldsymbol{\Omega}\}$ que transforma $\{P_1 \mid P_1 \in \rho\}$ em $\{R_1 \mid R \in \mathbf{R}\}$ e um elemento possivelmente distinto $\{\Omega_2 \mid \Omega_2 \in \boldsymbol{\Omega}\}$ que transforma $\{P_2 \mid P_2 \in \rho\}$ em $\{R_2 \mid R \in \mathbf{R}\}$ e existe alguma interseção entre R_1 e R_2.

Este raciocínio se aplica tanto à estrutura quanto aos dados das planilhas, mas de maneiras não excludentes. Portanto, podemos dizer que duas planilhas quaisquer podem se relacionar como explicitado na Tabela 1. É interessante ainda ressaltar que as planilhas podem ser razoavelmente similares, tanto em relação aos dados quanto em relação à estrutura.

A ideia aqui proposta é exatamente identificar o tipo de relacionamento entre duas planilhas quaisquer. Partindo do princípio que elas sejam no mínimo similares em relação a dados e estrutura, é também desejado que as duas planilhas sejam integradas em uma só representação.

Tabela 1 - Relacionamento entre planilhas

	Estruturalmente equivalente	**Estruturalmente similares**	**Estruturalmente independentes**
Informacionalmente equivalente	Estruturas e dados iguais	Mesmos dados e estruturas similares	Estruturas independentes, mas com os mesmos dados
Informacionalmente similares	Mesmas estruturas e dados similares	Estruturas e dados similares	Estruturas independentes, mas com dados similares
Informacionalmente independentes	Mesmas estruturas, mas dados independentes	Estruturas similares, mas dados independentes	Estruturas e dados independentes

2.7.2. DataUp

Voltado para o cenário acadêmico, o DataUp (STRASSER, 2013) é um ambiente web, voltado para cientistas e pesquisadores, que propõe uma camada intermediária de publicação de dados baseados em sistemas de planilhas. Como as plataformas baseadas na nuvem formam agora o local padrão de armazenamento de dados e plataforma para novos softwares, é necessário um formato comum aos utilizadores destes dados. Para os pesquisadores comuns, no entanto, não é tão fácil fazer uso da nuvem como se espera: para arquivar, compartilhar e publicar dados científicos, é necessário ter certeza de que outros terão acesso à mesma infraestrutura para fazer bom uso deles. O DataUp permite que cientistas tenham um formato padrão, baseado na nuvem, para compartilhamento de seus dados de pesquisa e auxilia a manutenção de conjuntos de dados dos mais variados tamanhos. Como 80% dos cientistas utilizam o Microsoft Excel no dia-a-dia (STRASSER, 2014), ele é considerado o software padrão para manipulação dados tabulares e serviu como base do projeto, que tem integração fácil e permite, através de um plugin, a sincronização de dados catalogados e indexados para pesquisas colaborativas. Além disso, o DataUp também permite o compartilhamento de dados tabulares em workflows científicos e atende às condições especificadas para que os pesquisadores rumem à nuvem junto com o mercado: gratuita, fácil de se configurar e de se usar e provê uma forma unificada de se armazenar dados e de acessar software. (BISHOP, 2013).

2.7.3. Senbazuru

Converter planilhas para modelos relacionais permitiria que ferramentas tradicionais de integração de dados pudessem operar sobre eles e facilitar (em alguns casos até viabilizar) a manipulação das informações. Porém, identificar tabelas em planilhas nem sempre é uma

tarefa trivial: planilhas não indicam explicitamente quais células são atributos, quais representam valores ou quais atributos representam quais valores, por exemplo. Foi proposto em (CHEN, 2013) o Senbazuru, uma ferramenta que permite pesquisa em grandes bases compostas por planilhas. É construída uma versão relacional dos dados, que permite algumas operações interessantes - por exemplo, as operações de seleção e cruzamento de tabelas. Além disso, automaticamente indexa planilhas disponíveis em datasets na Web, transforma seus dados em modelos relacionais e permite que os usuários corrijam possíveis erros na interpretação e extração destes dados. Além disso, também é possível realizar buscas por valores dos atributos, visualizar árvores dos atributos que formam uma planilha e relacionar planilhas distintas.

2.7.4. HaExcel

Em (CUNHA ET AL, 2009) é apresentado o framework HaExcel, que transforma planilhas em bases de dados relacionais e exporta bases de dados relacionais no formato de planilhas. Além disso, também executa um processo de otimização da planilha, buscando e eliminando redundâncias. Desenvolvida utilizando a linguagem de programação puramente funcional Haskell, a ferramenta conta ainda com funcionalidades que extraem as dependências funcionais das tabelas extraídas utilizando o algoritmo FUN, proposto em (NOVELLI, 2001).

2.7.5. SheetDiff e RowColAlign

Ao migrar planilhas para sistemas de informação, elas terão seus dados integrados. Logo, faz sentido agrupá-las por similaridade. Pensando em compará-las de forma eficiente, destacam-se 2 algoritmos: o SheetDiff e o RowColAlign.

O SheetDiff (CHAMBERS ET AL, 2010) é um algoritmo guloso, iterativo cuja ideia base é buscar modificações entre duas planilhas A e B. Uma evolução do SheetDiff, o RowColAlign (HARUTYUNYAN, 2012) analisa duas planilhas e retorna um valor de similaridade entre A e B. Esses e outros correlatos de comparação de planilhas serão melhor cobertos no capítulo 4.

2.8. Considerações finais

Neste capítulo foi apresentado ao leitor o cenário de uso de planilhas por usuários de computadores pessoais, destacando o seu uso em empresas. Foram traçadas algumas características deste uso e, a partir dela, foram tiradas algumas conclusões que balizaram a pesquisa aqui apresentada, a saber:

- As planilhas são de extrema importância nas empresas e são usadas frequentemente;
- Uma planilha é usada, replicada e editada por mais de um usuário, e o mesmo se aplica às réplicas;
- A maior parte das planilhas utilizadas tem até 1.000 células;

Foram também listadas as consequências e os tipos de falha decorrentes do uso descontrolado das planilhas. Em seguida foi introduzido o problema de versionamento e monitoramento de evolução de planilhas, além de levantadas algumas abordagens de comparações tradicionais de literatura.

Finalmente, foram listados e comentados alguns trabalhos correlatos ao tema tratado ao longo desta dissertação.

Capítulo 3 - Arquitetura proposta

Neste capítulo será proposta em alto nível uma arquitetura que deve ser capaz de atender as seguintes restrições:

- Ser de fácil utilização por usuários não-especialistas;
- Ser um repositório de arquivos de planilhas;
- Ter as vantagens de controle de uma aplicação baseada em SGBDR;
- Possibilitar a comparação de planilhas;
- Facilitar a unificação de planilhas semelhantes;
- Possibilitar a visualização da evolução de planilhas semelhantes;

3.1. Arquitetura

A arquitetura preliminar aqui proposta visa atender às restrições listadas no início deste capítulo, a saber:

- Ser de fácil utilização por usuários não-especialistas;
- Ser um repositório de arquivos de planilhas;
- Ter as vantagens de controle de uma aplicação baseada em SGBDR;
- Possibilitar a comparação de planilhas;
- Facilitar a unificação de planilhas semelhantes;
- Possibilitar a visualização da evolução de planilhas semelhantes;

A Figura 17 representa em alto nível a arquitetura, composta de três módulos principais e de alguns submódulos, descritos em seguida.

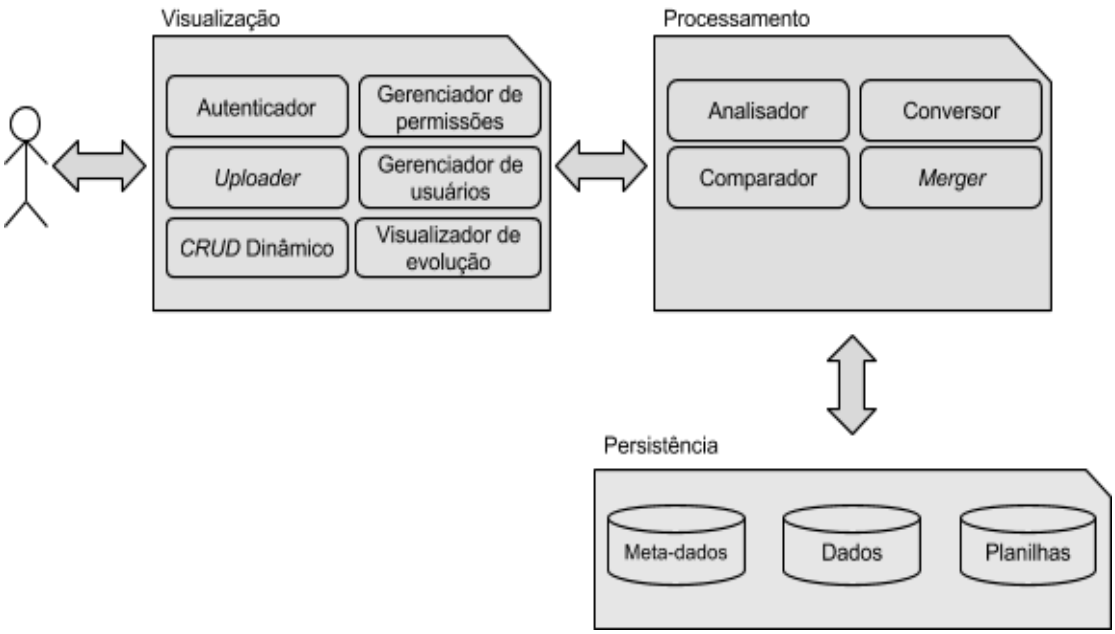


Figura 17 – Arquitetura em alto nível

3.1.1. Visualização

O módulo de visualização[17] é a interface principal entre o sistema e seus usuários. Tendo em vista que a maior parte de usuários de planilhas é formada por usuários leigos (não-especialistas no domínio de desenvolvimento de sistemas), este módulo deve comunicar de maneira clara e objetiva quais etapas necessárias para que se envie uma planilha e se realize as operações possíveis com ela. Deve ser, portanto, um sistema com interface clara e intuitiva, de forma a facilitar sua operação por parte do usuário final.

Por ser um sistema multiusuário, é razoável considerar que ele será capaz de gerenciar[18] usuários e suas respectivas permissões de acesso.

A seguir são detalhados os sub-módulos que compõem o módulo de visualização:

- **Autenticador**: responsável pelas funcionalidades de autenticação do usuário no sistema (através de identificação e senha, por exemplo) e de recuperação de senha. É necessário para garantir que somente usuários autenticados tenham acesso ao sistema;
- **Gerenciador de permissões**: responsável por permitir ou impedir que os usuários realizem determinadas operações sobre determinadas planilhas.

[17] Também poderia ser chamada de *GUI* (*Graphical User Interface*), como de praxe

[18] Entenda-se as operações básicas de criação, edição, listagem, busca e deleção

Presume-se que um usuário principal (o administrador do sistema) teria liberdade total para realizar estas concessões e bloqueios. Este sub-módulo é necessário para evitar que usuários façam operações não autorizadas sobre determinadas planilhas, como, por exemplo, apagar uma coluna importante ou mudar o tipo de dados de uma coluna, criando uma planilha inconsistente e/ou incorreta. Com este tipo de prevenção, diminui-se o risco de serem gerados erros por acesso não autorizado;

- **Gerenciador de usuários**: responsável por efetuar operações de criação, edição, listagem e bloqueio de usuários. Um usuário bloqueado não teria acesso ao sistema até que ele fosse novamente desbloqueado. Foi considerado o bloqueio, ao invés da tradicional deleção, para viabilizar eventuais auditorias de dados;
- ***Uploader***: responsável por receber dos usuários as planilhas enviadas por eles. Deve ser capaz de identificar arquivos válidos (arquivos que representem planilhas eletrônicas, seguindo formato especificado) para evitar que arquivos inválidos sejam enviados ao sistema - evitando brechas de segurança e garantindo economia de recursos de armazenamento e processamento.
- ***CRUD***[19] **dinâmico**: responsável por gerar uma interface dinâmica de criação, edição, listagem e deleção dos dados armazenados nas planilhas e convertidos para o modelo relacional de dados criado. Serão apresentados aos usuários formulários simples para as operações supracitadas e considera-se que ao restringir desta forma a entrada de dados, os usuários não terão tanta flexibilidade como têm ao mexer diretamente com as planilhas. Sendo assim, através da validação de dados feita por estes formulários, evita-se a entrada desregrada de dados e diminui-se o risco de ocorrerem erros mecânicos. As fórmulas identificadas nas planilhas devem ser aplicadas às regras de validação dos formulários gerados automaticamente;
- **Visualizador de evolução**: responsável por exibir ao usuário a evolução temporal das planilhas as quais ele tem acesso. Através desta visualização o usuário teria a capacidade de avaliar graficamente quais alterações ocorreram, em que momento ocorreram e quais usuários foram os responsáveis por cada uma das mudanças. Tal funcionalidade poderia ser fundamental para auxiliar auditorias de dados. Um exemplo de visualização de evolução de planilhas é

[19] Sigla tradicional do domínio de sistemas de informação, representa as letras iniciais dos termos *Create*, *Retrieve*, *Update* e *Delete*

exibido na Figura 18. Esse exemplo ilustra 4 eventos distintos ao longo do tempo, a saber:

1. 10/05/2009 14:56 - Envio de uma nova planilha pelo usuário Gustavo de O. Fernandes
2. 11/05/2009 13:18 - Alteração de planilha já existente pelo usuário Gustavo de O. Fernandes
3. 11/05/2009 13:38 - Alteração de planilha já existente pelo usuário Gustavo de O. Fernandes
4. 11/05/2009 15:34 - Alteração de planilha já existente pelo usuário Thiago B. Henriques

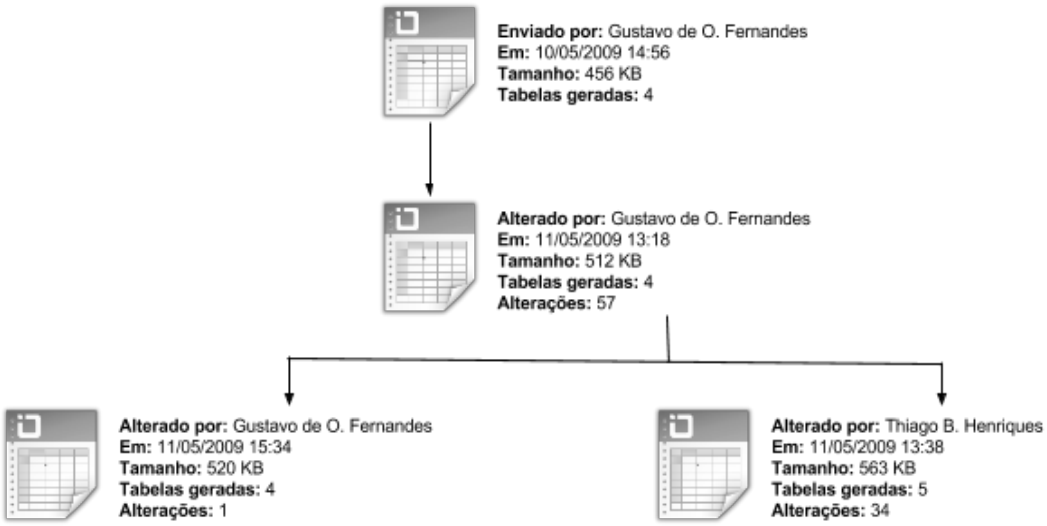


Figura 18 – Árvore do tempo de planilhas

3.1.2. Processamento

O módulo de processamento é o principal do sistema e é o responsável por todas operações de análise e processamento das planilhas.

- **Analisador**: responsável por analisar arquivos enviados através do sub-módulo *uploader* e definir se os arquivos contém planilhas válidas ou não. Caso seja válido, o módulo também é responsável por extrair e indexar todos os meta-dados e dados das planilhas existentes no arquivo. Os meta-dados devem incluir, mas não se limitar a:
    - Nome do arquivo e de cada uma das planilhas que o compõem

    - *Timestamp* do arquivo
    - Autor do arquivo[20]
- **Conversor**: responsável por converter as planilhas extraídas de um arquivo para um modelo relacional de dados e também pelo caminho contrário. Isto é, é ele que faz a transformação bi-direcional entre o formato de representação de dados em planilhas para o formato de representação de dados com entidades e relacionamentos. Este módulo pode utilizar uma das abordagens descritas na seção anterior desse capítulo;
- **Comparado**r: responsável por comparar duas planilhas e determinar o quão similares elas são. A métrica utilizada (ou conjunto de métricas utilizadas) para comparar os arquivos deve cobrir as características particulares das planilhas e deve ser flexível o suficiente para definir, dados dois arquivos quaisquer de planilhas, qual das opções abaixo se aplica a eles:
    - Os dois arquivos não são similares o suficiente;
    - Os dois arquivos são suficientemente similares e:
        - São iguais (são cópias entre si) ou;
        - São arquivos diferentes, mas que provavelmente surgiram a partir de um arquivo em comum (podendo, inclusive, um ser originado do outro);
- ***Merger***: responsável por fazer a união (processo conhecido como *merge*) de dois arquivos de planilhas. O resultado deste processo deve ser uma representação relacional contendo as estruturas e os dados combinados dos dois arquivos iniciais. O primeiro passo desse processo é adaptar a estrutura das duas tabelas em questão para que elas fiquem compatíveis. Em seguida, são comparados os dados das tabelas e feitas as operações necessárias para unificar em um só destino os dados de duas fontes distintas.

