Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Projeto de Final de Curso
Projeto e Análise de Algoritmos

Relatório Final

Alberto Hideki Ueda

Orientador: Berthier Ribeiro Neto

Belo Horizonte
1º de dezembro de 2014

Conteúdo

1 O Problema **2**
1.1 Contexto 2
1.2 Dificuldade do Problema 3
1.3 Objetivos do Trabalho 3
1.4 Abordagem Utilizada 4

2 Modelagem através de um Grafo **4**

3 Baseline **7**
3.1 WIRE 8
3.2 Algoritmo 8

4 Heurística **11**
4.1 Ideia 11
4.2 Implementação 12

5 Análise Teórica de Complexidade **13**
5.1 Complexidade de Tempo 13
5.2 Complexidade de Espaço 14

6 Análise Experimental dos Algoritmos **14**
6.1 Qualidade da Resposta 16

7 Trabalhos Futuros **19**

8 Conclusão **19**

1 O Problema

1.1 Contexto

Coletores de páginas da *Web* constituem o primeiro passo para a implementação de máquinas de busca modernas. De forma geral, um coletor - em inglês, *crawler* - é um sistema que faz requisições a servidores da *Web* de forma planejada e automática, coleta parte ou todo o conteúdo das páginas devolvidas pelas requisições e utiliza este novo conteúdo para realizar novas requisições [1]. Estima-se que hoje mais de 10% das visitas a *Websites* sejam feitas por coletores [11].

O primeiro coletor *Web* conhecido foi criado em 1993 por Matthew Gray - então graduando do MIT - e chamava-se WWWW (*World Wide Web Wanderer).* Comprovando a forte relação com a história dos sistemas de busca na *Web*, no mesmo ano foi lançada também a primeira máquina de busca conhecida, ALIWEB, criada por Martijn Koster [1]. Nesta época, um número razoável de servidores para se obter uma boa cobertura da rede girava em torno de apenas alguns milhares. Desde então, o número de *hosts* tem aumentado em alta velocidade (chegando a praticamente dobrar a cada ano, de 1993 a 1996 [7]), tornando as máquinas de busca ainda mais necessárias e, consequentemente, também os coletores de dados.

A figura a seguir mostra o ciclo de funcionamento de um sistema completo, desde a coleta até a busca em si.

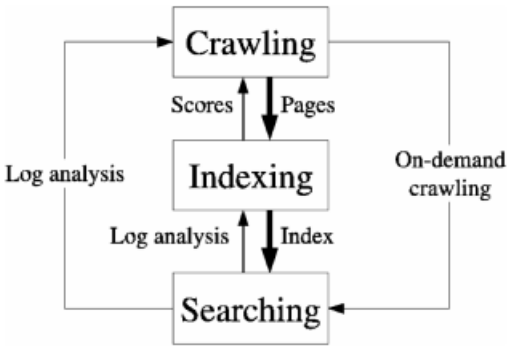

Figura 1: Ciclo de uma máquina de busca [4].

Deste modo, podemos fazer a seguinte pergunta: como coletar as páginas da *Web* de forma eficiente em termos de tempo e espaço, mantendo um nível mínimo de qualidade do resultado?

1.2 Dificuldade do Problema

Hoje, mesmo as principais máquinas de busca cobrem apenas uma parte da *Web* atual. Em 2005, foi demonstrado que o nível de cobertura das principais máquinas de buscas existentes está entre 58% e 76% da *Web* [8]. Além disso, o custo da utilização da rede também deve ser considerado. Em 2004, tal custo foi estimado em US$ 1,5 milhão para coletores de larga escala [6].

Outra característica que torna o problema difícil é a velocidade em que a *Web* se modifica. Devido à facilidade de acesso e à alta disponibilidade de servidores nos dias atuais, a cada minuto novas páginas são criadas, modificadas ou removidas da *Web*. Em todos os casos, a relevância de cada página existente pode aumentar ou diminuir, conforme as novas características da *Web*. A figura abaixo ilustra esta ideia.

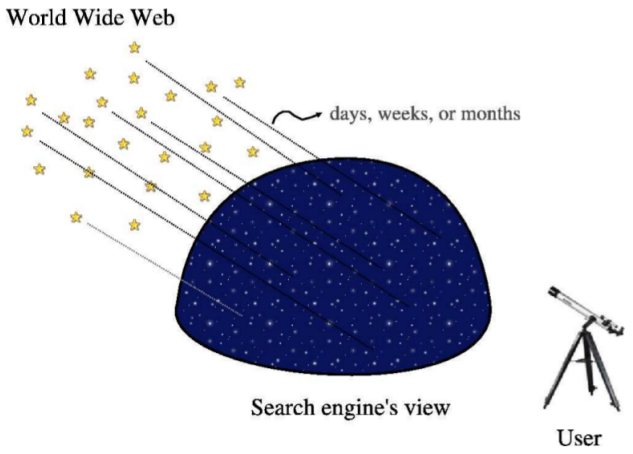

Figura 2: coletar documentos da *Web* pode ser visto como analisar um céu estrelado. No momento em que podemos visualizá-lo, o cenário real já se modificou [4].

Portanto, tal problema pode ser considerado de difícil resolução, dado a característica dinâmica da *Web* e o tamanho da entrada necessária para uma solução exata.

1.3 Objetivos do Trabalho

O objetivo deste trabalho é modificar um algoritmo existente de um *Web Crawler*, construindo uma heurística que visa aumentar a qualidade

das páginas coletadas em relação ao algoritmo original, nos casos específicos em que o coletor deve se concentrar em um determinado tópico de interesse.

