Influência dos Parâmetros de Controle no Desempenho de Algoritmos Adaptativos de Substituição de Páginas

Edson T. Midorikawa[1], Ricardo L. Piantola[1], Hugo Henrique Cassettari[1]

[1]Departamento de Engenharia de Computação e Sistemas Digitais
Escola Politécnica da Universidade de São Paulo
05508-900 – São Paulo – SP – Brasil

edson.midorikawa@poli.usp.br, piantola@uol.com.br, hugohc@terra.com.br

Abstract. *The research of new page replacement algorithms regained interest with the first adaptive algorithms, like SEQ, EELRU and LIRS. These algorithms characterizes in adapting its behavior according to the applications' memory access patterns and with some control parameters. The value of these control parameters are determined heuristically and, usually, are fixed during the execution of the algorithm or operating system operation. This paper presents a study of the influence of the behavior and the performance of adaptive page replacement algorithms according to a variation of the control parameters. The obtained results for the LRU-WAR algorithm show that a dynamic control of these parameters can provide some interesting benefits in the execution of some applications.*

Resumo. *A pesquisa de novos algoritmos de substituição de páginas se revigorou quando surgiram os primeiros algoritmos adaptativos, como o SEQ, EELRU ou LIRS. Estes algoritmos se caracterizam na adaptação de seu funcionamento com base no padrão de acessos à memória dos programas e em alguns parâmetros de controle. Os valores destes parâmetros de controle são determinados heuristicamente e, normalmente, são fixos durante a operação do algoritmo ou do sistema operacional. Este artigo procura estudar a variação de comportamento e o respectivo desempenho de algoritmos adaptativos de substituição de páginas em relação à variação destes parâmetros de controle. Os resultados obtidos com o algoritmo LRU-WAR mostram que um controle dinâmico destes parâmetros pode trazer benefícios interessantes na execução de certas aplicações.*

1. Introdução

Na década de 40, John von Neumann e seus colegas já discutiam o uso da hierarquia de memória no documento "*Preliminary report of the logical design of an electronic computing instrument*", onde diziam que foram "*forçados a reconhecer a possibilidade de construir uma hierarquia de memórias, cada qual com maior capacidade que o seu precedente, mas que fosse acessível menos rapidamente*." Desde então o conceito de hierarquia de memória fez com que gerações de projetistas se deparassem com o problema de buscar a melhor alternativa de gerência da memória.

No contexto da gerência de memória em sistemas operacionais, vários grupos de pesquisa têm desenvolvido trabalhos na área de algoritmos para gerência de memória virtual. Vários algoritmos para substituição de páginas têm sido propostos na literatura recente, em particular, os algoritmos adaptativos de substituição de páginas, como o SEQ, EELRU, LIRS, ARC, ClockPro, CAR, LRU-WAR e 3P [Cassettari and Midorikawa 2004b] [Cassettari and Midorikawa 2005]. Estes algoritmos impulsionaram novos trabalhos na área, com a busca de uma solução dinâmica e adaptativa para a

otimização do uso da memória pelos programas.

Cada um dos algoritmos adaptativos propostos procura solucionar o problema de um modo diferente e inovador. De uma maneira geral, estes algoritmos mudam seu comportamento com base no padrão observado de acessos à memória ou informação sobre a recência ou freqüência de acessos às páginas. Para isto, normalmente, estes algoritmos definem um ou mais parâmetros de controle para modificar seu comportamento ou estrutura de dados interna. Contudo é raro encontrarmos um estudo de como determinar o valor mais adequado destes parâmetros de controle.

O objetivo principal deste artigo é apresentar um estudo da variação do comportamento e do respectivo desempenho de algoritmos adaptativos de substituição de páginas com relação a estes parâmetros de controle.

Este artigo está organizado da seguinte forma. A seção 2 apresenta os algoritmos adaptativos de substituição de páginas, descrevendo algumas de suas características e exemplos. A seção 3 descreve os parâmetros de controle de alguns algoritmos adaptativos que são representativos na literatura recente. O estudo da variação dos parâmetros de controle sobre o desempenho e comportamento é realizado sobre o algoritmo LRU-WAR e apresentado na seção 4. A seção 5 finaliza o artigo apresentando as principais conclusões e indica alguns trabalhos futuros.

2. Algoritmos Adaptativos de Substituição de Páginas

Os algoritmos adaptativos de substituição atuam de forma dinâmica, adaptando seu comportamento de acordo com o padrão de acesso à memória em tempo de execução. Para tal, podem ser utilizadas várias técnicas adaptativas, como, por exemplo, modificar o critério de substituição, o tamanho da memória utilizada ou os parâmetros de controle do algoritmo.

O princípio da adaptação é encontrar um padrão a partir de uma informação coletada sobre as páginas usadas recentemente e então adequar o algoritmo ao comportamento destes acessos à memória. A idéia dos algoritmos adaptativos de substituição não é nova: o Atlas *loop detector* [Baylis et al 1968], predecessor dos mais recentes detectores de *looping* usados nos algoritmos SEQ [Glass and Cao 1997] e EELRU [Smaragdakis et al 1999], já poderia ser considerado uma política que se adapta ao comportamento do sistema.

