Universidade do Estado da Bahia

Departamento de Ciências Exatas e da Terra

Colegiado de Sistemas de Informação

Avaliação de uma Heurística de Peões Aplicada ao Programa de Xadrez Crafty

Augusto Artur Andrade da Costa

Salvador, Bahia, Brasil

14/07/2014

Augusto Artur Andrade da Costa

Avaliação de uma Heurística de Peões Aplicado ao Programa de Xadrez Crafty

Monografia submetida ao Colegiado de Sistemas de Informação da Universidade do Estado da Bahia como parte dos requisitos necessários para obtenção do grau de Bacharel em Sistemas de Informação.

Área de Concentração: Sistemas de Informação
Linhas de Pesquisa: Inteligência Artificial

Cláudio Alves de Amorim
(Orientador)

Salvador, Bahia, Brasil

14/07/2014

Augusto Artur Andrade da Costa

Avaliação de uma Heurística de Peões Aplicado ao Programa de Xadrez Crafty

Monografia submetida ao Colegiado de Sistemas de Informação da Universidade do Estado da Bahia como parte dos requisitos necessários para obtenção do grau de Bacharel em Sistemas de Informação.

Aprovada em: _____ / _____ / _____

Cláudio Alves de Amorim
Doutor em Ciência da Computação
Universidade do Estado da Bahia

Nome do membro da banca
Título do membro
Universidade do Estado da Bahia

Nome do membro da banca
Título do membro
Universidade do Estado da Bahia

Resumo

Além de serem um excelente campo para as pesquisas em Inteligência Artificial, os jogos como xadrez, damas, go possuem um grande atrativo pelo seu aspecto lúdico. Existe uma discussão ainda em aberto sobre qual a técnica é mais efetiva nestes jogos, aumento da busca ou da heurística. Sabe-se que o aumento da busca produz uma melhora imediata, mas o mesmo não se pode afirmar com a heurística, pois esta não possui métricas bem definidas. Este trabalho pretende entrar nessa discussão fazendo alguns refinamentos nas heurísticas de programas de xadrez e avaliando seus resultados. O programa escolhido foi o Crafty, por possuir um bom nível de jogo e não sofrer com o efeito horizonte. As tabelas de peões do Crafty foram alteradas com base nos conhecimentos enxadrísticos e após alguns refinamentos foi observado um aumento considerável do nível de jogo do programa. Posteriormente um segundo programa, Toga2, foi escolhido para complementar os testes e a melhora se confirmou.

Palavras-chave: IA, xadrez computacional, estrutura de peões.

Abstract

Games like chess, checkers, go are an excellent fiel for AI and are funny. There is a discussion about the best technique in this games, a better heuristic or an improvment in the search. It is known that an improvment in the search produces an imediate improvment in the game level, but the effect of a better heuristic is not known. This work wants to enter in this discussion doing some changes in heuristics of chess programs and evaluating the results. The chess program used is Crafty, a strong program who doesn't suffer the horizon effect. The Crafty's pawn tables were changed based on knowledge of chess and after some tests a significant increase was observed. After a second program, Toga2, was choosed to make new tests and the increase was confirmed.

Keywords: AI, chess, pawn structure.

Agradecimentos

Agradecimentos

Gostaria de agradecer aos amigos, família e professores que me apoiaram durante estes anos de estudo. Sem a ajuda de vocês, isso não teria sido possível. Muito obrigado a todos.

“O xadrez é uma ciência.” (Leibinitz)

”O xadrez é a ginástica da inteligência.“ (Goethe)

”O xadrez é muita ciência pra ser jogo e muito jogo pra ser ciência.“ (Montaigne)

”O xadrez é uma luta gostosa de emoções.“ (Lasker)

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Neste capítulo será feita uma breve introdução ao tema do trabalho, será exposto um histórico do xadrez computacional, os objetivos gerais, específicos e a justificativa.

1.1 Visão Geral

Em fevereiro de 1996, o atual campeão mundial de xadrez e considerado o melhor jogador de todos os tempos, Garry Kasparov foi derrotado pelo Deep Blue um super computador desenvolvido pela IBM. Essa partida foi um marco para a computação pois foi a primeira vez que um computador venceu o campeão do mundo nas regras normais de tempo.