Mais especificamente, este trabalho consistirá em alterações no escalonamento de longo prazo de um coletor, direcionando as requisições de *downloads* para páginas relevantes a um tópico pré-estabelecido, utilizando na análise de futuros *links* os termos mais frequentes das páginas já coletadas.

1.4 Abordagem Utilizada

Para isso, modificaremos a estratégia que o coletor utiliza para determinar quais são os próximos *links* que devem ser processados. No caso do *WIRE*, nosso *baseline*, devemos alterar seu escalonamento de longo-prazo (componente *manager*, descrito na seção 3, substituindo parte do código por um novo algoritmo de escolha das próximas páginas que deverão ser coletadas.

Em seguida, executaremos testes de coleta para os dois algoritmos - original e modificado - e faremos a comparação dos resultados obtidos. Dado um tópico-alvo para a coleta, espera-se que o algoritmo adaptado seja melhor em termos de espaço e memória que o original. Mais tecnicamente, nosso *crawler* deverá encontrar parte das páginas mais relevantes *antes* do coletor do *baseline*.

2 Modelagem através de um Grafo

Considerando as páginas da *Web* como vértices e os diversos *links* encontrados como arestas, pode-se aplicar diferentes algoritmos para a seleção *on-line* (em tempo de execução) dos *links* que o nosso *crawler* irá visitar em um nova coleta. Possíveis estratégias de escolha (*selection policy*) para o coletor baseiam-se, por exemplo, em: buscas em largura [10], indicador *PageRank* [5] ou até mesmo no uso de algoritmos genéticos [9].

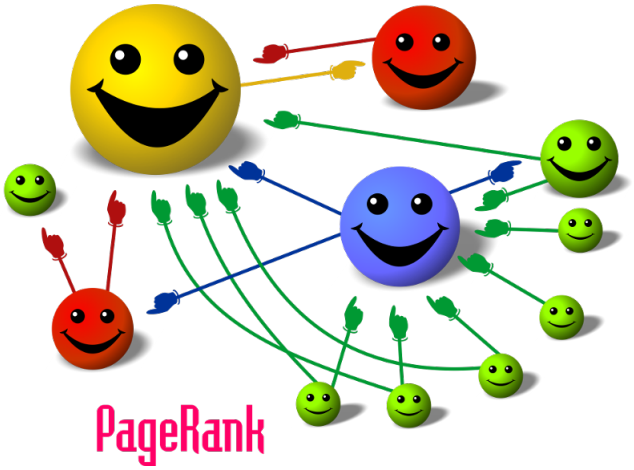

Figura 3: Um dos indicadores mais conhecidos em recuperação de informação, o *PageRank* mede a relevância de uma página da *Web* considerando, por exemplo, o número de *links* que apontam para esta página e a relevância das origens destes *links*. Na figura, quanto maior o tamanho do círculo (página da *Web*), maior sua relevância.

Levando-se em conta que não é possível coletar todas as páginas da *Web*, devido à enorme quantidade de páginas e sua característica mutável, é importante encontrar as páginas mais relevantes o quanto antes. Quanto mais rápido uma página muito relevante for encontrada (de acordo com o indicador considerado), melhor a qualidade dos resultados obtidos em um dado instante de tempo da execução do coletor.

Podemos então pensar no seguinte problema: dado um grafo $G(V, E)$ e um inteiro k, em que:

- os vértices de G são páginas da *Web*,
- as arestas representam os links entre estas páginas, e
- cada vértice possui uma pontuação de relevância não-conhecida a priori,

percorrer um caminho que maximiza as pontuações dos vértices visitando no máximo k páginas.

Desta forma, para a implementação de um *crawler* de alto desempenho, é interessante considerar a estrutura da *Web* atual. Tal estrutura pode ser analisada sob duas perspectivas: *microscópica* e *macroscópica*.

Sob uma perspectiva microscópica, percebe-se que a estrutura dos *links* não se assemelha a um arranjo aleatório. Ao invés disso, é comum identificar *hubs* e redes *scale-free* [2], como ilustrado na figura abaixo.

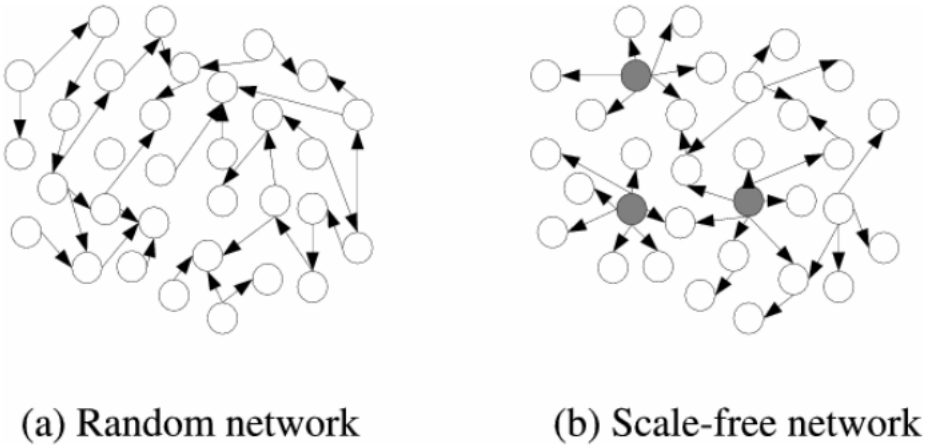

Figura 4: Exemplos de estruturas microscópicas da *Web*. Uma rede gerada aleatoriamente (à esquerda) e uma rede *scale-free* [2].