Existem alguns padrões bastante explorados por esses algoritmos, tais como a freqüência de acesso às páginas (localidade temporal), a recência (localidade espacial), a probabilidade de acesso individual, a previsão de acesso futuro baseado no reuso, os acessos seqüenciais e os *loopings*. As estratégias utilizadas pelos algoritmos variam. Nos últimos doze anos, porém, surgiram várias propostas baseadas apenas na mistura de freqüência e recência, tais como LIRS [Jiang and Zhang 2002], LRFU [Lee at al 2001] e ARC [Megiddo and Modha 2003].

Pesquisas recentes nesta área utilizam ferramentas da inteligência artificial para auxiliar a adaptação. Exemplos são os sistemas baseados em regras e lógica *fuzzy*. Quando uma política de substituição de página utiliza mais de um parâmetro para a decisão, achar a relação entre os parâmetros e combiná-los torna-se um problema. Sendo assim, uma forma de combinar esses parâmetros é modelá-los para a utilização de lógica *fuzzy* [Sabeghil and Yaghmaee 2006].

3. Parâmetros de Controle

Algoritmos adaptativos de substituição de páginas se caracterizam, sobretudo, por modificarem seu critério de substituição ao longo do tempo. Tal comportamento é gerenciado através de parâmetros de controle previamente definidos e delimitados, conforme o projeto e a estratégia particular de cada algoritmo no intuito de diminuir ao máximo a ocorrência de faltas de página. Os parâmetros de controle são, portanto, parte fundamental no projeto de qualquer algoritmo adaptativo.

A Tabela 1 descreve resumidamente os parâmetros de controle associados aos algoritmos SEQ, EELRU, LIRS e LRU-WAR, detalhados a seguir nesta seção.

Tabela 1. Parâmetros de controle definidos pelos principais algoritmos adaptativos.

Algoritmo	Parâmetro	Descrição Resumida
SEQ	L	Tamanho mínimo de uma seqüência para que possa abrigar um ponto de substituição.
	M	Posição do ponto de substituição em potencial de uma seqüência (posição MRU-M).
	N	Quantidade de páginas consideradas para se verificar a taxa de crescimento de uma seqüência.
EELRU	E	Ponto referencial de substituição antecipada (*early eviction point*).
	L	Ponto referencial de substituição LRU (*late eviction point*)
LIRS	L_{hirs}	Número porcentual máximo de páginas HIRs residentes em relação ao tamanho da memória.
LRU-WAR	C	Tamanho da região protegida, sendo C+1 o tamanho mínimo de uma área de trabalho.
	L	Tamanho da região seqüencial

3.1. Algoritmo SEQ

O algoritmo SEQ procura identificar padrões espacialmente lineares de acessos seqüenciais e seleciona uma das eventuais seqüências detectadas como alvo para substituições do tipo MRU-n, critério alternativo adotado pelo algoritmo quando o critério LRU é claramente ineficiente. Para tanto, uma lista de seqüências é criada e mantida pelo mecanismo de controle associado ao SEQ, o qual define três parâmetros operacionais, a saber: **L**, que é o tamanho mínimo para que uma seqüência identificada pelo algoritmo seja considerada elegível para abrigar um ponto de substituição de páginas em potencial; **M**, que indica o ponto de substituição dentro de uma seqüência qualquer de tamanho maior do que L (ponto MRU-M, ou seja, a M-ésima página mais recentemente acessada em tal seqüência); e **N**, utilizado para determinar a taxa pontual de crescimento das seqüências com tamanhos superiores a L; a seqüência cuja N-ésima página mais recentemente carregada tiver sido carregada por último será justamente o alvo de uma possível substituição do tipo MRU-n.

Os valores padrão especificados pelos criadores do algoritmo SEQ para os parâmetros L, M e N são, respectivamente, 20, 20 e 5. O parâmetro L tem a finalidade de descartar seqüências muito pequenas – que podem simplesmente refletir um caso favorável de localidade espacial, compatível com o tamanho da memória – ou seqüências em formação, que ainda não afetam significativamente a política de substituição. A importância do parâmetro M está no fato de que, por meio dele, o algoritmo evita que sejam retiradas da memória páginas carregadas muito recentemente. Por fim, o parâmetro N tem como objetivo verificar em qual seqüência houve mais faltas de página nos últimos acessos, isto é, qual é a seqüência que está mais propícia a aumentar de tamanho levando em consideração apenas um universo próximo de acessos.

3.2. Algoritmo EELRU

O algoritmo EELRU não utiliza parâmetros com valores pré-definidos para determinar seus possíveis pontos de substituição. Por outro lado, seu engenhoso controle é realizado através de cálculos dinâmicos que estabelecem a localização dos pontos referenciais **e** *(early eviction point)* e **l** *(late eviction point)*, os quais podem ser considerados parâmetros de controle para o algoritmo.

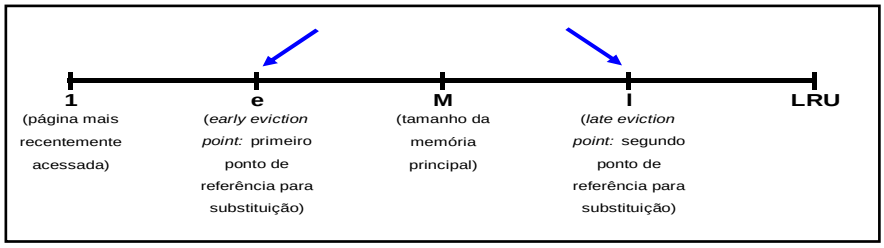


Figura 2. Eixo LRU representando a memória no contexto do algoritmo EELRU.