3.1.3. Persistência

O módulo de persistência é o responsável por armazenar três tipos de dados: as planilhas, os meta-dados e os dados em si.

- **Meta-dados:** são os dados que descrevem os arquivos de planilhas enviados (autor, data de envio, tamanho, etc.), as tabelas geradas a partir da transformação delas (seus identificadores, atributos, tipos, restrições, chaves

[20] Entende-se por autor do arquivo o usuário que enviou o arquivo para o sistema proposto

primárias etc.). Outros meta-dados podem ser armazenados, dependendo da implementação da arquitetura e da abordagem utilizada;

- **Dados:** são os esquemas relacionais criados dinamicamente, ao realizar a transformação dos arquivos de planilhas. Estes esquemas compreendem as tabelas criadas e os dados utilizados para alimentá-las. Tanto os dados informados *a priori*, através das planilhas enviadas, quanto os inseridos *a posteriori*, através do sub-módulo de *CRUD* dinâmico, devem ser armazenados nesta base;
- **Planilhas:** são os arquivos de planilhas que foram enviados pelos usuários ao longo do tempo. Eles podem ser persistidos no próprio SGBDR utilizado para os dados e os meta-dados, mas seria melhor otimizar o desempenho do SGBDR e poupar espaço de armazenamento com um sistema de arquivos (local ou remoto). Desta forma, seriam armazenados no SGBDR somente os ponteiros (*links*) para os arquivos.

3.2. Fluxograma

Nesta seção é ilustrado o fluxograma (Figura 19) que representa o processo principal do sistema proposto, que representa o envio de uma planilha para a aplicação, sua análise, extração de dados e meta-dados e posterior processamento.

O fluxograma considera que um usuário já esteja autenticado no sistema e queira enviar um novo arquivo de planilhas.

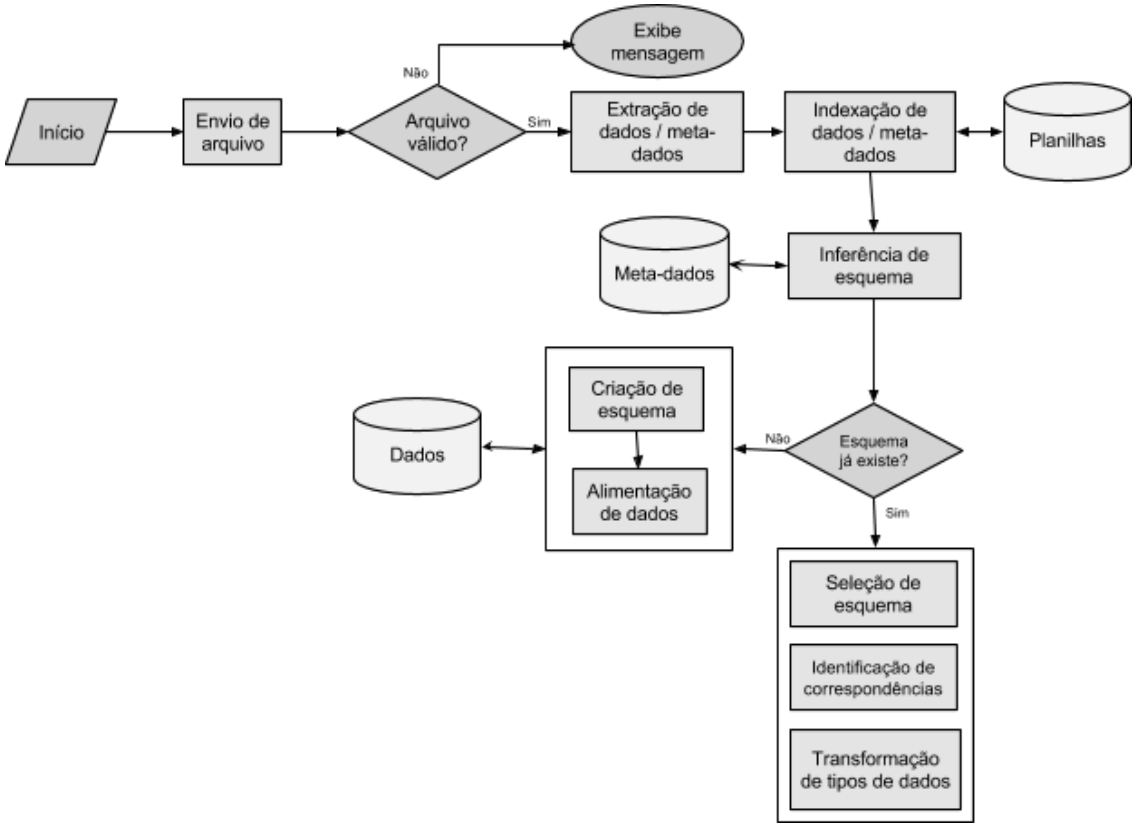


Figura 19 – Fluxo de ações para envio de nova planilha

1. Usuário envia arquivo;
2. Sistema armazena o no servidor e registra no BD meta-dados iniciais (usuário que salvou arquivo, data/hora, nome original do arquivo)
3. Sistema analisa arquivo, extraindo as planilhas contidas nele
4. Para cada planilha, sistema extrai estrutura das tabelas contidas na planilha
5. Sistema compara estrutura das tabelas da planilha com outras estruturas já salvas. Pode ser considerada diferente de todas as já cadastradas ou ser similar.
    a. Se for diferente, sistema avisa ao usuário que é diferente e que cadastrará a estrutura da nova planilha no sistema;
    b. Se for considerada similar a uma ou mais planilhas, sistema apresenta ao usuário uma lista destas planilhas - planilhas candidatas -, que são as mais similares para que usuário decida se irá fazer operação de merge ou não:
        i. Se usuário não quiser fazer operação de merge com nenhuma das planilhas apresentadas como candidatas, sistema registra a nova planilha no banco como uma nova estrutura;

ii. Se usuário decidir fazer operação de merge com uma das planilhas candidatas, sistema realiza a operação de merge fazendo as operações de transformações de dados pertinentes.

3.3. Considerações finais

Foi apresentada neste capítulo uma arquitetura em alto nível de um sistema capaz de atender às restrições listadas e identificadas como formas de auxiliar usuários a utilizar as planilhas eletrônicas de uma forma mais segura e menos propensa a erros.

Considerando os três módulos principais, o que apresenta maior desafio de implementação é o de processamento, que guarda toda a inteligência do sistema. Mais especificamente, o sub-módulo comparador tem significante importância, uma vez que ele é o responsável por identificar quais planilhas são consideradas mais ou menos similares e que deveriam ser mescladas.

Por este motivo a comparação de planilhas é o objetivo principal desta pesquisa e será detalhado no próximo capítulo como a comparação de duas planilhas pode ser feita e como ela pode auxiliar no controle de versão de uma planilha.

Capítulo 4 - Comparação e evolução de planilhas

Este capítulo se aprofunda numa das funcionalidades apresentadas na arquitetura proposta no capítulo anterior: a comparação de planilhas. A partir da comparação, é possível estabelecer relacionamentos cronológicos entre as planilhas, de forma a construir uma linha do tempo de planilhas.

Inicialmente são apresentadas algumas definições acerca das planilhas de dados. Em seguida, serão apresentadas novas abordagens de comparação. Finalmente serão comparadas as novas abordagens com algoritmos já publicados anteriormente como forma de avaliação do que foi proposto.

4.1. Definições

Neste momento, antes de efetivamente nos aprofundar no problema de comparação de planilhas, é necessária a definição formal de alguns termos que são usados no decorrer deste trabalho. O objetivo desta seção é garantir que o leitor tenha bem esclarecidos os conceitos referentes a planilhas de forma a compreender as técnicas que serão introduzidas em seguida.

4.1.1. Arquivo de planilhas

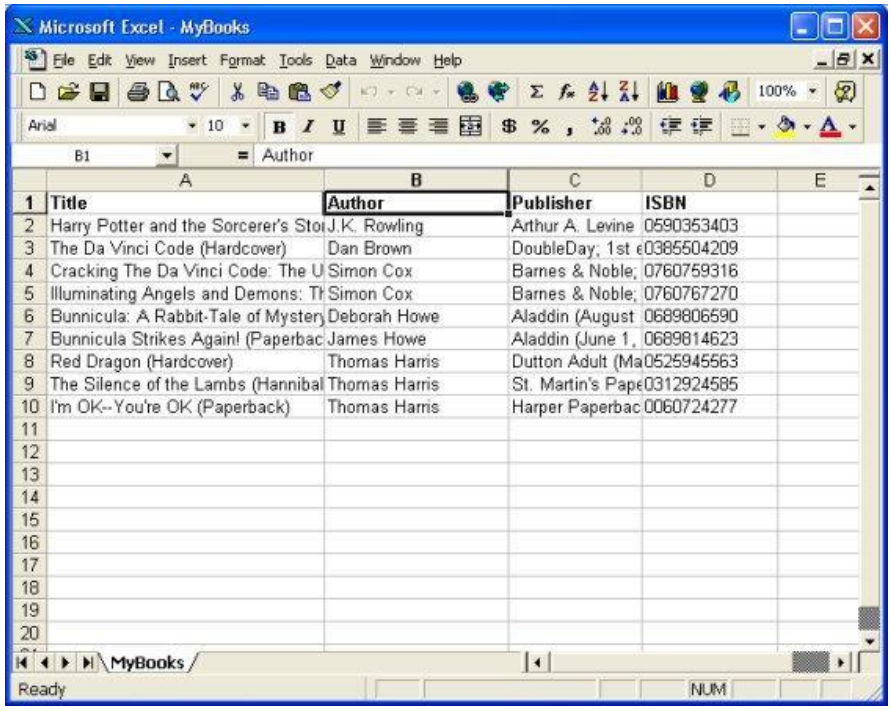
Um arquivo de planilhas é simplesmente um arquivo que contém uma ou mais planilhas armazenadas. Considerando arquivos compatíveis com o Microsoft Excel, ferramenta utilizada como referência nessa pesquisa, são geralmente arquivos salvos com extensão *.xls ou *.xlsx.

4.1.2. Planilha

Uma planilha é essencialmente um conjunto de células dispostas em formato matricial. As células são referenciadas de acordo com sua posição horizontal/vertical. Por estarem neste formato, as células são agrupadas em linhas e colunas. O formato de referência mais comum atualmente considera as colunas da matriz nomeadas com letras (de A a Z, AA a ZZ,

AAA a ZZZ etc.) e as linhas numeradas. Um exemplo de planilha é exibido[21] na Figura 20. Nela são listados os dados que dizem respeito a alguns livros em inglês.

Considerando esse exemplo, a célula contento o texto "Title" é referenciada como **A1**. Qualquer referência à célula **A1** é, portanto, diretamente associada ao texto "Title".



	A	B	C	D	E
1	Title	Author	Publisher	ISBN	
2	Harry Potter and the Sorcerer's Sto	J.K. Rowling	Arthur A. Levine	0590353403	
3	The Da Vinci Code (Hardcover)	Dan Brown	DoubleDay; 1st e	0385504209	
4	Cracking The Da Vinci Code: The U	Simon Cox	Barnes & Noble;	0760759316	
5	Illuminating Angels and Demons: Th	Simon Cox	Barnes & Noble;	0760767270	
6	Bunnicula: A Rabbit-Tale of Mystery	Deborah Howe	Aladdin (August	0689806590	
7	Bunnicula Strikes Again! (Paperbac	James Howe	Aladdin (June 1,	0689814623	
8	Red Dragon (Hardcover)	Thomas Harris	Dutton Adult (Ma	0525945563	
9	The Silence of the Lambs (Hannibal	Thomas Harris	St. Martin's Pape	0312924585	
10	I'm OK--You're OK (Paperback)	Thomas Harris	Harper Paperbac	0060724277	
11					
12					
13					
14					
15					
16					
17					
18					
19					
20					

Figura 20 - Exemplo de planilha eletrônica de dados

Os principais programas gerenciadores de planilha do mercado também podem conter outros elementos como imagens, gráficos dinâmicos e *scripts* de programação, além de conexão com banco de dados.

4.1.3. Célula

Uma célula é o principal elemento formador de uma planilha. Cada célula pode armazenar um caractere, uma palavra (conjunto de caracteres), textos (conjuntos de palavras) e dados numéricos (dígitos, percentuais, números inteiros, números com ponto flutuante, datas etc.). É como uma variável, que armazena o um valor inserido pelo(s) usuário(s).

[21] Fonte: http://www.primasoft.com/help/images/import_excel_file.jpg, disponível em 06/09/2014

As células podem conter valores diretos ou indiretos. Valores diretos são elementos simples que guardam por si só a informação desejada. A célula **B1**, da 13, guarda um valor direto – um texto representando a palavra “Author”. Valores indiretos são elementos complexos, que dependem de outros valores (diretos ou indiretos) para representar a informação. Essa dependência é feita através de fórmulas, onde é indicado que uma célula depende de outra(s) através de uma expressão matemática ou textual. A célula **C1**, exibida na Figura 25, indica uma dependência das células **A1** e **A2** feita através de uma expressão matemática. **A1** e **A2** são células cujos valores diretos são somados e armazenados em **C1**.

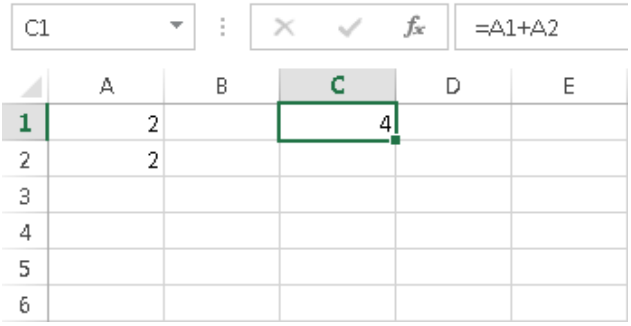


Figura 21 - Exemplo de valor indireto

Os valores armazenados pelas células podem tem significados diferentes em função do tipo da célula. O caractere *1*, por exemplo, pode ser visto como um caractere textual ou numérico, dependendo do tipo de célula que o armazene.

As fórmulas são operações que podem ser aplicadas aos valores das células. Podem manipular facilmente valores textuais ou numéricos e têm uma incrível flexibilidade. Muitos usuários de planilhas optam por seu uso exatamente por contar com toda a riqueza de operações que as fórmulas podem lhes proporcionar.

4.1.4. Linhas e colunas

Considera-se uma linha um conjunto horizontal de células e, analogamente, uma coluna é formada por um conjunto vertical de células. As linhas e as colunas estão limitadas, no escopo deste documento, pela última célula preenchida. Em outras palavras, na Figura 20 quando for feito comentário sobre a linha 5, deve-se considerar apenas os 4 elementos que a compõem, e não todas as células (incluindo as que são vazias). Caso fossem considerados todos os valores possíveis de uma linha, consideraríamos milhares de valores vazios - o que prejudicaria o desempenho dos algoritmos que serão apresentados adiante. O Microsoft

Excel 2010, por exemplo, permite operações com mais de 16.000 colunas e 1.000.000 de linhas[22].

4.1.5. Estrutura de planilha

Uma planilha é, como já dito, essencialmente um conjunto matricial de células. Mas é bastante comum que usuários organizem seus dados nas células das formas mais variadas, sem considerar uma matriz completa (uma matriz *m* x *n* contendo todas as suas *m***n* células preenchidas).

É portanto perfeitamente aceitável que alguém faça uso de uma planilha como a mostrada na Figura 22.

	A	B	C	D	E
1	**Preço do combustível (R$/l)**	R$ 3,199			
2				**Custo total**	
3				R$ 95,97	
4		**Litros abastecido:**	30		
5					
6					
7					
8					
9					
10					

Figura 22 – Planilha não organizada

Uma outra possibilidade é que estes mesmos dados fossem representados de acordo com a planilha da Figura 23.