Já sob a perspectiva macroscópica, podemos identificar uma estrutura *bow-tie* [3], conforme visto em um dos trabalhos práticos da disciplina. A figura a seguir apresenta uma possível visualização desta estrutura.

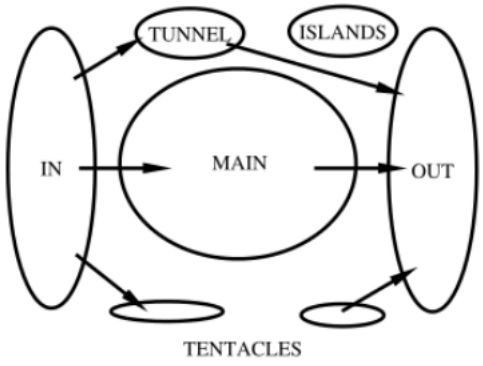

Figura 5: Estrutura macroscópia da *Web* [3].

Devido a estas perspectivas, muitos *crawlers* são implementados de modo

a beneficiar-se destas informações. Este é o caso do WIRE [4], o *web crawler* que é nosso *baseline* neste trabalho.

3 Baseline

De forma geral, os coletores podem ser divididos em diversas categorias, conforme mostra a figura a seguir. Em cada categoria é possível identificar quais critérios são os mais importantes em uma coleta.

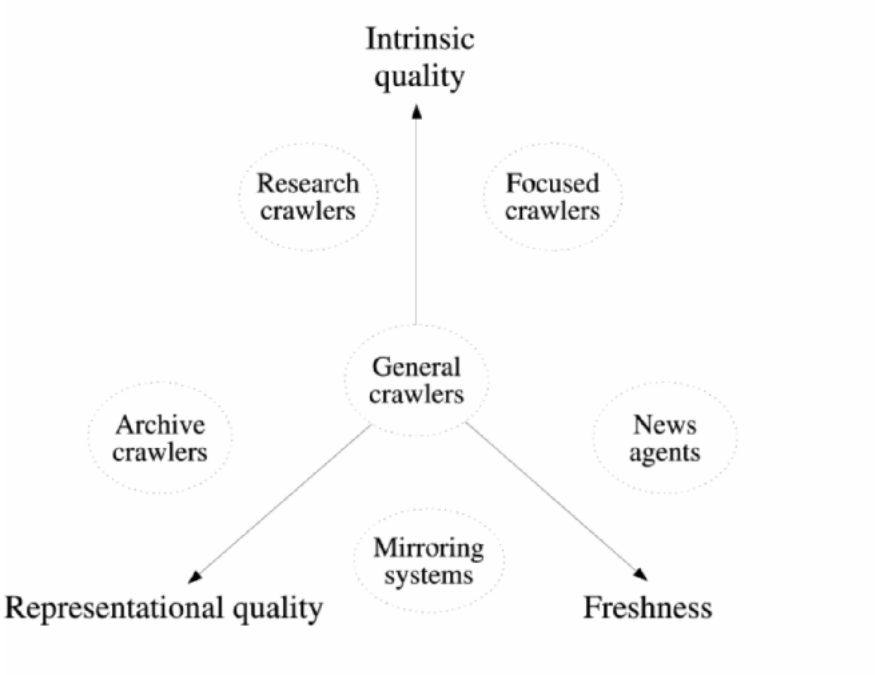

Figura 6: Tipos de coletores [4].

Por exemplo, em um coletor que visa reproduzir inteiramente o conteúdo das páginas em determinado instante de tempo, a *qualidade da representação* é o critério mais importante; já o *nível de atualização (freshness)* das páginas não é um critério tão relevante quanto o anterior.

Nosso baseline, descrito a seguir, é um coletor do tipo genérico. Ou seja, um coletor que tenta equilibrar os três critérios de qualidade, sem focar especificamente em algum destes critérios. Ao invés de apresentar os melhores resultados para um determinado critério, tal coletor tenta garantir um mínimo de qualidade em cada um deles.

3.1 WIRE

O *baseline* para este trabalho é o *WIRE* (*Web Information Retrieval Environment*), um *web crawler* desenvolvido por Carlos Castillo em seu doutorado na Universidade do Chile [4]. Este coletor foi construído visando-se alto desempenho e boa escalabilidade, respeitando as políticas de restricões a visitas de robôs a servidores *Web*.

O objetivo principal do *WIRE* é realizar a caracterização da *Web*, a partir dos dados coletados. A figura abaixo é uma representação em alto nível da função do coletor e possíveis integrações com outros objetivos de trabalho, representados em cinza claro.

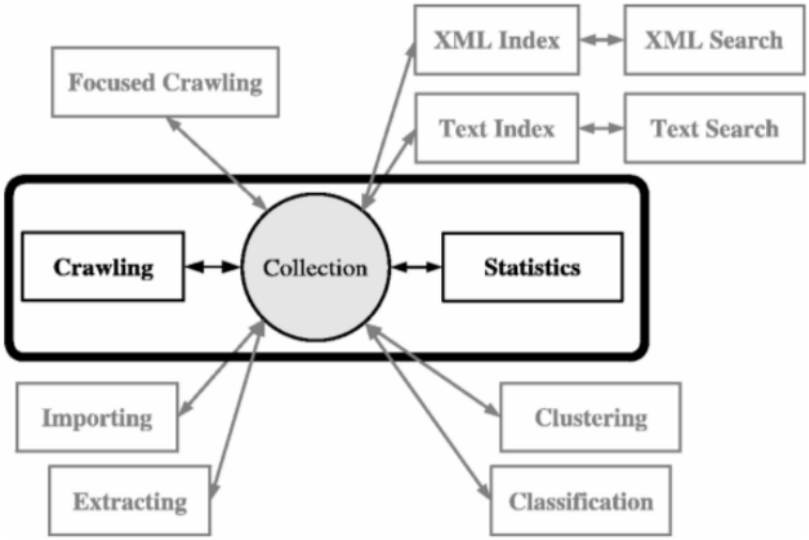

Figura 7: Áreas de atuação do *WIRE* (em preto) e trabalhos futuros (em cinza) [4].