Consideremos um eixo de representação em que certa quantidade de páginas de memória está disposta no sentido MRU → LRU (da esquerda para a direita, conforme a Figura 2). Sendo *M* o total de memória disponível e sendo *i* uma instância do conjunto de todas as possibilidades de combinações distintas para a localização dos pontos *e* e *l*, o algoritmo EELRU determina seus pontos de substituição através do cálculo do benefício dado por $MAX((M-e_i)/(l_i-e_i))$.

O algoritmo procura os pontos *e* e *l* que maximizam o desempenho do sistema de memória, isto é, as posições mais adequadas para que haja uma substituição, de acordo com o custo/benefício calculado por meio das freqüências de acesso agregadas em trechos do eixo LRU. A página *r(e)*, referente ao ponto *e* obtido, normalmente representa a melhor solução para substituição.

3.3. Algoritmo LIRS

O objetivo do algoritmo LIRS é minimizar as deficiências apresentadas pelo algoritmo LRU utilizando um critério interessante: a chamada IRR (*Inter-Reference Recency*), que representa o número de páginas referenciadas entre os dois últimos acessos consecutivos a uma mesma página. O algoritmo pressupõe uma inércia comportamental e, de acordo com as IRRs coletadas, substitui a página que provavelmente levará mais tempo para ser novamente acessada. Isto significa que o LIRS não substitui necessariamente a página acessada há mais tempo, mas ele utiliza o histórico recente desta informação para prever quais páginas têm maiores probabilidades de acesso em um futuro breve.

Desta forma, o algoritmo classifica as páginas dinamicamente em dois grupos: páginas do tipo HIR *(High Inter-reference Recency)* e páginas do tipo LIR *(Low Inter-reference Recency)*. Páginas do tipo HIR possuem prioridade de substituição e, portanto, tendem a permanecer residentes na memória por pouco tempo. Os dois únicos parâmetros de controle do algoritmo LIRS com valores pré-definidos, percentualmente complementares entre si, referem-se justamente à quantidade de cada tipo de página que deve permanecer residente na memória: $\mathbf{L_{hirs}}$, percentual máximo de páginas HIRs residentes em relação ao tamanho de memória disponível e $\mathbf{L_{lirs}}$, percentual mínimo de páginas LIRs residentes ($1-L_{hirs}$). O valor originalmente sugerido para L_{hirs} pelos autores do algoritmo é 1% do tamanho total de memória disponível.

3.5. Algoritmo LRU-WAR

O princípio de funcionamento do algoritmo LRU-WAR é baseado na estimativa e na análise da variação do tamanho máximo do *working set* do programa. Esta estimativa é validada entre faltas de página consecutivas. Ou seja, a dimensão máxima do *working set* temporário é usado como fator decisivo na adaptabilidade do LRU-WAR.

O LRU-WAR monitora os acessos à memória e verifica, entre duas faltas de página consecutivas, a página referenciada com menor recência em relação ao seu último acesso, ou seja, identifica a posição mais alta da fila LRU que recebeu acessos neste período. Tal posição W limita uma porção da fila LRU, conhecida como área de trabalho (*working area*).

Há três estados de execução definidos: tendência LRU, tendência seqüencial e operação seqüencial. Quando a área de trabalho é maior que o parâmetro L (tamanho da região seqüencial), considera-se que o programa está reutilizando as suas páginas em memória e então fica no estado de *tendência LRU*. Neste estado, o critério LRU é adotado, e permanece até que seja detectada uma tendência de acessos seqüencial.

Quando, entre duas faltas de página, o tamanho da área de trabalho se reduz, ficando menor que L, o algoritmo entra em *tendência seqüencial*, pois subentende-se que está ocorrendo um número adicional de faltas de página sem que haja uma boa reutilização das páginas em memória. Neste estado ainda se adota o algoritmo LRU para a substituição de páginas.

Se a tendência seqüencial se mantiver por um número de faltas de página proporcional ao tamanho da área de trabalho, desde o instante em que esta tendência começou, o LRU-WAR muda para o estado de *operação seqüencial*, onde se adota o algoritmo MRU-n para substituição. Para definir qual página substituir neste estado, o algoritmo define outro parâmetro: a carência mínima C. Ela define uma região inicial da fila LRU chamada região protegida. Esta região contém páginas que não podem ser removidas da memória, assim, a menor área de trabalho aceita pelo LRU-WAR é C+1. A figura 3 ilustra as divisões lógicas da fila LRU que orientam o algoritmo. O parâmetro C também é usado na determinação do número de faltas de página para o algoritmo passar do estado de tendência seqüencial para operação seqüencial, com o objetivo de evitar que o algoritmo detecte erroneamente um padrão seqüencial de acessos.

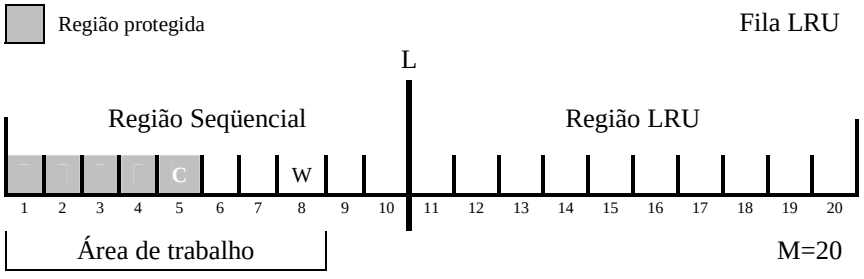


Figura 3. Divisão lógica da fila LRU utilizada pelo algoritmo LRU-WAR.