A vontade de criar uma máquina capaz de tal proeza não é de agora, em 1910, Lonardo Torres y Quevedo, um matemático e engenheiro espanhol desenvolveu um autômato para jogar xadrez(*El Ajedrecista*) que era capaz de executar um mate de rei e torre contra rei.

Em 1948 o matemático inglês Alan Turing criou o primeiro programa capaz de jogar xadrez conhecido como *Turochamp*. Como não existia uma máquina tão avançada para executar o programa Turing testou o algoritmo atuando como um computador humano. Turing levava em média meia hora para calcular cada movimento. Recentemente durante as comemorações do centenário do nascimento de Turing seu algoritmo foi derrotado por Garry Kasparov em pouco mais de dois minutos (COCHLIN, 2012).

1.2 Objetivo

1.2.1 Objetivo Geral

Este trabalho tem como objetivo geral avaliar o impacto do aperfeiçoamento de uma heurística de peões sobre o desempenho do Crafty, um programa de xadrez de código aberto.

1.2.2 Objetivo Específico

- Aperfeiçoar a tabela de valoração do Crafty 23.8 para finais de partida.
- Avaliar o desempenho com alterações ante o próprio Crafty 23.8 e ante o Toga.

1.3 Justificativa

Os jogos, principalmente os de tabuleiro como damas, xadrez e go além de apresentarem todas as características de um domínio adequado para a experimentação em IA exercem um grande atrativo devido ao seu aspecto lúdico (BITTENCOURT, 1998). Desde Alan Turing e Claude Shannon, o xadrez é considerado um importante campo para desenvolver algoritmos e testar as possiblidades e limites dos computadores, em atividades que, quando realizadas por humanos, requerem inteligência

Em 1950, Claude E. Shannon, no seu artigo *Programing a Computer for Playing Chess*, descreveu um algoritmo de xadrez para computadores de propósito geral. Para ele as idéias desenvolvidas no jogo ajudariam a resolver problemas como desenhos de circuitos, deduções lógicas, tradução de idiomas, execução de operações matemáticas simbólicas. Para Shannon a máquina de xadrez era ideal para estudos pois:

> 1)O problema, suas operações(movimentos) e a meta(xeque-mate) são bem definidos; 2)Não é nem tão simples nem tão difícil conseguir uma solução satisfatória; 3)A solução do problema nos fará a admitir a possibilidade de mecanizar o pensamento ou vai mudar a concepção do que é pensar; 4)

> A estrutura discreta do xadrez se encaixa bem com a natureza digital dos computadores modernos(SHANNON,1950).

Existe uma discussão ainda em aberto sobre os benefícios entre melhorar a heurística ou aumentar o cálculo de variantes através de uma busca mais profunda. Existem poucos estudos avaliando o impacto de uma heurística melhorada na performance dos programas (JUNGHANNS; SCHAEFFER, 2012). Por outro lado os benefícios de uma busca adicional estão muito bem documentados: há um ganho imediato quando a busca é aprofundada.

Entre essas duas dimensões, o aumento da busca é o mais fácil de alcançar, bastando escrever algoritmos mais rápidos ou apenas esperar que um hardware mais potente esteja disponível. A heurística é um pouco mais complexa. Onde existem métricas bem definidas para busca(profundidade, tempo de execução). Para a heurística não há nada parecido.

McCarthy acreditava que o xadrez computacional baseado no cálculo exaustivo de variantes não contribuía tanto para o estudo da inteligência artificial. A quantidade de variantes que um computador calcula é várias ordens de grandeza maior do que a calculada por um jogador humano. O que faz um jogador comum se tornar um grande mestre é sua habilidade em descartar as linhas de jogo improdutivas e se concentrar nas que realmente importam. Por isso McCarthy achava que os torneios deveriam permitir apenas computadores com capacidade limitada para o cálculo de variantes a fim de que os estudos se concentrassem nos avanços da inteligência artificial.

A intenção deste trabalho é adentrar na discussão sobre heurística versus força bruta e trazer novas conclusões sobre o assunto.