3.2 Algoritmo

O algoritmo deste *baseline* é representado em alto nível pela figura a seguir.

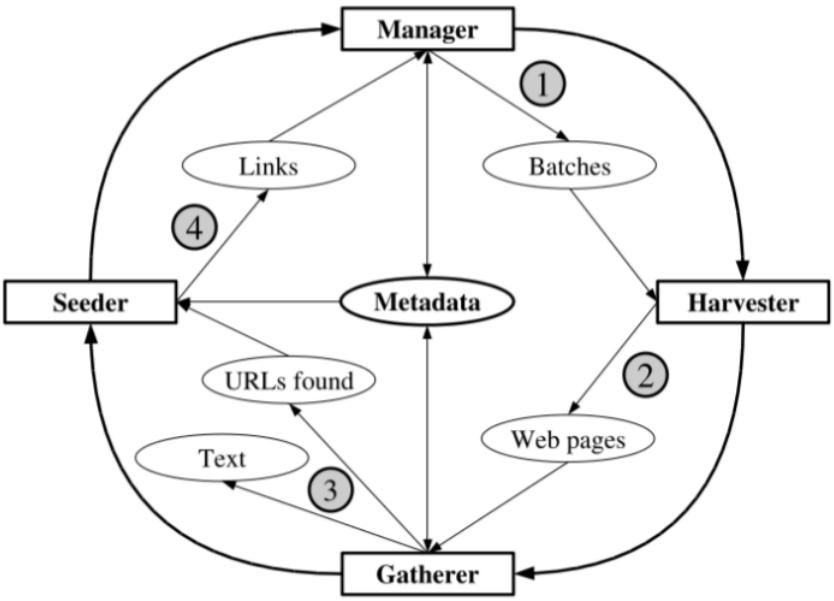

Figura 8: Representação do funcionamento do coletor *WIRE* [4].

Cada componente é descrito abaixo:

- *Manager*: realiza os cálculos da relevância de cada página coletada e as atualizações no escalonamento de longo-prazo;
- *Harvester*: responsável pelo escalonamento de curto prazo e pelas coletas de páginas;
- *Gatherer*: faz o *parse* e a extração de *links*;
- *Seeder*: responsável pela *resolução* de *URLs* e atualizações no grafo de *links*;

O algoritmo utiliza listas de adjacências para representar a estrutura de *links* encontrada entre as páginas, podendo ser descrito da seguinte forma:

1. (*Seeder*) A partir de um arquivo inicial de *URLs*, gera-se a lista das próximas páginas que devem ser coletadas;
2. (*Manager*) Se uma página da lista gerada já foi visitada recentemente, a mesma é ignorada;

3. O valor intrínseco de cada página da lista é calculado, por meio de um indicador. Por exemplo, o *PageRank* [12];

4. Calcula-se o quanto as páginas estão atualizadas;

5. Com o valor intrínseco e o nível de atualização das páginas, calcula-se a qualidade geral de cada página;

6. Extrai-se então da lista as k páginas de maior qualidade, k determinado em um arquivo de configuração;

7. (*Harvester*) Determina-se como as k páginas serão coletadas (observando regras de "bom comportamento"para coletores);

8. Coleta-se todas as páginas de acordo com as regras determinadas no item anterior;

9. (*Gatherer*) As informações coletadas são tratadas, armazenando-se o que é relevante e descartando o restante do material coletado;

10. São enviadas para o *Seeder* as novas *URLs* encontradas nas páginas coletadas e reinicia-se o processo, agora sem a necessidade de arquivos de inicialização.

A seguir, um exemplo de como o *manager* calcula as pontuações de cada *URL* e escolhe as próximas páginas para *download*.

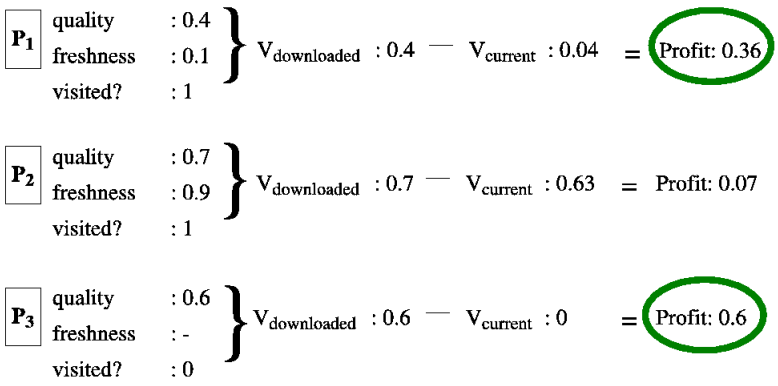

Figura 9: Exemplo de escolha do *Manager*. O lucro (*profit*) é a relevância estimada para a página, calculada a partir de seu nível de atualização e indicadores de qualidade como, por exemplo, a relevância das páginas de origem.