	A	B	C	D	E
1	**Preço do combustível (R$/l)**	R$ 3,199			
2	**Litros abastecidos**	30			
3	**Custo total**	R$ 95,97			
4					
5					
6					
7					
8					
9					

Figura 23 – Planilha organizada

[22] http://office.microsoft.com/en-001/excel-help/excel-specifications-and-limits-HP010342495.aspx, disponível em 06/09/14

Aparentemente a planilha representada na Figura 22 passa a impressão de estar mais organizada que a representada em Figura 23, mas ambas têm exatamente a mesma informação apresentada. Seria igualmente plausível que outro usuário apresentasse a mesma informação através de colunas e não de linhas. Ou ainda que um outro usuário adicionasse uma linha em branco entre cada linha preenchida - e todas essas possibilidades representam exatamente a mesma informação.

A estrutura de uma planilha pode ser definida, em termos gerais, como sendo a disposição das informações nela armazenada em função da posição de suas células. De forma intuitiva, a estrutura da planilha exibida na Figura 22

4.1.6. Modificação de planilha

Uma planilha pode sofrer ao longo do tempo duas transformações básicas: uma duplicação ou uma alteração.

Uma duplicação é a cópia pura e simples do arquivo original que contêm a planilha. Pode ser feita com o intuito de backup de dados (um usuário copia um arquivo para mantê-lo em segurança) ou com o intuito de réplica de dado: um usuário envia para outro o arquivo original. Um novo arquivo foi gerado, duplicando o original. Seja qual for o intuito do usuário, um novo arquivo foi gerado e passa a ser isolado do original, podendo ser replicado inúmeras outras vezes.

Uma alteração de planilha é considerada nesse escopo como uma alteração de uma célula, o elemento atômico da planilha. Essa alteração pode ser uma substituição da célula (o valor original é substituído por outro completamente diferente) ou uma modificação da célula (o valor original sofreu pequenas alterações, mas ainda se parece com o original). A modificação de uma célula pode ser considerada para comparações envolvendo textos, onde computar a similaridade entre células pode indicar a similaridade total da planilha.

Se uma planilha é essencialmente uma disposição matricial de células, é razoável considerar que uma planilha é uma matriz. Uma métrica de distância de edição de matrizes idealmente deve ser capaz de detectar as seguintes modificações:

- Células: substituição, alteração;
- Linhas e colunas: deleção, inserção, substituição, alteração, transposição.

4.2. Métricas de similaridade de planilhas

Nesta seção serão apresentadas métricas de similaridade de planilhas, considerando a fundamentação teórica e levantamento de literatura apresentados anteriormente.

4.2.1. Restrições

Para validar os conceitos não foram considerados gráficos, macros, etc. Foram consideradas apenas as células e seus valores após terem sido calculados pelas fórmulas. Ou seja, para fins práticos, a célula **A1** como a da imagem Figura 24 e a célula **A6** da Figura 25 são consideradas iguais neste contexto.

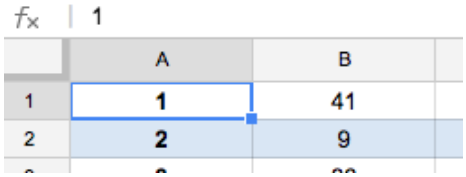


Figura 24 – Exemplo de valor direto

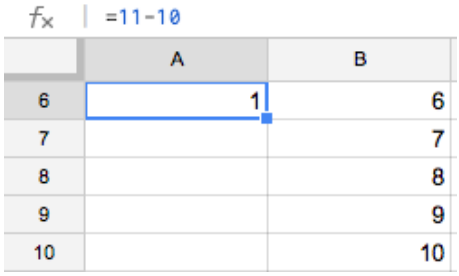


Figura 25 – Exemplo de valor indireto, calculado por fórmula

4.2.2. Comparação MS Excel™ ou de terceiros

A primeira abordagem que pode ser considerada para controle de planilhas é a própria funcionalidade nativa do MS Excel™, principal software editor de planilhas, para controle de versões. Através dele é possível identificar graficamente qual usuário fez qual alteração em que momento.

Outra opção seria usar um dos programas oferecidos por terceiros para comparação par-a-par de planilhas. De forma geral, esses programas oferecem uma funcionalidade de comparação de duas planilhas informadas pelo usuário informando quais células foram alteradas de uma para a outra – uma lógica bastante parecida com o algoritmo de distância de edição.

O problema as duas opções anteriores é que o usuário já deve saber, de antemão, que uma planilha A deu origem a outra planilha B. Para situações em que esta informação já é conhecida (por exemplo, quando um usuário sabe que dois arquivos são parecidos e ele quer apenas destacar as diferenças célula-a-célula) ambas opções são úteis. No entanto, quando não se sabe a priori qual arquivo deu origem a outro ou quando há vários arquivos (e não somente dois), elas não são opções viáveis – uma vez que exigiriam que um usuário realizasse a comparação par-a-par de dezenas, centenas ou até milhares de planilhas.

4.2.3. Damerau-Levenshtein

Mais genérica que a proposta originalmente por Levenshtein, a distância Damerau-Levenshtein (DAMERAU, 1964) considera também a transposição dos dados (o que pode ser útil quando tratarmos de linhas ou colunas ou grupos de células deslocados). Ela calcula o número de alterações necessárias para transformar uma planilha A em uma planilha B. Uma planilha tem suas células lidas sequencialmente de forma a reduzir a dimensionalidade e poder tratar a matriz como uma string. Uma implementação de Damerau-Levenshtein aplicada a matrizes foi proposta por SCHAUERTE (2010). A desvantagem de se realizar a redução de dimensões é que se perde a informação do posicionamento relativo das células. As duas planilhas exibidas abaixo (Figura 26 e Figura 27), por exemplo, poderiam ser consideradas iguais considerando essa métrica.

18	8370	8388	1,7135	5,1249
28	8370	8398	2,345	4,6426
34	8370	8404	3,3096	5,0519
38	8370	8408	3,5197	4,6234

Figura 26 - Exemplo de planilha

18	8370	8388	1,7135	5,1249	28	8370	8398	2,345	4,6426

Figura 27 - Exemplo de planilha

Como este algoritmo foi proposto para cadeias de caracteres, ele leva em consideração parâmetros de uma dimensão. Foi implementado durante esta pesquisa uma métrica de similaridade baseada neste algoritmo. Como as matrizes são bi-dimensionais, primeiro elas são linearizadas através de um método *zigzag*. Em seguida foi aplicado o algoritmo de damerau-levenshtein para comparar as representações lineares das matrizes. Por ter complexidade $O(n^2)$, esse algoritmo não escala bem. Inicialmente concebido para comparação de palavras (que têm tamanho inferior a 20 caracteres), ele não parece ser o mais adequado para trabalhar com matrizes, que costumam ter centenas de células.

4.2.4. ReferenceSet

Durante essa pesquisa, foi tentada a abordagem chamada de *ReferenceSet* (conjunto de referências). Esse método busca comparar duas planilhas quaisquer transformando cada uma delas em um conjunto de referências. As referências são feitas entre pares de células na mesma planilha. Cada célula à esquerda de outra é representada por "CÉLULA1:LEFT:CÉLULA2". Analogamente, cada célula acima de outra é representada por "CÉLULA1:TOP:CÉLULA2". São listadas todas as referências possíveis desses tipos numa mesma planilha e, em seguida, na planilha a ser comparada. As referências são então colocadas em um set (não importa a ordem em que elas apareçam, e podem ocorrer duplicadas). Em seguida, é comparado o set de referências de cada uma das duas planilhas a ser considerada. Para realizar a comparação, pode-se partir de uma abordagem mais trivial (como considerar a interseção dos sets).

O custo computacional e o espaço de armazenamento necessário para esse método são altos, mas as referências podem ser compactadas (há bastante *overhead* textual) e armazenadas em um banco de dados. Para futuras comparações, basta resgatar do banco de dados o conjunto de referências da planilha em questão.

A desvantagem clara é que esse algoritmo dificilmente seria útil para casos práticos, pois seu tempo cresceria exponencialmente em relação ao número de células. Considerando uma planilha com L linhas e C colunas, teria um total de X = L*C células. Como as comparações relativas seriam feitas entre todas as células, a primeira célula seria comparada às (X-1) outras. A segunda célula, às (X-2) restantes e assim por diante. É fácil perceber que esse algoritmo é fatorial e, portanto, seu custo computacional seria de fato um obstáculo para implementação.

4.2.5. DiagonalAlign

Baseada com o RowColAlign, foi desenvolvida nesta pesquisa uma abordagem que visa alinhar as matrizes que representam as planilhas através de suas diagonais principais. Dadas as planilhas A e B, calcula-se três matrizes auxiliares de comparação: rowsM, colsM e cellsM - todas também quadradas de tamanho m x m. Elas representam, nesta ordem, as comparações entre linhas, colunas e célula-a-célula das matrizes A e B.

Cada elemento rowsM(i,j) é o resultado da comparação entre as linhas A(i) e B(j). Duas linhas L1 e L2 quaisquer são comparadas utilizando uma distância de edição de strings (onde cada célula é análoga a um caractere). Cada elemento colsM(i,j) é o resultado da comparação entre as colunas A(i) e B(j), onde a comparação é semelhante à feita com as

linhas. Finalmente, cada elemento cellsM(i,j) é o resultado da comparação direta entre A(i,j) e B(i,j): se A(i,j) for igual B(i,j), então cellsM(i,j) = 0; caso contrário, cellsM(i,j) = 1.

Tanto rowsM quanto colsM guarda uma propriedade interessante: caso A seja igual a B, as diagonais principais de rowsM e colsM terão seus valores iguais a 0. Logo, a distância de edição final das planilhas A e B pode ser dada em função da comparação do traço (soma da diagonal principal) dessas duas matrizes. A vantagem em relação ao RowColAlign é que é possível obter o número de edições necessárias para se transformar A em B. Como exemplo, consideremos as matrizes exibidas na Figura 28 e na Figura 29.

1	2	3
4	5	6
7	8	9

Figura 28 - Matriz X

1	2	3
4	5	6
0	0	0

Figura 29 - Matriz Y

A distância de edição delas seria dada pelas 3 matrizes ilustradas na

Figura *30*, onde foram destacados os valores das diagonais principais de rowsM e colsM.

Tabela 2- rowsM

0	3	3
3	**0**	3
3	3	**3**

Tabela 3 - colsM

1	3	3
3	**1**	3
3	3	**1**

Tabela 4 – cellsM

0	0	0
0	0	0
1	1	1

Figura 30 - Distância de edição entre X e Y

As tabelas acima podem ser lidas também como:

- Foram necessárias 3 modificações em uma única linha ou
- Foram necessárias 3 modificações de coluna, 1 em cada uma das 3 colunas ou
- Foram necessárias 3 modificações isoladas de célula.

Essas informações nos levam para um grafo de alterações de planilha, como o exibido na Figura 15. Basta aplicar um algoritmo de menor caminho deste grafo para se indicar qual a melhor forma de indicar a distância de edição das planilhas X e Y.

A desvantagem é deste algoritmo é o fato dele somente trabalhar com matrizes quadradas de mesmo tamanho.

4.2.6. DSJ – Dicionário de similaridade de Jaccard

Foi também desenvolvido como resultado da pesquisa o método de Dicionário de similaridade de Jaccard – DSJ. Esse método retorna uma similaridade entre duas planilhas analisando tanto os valores quanto as estruturas delas através do coeficiente de Jaccard aplicado a dicionários que armazenam os dados. O algoritmo é dado a seguir.

Inicialmente a similaridade é nula. Cada planilha dá origem a um dicionário, onde as chaves são os dados encontrados na planilha e os valores são listas contendo todas as posições daquele dado naquela planilha. Caso um dado de uma planilha não exista na outra planilha, a similaridade entre eles é nula. Caso as chaves dos dicionários sejam encontradas em ambas as planilhas, seus valores (as listas com as posições) são comparados através do coeficiente de Jaccard.

Em cada iteração, a similaridade é acrescida do valor da similaridade calculada pelo coeficiente de Jaccard entre as duas listas. Como o coeficiente é um número real no intervalo (0, 1), o máximo valor de similaridade entre duas listas quaisquer é 1 (quando elas são totalmente iguais).

Logo, caso uma planilha A seja exatamente igual a uma planilha B, a similaridade final será igual a N * 1 = N, onde N é o número de elementos (células) na planilha. Como forma de ponderar as similaridades locais entre as listas, é calculada a similaridade final como sendo a a razão entre a similaridade total (soma das similaridades locais) e N.

Esta abordagem simples é bastante eficiente e seu pseudo-algoritmo é dado pelo texto na Figura 31.

```
Dadas 2 planilhas: planilha1 e planilha2;
Início
        similaridade = 0;
        dicionario1 <- ExtrairDicionario(planilha1);
        dicionario2 <- ExtrairDicionario(planilha2);
        Para cada chave em união(dicionario1, dicionário2):
               Se chave existe em dicionario1 e existe em dicionario2:
                      lista1 = dicionario1(chave);
                      lista2 = dicionario2(chave);
                      similaridade += jaccard(lista1, lista2);
        similaridade_maxima = tamanho(união(dicionario1, dicionário2));
        similaridade = similaridade/similaridade_maxima;
 Fim.
```

Figura 31 - Pseudoalgoritmo do algoritmo "Similaridade por dicionários de Jaccard"

Como exemplo, tomemos as matrizes X e Y exibidas nas Figura 28 e na Figura 29. Os dicionários criados seriam:

- Dicionário 1
  - { 1: [(1,1)]; 2: [(1,2)]; 3: [(1,3)]; 4: [(2,1)]; 5: [(2,2)]; 6: [(2,3)]; 7: [(3,1)]; 8: [(3,2)]; 9: [(3,3)]; }
- Dicionário 2
  - { 1: [(1,1)]; 2: [(1,2)]; 3: [(1,3)]; 4: [(2,1)]; 5: [(2,2)]; 6: [(2,3)]; 0: [(3,1), (3,2), (3,3)]; }

A similaridade computada entre X e Y é igual a 0.60, pois existem 10 chaves diferentes somando os dois dicionários e a similaridade par-a-par entre as chaves é máxima (1) em seis deles e mínima (0) em quatro, como pode ser visto na Tabela 5.

Tabela 5 - Similaridade dos dicionários

Chave	Similaridade dos valores
0	0
1	1
2	1
3	1
4	1

5	1
6	1
7	0
8	0
9	0

4.2.7. LCS – Linear-Column Similarity

Foi elaborada também uma abordagem de linearização das planilhas utilizando as colunas. Isso quer dizer que as planilhas avaliadas tiveram sua dimensão reduzida e tiveram comparadas suas representações lineares. A redução de dimensão escolhida para esta abordagem foi a de linearização de colunas, onde uma matriz X de tamanho N x M é representada linearmente por um vetor de mesmo tamanho, mas cujos elementos são dispostos através da primeira coluna de X, seguidos da segunda coluna de X e assim por diante.

Com essa abordagem de linearização, é possível aplicar algoritmos de comparação de *strings*, pois as representações matriciais têm apenas 1 dimensão. Foram utilizadas três métricas de comparação, todas com implementação padrão do software Matlab:

- Coseno;
- Hamming;
- Jaccard.

As métricas de Coseno e Jaccard já foram explicadas anteriormente. A métrica de Hamming calcula a distância de edição somente de substituições de elementos em uma cadeia, e não leva em consideração adições ou remoções. Portanto, ele é limitado a casos onde os vetores sejam de mesmo tamanho.

4.3. Avaliação

Nessa seção será apresentada a avaliação das abordagens apresentadas na seção anterior. A avaliação principal tem como objetivo validar o algoritmo DSJ, desenvolvido nesta pesquisa, e compará-lo com o RowColAlign[23], algoritmo já publicado. O algoritmo DiagonalAlign não pôde ser utilizado para os testes uma vez que ele tem a restrição de

[23] Como o RowColAlign é uma evolução do SheetDiff, não foi incluído o SheetDiff nas comparações.

apenas comparar matrizes de tamanhos iguais – o que não é verdade para a maioria das planilhas analisadas.

4.3.1. Dataset utilizado

A avaliação foi feita utilizando o mesmo dataset utilizado por Harutyunyan (2012), um subset do EUSES, proposto por Fisher (2005). Com mais de 5.600 itens, o dataset original conta com muitos arquivos inválidos (que não podem ser lidos sem senha e/ou com falha de leitura). Os arquivos inválidos foram eliminados e o subset resultante contém 3.620 arquivos com planilhas de tamanhos variados que chegam até 6,5 milhões de células. A distribuição cumulativa das planilhas[24] considerando seu número de células pode ser vista na Figura 32.

