# Utilizando IA para jogar blackjack

Gustavo Lopes Rodrigues
Instituto de Ciências Exatas e Informática
Pontifícia Universidade Católica
de Minas Gerais (PUC-MG)
Belo Horizonte, Brasil
Email: gustavolr@gmail.com

Homenique Vieira Martins
Instituto de Ciências Exatas e Informática
Pontifícia Universidade Católica
de Minas Gerais (PUC-MG)
Belo Horizonte, Brasil
Email: homenique.t@gmail.com

***Abstract*—Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.**

***Index Terms*—Keywords, here, separated, by, comma**

## I. Introdução

[1].

A Cache é um dispositivo de acesso rápido, que fica localizado dentro do processador, com a intenção de reduzir o acesso do processador à memória principal, que demanda um tempo de acesso muito superior à cache. Criando uma referência da localidade do dado na memória principal, até mesmo gravando esses dados, porém existe um limite de quantos dados podem ser armazenados. Para isso é necessário uma política de armazenamento para reocupar espaço quando necessário.

Desde a invenção dos computadores, a busca por otimizar os processos que são executados nos hardwares se tornou foco, considerando que simples ajustes podem render grandes ganhos de desempenho. Dessa forma, um algoritmo eficiente na leitura e escrita de dados na cache é de grande importância.

Com isso em mente, pensamos em políticas de substituição clássicos, LRU e o FIFO, para averiguar qual possui melhor desempenho em diferentes cenários. Começamos pensando em aumentar gradativamente o número de instruções que acessam as palavras na RAM para serem escritas na cache. Dessa forma, seria simples visualizar como o número de instruções influencia no tempo de acesso aos dados. Além disso, pensamos também em vários cenários com caches de diferentes tamanhos, desde o mínimo de espaço possível, até uma cache de mesmo tamanho que a memória RAM.

O resto do papel está organizado da seguinte maneira. Seção 2 irá mostrar pesquisas correlatas, apresentar de maneira breve o assunto de cada e se existe algo que possa contribuir com a pesquisa atual. Seção 3 apresenta as arquiteturas utilizadas para a realização dos testes, assim também com os cenários que foram testados. Resultados experimentais são colocados na Seção 4. Por fim, a Seção 5 dá conclusão a nossa pesquisa.

## II. Trabalhos Correlatos

A partir de James E. Smith e James R. Goodman [**?**], tanto os algoritmos de LRU, quanto o de FIFO para uma cache, possuem o pior desempenho possível, quando comparados a qualquer outro algoritmo. O exemplo principal deles foi mostrar como mapeamento aleatório consegue ser várias vezes mais eficaz, uma vez que programas que precisam reescrever muitas vezes na cache e remover um valor que será reutilizado em um futuro muito próximo.

Considerando que [**?**] é um artigo escrito em 1985, muitas coisas mudaram e não pudemos usar mais esses resultados como verdadeiros em situações do dia-a-dia. Tanto o LRU quanto o FIFO se mostram ineficientes quando os programas se tornam maiores do que o tamanho de toda a cache do processador (resultado apresentado pelo mesmo artigo). Hoje, temos caches muito maiores do que naquela época, com isso, o principal foco da pesquisa do Smith e do Goodman acabou perdendo o seu valor nos dias de hoje, já que estava sendo considerando caches extremamente pequenas, na maior parte das vezes inferior ao tamanho de aplicações comuns do cotidiano.

Por outro lado, o artigo [**?**], por ser de 2008 poderia ser mais promissor em apresentar dados que ajudariam em nossa pesquisa, porém não foi isso que aconteceu. A pesquisa da Universidade de Rennes faz também análise da memória Cache em relação as políticas de substituição. Entretanto, as políticas utilizadas(fora a LRU) são diferentes e não há a utilização da FIFO. Além de que, o objetivo da pesquisa é de usar dados teóricos para melhorar um método de análise de instrução estática de cache usando Pseudo LRU e a política aleatória de troca. E mesmo assim, o próprio artigo provou que métodos não LRU demonstraram uma significante perca de precisão.