O valor padrão para C é igual a 5 e o parâmetro L é especificado como o MIN(50, M/2), onde M é o tamanho total da memória em número de páginas.

4. Análise da Variação dos Parâmetros de Controle

Nesta seção é apresentada a análise realizada sobre o algoritmo adaptativo de substituição de páginas LRU-WAR. Iniciamos com a descrição dos programas usados para a avaliação e, em seguida, com a apresentação e análise dos resultados obtidos.

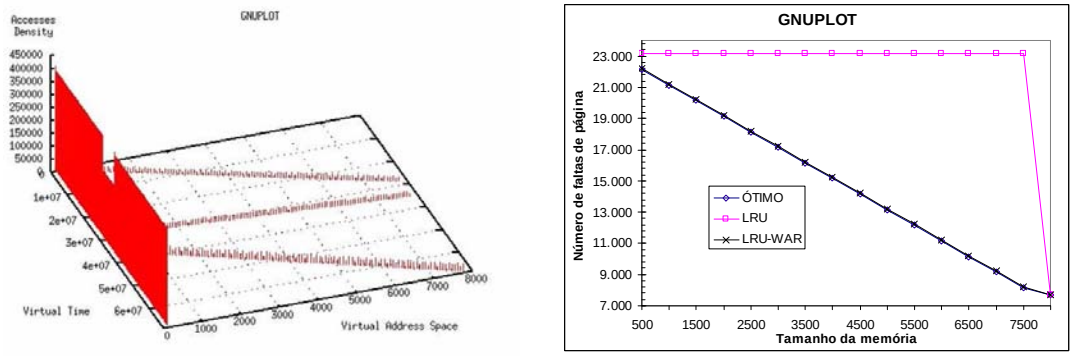
4.1. Configuração dos Testes

A avaliação dos algoritmos foi executada usando-se três diferentes *traces* de programas, a saber, gnuplot, grobner e sprite (Tabela 2). Estes foram escolhidos com base no seu padrão de acessos e desempenho dos diversos algoritmos de substituição de páginas estudados [Cassettari, 2004]. As simulações foram realizadas com as ferramentas disponibilizadas pelo ambiente Elephantools [Cassettari and Midorikawa, 2004a].

Tabela 2. Descrição dos *traces* utilizados.

Trace	Descrição	Origem	Total de páginas
gnuplot	Trace relativo à geração de um gráfico em *postscript*.	VMTrace	7718
grobner	Programa para reorganização de fórmulas baseado em funções base de Grobner.	VMTrace	67
sprite	Proveniente do sistema de arquivos de rede Sprite. Contém requisições a um servidor de arquivos a partir de várias estações de trabalho cliente em um período de dois dias.	LIRS	7075

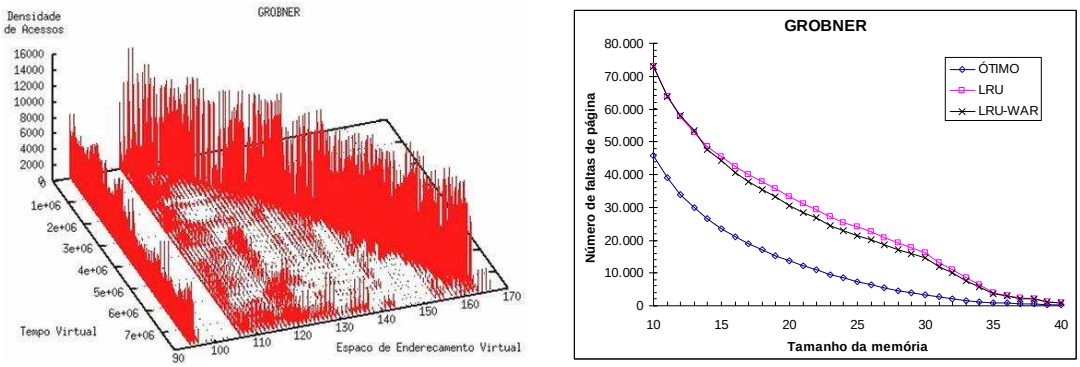
O programa **gnuplot** se caracteriza por apresentar dois padrões de acessos bem definidos: um conjunto de páginas com forte localidade temporal ao longo da execução (com endereços virtuais baixos) e um padrão de acessos seqüencial devido à execução de laços que percorrem três vezes quase todo o espaço de endereçamento virtual (figura 4.a). Esta característica faz com que o algoritmo LRU apresente um desempenho "em forma de degrau" porque, devido ao fato deste não ser adequado a este padrão de acessos seqüencial, o algoritmo sempre faz uma escolha ruim, até que o espaço de memória disponível seja igual ou maior ao tamanho total do programa (figura 4.b). O gnuplot apresenta o padrão de acessos mais adequado ao LRU-WAR, que apresenta um desempenho muito próximo do algoritmo Ótimo. O algoritmo Ótimo, também conhecido como OPT ou MIN, se refere ao algoritmo teórico que apresenta o melhor desempenho possível entre os algoritmos on-line.



(a) Padrão de acessos à memória. (b) Desempenho do LRU-WAR, LRU e Ótimo.