Pode-se observar que a distribuição do subset contém 2.135 planilhas com até 1.000 células (58,98% do total) e 3.361 (92,85% do total) com até 10.000. Como apresentado no capítulo 2, a maior parte das planilhas (61%) contém 1.000 células ou menos e quase todas (89%), menos de 10.000. Logo, é razoável assumir que esse conjunto, por possuir aproximadamente a mesma distribuição de planilhas do mundo real, é uma boa amostra da população total de planilhas existentes.

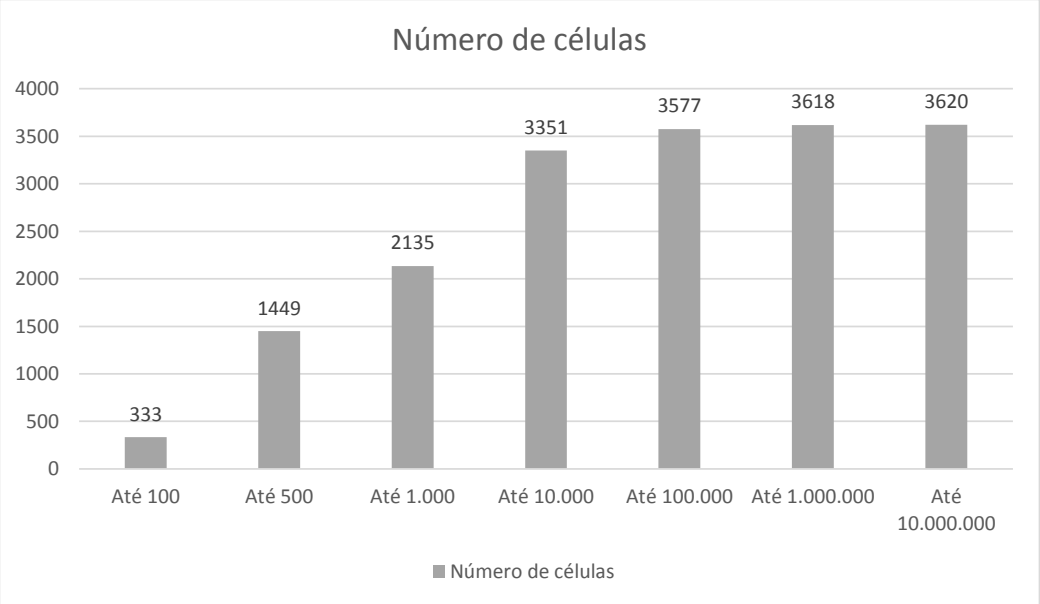


Figura 32 - Distribuição cumulativa das planilhas

4.3.2. Implementação

[24] Foram desconsideradas planilhas vazias ou com apenas 1 célula preenchida

Como o RowColAlign foi implementado em Matlab[25], a implementação dos algoritmos apresentados também fez uso da mesma ferramenta. Durante o desenvolvimento dos algoritmos, foram selecionados aleatoriamente diversos arquivos do subset descrito anteriormente apenas como forma de validar as implementações. Como plataforma de execução dos testes foi escolhido um *notebook* padrão de mercado com processador Intel Core i5 @2.3GHz, 6GB RAM, Windows 64-bit.

4.3.3. Avaliação de tempo

Foram selecionados aleatoriamente 100 arquivos (28% do total de 3.620 arquivos) e extraídas suas planilhas. As planilhas com menos de 10 células foram eliminadas por serem consideradas muito pequenas e não relevantes para a avaliação. Foram gerados 3.969 pares de planilhas e cada um desses pares foi analisado pelos algoritmos listados abaixo. Cada algoritmo foi executado 4 vezes para cada par e foi registrada a média temporal dessas 4 iterações.

O resultado da avaliação foi consolidado nos gráficos ilustrados a seguir (Figura 33 e Figura 34), que mostram a média (em segundos) de cada algoritmo para analisar o par de planilhas em função do número células nas duas planilhas.

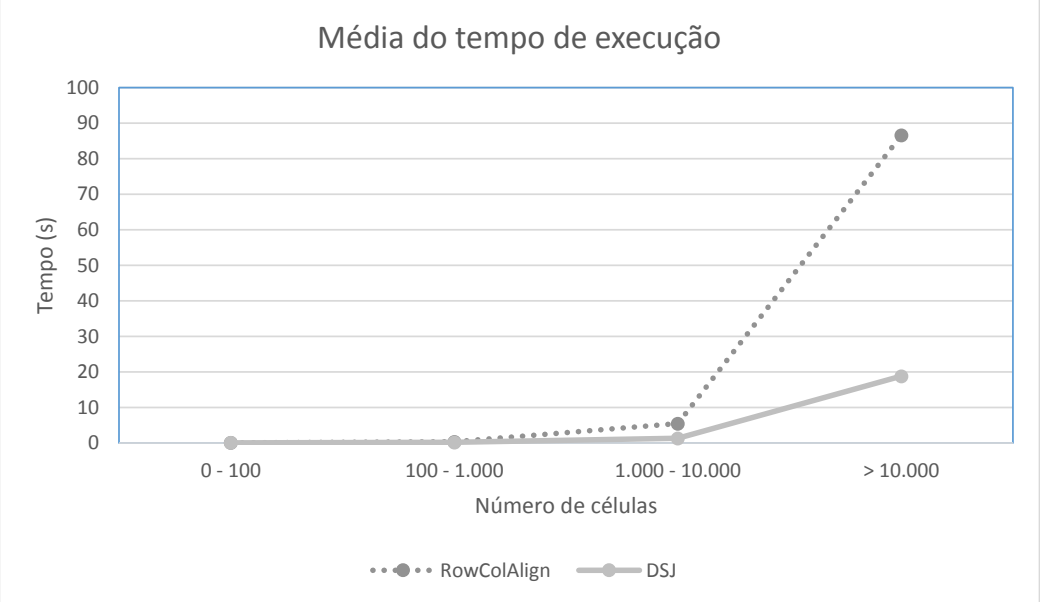


Figura 33 - Média do tempo de execução dos algoritmos

[25] http://www.mathworks.com/products/matlab/, disponível em 06/09/2014

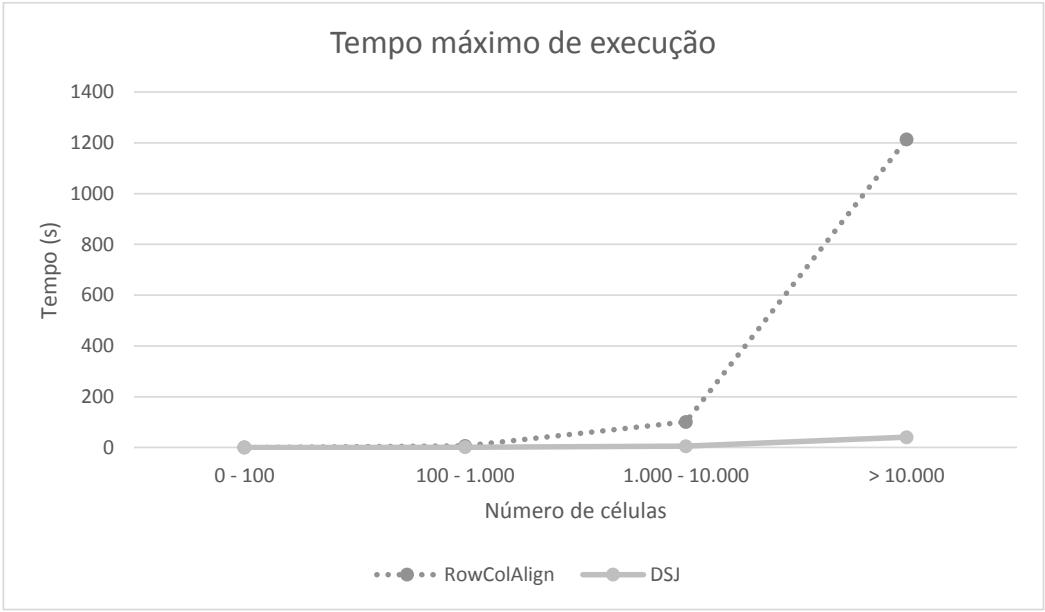


Figura 34 - Tempo máximo de execução dos algoritmos

O algoritmo proposto (DSJ) é na maior parte das vezes mais eficiente que o RowColAlign. Os casos em que o RowColAlign é mais eficiente são ocasionados pela disparidade extrema entre o número total de células das duas planilhas analisadas. Ou seja, se duas planilhas possuem números de células muito diferentes (por exemplo, uma com 20 células e outra com 8.000), o RowColAlign é executado mais rapidamente. No entanto, é importante ressaltar conforme os números de células crescem, o RowColAlign perde rapidamente sua vantagem.

Tomando como exemplo o conjunto de comparações abaixo (Tabela 6), onde é comparada uma planilha de 8.370 células com outras planilhas que variam de 18 a 1.408 células, observamos que nos primeiros casos (onde a planilha menor tem entre 18 e 38 células) o RowColAlign roda quase 50% mais rapidamente que o DSJ. Porém, basta que o tamanho da planilha menor passe de 100 células para que o DSJ seja significativamente mais rápido. Nas últimas comparações, onde as planilhas têm 1.408 e 8.370 células, o RowColAlign roda quase 22 vezes mais devagar que o DSJ.

Tabela 6 - Conjunto de medidas de tempo de execução

Células na planilha 1	Células na planilha 2	Soma de células	Tempo de execução	
			RowColAlign	DSJ
18	8370	8388	1,7135	5,1249
28	8370	8398	2,345	4,6426
34	8370	8404	3,3096	5,0519
38	8370	8408	3,5197	4,6234

112	8370	8482	8,7181	4,518
140	8370	8510	7,5801	3,0848
220	8370	8590	19,48	5,727
231	8370	8601	19,263	4,7566
568	8370	8938	51,582	5,6791
918	8370	9288	83,138	6,4112
950	8370	9320	71,632	5,3519
1408	8370	9778	105,88	4,9033

O comportamento dos algoritmos fica mais claro ao se analisar a Figura 35, que mostra que o tempo de execução do RowColAlign cresce rapidamente em função do número de células da planilha 1 (fixado o número de células da planilha 2), enquanto o tempo de execução do DSJ permanece quase constante.

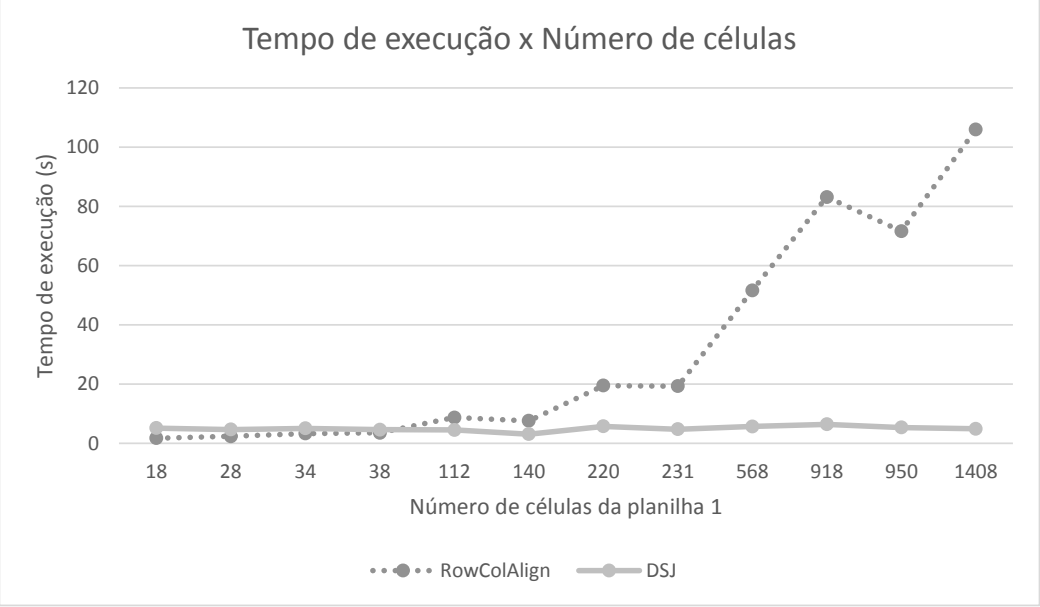


Figura 35 - Tempo de execução x Número de células

4.3.4. Avaliação de similaridade

Para analisar o quesito similaridade, foi elaborado outro conjunto de testes. Esse segundo conjunto tinha como objetivo comparar unicamente as similaridades computadas pelos algoritmos selecionados.

Para realizar esta primeira análise o ambiente de testes foi composto por 4 planilhas com as seguintes características:

- Planilha 1: Original[26] com 9.278 células;

[26] Planilha com dados históricos de preço de gasolina nos Estados Unidos.

- Planilha 2: Planilha 1 com algumas (10% ou menos) modificações (linhas removidas, conjuntos de dados alterados);
- Planilha 3: Planilha 1 com muitas modificações (linhas removidas, conjuntos de dados alterados);
- Planilha 4: Planilha[27] distinta, com 55 células, sem qualquer relação com a planilha 1;

Foram então comparadas todas as 4 planilhas, par a par, com cada um dos algoritmos. O valor computado de similaridade entre cada par de planilhas varia de 0 (similaridade mínima) a 1 (similaridade máxima). Ou seja, dadas duas planilhas A e B, o valor de similaridade entre elas será mais próximo de 0 quando A e B forem diferentes e se aproximará de 1 quando A e B forem parecidas. Os resultados dos testes são apresentados a seguir.

Tabela 7 - Testes de Similaridade com RowColAlign até 10.000 células

Similaridade – RowColAlign				
	Planilha 1	**Planilha 2**	**Planilha 3**	**Planilha 4**
Planilha 1	1	0,9978	0,9784	NaN
Planilha 2	-	1	0,9808	NaN
Planilha 3	-	-	1	NaN
Planilha 4	-	-	-	1

Tabela 8 - Testes de Similaridade com DSJ até 10.000 células

Similaridade – DSJ				
	Planilha 1	**Planilha 2**	**Planilha 3**	**Planilha 4**
Planilha 1	1	0,9846	0,7721	1,23E-02
Planilha 2	-	1	0,7851	1,23E-02

[27] Planilha com relatório financeiro de um departamento de empresa francesa.

Planilha 3	-	-	1	1,45E-02
Planilha 4	-	-	-	1

Cada célula da planilha mostra a similaridade da planilha representada naquela linha com a planilha representada naquela coluna. Como os algoritmos são simétricos, só foram exibidos os valores da metade superior da planilha. As diagonais principais das matrizes de similaridade são iguais a 1 pois as planilhas são comparadas com elas mesmas - e logicamente a semelhança é a máxima possível.

Os testes apresentados (Tabela 7 e

Tabela 8) indicam o esperado de ambos algoritmos:

- Em planilhas iguais (no caso das diagonais), a similaridade é a máxima (igual a 1);
- Em planilhas muito parecidas (planilhas 1 e 2), a similaridade é muito alta;
- Em planilhas originadas da mesma fonte, mas razoavelmente alteradas (planilhas 1 e 3; planilhas 2 e 3), a similaridade é significante, porém menor que no caso anterior;
- Finalmente, no caso de planilhas completamente distintas (planilha 4 e todas as outras), a similaridade é muito pequena, praticamente desprezível.

É interessante notar ainda que o RowColAlign não resulta um valor pequeno para similaridades baixas. Nesse caso, ele retorna um valor *NaN* (*Not a Number* - tipo de dado do Matlab que indica número indefinido). Esse comportamento não é algo desejável, uma vez que não deixa claro o real valor resultante. Além disso, o valor de similaridade entre os pares (Planilha 1, Planilha 3) e (Planilha 2, Planilha 3) parecem ser muito altos para refletir a real mudança aplicada a estes arquivos – refletida de fato pelo DSJ.

Uma outra rodada de testes foi realizada, nos mesmos moldes que o anterior, com planilhas menores, para comparar as similaridades obtidas ao analisar diferentes tamanhos. Para realizar esta análise o ambiente de testes foi composto por 4 planilhas com as seguintes características:

- Planilha 1: Original[28] com 334 células;
- Planilha 2: Planilha 1 com algumas (10% ou menos) modificações (linhas removidas, conjuntos de dados alterados);
- Planilha 3: Planilha 1 com muitas modificações (linhas removidas, conjuntos de dados alterados);
- Planilha 4: Planilha[29] distinta, com 199 células, sem qualquer relação com a planilha 1;

Os resultados dos testes são apresentados a seguir.