Capítulo 2

Programas de Xadrez

Neste capítulo serão expostas as principais técncias utilizadas pelos computadores para jogar xadrez

2.1 Cálculo de Variantes

Em um motor de xadrez o módulo responsável pelo cálculo de variantes é o mais importante. É ele que cria uma árvore com todos os lances possíveis para o momento. O rápido avanço do nível de jogo das engines é devido principalmente à criação e o refinamento das técnicas de busca utilizadas no cálculo de variantes.

2.1.1 A Árvore de Xadrez e o Algoritmo Mini-Max

Este é um dos algoritmos principais utilizados pelos programas. Foi desenvolvido por Claude Shannon e Alan Turing nos anos 40 para ser utilizado no jogo de xadrez. Quando um jogo começa o primeiro jogador tem a sua disposição 20 lances possíveis, seu adversário outros 20. Só no primeiro lance 400 posições são possíveis, após o segundo lance já são 160 000, após poucos movimentos esse número se torna intratável.

Existe uma regra que se após 50 movimentos de cada lado nenhum peão for movido ou

nenhuma peça capturada o jogo termina empatado. Graças à essa regra a quantidade de movimentos possíveis é bastante reduzida. Uma partida pode durar no máximo 3150 lances.(LEVY; NEWBORN, 2009) Esse número ainda é muito grande para se examinar todas as posiçoes possíveis. Normalmente se trabalha em uma pequena árvore esperando reunir conhecimento suficiente para fazer um bom lance. Essa estratégia é utilizada tanto pelos computadores quanto pelos humanos. Como os humanos fazem para descartar partes da árvore de busca ainda não está claro para a ciência(MCCARTHY, 1997).

Nos anos 70 os programas podiam calcular cerca de 200 posições por segundo (LEVY; NEWBORN, 2009), em 1997 o Deep Blue calculava 200 milhões. Os melhores programas dessa época examinavam entre 8 e 10 lances à frente. Em alguma linhas de jogo mais críticas a profundidade da busca era ainda maior. Para uma árvore de xadrez dada, o algoritmo minimax decide qual movimento o primeiro jogador deve fazer. O algoritmo começa calculando um valor numérico para as posições terminais. Para fazer esse cálculo é utilizada uma função de avaliação, um resultado positivo significa que o computador está ganhando, negativo que o oponente está melhor. Quanto maior o número melhor a posição e vice versa. O minimax entende que o objetivo do primeiro jogador é chegar numa posição terminal que tenha o maior valor possível enquanto a meta do oponente é chegar ao menor valor. Isso equivale a dizer que nos níveis pares da árvore que sejam do turno do computador deve ser dado um valor igual ao valor máximo dos seus sucessores. Enquanto no nível ímpar, turno do oponente, o minimax deve escolher uma posição não terminal que tenha o escore mínimo nos seus sucessores. A avaliação das posições terminais deve ser feita antes e através de uma técnica chamada escores armazenados os valores são determinados até chegar à raiz da árvore.(LEVY; NEWBORN, 2009)

Em (FRAYN, 2005) é apresentada esta tabela de tempo e quantidade de posições analisadas, considerando que o programa consiga examinar 5 000 000 posições por segundo:

Níveis Gerados	Número de posiçoes	Tempo de busca
2	900	0.00018 segundos
3	27.000	0.0054 segundos
4	810.000	0.162 segundos
5	24.300.000	4,86 segundos
6	729.000.000	2.43 segundos
7	21.870.000.000	1.215 hora

Tabela 2.1: Geração do algoritmo minimax

Uma pequena árvore de três níveis níveis é mostrada na figura, o algoritmo minimax primeiro faz o cálculo do escore das posições finais(D,E,F,G), em seguida os escores das posiçoes B e C são calculados levando em consideração que o oponente vai fazer o movimento que tenha a menor pontuação possível no nível 2. Na posição B o movimento seria para a posição D, logo essa pontuação é definida para B. Em C o oponente escolheria a posição F, então C fica com esse valor.