Figura 4. Características do programa Gnuplot.

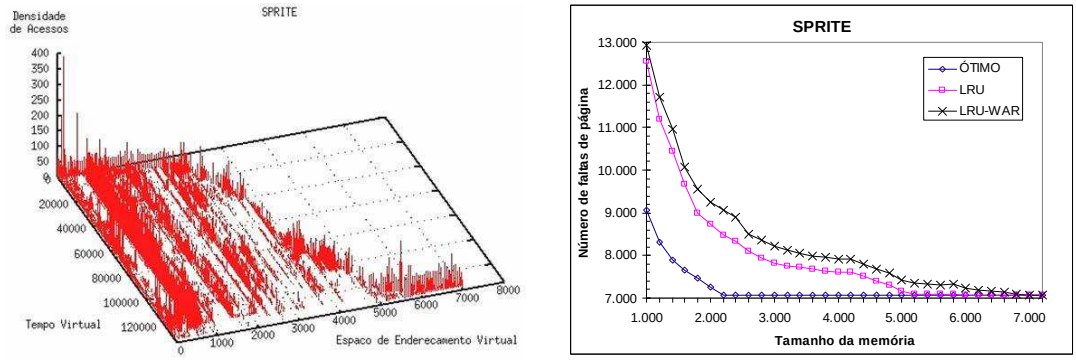
O programa **grobner** também apresenta um padrão de acessos seqüencial bem definido, como mostrado na figura 5.a. Contudo, ao contrário do gnuplot, este padrão seqüencial está intercalado com outros padrões de acesso à memória. Entre os outros padrões observa-se um com acessos a poucas páginas com forte localidade temporal. Esta característica faz com que o LRU-WAR apresente um ganho modesto de desempenho em relação ao LRU, não chegando a mais de 12% para tamanho de memória igual a 28 páginas (figura 5.b). Este programa foi escolhido, pois se trata de um caso em que o LRU-WAR poderia explorar melhor o padrão seqüencial presente.



(a) Padrão de acessos à memória. (b) Desempenho do LRU-WAR, LRU e Ótimo.

Figura 5. Características do programa Grobner.

O terceiro programa escolhido foi o **sprite**. O padrão de acessos observado para este programa mostra um grande conjunto de páginas que são acessadas com uma certa freqüência, mas sem apresentar um padrão destacado e com intervalos irregulares (figura 6.a). O LRU-WAR comete decisões equivocadas, cujas conseqüências são amenizadas pelo mecanismo de detecção de erros. Contudo, o desempenho final do algoritmo é pior que o LRU, chegando a cerca de 7% para tamanho de memória igual a 2200 páginas (figura 6.b).



(a) Padrão de acessos à memória. (b) Desempenho do LRU-WAR, LRU e Ótimo.

Figura 6. Características do programa Sprite.

A seguir apresentamos os resultados das análises da variação de desempenho do LRU-WAR com a modificação dos valores dos seus parâmetros de controle C e L.

4.2. Resultados Obtidos e Análises

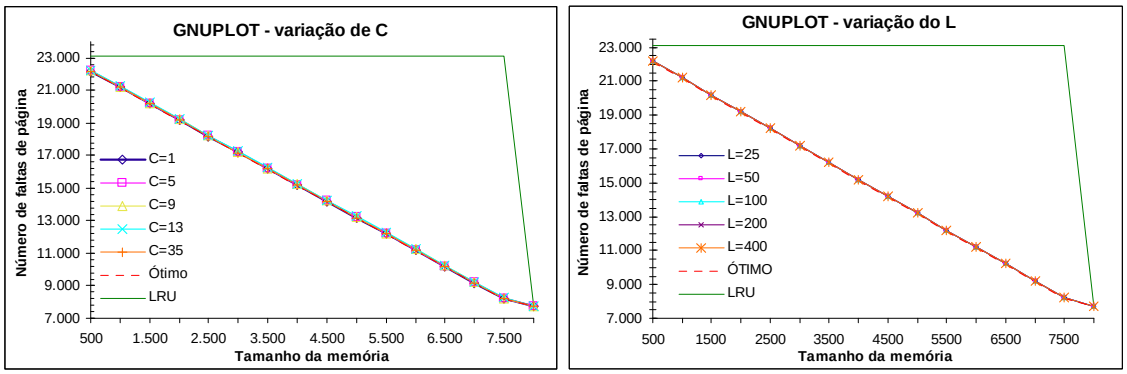
Para cumprir os objetivos definidos neste estudo, apresentamos aqui os resultados e

análises do estudo do desempenho do algoritmo LRU-WAR com a variação dos seus parâmetros de controle.

Gnuplot

O LRU-WAR é um algoritmo que alcança excelente desempenho quando um programa apresenta padrões de acessos seqüenciais predominantes. Análises anteriores já demonstraram este comportamento [Cassettari and Midorikawa 2004]. Como o programa gnuplot apresenta acessos seqüenciais do inicio ao fim, o LRU-WAR apresentou desempenho muito próximo do algoritmo Ótimo: apenas uma diferença constante em relação ao número de faltas de página.

Apesar do excelente desempenho alcançado pelo LRU-WAR, foi conduzido um experimento com a alteração dos valores dos parâmetros C e L do algoritmo (figura 7).



(a) Variação da carência mínima C. (b) Variação do tamanho da região seqüencial (L).

Figura 7. Variação dos parâmetros para o gnuplot.