Tabela 9 - Testes de Similaridade com RowColAlign até 1.000 células

Similaridade – RowColAlign – Até 1.000 células				
	Planilha 1	**Planilha 2**	**Planilha 3**	**Planilha 4**
Planilha 1	1	0,9303	1	NaN
Planilha 2	-	1	1	NaN
Planilha 3	-	-	1	NaN
Planilha 4	-	-	-	1

Tabela 10 - Testes de Similaridade com DSJ até 1.000 células

Similaridade – DSJ – Até 1.000 células				
	Planilha 1	**Planilha 2**	**Planilha 3**	**Planilha 4**
Planilha 1	1	0,6627	0,0455	0
Planilha 2	-	1	0,0376	0
Planilha 3	-	-	1	0
Planilha 4	-	-	-	1

Desta vez, os resultados (Tabela 9 e Tabela 10) mostraram-se diferentes dos anteriores, como mostram as avaliações abaixo:

- Em planilhas iguais (no caso das diagonais), a similaridade é a máxima (igual a 1);

[28] Planilha de controle de programas utilizados em um laboratório de software.
[29] Planilha de notas de alunos.

- Em planilhas muito parecidas (planilhas 1 e 2), a similaridade é muito alta, mas o algoritmo DSJ mostrou-se mais sensível às alterações;
- Em planilhas originadas da mesma fonte, mas razoavelmente alteradas (planilhas 1 e 3; planilhas 2 e 3), o RowColAlign falhou (pois a similaridade obtida foi máxima) e o DSJ apresentou-se mais sensível às alterações;
- Finalmente, no caso de planilhas completamente distintas (planilha 4 e todas as outras), a similaridade obtida pelo DSJ é 0, e o RowColAlign obteve novamente valores NaN.

Finalizando, foram executados o mesmo raciocínio de testes para planilhas com até 100 células. Seguem as características das planilhas e os resultados dos testes:

- Planilha 1: Original[30] com 55 células;
- Planilha 2: Planilha 1 com algumas (10% ou menos) modificações (linhas removidas, conjuntos de dados alterados);
- Planilha 3: Planilha 1 com muitas modificações (linhas removidas, conjuntos de dados alterados);
- Planilha 4: Planilha[31] distinta, com 37 células, sem qualquer relação com a planilha 1;

Tabela 11 - Testes de Similaridade com RowColAlign até 100 células

Similaridade – RowColAlign – Até 100 células				
	Planilha 1	**Planilha 2**	**Planilha 3**	**Planilha 4**
Planilha 1	1	0,944	0,9306	NaN
Planilha 2	-	1	0,9524	NaN
Planilha 3	-	-	1	NaN
Planilha 4	-	-	-	1

Tabela 12 - Testes de Similaridade com DSJ até 100 células

Similaridade – DSJ – Até 100 células				
	Planilha 1	**Planilha 2**	**Planilha 3**	**Planilha 4**
Planilha 1	1	0,3731	0,7094	3,40E-03
Planilha 2	-	1	0,0332	4,50E-02

[30] Planilha com formulário de dados de projeto de um banco.
[31] Planilha com dados sobre o aquecimento global.

Planilha 3	-	-	1	3,50E-03
Planilha 4	-	-	-	1

E, finalmente, segue a análise dos resultados dos testes feitos com planilhas de até 100 células (Tabela 11 e Figura 12):

- Em planilhas muito parecidas (planilhas 1 e 2), a similaridade é muito alta, mas o algoritmo DSJ mostrou-se mais sensível às alterações;
- Em planilhas originadas da mesma fonte, mas razoavelmente alteradas (planilhas 1 e 3; planilhas 2 e 3), o RowColAlign apresentou como o esperado (pois a similaridade obtida foi considerável, mas não tão alta) e o DSJ falhou (pois apresentou similaridade muito baixa pro esperado);
- Finalmente, no caso de planilhas completamente distintas (planilha 4 e todas as outras), o DSJ apresentou similaridade muito pequena, praticamente desprezível (como o esperado), enquanto o RowColAlign retornou valores NaN.

4.3.5. Avaliação de comportamento

Foi também realizado um conjunto de testes com o objetivo de analisar o comportamento dos algoritmos em relação a um conjunto maior de planilhas contendo as planilhas originais e variações intencionalmente modificadas. Foi selecionado um conjunto de 62 planilhas a partir do dataset, cujos tamanhos (número de linhas x número de colunas) são ilustrados na Figura 37. As planilhas possuem uma quantidade de células com uma distribuição cumulativa (Figura 36) razoavelmente similar à de todo dataset (que foi ilustrada na Figura 32), com maioria das planilhas tendo até 1.000 células.

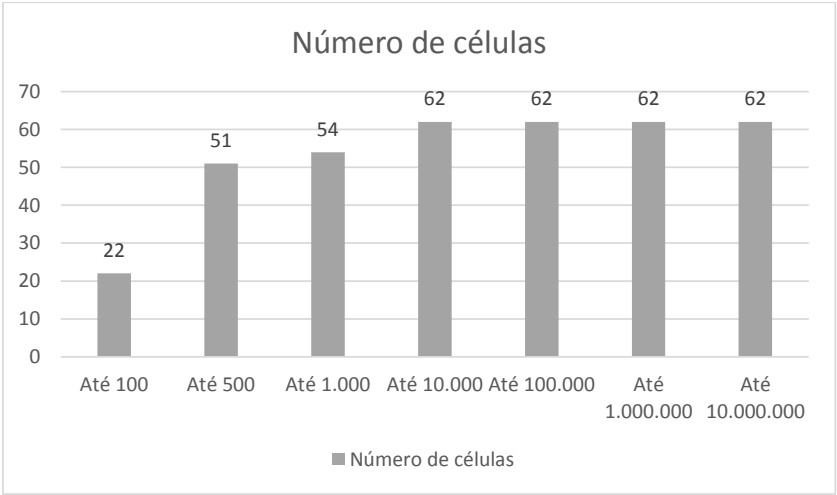


Figura 36 - Número de células por planilha

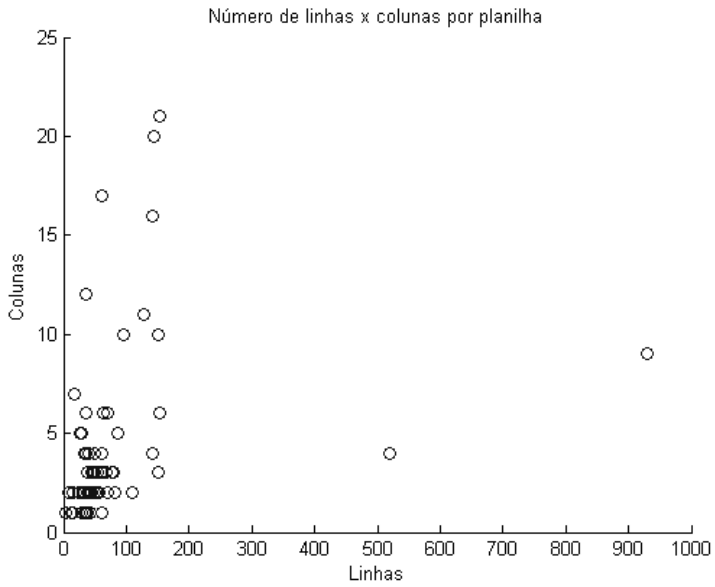


Figura 37 - Linhas x Colunas nas planilhas

Para cada uma dessas planilhas foi gerado um conjunto composto de quatro outras planilhas, cada um com N ($10 < N < 50$) alterações:

1. RCE (Random Cell Edits). Uma com N edições aleatórias em células: N células aleatórias têm seus valores alterados para um outro valor aleatório;
2. RLE (Random Line Edits). Uma com N edições aleatórias em linhas: N linhas aleatórias tinham todos seus valores alterados para outros valores aleatórios;
3. ROE (Random cOlumn Edits). Uma com N edições aleatórias em colunas: N colunas aleatórias tinham todos seus valores alterados para outros valores aleatórios;
4. RLOE (Random Line/cOlumn Edits). Uma com N edições aleatórias em linhas ou colunas: N linhas ou colunas aleatórias tinham todos seus valores alterados para outros valores aleatórios;

Foram gerados 5 conjuntos destes com N assumindo os valores 10, 20, 30, 40 e 50. Ou seja, para cada uma das planilhas selecionadas, foi gerado um grupo com 10 edições aleatórias, em células, outro com 10 edições aleatórias em linhas, outro com 10 edições aleatórias em colunas e outro com 10 edições aleatórias em linhas ou colunas. Em seguida, foram gerados os mesmos grupos com 20 edições, 30 edições, 40 edições e 50 edições para cada um dos tipos de alterações.

Para cada planilha, foi computada a similaridade entre ela e cada uma das planilhas modificadas utilizando os algoritmos DSJ, RowColAlign, LCS-Cosine, LCS-Hamming e LCS-

Jaccard. Uma planilha maior, com mais células, deve ter similaridade alta quando feitas poucas alterações. Uma menor, no entanto, teria um valor de similaridade relativamente menor. Conforme o número de alterações aumenta (isso é, caminha-se para a direita no eixo horizontal), o valor de similaridade diminui. As planilhas com mais células (círculos maiores) descem mais suavemente, enquanto as planilhas com menos células (círculos menores) descem mais rapidamente. Os resultados são exibidos nos gráficos do tipo *scatter* nos apêndices deste trabalho. O tamanho dos círculos indica o tamanho de cada planilha.

Para os algoritmos DSJ, LCS-Jaccard e LCS-Hamming, de fato é o que acontece, principalmente para as modificações do tipo RCE. Esse efeito nas modificações RLE também ocorre, mas de forma mais sutil. Nas dos tipos ROE e RLOE esse efeito não é percebido, uma vez que as similaridades caem muito mais rapidamente. Um possível motivo para esse comportamento é a forma como as planilhas foram linearizadas – através das colunas. Sendo assim, as modificações em colunas acabam criando um efeito maior no cálculo da similaridade.

Para o algoritmo LCS-Cosine, o comportamento é ligeiramente mais lento que os anteriores, mas ele também comporta-se como o esperado (indicando que ele é menos sensível às modificações).

Já o algoritmo RCA mostrou-se quase indiferente às alterações RLE, ROE e RLOE, computando similaridade quase máxima para maioria dos casos. Apenas para os casos RCE este algoritmo apresentou o comportamento dentro do previsto.

Como forma de consolidar os resultados e facilitar a representação, pode-se observar na Figura 38, Figura 39, Figura 40, Figura 41 e na Figura 42 como cada algoritmo é mais ou menos sensível às edições feitas. Os gráficos de linha indicam no eixo horizontal o número de alterações feitas nas planilhas e no eixo vertical, o valor médio de similaridade computado entre a matriz original e a matrizes modificadas, com o erro padrão indicado pelos marcadores.

Como observado, os algoritmos DSJ, LCS-Hamming e LCS-Jaccard caem ao longo do tempo de forma suave, sendo DSJ e LCS-Jaccard os que têm menor desvio padrão. Os algoritmos RCA e LCS-Cosine caem de forma muito mais lenta, indicando serem menos sensíveis às alterações feitas.

Uma conclusão possível é que os algoritmos LCS-Cosine e RCA não ilustram a similaridade de planilhas modificadas adequadamente, uma vez que planilhas que sofreram muitas modificações tiveram similaridade muito alta quando comparadas com as planilhas originais.

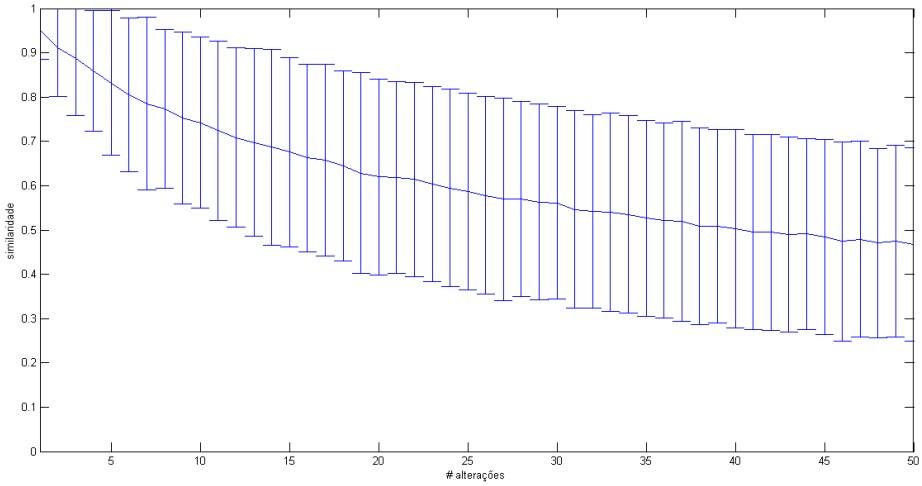


Figura 38 - Comportamento DSJ

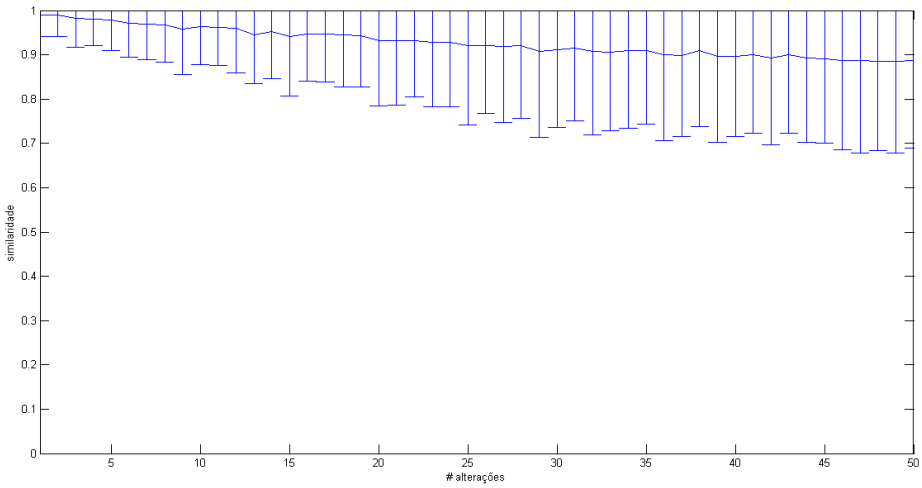


Figura 39 - Comportamento LCS-cosine

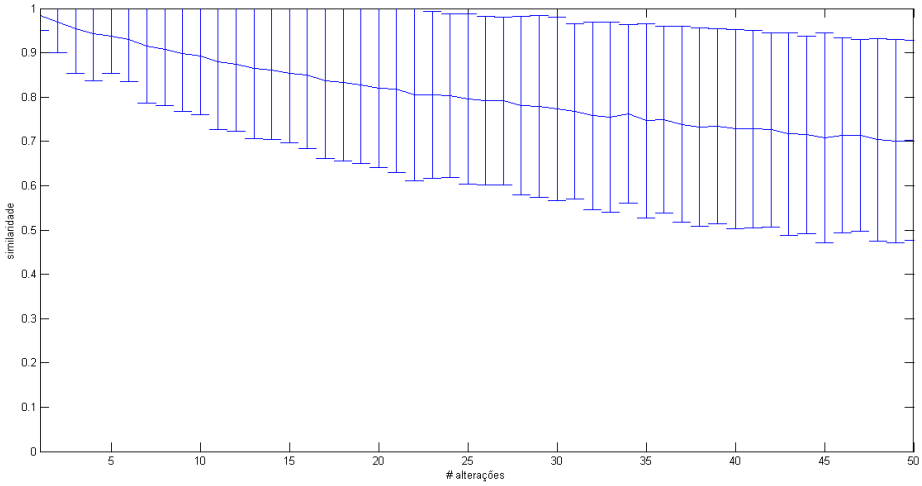


Figura 40 - Comportamento LCS-Hamming

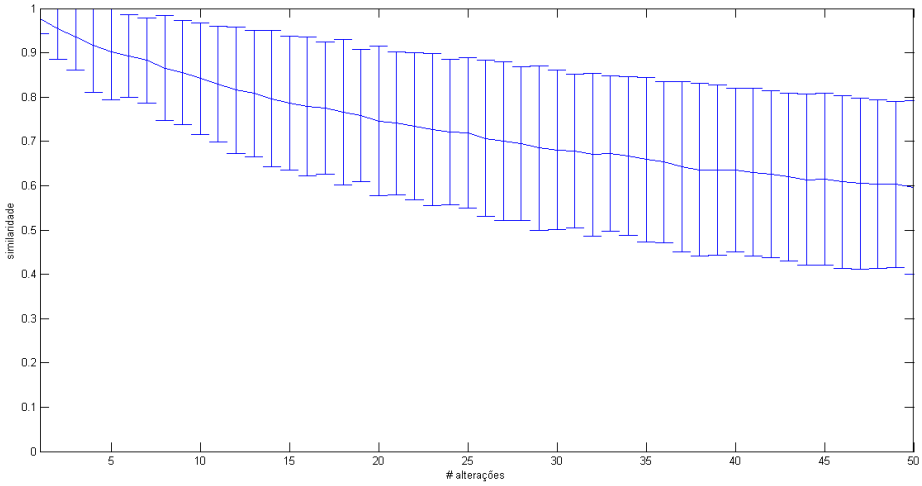


Figura 41 - Comportamento LCS-Jaccard

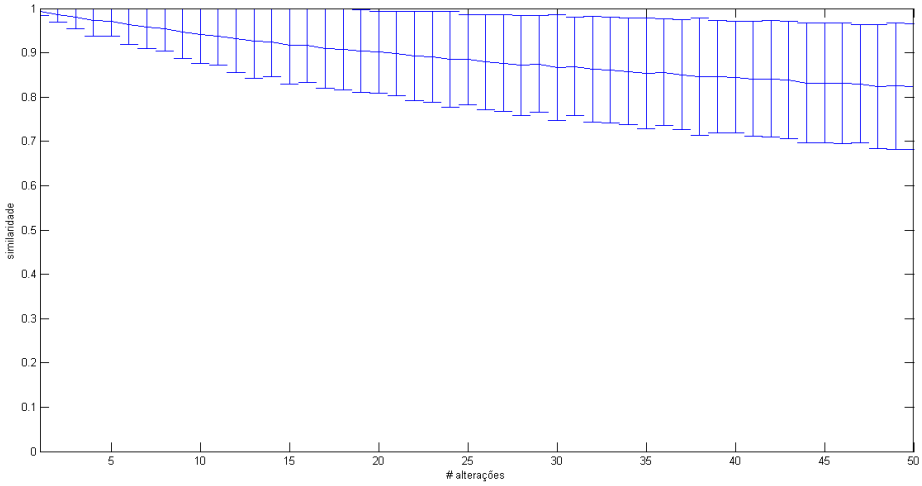


Figura 42 - Comportamento RCA

4.4. Considerações finais

Neste capítulo foram apresentadas as definições necessárias ao entendimento dos conceitos que se seguiram, como o ciclo de vida de uma planilha.