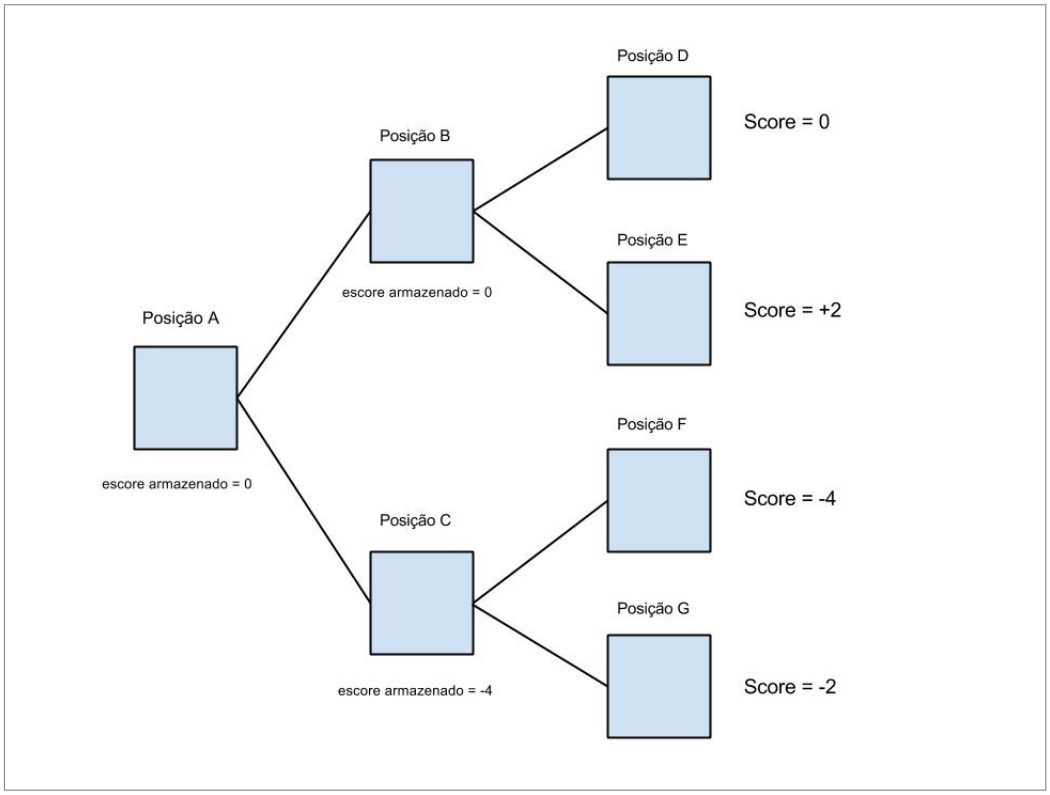

Figura 2.1: Arvore minimax.

2.1.2 Algoritmo Alfabeta com Poda

Um estudo cuidadoso do minimax demonstra que muitos ramos da árvore de busca não precisam ser avaliados. Quem primeiro pensou sobre isso foi John McCarthy, em 1956. Na prática o alfa-beta com poda é apenas o minimax com algumas linhas de código extras que decidem quais caminhos da árvore podem ser deixados de lado na hora da pesquisa. Utilizando o mesmo exemplo anterior veremos que uma posição final não precisa ser verificada. É definido um valor igual a 0 para posição D e B. Se o próximo valor analisado for menor que 0 então B recebe esse valor. Não é esse o caso pois E vale +2. Passamos para F, onde encontramos o valor -4 que também é definido para a posição C. Analisando percebemos que seria um erro o jogador escolher a posição C pois mesmo sem analisar a última posição final sabemos que o máximo de pontuação que ele pode chegar é -4, enquanto indo pelo outro caminho ele vai marcar 0. Com isso obtemos o mesmo resultado do minimax analisando uma posição a menos. Dizemos que o movimento de A para C foi refutado pelo movimento de C para F.

2.1.3 Efeito Horizonte

Um problema que pode acontecer com a busca em árvore com uma profundidade fixa é o efeito horizonte (FRAYN, 2005). Este acontece quando após uma busca o algoritmo devolve uma avaliação positiva, mas se fizesse uma pesquisa um pouco mais profunda descobriria na verdade uma grande desvantagem. Usando um lance como exemplo seria como se o primeiro jogador capturasse uma dama e um lance depois recebesse um xeque-mate. A árvore de busca não pode prever o lance porque a pesquisa não fui suficientemente profunda. Para tentar resolver o problema do efeito horizonte foi criada a busca quiescente, que consiste em continuar a pesquisa quando a última posição avaliada for uma captura ou ataque ao rei. A pesquisa termina quando uma posição tranquila é encontrada. Um problema da busca quiescente é que ela pode aumentar bastante a profundidade da árvore de busca.