Como pode ser observado, o desempenho do LRU-WAR permaneceu bem próximo do algoritmo Ótimo, mesmo com uma variação ampla dos parâmetros. Apesar disto, foi possível obter ganhos modestos em relação aos valores padrão: para C=35 e C=1, obtivemos um número menor de faltas de página de 23 e 21, respectivamente. Isto fez com que ficássemos, respectivamente, a uma distância de apenas 42 e 44 faltas de páginas do algoritmo Ótimo de um total de 68.458.509 acessos à memória.

O estudo realizado do efeito da variação da carência mínima C mostrou que, para valores maiores que 50, o algoritmo LRU-WAR se comportou como o algoritmo LRU para o gnuplot. A razão deste efeito é devido ao fato que quanto maior a carência, maior é o tamanho da região protegida. Assim o algoritmo LRU-WAR, mesmo em operação seqüencial, irá substituir uma página MRU em uma posição mais alta da fila (W+1, com W»C), o que é próxima da página que seria substituída caso estivesse usando a algoritmo LRU.

As variações do L não surtiram efeito para esse programa e o LRU-WAR apresentou o mesmo desempenho para os valores analisados (figura 7.b). Uma razão para este resultado é o fato de que, para o gnuplot, o LRU-WAR normalmente está no estado de operação seqüencial. Uma variação de L somente muda o momento em que há a mudança do estado de tendência LRU para a operação seqüencial, resultando em um comportamento bastante similar em todos os casos.

Grobner

O programa grobner se caracteriza por possuir trechos com padrão seqüencial intercalado com referências a muitas outras páginas, utilizando quase todas as páginas do seu espaço de endereçamento virtual. O desempenho do LRU-WAR com parâmetros padrão é ligeiramente melhor que o do LRU (figura 5.b). Desta forma, algoritmos que procuram detectar padrões de acessos seqüenciais podem obter bons resultados.

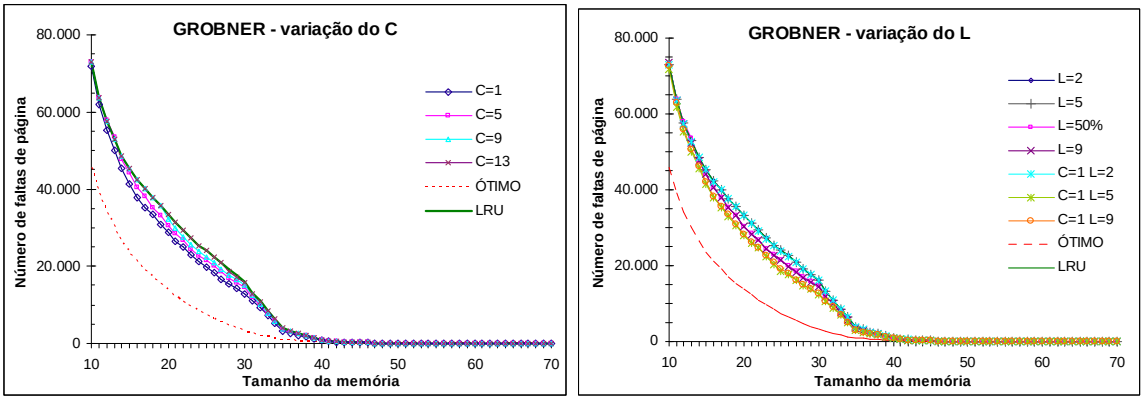
Como o Grobner possui apenas 67 páginas referenciadas, neste estudo, vamos desconsiderar tamanhos de memória muito pequenos (menores que 10) e muito grandes (maiores que 67) pelo fato de não agregar informações relevantes para as comparações.

Ao analisarmos os efeitos da variação da carência mínima C, verificamos que quanto menor o valor de C, o LRU-WAR detecta mais rapidamente os trechos seqüenciais do grobner e, por conseqüência, melhor o seu desempenho (figura 8.a). A razão para este comportamento é que, depois de estar em tendência seqüencial, quanto menor a carência mínima, mais rapidamente o algoritmo entra em operação seqüencial.

Quando C=1, o LRU-WAR supera o LRU em todos os tamanhos de memória até 44 páginas, quando então as faltas de página se convergem para todos os valores de C estudados (figura 8.a). O maior ganho de desempenho foi obtido para uma memória de 35 páginas, com um ganho de quase 21% em relação ao LRU. Em comparação com os valores padrão do LRU-WAR, uma carência C=1 também apresenta desempenho melhor, com um ganho de até 13% para uma memória de 38 páginas.

Para C=9 e C=13, o LRU-WAR apresenta uma tendência de comportamento para o LRU, porém ainda obtendo resultados satisfatórios em alguns tamanhos de memória.

Para as simulações para estudo do parâmetro L, foram definidos valores fixos do tamanho da região seqüencial até o valor 10, devido ao tamanho total de páginas do programa. Os resultados são apresentados na figura 8.b.



(a) Variação da carência mínima C. (b) Variação do tamanho da região seqüencial (L).

Figura 8. Variação dos parâmetros para o grobner.

Quando L=9, o LRU-WAR superou o LRU em todos os tamanhos de memória até 43 páginas, exceto quando M=10. Obteve o maior ganho de desempenho para M=35, superando o LRU em 13,2%. De uma maneira geral, valores baixos de L não apresentaram melhor desempenho que o LRU. Este comportamento pode ser explicado pelo fato do padrão de acessos predominante do grobner estar intercalado com outros padrões. Assim, um valor muito pequeno de L levaria o LRU-WAR a entrar em operação seqüencial de forma equivocada.