A condição de parada depende do objetivo do usuário, que pode priorizar cobertura (continuar até que não haja páginas no domínio de interesse, demandando potencialmente um tempo inviável) ou por critério de tempo (encerra-se quando atingir um nível determinado de cobertura). O estudo do *WIRE* mostrou que a partir de uma coleta com 50% de uma grande coleção de páginas, já é possível atingir 80% do valor total do *PageRank* dessa coleção [4].

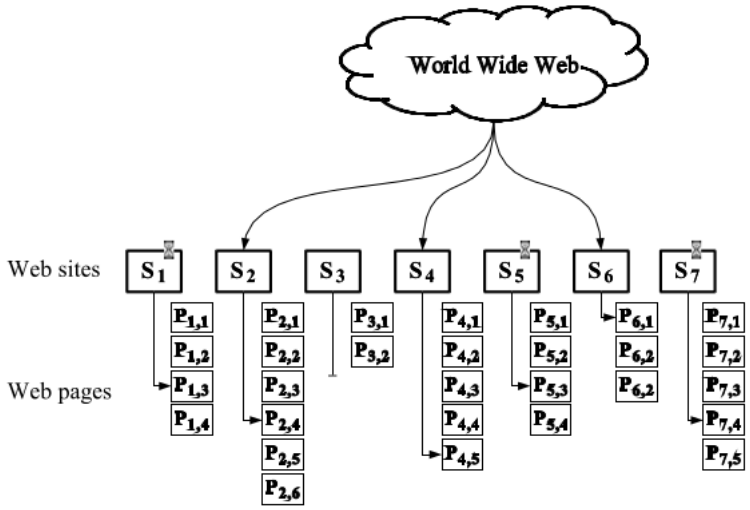

Figura 10: Representação da coleta de *Websites* e suas respectivas páginas, realizada pelo *Wire*.

4 Heurística

4.1 Ideia

Nossa heurística utiliza a seguinte hipótese: dado um conjunto inicial de *websites* relevantes para um determinado tópico de interesse, os termos mais frequentes destes domínios são termos relevantes para o tópico de interesse.

Podemos então construir tal lista de palavras frequentes e procurar no texto das *URLs* a existência ou não destes termos, adicionando uma pontuação maior às *URLs* que os possuem, proporcionalmente ao número de termos encontrados. À medida que a coleta prossegue, tal lista é atualizada de modo a conter sempre as palavras mais frequentes nas páginas encontradas, aumentando assim o "conhecimento"do coletor.

São exemplos de tópicos de interesse: futebol, esportes, política, economia, concursos públicos, etc.

A seguir, um exemplo ilustrativo de nuvem de palavras construída para o tópico *futebol*, por meio de algumas páginas de notícias sobre o assunto. Quanto maior uma palavra na nuvem, mais ela é frequente nos documentos da *Web* encontrados.

Figura 11: Nuvem de palavras para o tópico "futebol", gerada no *website*: http://jasondavies.com/wordcloud.

4.2 Implementação

Há duas intervenções necessárias para a implementação desta heurística.

A primeira é a contagem de palavras, que é feita no componente *Gatherer*. Esta é a fase da coleta em que são extraídos os *links* do conteúdo das páginas, que serão posteriormente utilizados em novas coletas. Com o conteúdo disponível, analisamos o texto e atualizamos a lista de palavras mais frequentes. Tal contagem de palavras é feita da maneira mais direta possível, atualizando o contador conforme cada nova palavra da página em *html*. A única sofisticação é a utilização de uma lista de palavras não relevantes da língua (*stopwords*), como artigos, preposições, advérbios, etc.

A segunda intervenção é no componente *Manager*, que atribui pontuações para *URLs* baseado nas informações que o coletor possui até então. Dado um número n (configurável), busca-se na *URL* cada uma das n palavras mais frequentes. Quanto mais palavras forem encontradas, maior é a pontuação da *URL*.

5 Análise Teórica de Complexidade

Partindo do problema modelado em um grafo G(V,E) em que os vértices são páginas da *Web* e as arestas representam os links entre estas páginas, o algoritmo do baseline tem a mesma característica da busca em largura, que é a de não repetir vértices nem arestas durante sua execução. Dada uma página cujos *links* já foram extraídos, o coletor não repete o processamento para esta página novamente. O mesmo raciocínio aplica-se às arestas: um mesmo *link* não é analisado novamente após já ter sido pontuado.

Como é explicado nas próximas subseções, a heurística adotada não altera as complexidades de tempo e espaço do baseline, por se tratarem de modificações que possuem complexidade constante em relação à V e E.

Portanto, as complexidades de ambos os algoritmos podem ser descritas como a seguir:

Algoritmo	Complexidade de Tempo	Complexidade de Espaço
Baseline (WIRE)	O(V+E)	O(V+E)
Heurística	O(aV+bE) = O(V+E)	O(V+E+c) = O(V+E)

Figura 12: a, b, c: constantes em relação a V e E.

5.1 Complexidade de Tempo

Há duas intervenções no algoritmo: uma para contagem de palavras e outras para análise de *URLs*. A contagem é feita de maneira direta, comparando-se com uma lista *stopwords* e atualizando a lista incrementando a frequência de um termo cada vez que ele é encontrado em uma página. Tal algoritmo não depende do tamanho de V ou de A e portanto possui uma complexidade de tempo a constante em relação à instância do problema.

O mesmo ocorre com a análise e pontuação de *URLs*. Dadas as n palavras mais frequentes da lista de termos, cada um é procurado diretamente na *URL*, sem utilização de análises semânticas ou ontologias, como alguns estudos encontrados durante a pesquisa sugerem. Esta análise não depende das características do grafo do problema, logo também possui uma complexidade de tempo b contante em relação à V e E.