De posse das definições e dos conceitos, foram apresentadas algumas abordagens de comparações de planilhas como resultado da pesquisa efetuada, com destaque para o algoritmo DSJ para comparação com um algoritmo já publicado anteriormente, o RowColAlign. As comparações levaram em consideração o tempo de execução e os resultados de similaridade obtidos com os dois algoritmos.

Em relação ao tempo de execução, o DSJ foi consideravelmente melhor que RowColAlign na maior parte dos casos. Apenas para casos onde os números de células são muito pequenos, o RowColAlign mostra-se mais rápido.

Em relação aos resultados de similaridade, foram feitas comparações com conjuntos de planilhas com até 10.000, até 1.000 e até 100 células – tamanhos de amostra balizadas pelo levantamento no capítulo 2. Na maior parte das comparações o DSJ mostrou um comportamento melhor do que o RowColAlign, uma vez que o DSJ não retornou similaridades máximas em caso de planilhas distintas, não retornou um valor dúbio (NaN, como o RowColAlign) e retornou um valor de similaridade menor para o caso de planilhas com um razoável número de alterações. O DSJ, no entanto, mostrou-se muito sensível a alterações

feitas em planilhas pequenas e médias (até 100 e 1.000 células, respectivamente), e a similaridade resultante foi muito mais baixa do que o RowColAlign. De uma forma geral, pode-se considerar que o DSJ mostrou-se mais rápido e melhor que o RowColAlign.

Finalmente, foi elaborado nesta pesquisa um conjunto de testes para analisar o comportamento dos algoritmos aqui apresentados (LCS-Cosine, LCS-Hamming, LCS-Jaccard e DSJ) em comparação com outro já publicado (o RowColAlign). Esse experimento indicou que os algoritmos comparados comportam-se como o esperado (a similaridade média cai continuamente quando o número de alterações aumenta), mas que LCS-Cosine e RCA são algoritmos pouco sensíveis às edições de células enquanto os outros algoritmos apresentaram curvas suaves de decaimento.

Capítulo 5 - Conclusão

O objetivo deste capítulo é revisar o que foi abordado ao longo do texto desta dissertação, apresentar as contribuições feitas à área de pesquisa acadêmica e listar os trabalhos futuros a serem feitos como continuidade.

5.1. Revisão do problema e da proposta

No primeiro capítulo o leitor foi contextualizado com a história da criação e evolução dos programas de computador utilizados para manipular as planilhas eletrônicas, a importância desse tipo de software para o mercado – e para a história da computação.

Foi apresentado no segundo capítulo o cenário atual do uso de planilhas eletrônicas por usuários de computadores pessoais, com destaque para os usuários corporativos. De posse de estatísticas de uso das planilhas que guiaram o escopo do trabalho e de trabalhos correlatos que foram listados, o leitor foi inserido no estado da arte da pesquisa relacionada ao tema. Foram também listados alguns tipos de erros e falhas causados pela má utilização de planilhas no ambiente de trabalho, visto que os arquivos sofrem diversas modificações e são replicados de forma descontrolada. Listou-se algoritmos já usados na literatura para comparação de documentos, grafos e sua possível aplicação no contexto dos dados tabulares.

No capítulo seguinte foi apresentada, em alto nível, uma arquitetura de sistema que tem como principais objetivos a conversão de planilhas para o modelo relacional e a integração delas em um sistema de informação baseado na web (de forma a facilitar o uso colaborativo dos dados). Fazer a conversão das planilhas de dados para um sistema de informação baseado em bancos de dados relacionais é uma das abordagens mais aceitas para se evitar a inserção desregrada de dados e para se manter o controle de acesso de usuários aos dados armazenados.

No quarto capítulo foi explorado o desafio de se realizar a comparação de planilhas: foi apresentada uma abordagem para cálculo de similaridade de arquivos de planilhas, bem como foi estabelecido um *benchmark* com outro algoritmo já publicado.

5.2. Contribuições

Dentre as contribuições resultantes desta pesquisa, destacam-se:

- A proposta conceitual de uma arquitetura de migração e controle de planilhas para um sistema de informação baseado na web;
- Levantamento de técnicas de comparação existentes e sua aplicação às planilhas;
- Apresentação de abordagens de comparação de planilhas, incluindo melhoria em relação um algoritmo já existente;
- Uma visualização de ciclo de vida de planilhas utilizando a similaridade entre elas;
- O resultado de testes de comparação de novos algoritmos com o estado da arte atual.

5.3. Limitações e trabalhos futuros

Por uma questão de escopo, foi definido que a arquitetura proposta seria limitada ao seu projeto conceitual, e que apenas o módulo de comparação e evolução de planilhas seria explorado. Por esta razão foi implementada apenas uma prova de conceito dela, sem validade prática.

Como trabalhos futuros, pode-se listar:

- A implementação completa da arquitetura proposta no capítulo 3, testes e a validação da mesma em um ambiente real (uma empresa que faça uso de planilhas de forma desregrada, por exemplo).
- Adaptação ao DiagonalAlign para que ele passe a considerar planilhas de tamanhos variados;
- Melhorias ao DSJ: pode-se considerar a comparação das listas de posições ocupadas pelas células. Dessa forma, será possível ponderar as similaridades entre valores e conteúdo
- Novas abordagens de comparação de planilhas: novas métricas e novos conceitos podem ser adicionados ao que foi proposto de forma a tornar o algoritmo mais robusto;

Capítulo 6 - Referências bibliográficas

Abraham, R., Erwig, M., Kollmansberger, S., & Seifert, E. (2005, Setembro). Visual specifications of correct spreadsheets. In *Visual Languages and Human-Centric Computing, 2005 IEEE Symposium on* (pp. 189-196). IEEE.

Ahmad, Y., Antoniu, T., Goldwater, S., & Krishnamurthi, S. (2003, Outubro). A type system for statically detecting spreadsheet errors. In *Automated Software Engineering, 2003. Proceedings. 18th IEEE International Conference on* (pp. 174-183). IEEE.

Arslan, A. N. (2007, January). A largest common d-dimensional subsequence of two d-dimensional strings. In *Fundamentals of Computation Theory* (pp. 40-51). Springer Berlin Heidelberg.

Aumueller, D., Do, H. H., Massmann, S., & Rahm, E. (2005, Junho). Schema and ontology matching with COMA++. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data* (pp. 906-908). ACM.

Baeza-Yates, R. A. (1998). Similarity in two-dimensional strings. In*Computing and Combinatorics* (pp. 319-328). Springer Berlin Heidelberg.

Baeza-Yates, R., & Ribeiro-Neto, B. (1999). *Modern information retrieval* (Vol. 463). New York: ACM press.

Baker, K. R., Foster-Johnson, L., Lawson, B., & Powell, S. G. (2006). *A survey of MBA spreadsheet users*.

Bishop, J. (2013). Industry's role in data and software curation in the cloud.*Journal of Systems and Software*, *86*(9), 2327-2329.

Booch, G., Rumbaugh, J., & Jacobson, I. (2006). *UML: guia do usuário*. Elsevier Brasil.

Bunke, H. (1997). On a relation between graph edit distance and maximum common subgraph. *Pattern Recognition Letters*, *18*(8), 689-694.

Cale, E. G. (1994). Quality issues for end-user developed software. *Journal of Systems Management*, *45*, 36-36.

Campbell-Kelly, M. (2007). Number crunching without programming: The evolution of spreadsheet usability. *IEEE Annals of the History of Computing*,*29*(3), 6-19.

Ceruzzi, P., & Grad, B. (2007). PC software: Spreadsheets for everyone.*IEEE Annals of the History of Computing*, *29*(3), 0004-5.

Chambers, C., Erwig, M., & Luckey, M. (2010, Setembro). SheetDiff: A tool for identifying changes in spreadsheets. In *Visual Languages and Human-Centric Computing (VL/HCC), 2010 IEEE Symposium on* (pp. 85-92). IEEE.

Chen, Z., & Cafarella, M. (2013, Agosto). Automatic web spreadsheet data extraction. In *Proceedings of the 3rd International Workshop on Semantic Search Over the Web* (p. 1). ACM.

Chen, Z., Cafarella, M., Chen, J., Prevo, D., & Zhuang, J. (2013). Senbazuru: A prototype spreadsheet database management system.*Proceedings of the VLDB Endowment*, *6*(12), 1202-1205.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2001). *Introduction to algorithms* (Vol. 2, pp. 531-549). Cambridge: MIT press.

Cunha, J., Saraiva, J., & Visser, J. (2009, Janeiro). From spreadsheets to relational databases and back. In *Proceedings of the 2009 ACM SIGPLAN workshop on Partial evaluation and program manipulation* (pp. 179-188). ACM.

Damerau, F. J. (1964). A technique for computer detection and correction of spelling errors. *Communications of the ACM*, *7*(3), 171-176.

Dasgupta, S., Papadimitriou, C. H., & Vazirani, U. (2006). *Algorithms*. McGraw-Hill, Inc..

Davies, N., & Ikin, C. (1987). Auditing spreadsheets. *Australian Accountant*,*57*(11), 54-56.

Do, H. H., Melnik, S., & Rahm, E. (2003). Comparison of schema matching evaluations. In *Web, Web-Services, and Database Systems* (pp. 221-237). Springer Berlin Heidelberg.

Engels, G., & Erwig, M. (2005, Novembro). ClassSheets: automatic generation of spreadsheet applications from object-oriented specifications. In *Proceedings of the 20th*

IEEE/ACM international Conference on Automated software engineering (pp. 124-133). ACM.

Fisher, M., & Rothermel, G. (2005). The EUSES spreadsheet corpus: a shared resource for supporting experimentation with spreadsheet dependability mechanisms. *ACM SIGSOFT Software Engineering Notes*,*30*(4), 1-5.

Gansel, B. B. (2008). About the Limitations of Spreadsheet Applications in Business Venturing. In *Operations Research Proceedings 2007* (pp. 219-223). Springer Berlin Heidelberg.

Gao, X., Xiao, B., Tao, D., & Li, X. (2010). A survey of graph edit distance.*Pattern Analysis and applications*, *13*(1), 113-129.

Grad, B. (2007). The creation and the demise of VisiCalc. *IEEE Annals of the History of Computing*, *29*(3), 20-31.

Harutyunyan, A., Borradaile, G., Chambers, C., & Scaffidi, C. (2012, Setembro). Planted-model evaluation of algorithms for identifying differences between spreadsheets. In *Visual Languages and Human-Centric Computing (VL/HCC), 2012 IEEE Symposium on* (pp. 7-14). IEEE.

Hung, V., Benatallah, B., & Saint-Paul, R. (2011, October). Spreadsheet-based complex data transformation. In *Proceedings of the 20th ACM international conference on Information and knowledge management* (pp. 1749-1754). ACM.

Hurst, M. (2001, Setembro). Layout and language: Challenges for table understanding on the web. In *Proceedings of the International Workshop on Web Document Analysis* (pp. 27-30).

Jaccard, P. (1901). Distribution de la flore alpine dans le Bassin des Drouces et dans quelques regions voisines. *Bulletin de la Société Vaudoise des Sciences Naturelles,* 37(140), 241–272.

Jha, P. (2008). Wang Notation Tool: A Layout Independent Representation of Tables (Doctoral dissertation, Rensselaer Polytechnic Institute).

Kelly, M. C., Ceruzzi, P. (2004). *An Interview with Dan Bricklin and Bob Frankston*. Minneapolis: Charles Babbage Institute. Disponível em: http://www.cbi.umn.edu/oh/pdf/oh402b&f.pdf.

Kelly, M. C., Croarken, M., Flood, R., & Robson, E. (2003). *The rise and rise of the spreadsheet* (pp. 323-347). OXFORD UNIV PRESS.

Layland, R. (2007, Maio). *Business Communications Review* (pp. 44-49).

Levenshtein, V. I. (1966, Fevereiro). Binary codes capable of correcting deletions, insertions and reversals. In *Soviet physics doklady* (Vol. 10, p. 707).

Lim, S. J., & Ng, Y. K. (1999, Novembro). An automated approach for retrieving hierarchical data from HTML tables. In *Proceedings of the eighth international conference on Information and knowledge management* (pp. 466-474). ACM.

Lopresti, D., & Nagy, G. (1999, Setembro). Automated table processing: An (opinionated) survey. In *Proceedings of the Third IAPR Workshop on Graphics Recognition* (pp. 109-134).

Maedche, A., & Staab, S. (2002). Measuring similarity between ontologies. In *Knowledge engineering and knowledge management: Ontologies and the semantic web* (pp. 251-263). Springer Berlin Heidelberg.

Messmer, B. T., & Bunke, H. (1994). Efficient error-tolerant subgraph isomorphism detection. In *Shape, Structure and Pattern Recognition* (pp. 231-240). World Scientific.

Messmer, B. T., & Bunke, H. (1998). A new algorithm for error-tolerant subgraph isomorphism detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, *20*(5), 493-504.

Muniswamy-Reddy, K. K., Wright, C. P., Himmer, A., & Zadok, E. (2004, March). A Versatile and User-Oriented Versioning File System. In *FAST* (Vol. 4, pp. 115-128).

Myers, R., Wison, R. C., & Hancock, E. R. (2000). Bayesian graph edit distance. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*,*22*(6), 628-635.

Nierman, A., & Jagadish, H. V. (2002, Junho). Evaluating Structural Similarity in XML Documents. In *WebDB* (Vol. 2, pp. 61-66).

Novelli, N., & Cicchetti, R. (2001). Fun: An efficient algorithm for mining functional and embedded dependencies. In *Database Theory—ICDT 2001*(pp. 189-203). Springer Berlin Heidelberg.

Panko, R. R. (1998). What we know about spreadsheet errors. *Journal of Organizational and End User Computing (JOEUC)*, *10*(2), 15-21.

Panko, R. R. (2008). Spreadsheet errors: What we know. what we think we can do. *arXiv preprint arXiv:0802.3457*.

Panko, R. R., & Aurigemma, S. (2010). Revising the Panko–Halverson taxonomy of spreadsheet errors. *Decision Support Systems*, *49*(2), 235-244.

Panko, R. R., & Halverson Jr, R. P. (1996, Janeiro). Spreadsheets on trial: a survey of research on spreadsheet risks. In *System Sciences, 1996., Proceedings of the Twenty-Ninth Hawaii International Conference on,* (Vol. 2, pp. 326-335). IEEE.

Parberry, I. (1995). Problems on algorithms. *ACM SIGACT News*, *26*(2), 50-56.

Pryor, L., Evans, R., Foley, B., Garner, M., Hilary, N., Skinner, J., & Tanser, J. (2006, Setembro). Actuaries excel: but what about their software?. In *Paper presented at the 33rd Annual GIRO conference*.

Rahm, E., & Bernstein, P. A. (2001). A survey of approaches to automatic schema matching. *the VLDB Journal*, *10*(4), 334-350.