Em seguida, dados os ganhos de desempenho obtidos com a variação individual de C e L, resolvemos testar a variação conjunta de ambos os parâmetros. O melhor resultado obtido foi com C=1 e L=5, com um ganho máximo de quase 24% em relação ao LRU para M=35 páginas. Este valor representou um ganho de mais de 15% em relação aos valores padrão do LRU-WAR (figura 9). Estes resultados mostram que é possível otimizar o desempenho do LRU-WAR, de acordo com o padrão de acessos do programa.

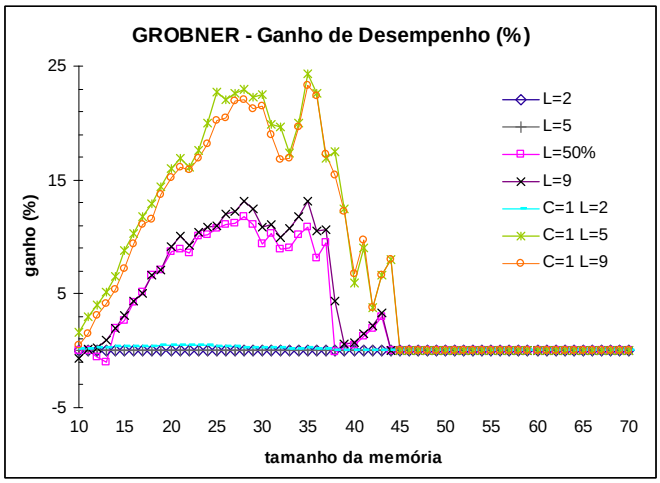


Figura 9. Ganho de desempenho do LRU-WAR no grobner em relação ao LRU.

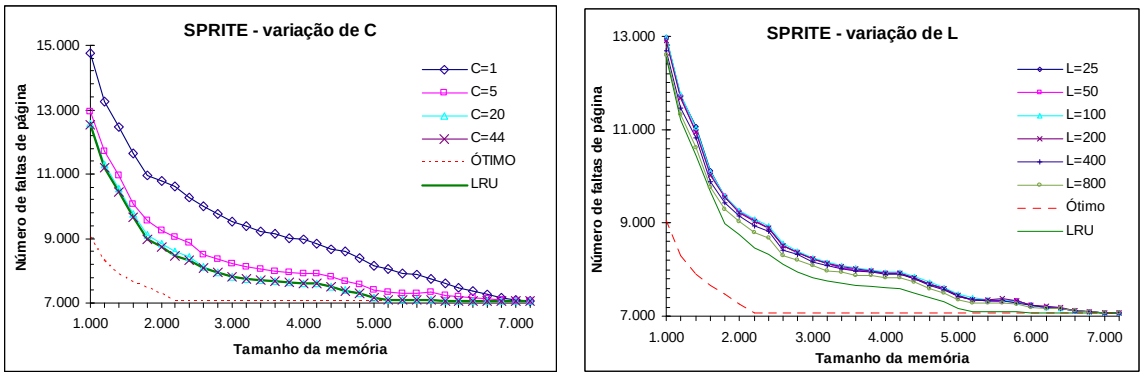
Sprite

O programa sprite tem baixa localidade temporal e acessa um grande conjunto de páginas com alta freqüência. Neste caso o algoritmo LRU-WAR toma decisões equivocadas entrando muitas vezes em operação seqüencial, e assim apresenta desempenho pior que o LRU. A freqüência de acessos a vários conjuntos de páginas faz com que a área de trabalho do LRU-WAR diminua, o que faz o algoritmo entrar em tendência seqüencial e, posteriormente, em operação seqüencial. Como a localidade temporal não é uniforme, páginas que serão acessadas em um futuro próximo são logo descartadas. O LRU-WAR com valores padrão para os parâmetros de controle teve um desempenho médio pior de cerca de 4% em relação ao LRU (figura 6.b).

A análise da variação de C mostrou que quanto maior o valor da carência mínima C, o desempenho apresentado pelo LRU-WAR fica mais próximo do LRU (figura 10.a). Nos testes realizados, o melhor desempenho foi obtido para C=44, onde o LRU-WAR igualou os resultados do LRU.

Uma explicação deste resultado é que um valor alto de C faz com que o LRU-WAR diminua a freqüência com que entra em operação seqüencial, assim permanecendo mais tempo em tendência LRU. Além disto, um valor alto de C faz com que a região protegida contenha páginas que saíram recentemente do *working set* e que podem ser reutilizadas em breve.

A análise da variação do comportamento do LRU-WAR com o tamanho da região seqüencial L mostra que à medida que aumentamos este tamanho, melhor é o desempenho observado (figura 10.b). Contudo, mesmo com L=800, o LRU-WAR não igualou o desempenho do LRU.



(a) Variação da carência mínima C. (b) Variação do tamanho da região seqüencial (L).

Figura 10. Variação dos parâmetros de controle para o sprite.

Este comportamento pode ser explicado pelo fato de que um valor muito alto de L faz com que o algoritmo demore mais para sair do modo de operação seqüencial. Desta forma, a área de trabalho irá conter muitas páginas, ou seja, apresentando um valor relativamente alto para W. Então, embora o algoritmo adote a política de substituição MRU-n, como o ponto de substituição é definido pelo valor de W, a página selecionada para substituição será uma página próxima ao extremo LRU da fila LRU.