Como a multiplicação por constantes não muda a complexidade de um algoritmo, a complexidade de tempo da heurística também é $O(V + E)$.

5.2 Complexidade de Espaço

Em relação ao *baseline*, o espaço adicional utilizado se refere à lista para contagem de palavras. O tamanho desta lista depende diretamente do número de palavras possíveis em um idioma, subtraindo-se preposições, artigos, advérbios, etc (*stopwords*). Logo, o tamanho deste espaço não depende da instância do grafo e é limitado superiormente por uma constante c, por exemplo.

Novamente, como a multiplicação por constantes não muda a complexidade de um algoritmo, a complexidade de espaço da heurística também é $O(V + E)$.

6 Análise Experimental dos Algoritmos

A seguir alguns dos resultados obtidos pela heurística e pelo *baseline*. De um modo geral, o tempo de execução foi maior na heurística, conforme esperado devido às mudanças no algoritmo. O espaço utilizado é ligeiramente maior, devido ao fato do espaço extra utilizado (lista de termos comuns) ser pequeno em relação à grande quantidade de memória que já é alocada para armazenamento dos dados coletados.

Em relação à qualidade da resposta, os experimentos apresentaram tanto casos em que o resultado da heurística supera o do *baseline* quanto alguns casos em que a coleta realizada pela heurística fica abaixo do algoritmo de referência, em termos de qualidade. Mais detalhes são descritos na subseção a seguir.

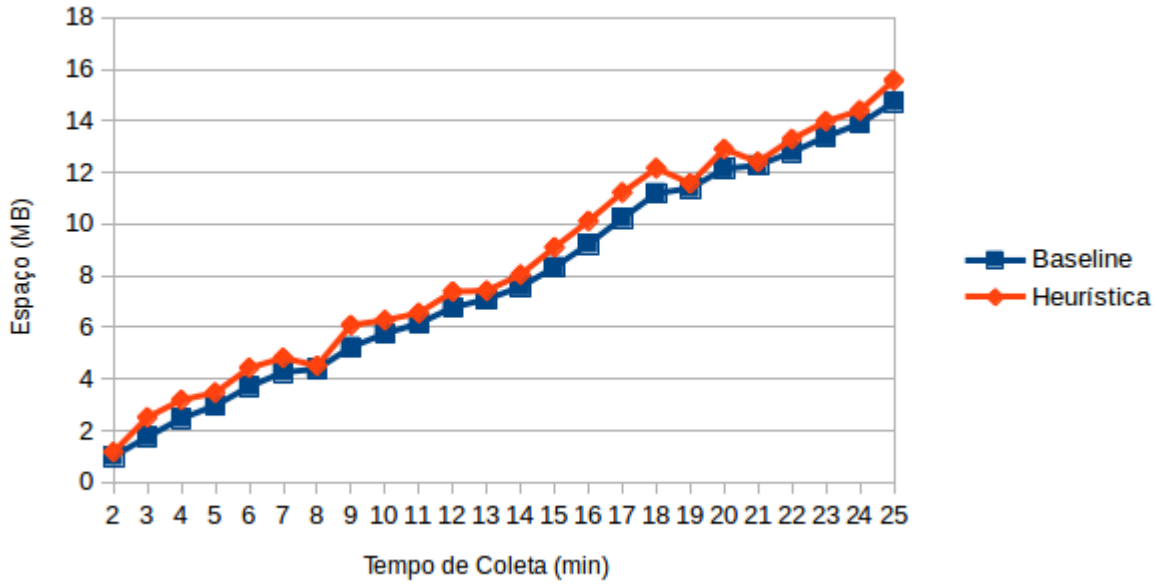

Figura 13: Memória utilizada pelos algoritmos.

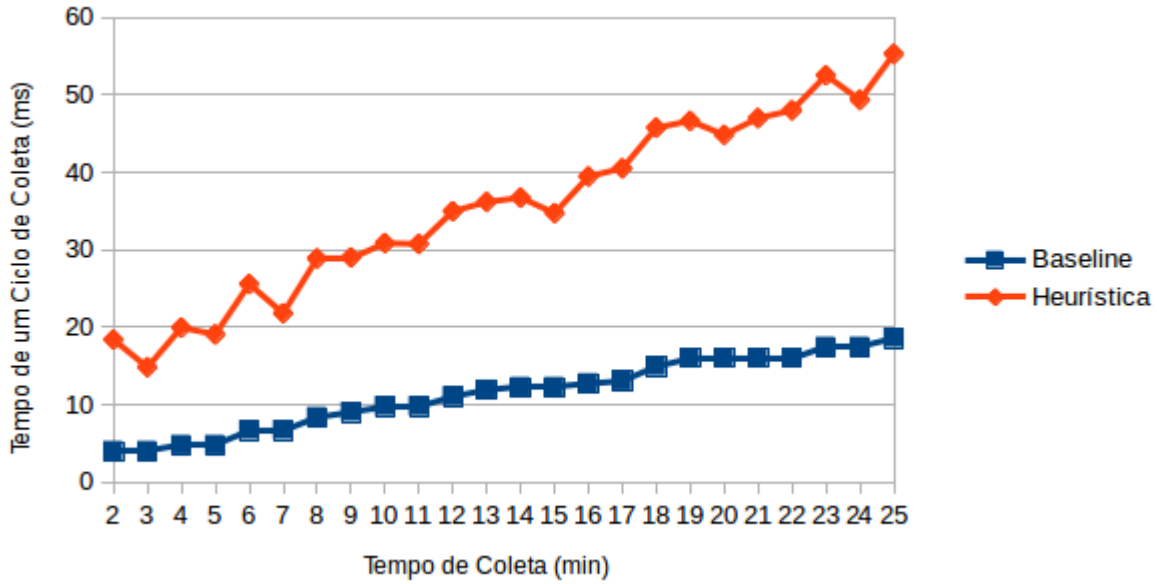

Figura 14: Tempo gasto pelos algoritmos para um ciclo de coleta (intervalo de tempo entre duas execuções do componente *manager*).