Em contrapartida, o artigo de Samira Mirbagher Ajorpaz [**?**] faz uma comparação qualitativa entre algumas políticas de substituição com a GHRP, a qual o artigo pretende discorrer sobre, e mostrar a sua eficiência quanto às demais política entre ela e a LRU. A partir desse cenário, o LRU se mostrou uma política bem intermediária, e em alguns casos seus resultados sendo próximos a valores de uma política de valores aleatórios. Podemos até analisar que: pela quantidade de 1.000 instruções e o tamanho da cache este é um resultado esperado,

TABLE I
CARACTERÍSTICAS GERAIS DA ARQUITETURA I

| Especificações da Arquitetura | Partes da Arquitetura | | |
|---|---|---|---|
| | *Processador* | *CPU* | *Trace* |
| Tamanho da palavra | — | 4 | 4 |
| processorContains | 0 | — | — |
| Ciclos por escrita | 0 | — | — |

TABLE II
CARACTERÍSTICAS GERAIS DA ARQUITETURA II

| Especificações da Arquitetura | Partes da Arquitetura | |
|---|---|---|
| | *Memória Principal* | *Cache* |
| Tamanho da linha / bloco | 1 | 1 |
| Ciclos por leitura | 1 | 1 |
| Ciclos por escrita | 2 | 2 |
| Tempo do ciclo | 10 | 1 |
| Tamanho da memória | 16 | *2 |
| Associatividade | — | 2 |
| Politica de escrita | — | Write-Through |
| Politica de substituição | — | *FIFO |

* Os valores com o asterisco foram os valores modificados.

TABLE III
DADOS MODIFICADOS

| Especificações da Arquitetura | Valores | |
|---|---|---|
| | Tamanho da Memoria | Politica de Substituição |
| Cenário 1 | 2 | FIFO |
| Cenário 2 | 2 | LRU |
| Cenário 3 | 4 | FIFO |
| Cenário 4 | 4 | LRU |
| Cenário 5 | 8 | FIFO |
| Cenário 6 | 8 | LRU |
| Cenário 7 | 16 | FIFO |
| Cenário 8 | 16 | LRU |

pois a LRU é um tipo de política que não é eficiente em gerenciar valores que vão ser mantidos na memória em caches muito pequenas. O que podemos notar no nosso trabalho é que os resultados não diferenciam muitos da FIFO, o que era inesperado, pois acreditava-se que o resultado fosse ser muito pior em relação ao LRU Image[1] [2] [3] [4]

## III. METODOLOGIA OU PROPOSTA DE ARQUITETURAS

Para a elaboração de teste utilizamos o simulador-Amnésia para elaborar algumas arquiteturas (tabelas I e II) que a partir desses cenários serem executados executamos os testes (tabela III)

Ao modificar o valor do tamanho da memória e da política de substituição da cache, é possível criar 8 cenários distintos para se avaliar o impacto do tamanho da cache no desempenho dos métodos de substituição. Sendo assim, com os cenários propostos foram realizados diversos testes com o intuito de analisar a localidade temporal e localidade espacial nas arquiteturas.

## IV. AVALIAÇÃO DOS RESULTADOS

Nos gráficos subsequentes, há uma demonstração de como a memória cache se comporta ao ser usada com um grande conjunto de instruções de *load* e *store* (maior que sua capacidade) em cenários com caches de 2, 4, 8 e 16 linhas. Para