Sanfeliu, A., & Fu, K. S. (1983). A distance measure between attributed relational graphs for pattern recognition. *Systems, Man and Cybernetics, IEEE Transactions on*, (3), 353-362.

Schauerte, B., & Fink, G. A. (2010, November). Focusing computational visual attention in multi-modal human-robot interaction. In *International Conference on Multimodal Interfaces and the Workshop on Machine Learning for Multimodal Interaction* (p. 6). ACM.

Soules, C. A., Goodson, G. R., Strunk, J. D., & Ganger, G. R. (2003, March). Metadata Efficiency in Versioning File Systems. In *FAST* (Vol. 3, pp. 43-58).

Strasser, C., & Cruse, P. (2013). The DMPTool and DataUp: Helping Researchers Manage, Archive, and Share their Data. In *Research Data Management Implementations Workshop*.

Strasser, C., Kunze, J., Abrams, S., & Cruse, P. (2014). DataUp: A tool to help researchers describe and share tabular.

Tao, C. (2003). *Schema matching and data extraction over HTML tables*(Doctoral dissertation, Brigham Young University).

Teo, T. & Tan, M. (1997) "Quantitative and Qualitative Errors in Spreadsheet Development." *Proceedings of the 30th Annual Hawaii International Conference on System Sciences*, Wailea, 149-156.

Teo, T. S., & Lee-Partridge, J. E. (2001). Effects of error factors and prior incremental practice on spreadsheet error detection: an experimental study.*Omega*, *29*(5), 445-456.

Teo, T. S., & Tan, M. (1999). Spreadsheet development and 'what-if'analysis: quantitative versus qualitative errors. *Accounting, Management and Information Technologies*, *9*(3), 141-160.

Tubbs, K. M., & Embley, D. W. (2002, Novembro). Recognizing records from the extracted cells of microfilm tables. In *Proceedings of the 2002 ACM symposium on Document engineering* (pp. 149-156). ACM.

Wang, X., & Wood, D. (1996). *Tabular abstraction, editing, and formatting*. Canada: University of Waterloo.

Wei, J. (2004). Markov edit distance. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, *26*(3), 311-321.

Wilson, R. C., & Hancock, E. R. (1997). Structural matching by discrete relaxation. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 19(6), 634-648.

Zhang, K. (1996). A constrained edit distance between unordered labeled trees. *Algorithmica*, *15*(3), 205-222.

Zhang, K., & Shasha, D. (1989). Simple fast algorithms for the editing distance between trees and related problems. *SIAM journal on computing*,*18*(6), 1245-1262.

Zynda, M. R. (2013). The first killer app: a history of spreadsheets.*interactions*, *20*(5), 68-72.

Apêndice A

Lista de gráficos scatter para os testes de comportamento feitos com o algoritmo DSJ.

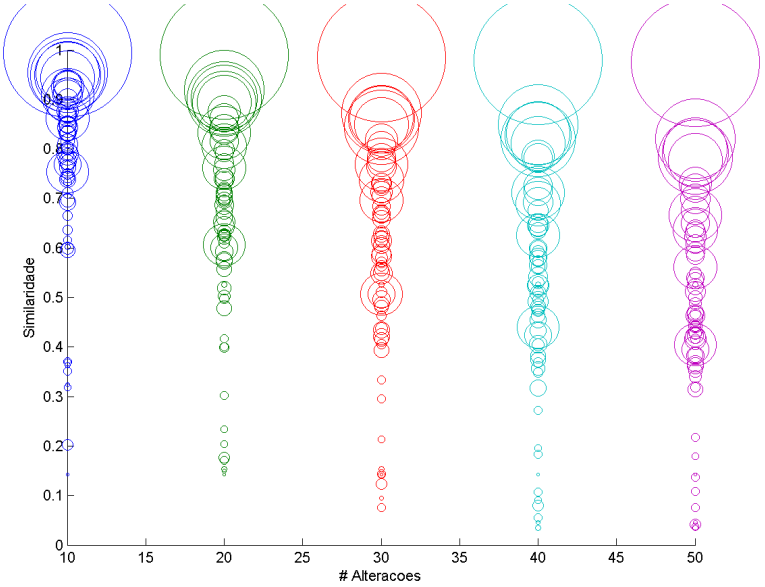


Figura 43 - RCE (DSJ)

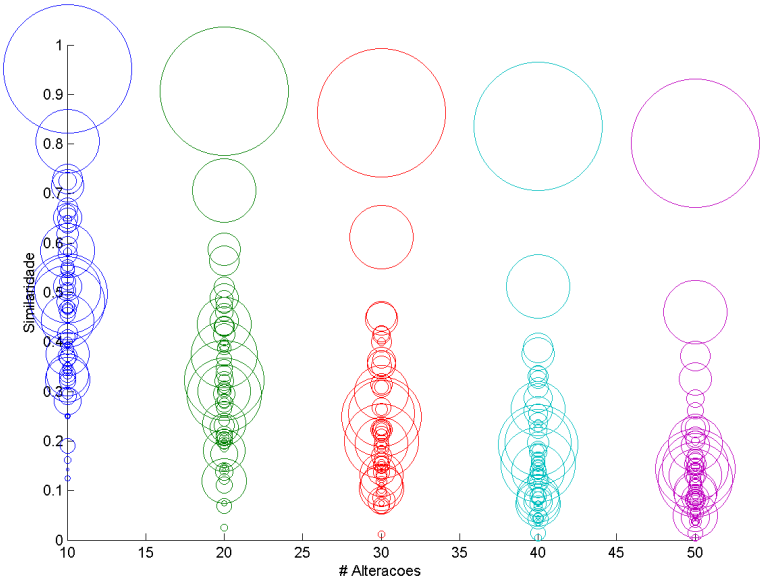


Figura 44 - RLE (DSJ)

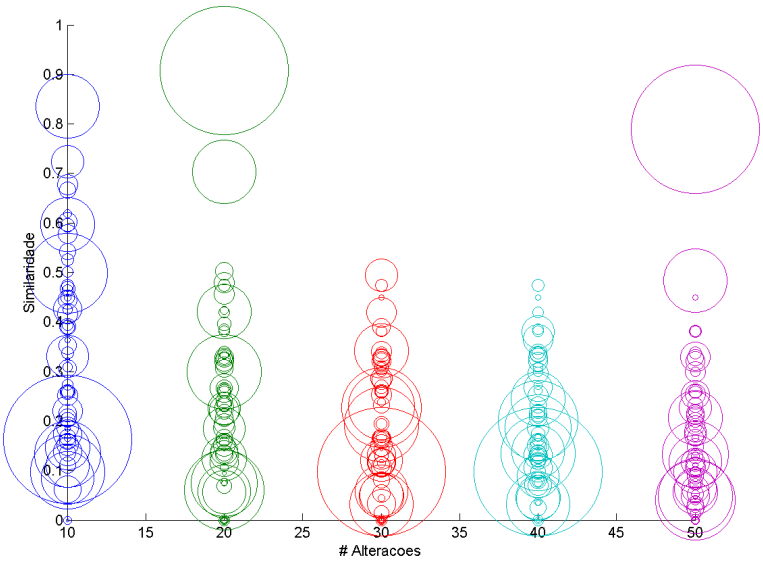


Figura 45 - RLOE (DSJ)

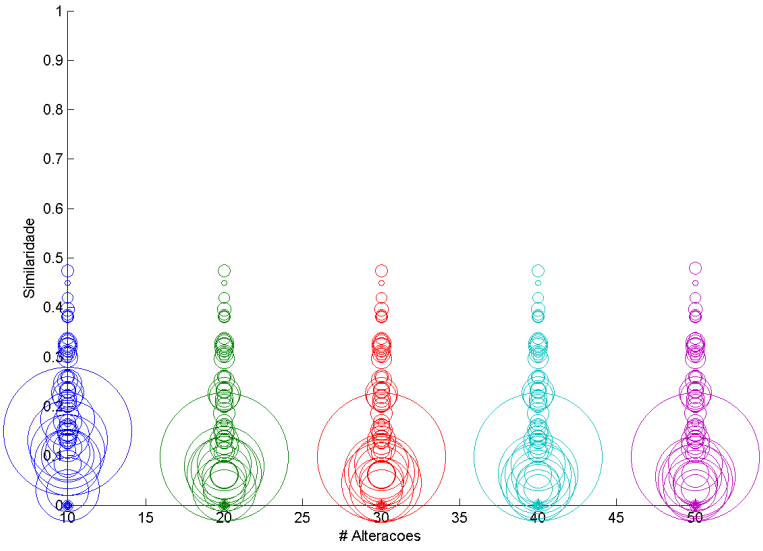


Figura 46 - ROE (DSJ)

Apêndice B

Lista de gráficos scatter para os testes de comportamento feitos com o algoritmo LCS-Cosine.

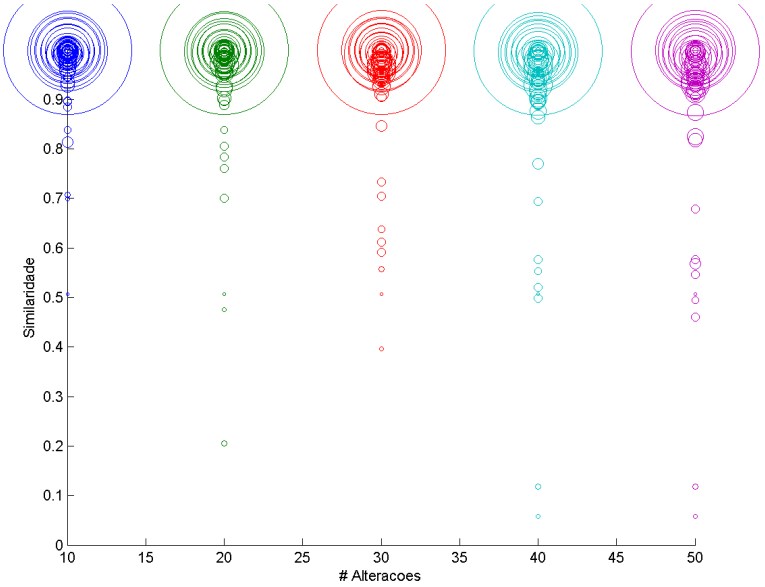


Figura 47 - RCE (LCS-Cosine)

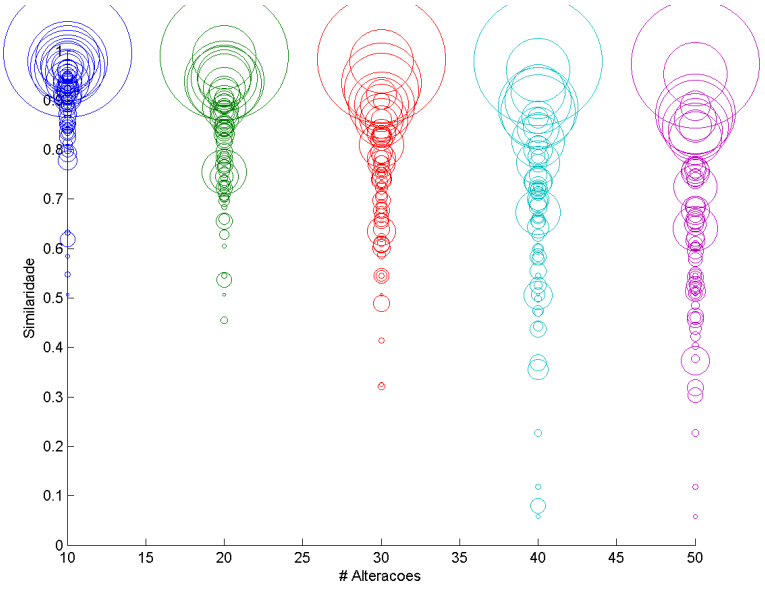


Figura 48 - RLE (LCS-Cosine)

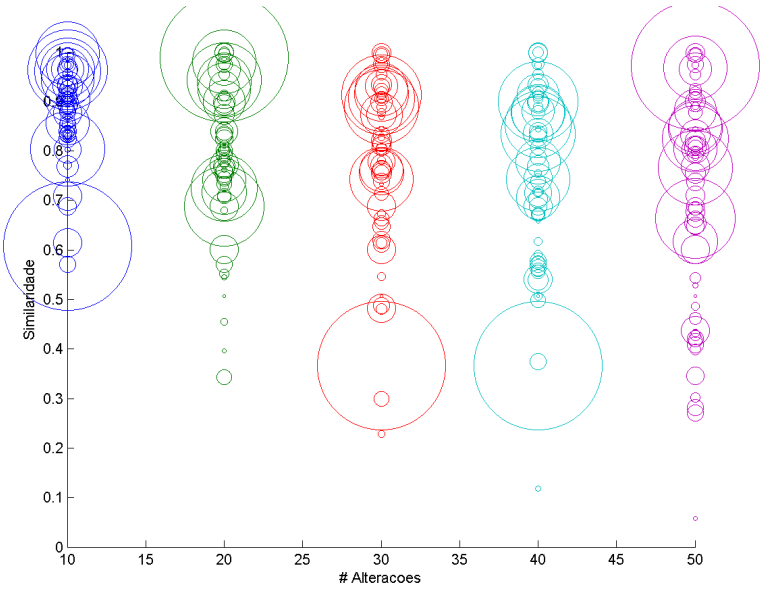


Figura 49 - RLOE (LCS-Cosine)

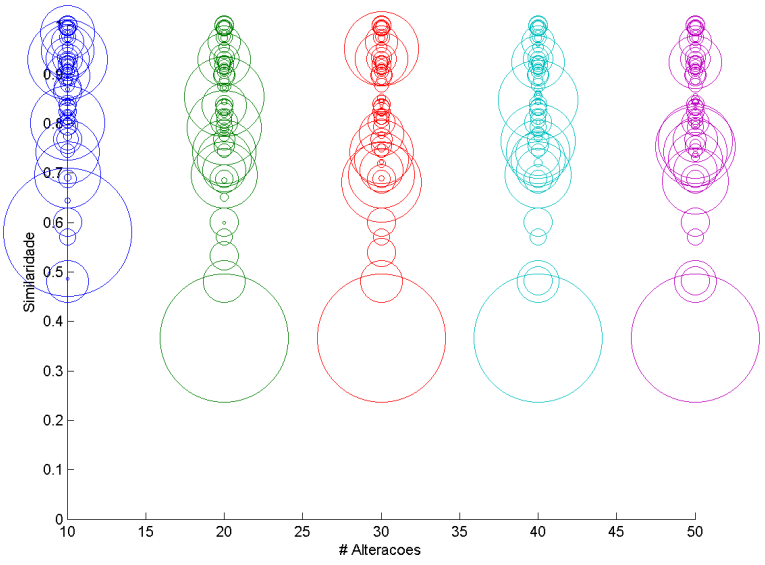


Figura 50 - ROE (LCS-Cosine)

Apêndice C

Lista de gráficos scatter para os testes de comportamento feitos com o algoritmo LCS-Hamming.