6.1 Qualidade da Resposta

Um modo de avaliar a qualidade da coleta é utilizar um ambiente controlado de teste, em que os *pageranks* das páginas já são conhecidos. Neste ambiente, podemos então comparar diversos resultados de coletas. Quanto mais rápido o *pagerank* cumulativo cresce durante uma coleta, mais páginas relevantes foram encontradas durante um determinado instante do tempo. Logo, maior a qualidade do coletor.

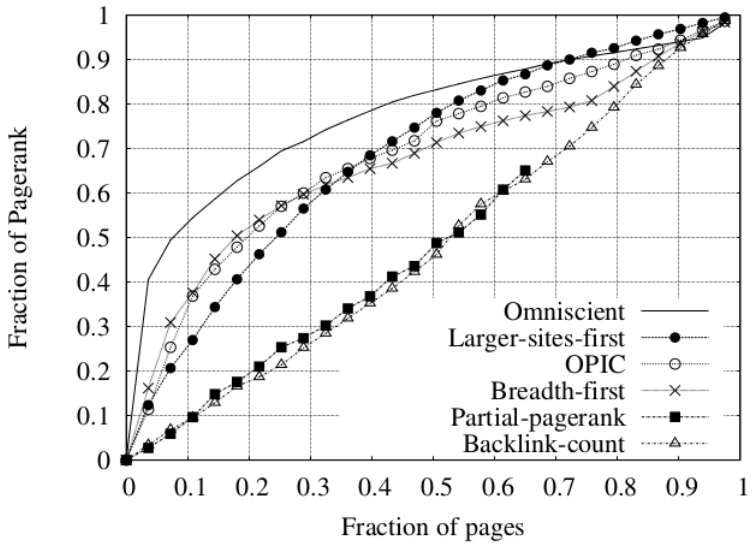

Figura 15: Gráfico com os desempenhos de diferentes estratégias para escalonamento de longo prazo [4]. A linha contínua representa a estratégia omnisciente, em que o coletar já conhece todas os *pageranks* e toma decisões baseando-se nesse conhecimento. Nenhuma estratégia pode ser melhor que a omnisciente. A estratégia *larger-sites-first* (páginas com mais *links* primeiro) é a estratégia utilizada pelo *baseline WIRE*.

As seguintes instâncias apresentaram resultados de qualidade diferentes para os algoritmos.