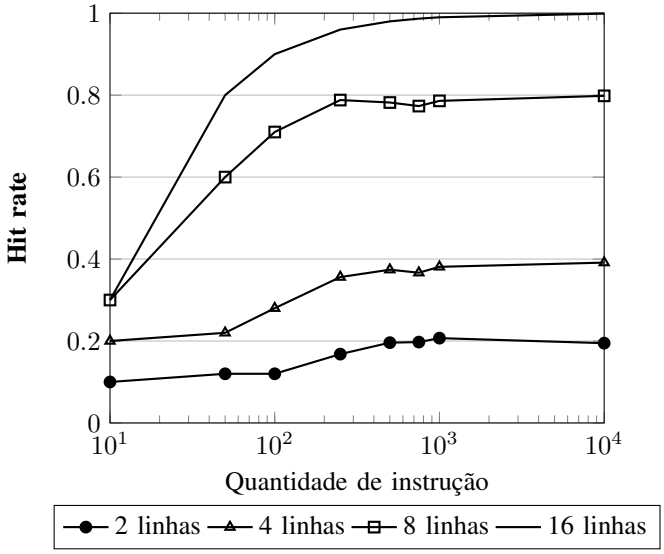


Fig. 1. ***Cache com algoritmo de substituição FIFO***

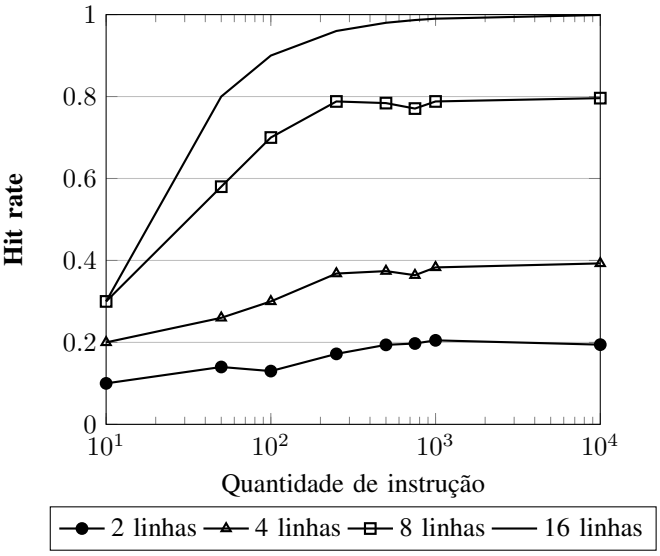


Fig. 2. ***Cache com algoritmo de substituição LRU***

fazer esses testes utilizamos um conjunto de 10, 50, 100, 250, 500, 750, 1.000 e 10.000 instruções.

### A. Gráficos de hit rate por número de instruções

Os dois gráficos mostrados foram montados usando os valores que extraímos. Nota-se que os resultados foram muito próximos, aos resultados de James E. Smith e James R. Goodman [**?**]. Considerando que no início, pensávamos que haveria uma diferença significativa entre os algoritmos LRU e FIFO para substituição de valores na cache, achamos que nossos testes estavam errados e que precisaríamos refazê-los, ao ler o artigo vimos que os nossos resultados na verdade estavam bem dentro do esperado.

Nossos testes apenas foram feitos usando caches completamente associativos, pensamos em fazer com a associatividade