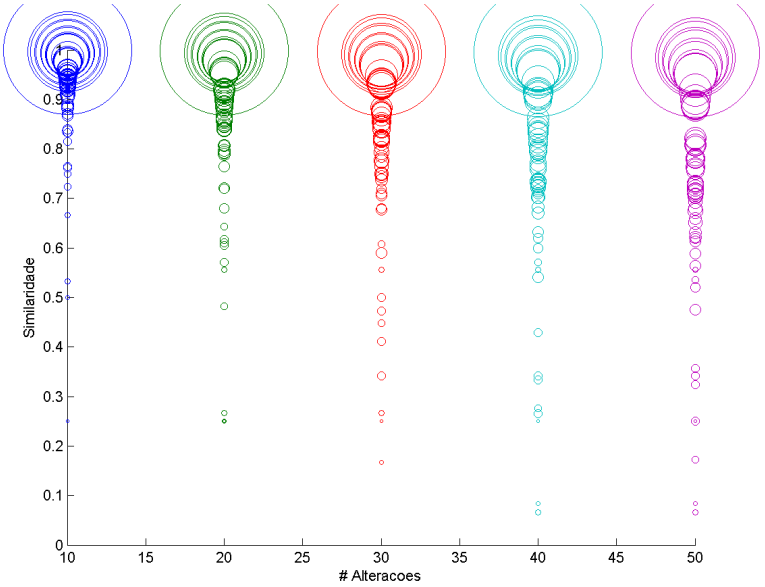


Figura 51 – RCE (LCS-Hamming)

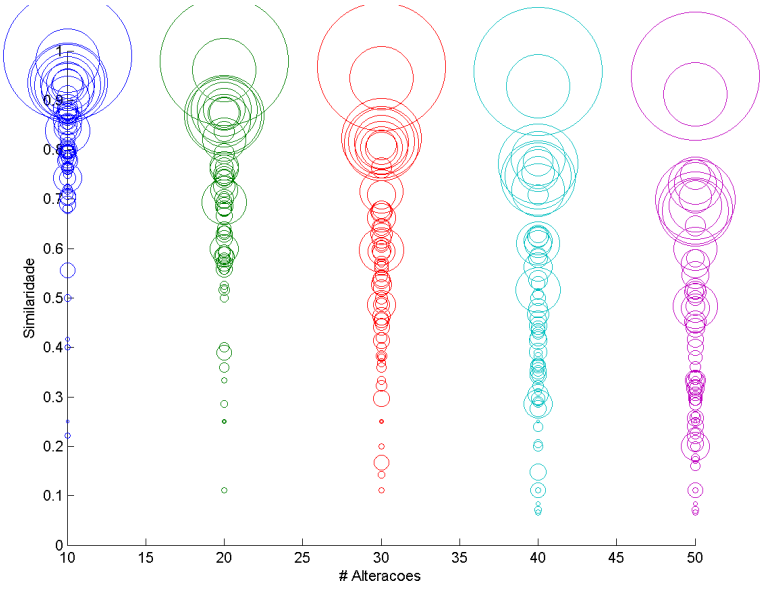


Figura 52 – RLE (LCS-Hamming)

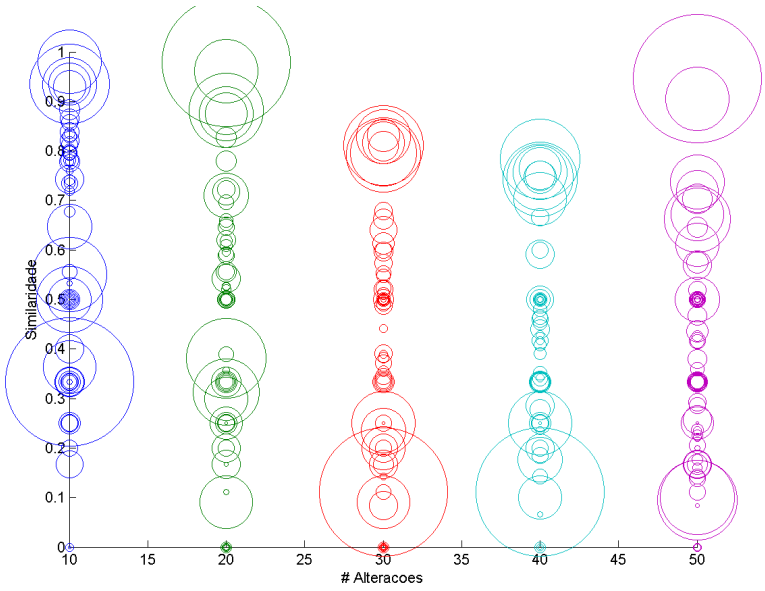


Figura 53 – RLOE (LCS-Hamming)

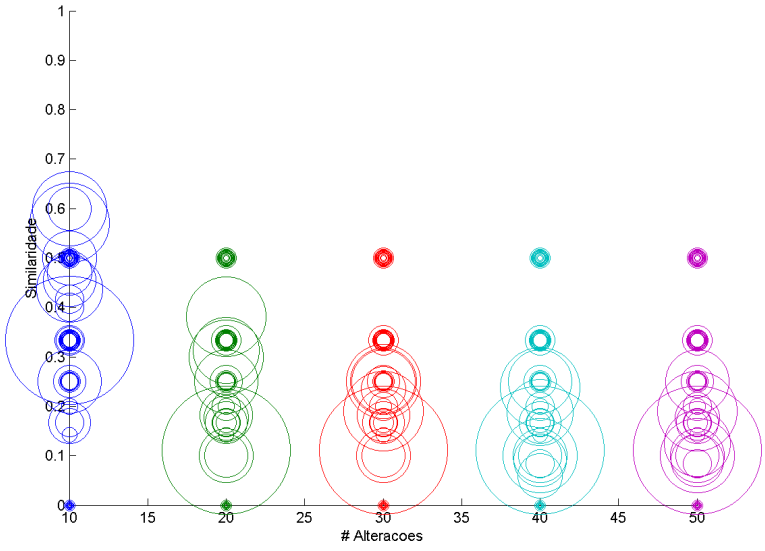


Figura 54 – ROE (LCS-Hamming)

Apêndice D

Lista de gráficos scatter para os testes de comportamento feitos com o algoritmo LCS-Jaccard.

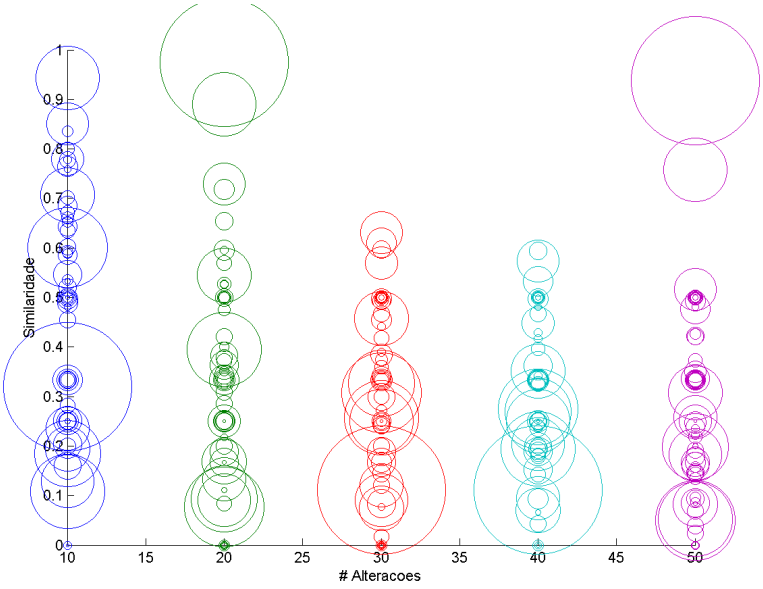


Figura 55 – RCE (LCS-Jaccard)

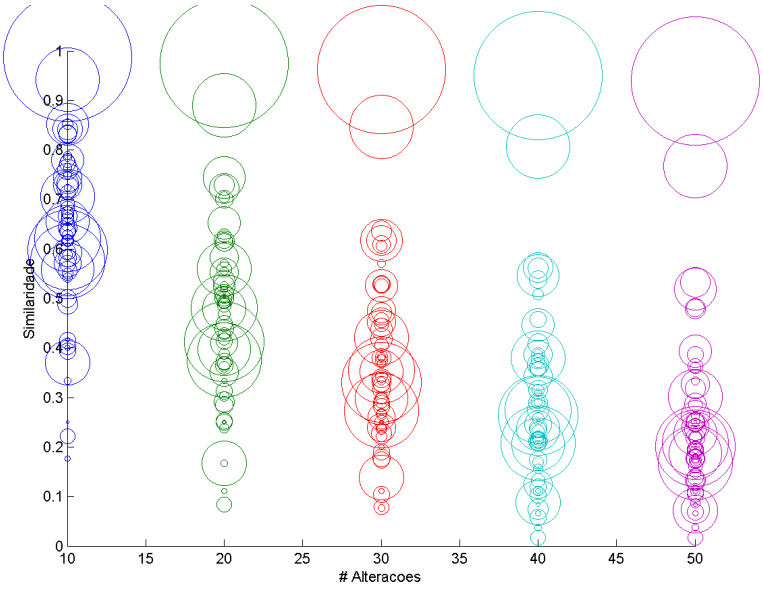


Figura 56 - RLE (LCS-Jaccard)

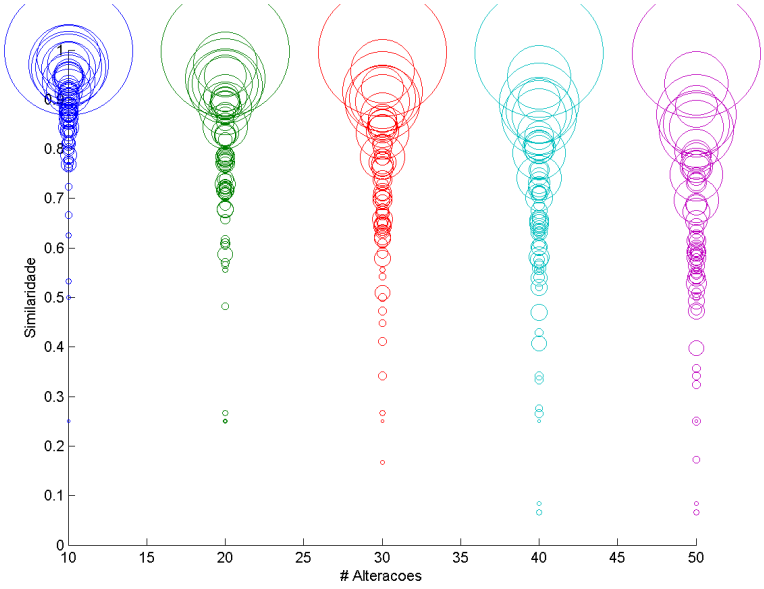


Figura 57 - RLOE (LCS-Jaccard)

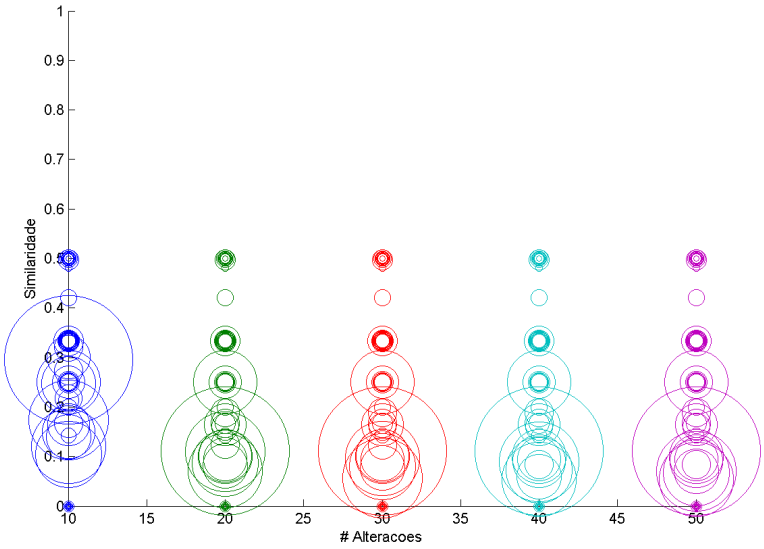


Figura 58 - ROE (LCS-Jaccard)

Apêndice E

Lista de gráficos scatter para os testes de comportamento feitos com o algoritmo RCA.

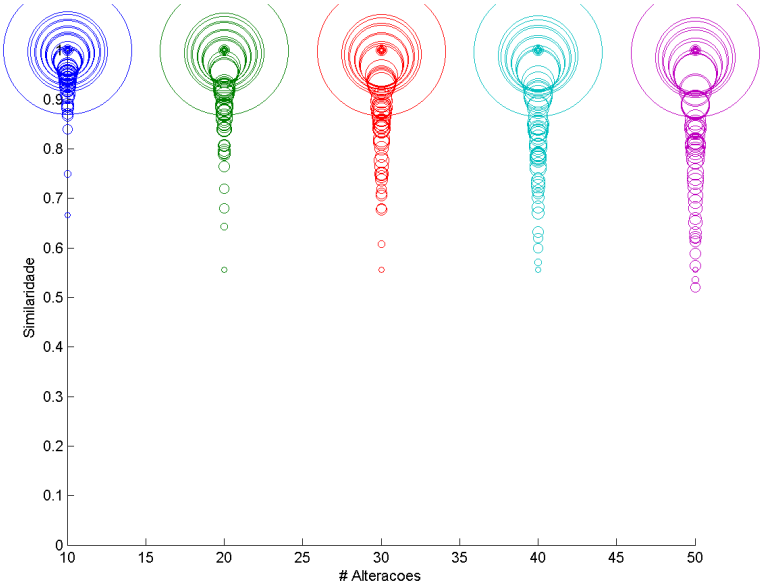


Figura 59 – RCE (RCA)

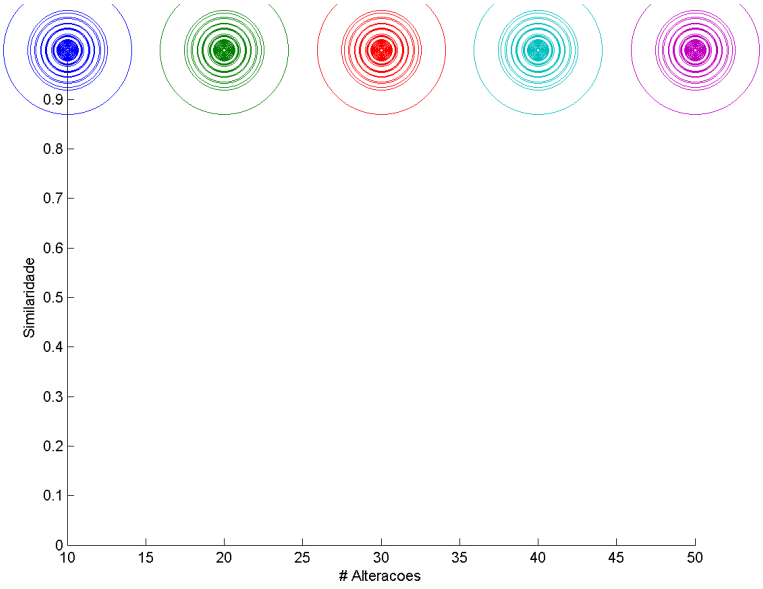


Figura 60 – RLE (RCA)

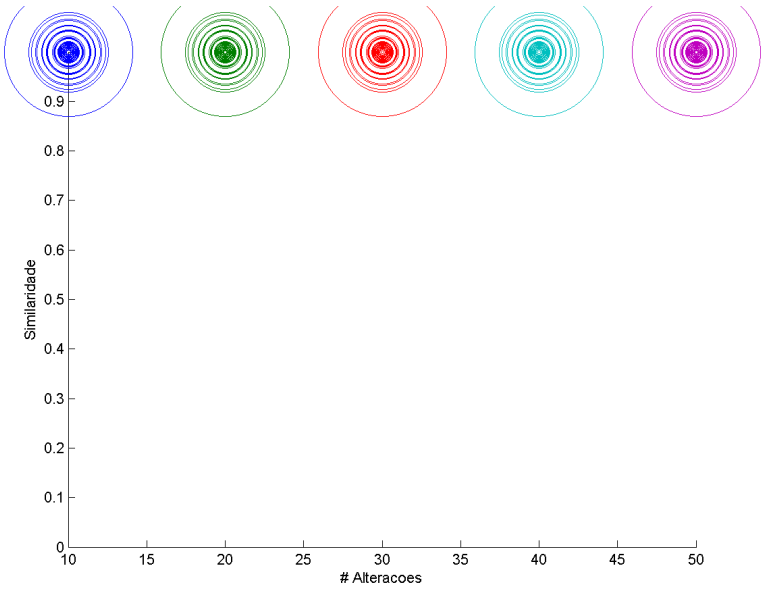


Figura 61 – RLOE (RCA)

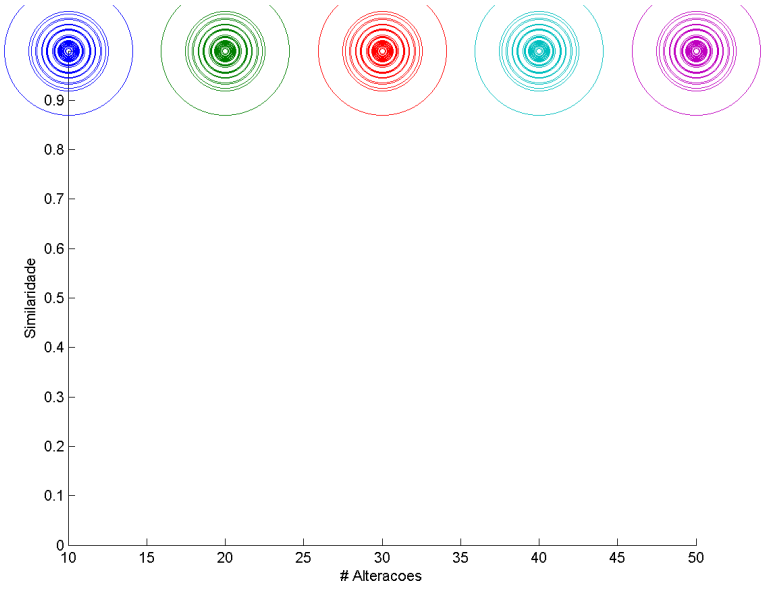


Figura 62 - ROE (RCA)