2.1.4 Geração de Movimentos e Heurística Assassina

A ordem exata em que os movimentos são gerados é muito importante para que o algoritmo alfabeta seja eficiente. É desejável que bons movimentos sejam colocados no topo da lista. Com uma boa ordenação o número de refutações é maximizado e consequentemente o número de movimentos que são cortados da árvore. A geração de movimentos pode ser vista como dois processos: (1) gera-se uma lista de movimentos, (2) a lista é ordenada para que os melhores movimentos fiquem próximos ao topo. Os programas podem fazer isso de diversas formas, por exemplo, o gerador de movimentos pode deixar os movimentos do rei por último durante o início do jogo e fazer o inverso no fim. A heurística assassina é utilizada para identificar bons movimentos rapidamente, na prática quando descobre-se que um movimento é uma refutação em um nível k da árvore ele é colocado na lista assassina de k.

A figura ilustra a heurística assassina. É a vez das brancas jogarem um programa poderia tentar o movimento **Cg4xh6** pois ganha uma torre. Após examinar resposta das negras descobriria **Ta3-a1** mate. Agora cada vez que o programa for examinar um movimento para as brancas a resposta imediata será **Ta3-a1**.

2.1.5 Tabelas Hash(tabela de transposição)

Note que os movimentos **1d2-d4 d7-d5 2e2-e4** leva a uma posição P que também é encontrada através de **1e2-e4 d7-d5 2d2-d4**. Torna-se desnecessário avaliar a mesma posição mais de uma vez se o resultado já estiver previamente guardado. Para isso os programas utilizam as tabelas *hash*, também chamadas tabelas de transposição para poupar tempo com posições já examinadas. Quando uma busca chega numa posição, a tabela *hash* é examinada para saber se já existe a posição e se a informação sobre ela é suficiente. Se já existe a posição ela não precisa ser examinada novamente. Se não existe, cria-se um novo registro na tabela. Com o andamento do jogo a tendência é que as posições se repitam cada vez mais.

2.1.6 Finais de Jogo

Nos anos 60, quando um jogo entre dois programas chegava à fase final era normal que os programadores sugerissem um empate para evitar que ficasse evidente a fraqueza dos computadores da época para jogar essa fase do jogo (LEVY; NEWBORN, 2009). Os programadores sabiam que alguns conceitos simples de entender para jogadores humanos eram totalmente desconhecidos pelos programas enxadristas. Isso se devia em parte por estes conceitos não terem sido programados e também porque muitas posições de fim de jogo necessitavam de buscas muito mais profundas que os computadores da época podiam fazer. Com o passar do tempo, muitos conceitos de fim de jogo foram adicionados às rotinas de avaliação, como por exemplo, manter o rei defendendo a casa de um peão avançado. Os programas também entenderam a vantagem de controlar as casas onde ocorrem a promoção dos peões e também que alguns finais de jogo como rei e bispo contra rei ou rei e cavalo contra rei são empate. Conceitos como oposição e zugzwang eram problemas sérios para os computadores. A oposição acontece quando dois reis estão na mesma linha, coluna ou diagonal a um número ímpar de casas de distância. O rei que ganha a oposição força o rei inimigo a se distanciar e com isso pode conseguir uma posição melhor para outra peça da sua cor.

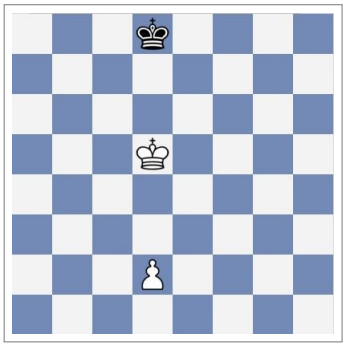

Figura 2.2: Oposicao.

Zugzwang é uma palavra alemã que significa obrigação de se movimentar. O termo é usado para uma posição na onde qualquer movimento piora a situação do jogador. É uma situação em que seria melhor passar a vez(se fosse possível).

Quando realmente começa o final do jogo é uma decisão totalmente arbitrária, alguns programas contam o número de peças, outros verificam a existência de certas peças.