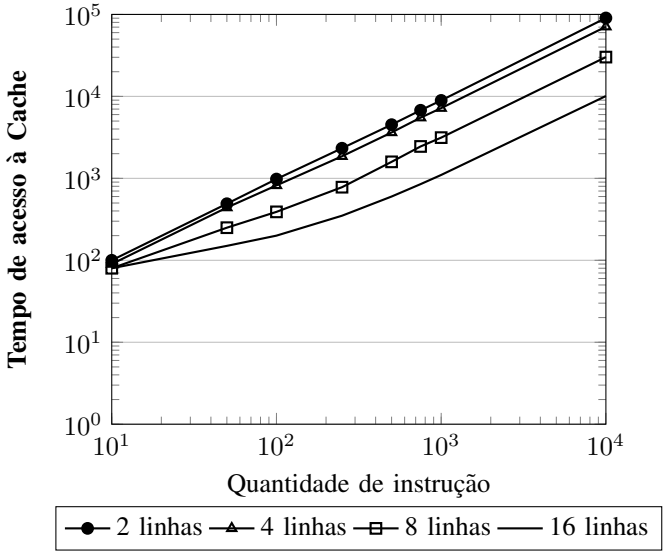


Fig. 3. ***Tempo total de acesso aos dados com algoritmo FIFO***

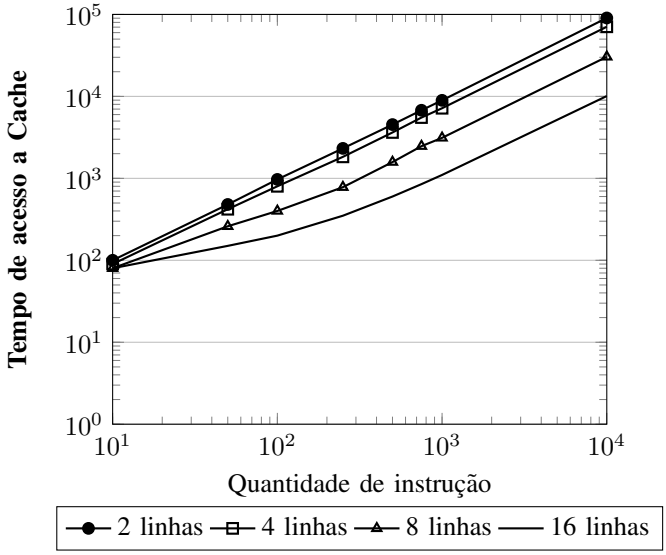


Fig. 4. ***Tempo total de acesso aos dados com algoritmo LRU***

por conjunto e direta, mas depois de resultados tão inesperados, pensamos em pesquisar outros artigos relacionados que já tivessem feito esses testes para comparar os resultados e ver se os resultados seriam diferentes. Ao ver os dados do artigo [**?**], descobrimos que teríamos valores bem semelhantes dentro dessas três arquiteturas.

Para colocar em perspectiva nossos resultados, os dados em [**?**] foram medidos usando benchmark mantido pelos grupos de pesquisa, considerando uma cache de 2KB com 32 bytes de linhas. Eles são expressos em termos da porcentagem de hits na cache detectada pela análise feita pelo grupo. A política de LRU se demonstrou mais eficiente do que a PLRU e a política aleatória de troca, em apenas alguns casos que não houve diferença entre as análises, porém na pesquisa d [**?**] foi utilizado também como parâmetro o tempo de vida mínimo de um elemento na cache(*Minimum life span)*)

### *B. Tempo total de acesso aos dados por quantidade de instruções*

Esses dois últimos gráficos, seguindo a mesma linha do hit rate, mantêm proximidade dos valores no tempo de acesso dos dados, desde copiar da RAM para a cache até acessar esse valor no processador. Dentro dos valores que experimentamos, os valores se mantiveram bem próximos. O tempo de acesso aos dados se mostrou fora do padrão apenas no caso da cache com 16 linhas (mesmo tamanho que o processador) houve uma grande queda no tempo de acesso dos dados. Em todos os outros casos, o tempo se manteve muito próximo, mesmo que entre caches de 2 a 8 linhas de tamanho.

Esses dadas nos mostram que para ter algum ganho significativo de performance seria necessário ter uma cache de tamanho próximo ao da RAM. Seria algo completamente impraticável considerando a diferença de preço por unidade de memória entre as duas. O cache se apresenta com gasto de tempo bastante próximos quando em tamanhos usados em processadores comerciais. Demonstrando que independente desses algoritmos possuem um maior hit hate em caches maiores. Dessa forma, aumentar a cache importa mais do que o algoritmo escolhido.

## V. Conclusões

Após a realização dos testes em todos os cenários propostos, e uma análise dos gráficos apresentados, houve um receio de que o ponto da pesquisa não havia sido alcançado, já que os resultados se mostraram pouco expressivos. Em nosso caso, tanto o LFU quanto o FIFO tiveram um desempenho com diferença menor que 5% entre eles, em alguns casos nem havendo diferença alguma. Porém, após uma análise e estudo dos artigos correlacionados, foi possível observar que a política de substituição LRU se mostra de forma pouco significativa mais eficiente do que a política FIFO.

Tudo que descobrimos esteve desde Smith e Goodman [**?**] que disseram que não há diferença de desempenho entre os algoritmos e que aleatoriedade é mais eficiente. Enquanto no artigo da Samira Mirbagher Ajorpaz, Elba Garza, Sangam Jindal, e Daniel A. Jiménez [**?**], mostrou que o desempenho do LRU é pouco superior ao FIFO, algo que consegue ser detectado, mas seu uso diário não teria melhora perceptível e que seria mais importante considerar outros tantos algoritmos que possuem um desempenho significativamente superior aos dois anteriores. Por fim, o artigo de Aurore Junier, Damien Hardy, Isabelle Puaut [**?**], mostrou algo inusitado, indo na direção oposta dos outros dois artigos, apresentou dados de que o LRU pode sim ter um desempenho superior à inserção aleatória de dados na cache.

Concluindo, nossos testes demonstraram que não há grande diferença desses algoritmos quando usados em caches pequenas, os dois primeiros artigos concordam com nosso posicionamento, enquanto o outro artigo discordou. Dessa forma, é importante manter essa discussão aberta para novos

experimentos. Considerando que cenários de testes diferentes apresentam resultados completamente dispersos.

## REFERENCES

[1] *Proceedings of the 34th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*. Gramado, RS, Brazil: IEEE, October 2021.