5. Conclusões e Trabalhos Futuros

Este artigo apresentou um estudo sobre a variabilidade do desempenho de algoritmos adaptativos de substituição de páginas em relação aos seus parâmetros de controle. Este estudo foi conduzido sobre o algoritmo LRU-WAR, usando três programas com características bem diferentes de seus respectivos padrões de acesso à memória.

A principal contribuição deste trabalho é mostrar que, apesar dos algoritmos adaptativos sejam bem configurados e apresentarem desempenhos bem melhores que os algoritmos tradicionais, é possível melhorar o seu comportamento para situações específicas. As análises da variação dos parâmetros C e L do LRU-WAR mostram que os valores padrão do algoritmo foram bem definidos, levando a um desempenho muito bom em geral. Contudo, para cada programa estudado, foi possível ajustar os valores destes parâmetros de forma a melhorar o desempenho. Mesmo para programas que exibem um padrão de acessos mais adequado ao algoritmo, como o gnuplot, um ajuste dos parâmetros resultou na diminuição do número total de faltas de páginas.

Para o programa grobner, que apresenta um padrão de acessos seqüencial predominante mas intercalado a outros, um ajuste dos parâmetros possibilitou uma melhora de desempenho de até 15% em relação aos valores padrão. Com isto, o LRU-WAR apresentou um desempenho até 24% melhor que o algoritmo LRU. A análise feita com o programa sprite foi bastante interessante: mesmo para programas com padrões de acesso à memória não adequados ao LRU-WAR, foi possível ajustar os parâmetros de forma a, no pior caso, aproximar o seu desempenho ao do LRU.

Os resultados mostram que o LRU-WAR é um algoritmo adaptativo de substituição de páginas com muitos recursos para o controle do seu comportamento, de acordo com o padrão de acessos à memória das aplicações.

Para dar continuidade a este trabalho, alguns projetos complementares podem ser desenvolvidos. Uma primeira sugestão é conduzir este mesmo estudo para um conjunto maior de aplicações, de forma a obtermos uma correlação mais detalhada dos valores dos parâmetros de controle e o comportamento exibido pelo algoritmo. Contudo, uma atividade que está diretamente ligada aos resultados obtidos é o desenvolvimento de um algoritmo dinâmico de ajuste dos parâmetros de controle de acordo com certas informações da aplicação em execução. Dados sobre o padrão de acessos à memória, tamanho e composição do *working set* ou tipo de localidades presentes no programa poderiam ser utilizados por este algoritmo para otimizar o desempenho do LRU-WAR.

Referências

Baylis, M. H. J., Fletcher, D. G., and Howarth, D. J. (1968). *Paging studies made on the I.C.T. ATLAS Computer*. In Information Processing, IFIP Congress Booklet D, pages 831-837.

Cassettari, H. H. (2004). *Análise da Localidade de Programas e Desenvolvimento de Algoritmos Adaptativos para Substituição de Páginas*. Dissertação de Mestrado. Escola Politécnica da Universidade de São Paulo, 2004.

Cassettari, H.H. and Midorikawa, E.T. (2004) "*Caracterização de Cargas de Trabalho em Estudos sobre Gerência de Memória Virtual*", In Anais do III Workshop em Desempenho de Sistemas Computacionais e de Comunicação (WPerformance 2004), Salvador, BA.

Cassettari, H.H. and Midorikawa, E.T. (2004) "*Algoritmo Adaptativo de Substituição de Páginas LRU-WAR: Exploração do Modelo LRU com Detecção de Acessos Seqüenciais*". In: Anais do I Workshop de Sistemas Operacionais (WSO 2004), Salvador, BA.

Cassettari, H.H. and Midorikawa, E.T. (2005) "*Algoritmo de Substituição de Páginas 3P: Acrescentando Adaptatividade ao Clock*". In: Anais do II Workshop de Sistemas Operacionais (WSO 2005), São Leopoldo, RS.

Glass, G. and Cao, P. (1997) "*Adaptive Page Replacement Based on Memory Reference Behavior*", In Proc. of the ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS'97), Seattle, pp.115-126.

Jiang, S. and Zhang, X. (2002) "*LIRS: An Efficient Low Inter-Reference Recency Set Replacement Policy to Improve Buffer Cache Performance*", In Proc. of the ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS'02), Marina Del Rey, pp.31-42.

Lee, D. et al. (2001) "*LRFU: a spectrum of policies that subsumes the Least Recently Used and Least Frequently Used policies"*. IEEE Transactions on Computers, vol.50, n.12, p.1352-1361.

Megiddo, N. and Modha, D. S. (2003) "*ARC: A Self-Tuning, Low Overhead Replacement Cache*", In Proc. of the USENIX Conference on File and Storage Technologies (FAST'03), San Francisco, pp.115-130.

Sabeghil, M. and Yaghmaee (2006), M. H. "*Using fuzzy logic to improve cache replacement decisions*". IJCSNS International Journal of Computer Science and Network Security, Seoul, v.6, n.3A, pages182-188.

Smaragdakis, Y., Kaplan, S., and Wilson, P. (1999) "*EELRU: Simple and Effective Adaptive Page Replacement*", In Proc. of the ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS'99), Atlanta, pp.122-133.