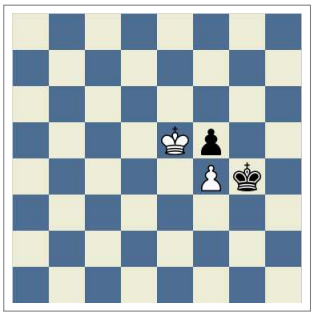

Figura 2.3: Zugzwang.

Qualquer que seja a escolha, é uma decisão muito importante para o motor do jogo, pois algumas políticas utilizadas durante a partida se invertem no final, por exemplo, a entrada do rei no jogo é incentivada. Se o programa avaliar erroneamente pode expor o rei antes da hora e terminar perdendo o jogo. O aumento da capacidade de processamento dos computadores tornou possível que certos finais pudessem ser totalmente analisados. Grandes bancos de dados foram criados permitindo que os programas "entendessem" como jogar finais em grande nível. Para construir um banco de dados de final considere o exemplo simples de rei e rainha contra rei (LEVY; NEWBORN, 2009). Primeiro o rei preto pode ser colocado em 64 casas, embora do ponto de vista de simetria apenas 10 casas são necessárias. Depois que o rei negro já está alocado existem 64 x 64 diferentes posiçoes de alocar as outras peças. O banco de dados de rei-rainha contra rei deve conter 10x64x64 = 40960 posições, para cada uma o programa deve saber em quantos movimentos as brancas ganham. O banco é criado por um processo chamado análise retrógrada, para começar cada uma das 40960 posições é inicializada com o valor 0. A primeira posição corresponde a <a1-a1-a1> e por fim a última posição <d4-h8-h8>. A análise retrógrada executa os três passos seguintes até completar a tabela:

- Identifica todas as posições em que o rei negro está em xeque e todas as posições impossíveis. Posições impossíveis são aquelas em que dois reis ocupam a mesma casa ou casas adjacentes.
- Depois todas as posições que são mate em 1 lance são determinadas
- Iniciando com n igual a 1, recursivamente determina todas as posições que são mate em (n+1) lances a partir das posições que são mate em n lances.

2.2 Heurísticas

As funções heurísticas determinam o valor relativo de uma posição, tentando formalizar as técnicas dos grandes jogadores humanos. Os programas de xadrez atuais possuem milhares de regras.O Deep Blue possui cerca de 8000 (HSU et al., 1990) estas variam conforme o conhecimento enxadrístico e, até certo ponto, as preferências do programador por determinado estilo de jogo. Abaixo serão apresentadas alguns dos aspectos mais comuns utilizados nas heurísticas dos programas de xadrez.

2.2.1 Valor Comparativo das peças

Esse é um dos primeiros conceitos aprendidos por quem começa a jogar xadrez. Nele o valor das peças é calculado tendo o peão como unidade.

- Peão = 1
- Cavalo e Bispo = 3
- Torre = 5
- Dama = 10

O valor do rei é infinito pois é a peça principal do jogo. Alguns autores consideram que o valor da Dama seja 9. Os motores de xadrez tem que levar em consideração que esses valores tendem a mudar conforme o andamento do jogo. Por exemplo o bispo no fim do jogo, com menos peças no tabuleiro bloqueando as diagonais, possui um raio de ação muito maior que o do cavalo, enquanto este por sua característica de movimentação é bastante utilizado no início do jogo. A função que utiliza esse aspecto deve varrer o tabuleiro contando as peças e retornar esse valor.

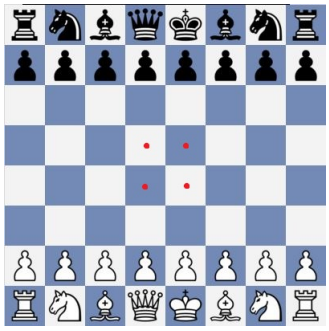

Figura 2.4: Casas centrais do tabuleiro

2.2.2 Procurar ocupar as casas centrais no início

Na maioria das vezes essas casas são ocupadas com peões. Esse movimento é indicado porque as partidas analisadas mostram que o jogador que tiver as peças posicionadas no centro tem mais opções de jogada e restringe os movimentos do adversário.