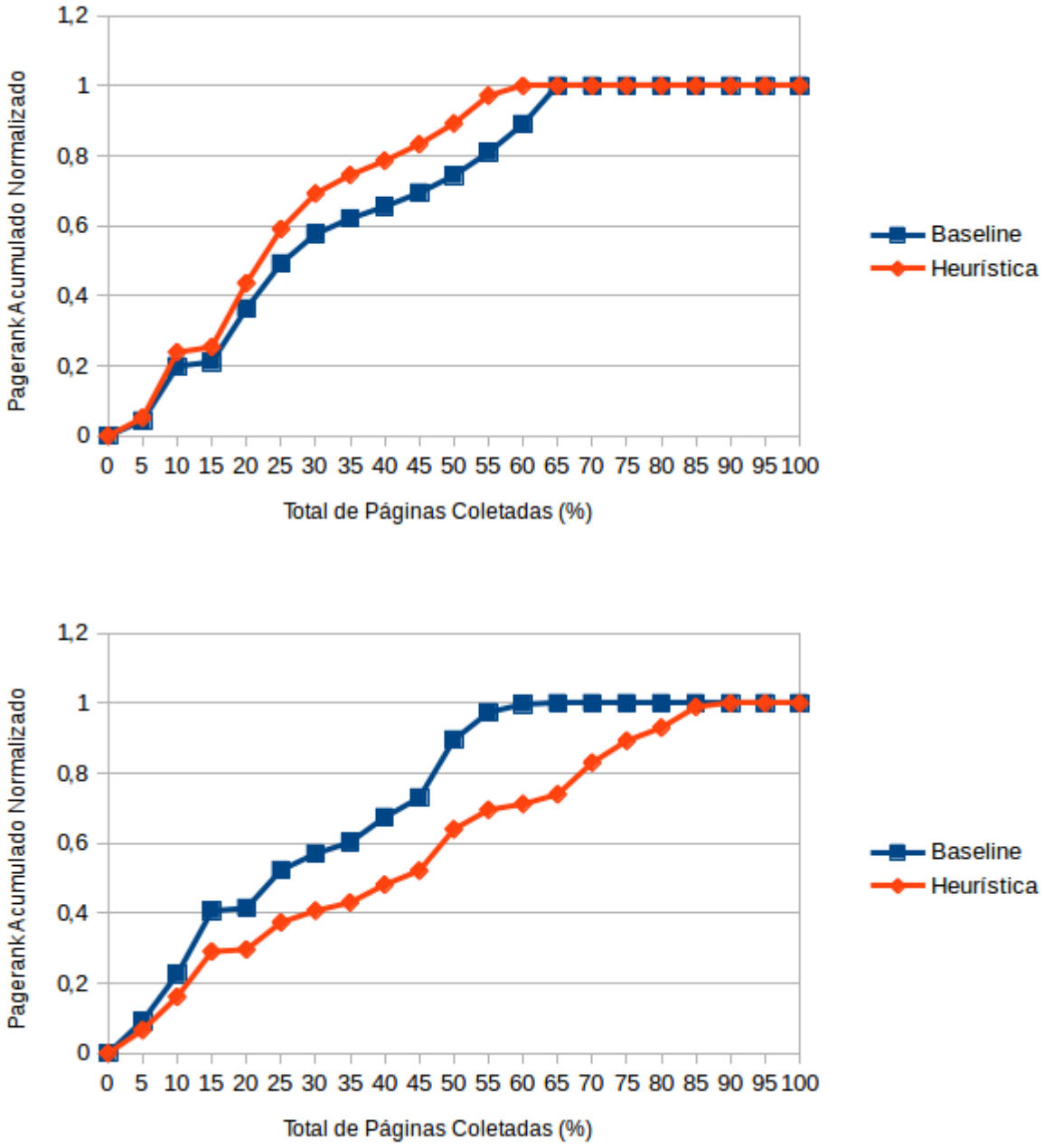

Figura 16: Qualidade dos resultados para dois grupos diferentes de páginas iniciais. Acima, o resultado da heurística teve maior qualidade. Abaixo, o algoritmo do *baseline* teve melhor resultado.

7 Trabalhos Futuros

Ao longo do projeto foram percebidas melhorias ou ideias que podem ser implementadas em trabalhos futuros. Entre elas, destacam-se: utilização de uma ontologia ao invés da contagem simples de palavras, passando a utilizar as relações entre as palavras encontradas para determinar as sentenças mais relevantes, por exemplo; detecção de *spams* (e outras informações não relevantes) durante a coleta, a fim de evitar que palavras que são comuns mas não são relevantes direcionem incorretamente a coleta para outros tópicos de interesse; e por último, reduzir, se possível, o problema da coleta da *Web* para um problema conhecido de grafos - como por exemplo, o problema do Clique - para então utilizar as técnicas mais modernas para resolução deste problema.

8 Conclusão

A coleta de páginas da *Web* mostra-se ser um problema real, moderno e desafiador. Devido ao grande volume de dados, dinamismo de conteúdo e a existência de diferentes características que podem ser aproveitadas na implementação de coletores, o problema pode ser atacado de diversas formas distintas, buscando-se o resultado mais próximo possível de uma solução exata. Durante o trabalho, tive um grande aprendizado neste campo de pesquisa da área de Recuperação de Informação, que é a área onde pretendo seguir em meu futuro acadêmico.

A pesquisa e implementações de heurísticas mostraram que mesmo ideias simples como a deste trabalho podem ter resultados de boa qualidade em relação aos *baselines* utilizados. Porém, deve-se atentar também para os casos em que tais heurísticas não superam - ou nem mesmo ficam próximas - dos resultados de um algoritmo de referência, em geral, um que já tenha sido publicado em conferências ou em revistas científicas.

Os conceitos, as técnicas e a experiência adquiridas na disciplina de *Projeto e Análise de Algoritmos* foram bastante úteis para elaboração deste trabalho, tanto em termos individuais como nas discussões em grupo com outros colegas da disciplina. Por meio deste trabalho prático, muito do aprendizado obtido teve de ser testado, revisto ou mesmo alterado durante todas as fases do projeto.

Referências

[1] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval: The concepts and technology behind search*. Pearson Education Limited, 2nd edition, 2011.

[2] Albert-Laszlo Barabasi. Linked the new science of networks, 2002.

[3] Andrei Broder, Ravi Kumar, Farzin Maghoul, Prabhakar Raghavan, Sridhar Rajagopalan, Raymie Stata, Andrew Tomkins, and Janet Wiener. Graph structure in the web. *Comput. Netw.*, 33(1-6):309–320, June 2000.

[4] Carlos Castillo. *Effective Web Crawling*. PhD thesis, School of Engineering, Santiago, Chile, November 2004.

[5] Junghoo Cho, Hector Garcia-Molina, and Lawrence Page. Efficient crawling through url ordering. In *Proceedings of the Seventh International Conference on World Wide Web 7*, WWW7, pages 161–172, Amsterdam, The Netherlands, The Netherlands, 1998. Elsevier Science Publishers B. V.

[6] Nick Craswell, Francis Crimmins, David Hawking, and Alistair Moffat. Performance and cost tradeoffs in web search. In *Proceedings of the Australasian Database Conference ADC2004*, pages 161–170, Dunedin, New Zealand, January 2004. `http://research.microsoft.com/users/nickcr/pubs/craswell_adc04.pdf`.

[7] M. Gray. Web growth, 1996.

[8] A. Gulli and A. Signorini. The indexable web is more than 11.5 billion pages. In *Special Interest Tracks and Posters of the 14th International Conference on World Wide Web*, WWW '05, pages 902–903, New York, NY, USA, 2005. ACM.

[9] Judy Johnson, Kostas Tsioutsiouliklis, and C. Lee Giles. Evolving strategies for focused web crawling. In *Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003), August 21-24, 2003, Washington, DC, USA*, pages 298–305, 2003.

[10] Marc Najork and Janet L. Wiener. Breadth-first crawling yields high-quality pages. In *Proceedings of the Tenth Conference on World Wide Web*, pages 114–118, Hong Kong, May 2001. Elsevier Science.

[11] J. Nielsen. Statistics for traffic referred by search engines and navigation directories to useit. http://www.useit.com/about/searchreferrals.html, 2004.

[12] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford University, 1998.