2.2.3 Tentar ampliar o raio de ação das peças

Nessa duas imagens vemos como o posicionamento do cavalo pode alterar seu raio de ação. Na sua casa inicial o cavalo só pode alcançar três casas, enquanto na segunda posição ele alcança oito, ou seja, apenas um cavalo pode atacar ou defender oito peças.

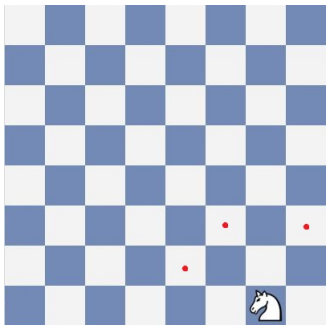

Figura 2.5: Cavalo na posição inicial

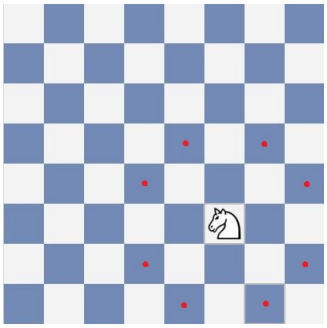

Figura 2.6: Cavalo numa das melhores posições

2.2.4 Evitar enfraquecer o roque

O roque é uma jogada especial do xadrez, nela o rei anda duas casas na horizontal e a torre outras duas no sentido oposto ao rei. é considerada uma jogada de defesa e de desenvolvimento pois protege o rei e coloca a torre em jogo. O roque pode ser feito com qualquer uma das torres. A estrutura dos peões do roque deve ser mantida para garantir a segurança do rei. Os três peões do roque tem força defensiva máxima nas suas casas iniciais (D'AGOSTINI, 1955). Por isso qualquer lance envolvendo a movimentação desses peões deve ser muito bem pensado.

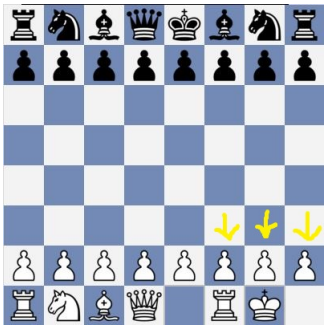

Figura 2.7: peões do roque

2.2.5 A exposição do rei deve ser evitada(antes do fim do jogo)

Deve-se evitar durante o maior tempo possível qualquer jogada com o rei a não ser o roque que se aconselha fazer o mais rápido possível. Fora de sua casa inicial o rei se torna um alvo fácil para o ataque do inimigo.

2.3 Heurísticas de Peões

2.3.1 Peões a Alma do Jogo

O foco principal do trabalho será desenvolver heurísticas baseadas no jogo de peões, por isso essa parte merece um estudo maior. Embora a literatura enxadrística seja muito extensa, poucos são os autores que dão a devida importância aos peões (SOLTIS, 1995).André Philidor [1], já dizia: "os autores deveriam escrever, os peões são a alma do jogo." Quem dedicar um pouco do seu tempo ao estudo verá que entre o fim da abertura e o início do fim do jogo a posição dos peões pouco se altera. A estrutura dos peões se torna o campo de batalha, um centro forte equivale ao terreno alto que todo general desejaria numa guerra. A falta dos peões abre filas e diagonais que se tornam rotas de acesso para o ataque do exército inimigo. Philidor também explicou porque os peões permitem um bom meio de jogo, segundo ele: "peões são capazes de atacar e de defender." Existe uma tendência natural de trocar uma estrutura de peões sólida por avanços e capturas mas esta é uma ação que deve ser evitada. Uma das diferenças entre um amador e um mestre é que o último percebe quando tem uma estrutura de peões que não deve ser alterada. É interessante lembrar que a estrutura de peões não é uma característica isolada de uma posição, e sim uma parte intrínseca a ela.

2.3.2 Peões Dobrados

Peões dobrados são aqueles que são da mesma cor e estão na mesma coluna. são evitados porque dificultam a criação de peões passados e se estiverem isolados são facilmente atacáveis e difíceis de defender.

2.3.3 O peão passado

Apesar do seu poder bastante limitado o peão possui uma grande vantagem sobre as outras peças que é a sua promoção para qualquer peça quando atinge a oitava casa do tabuleiro. Um

[1] François André Danican Philidor foi um dos mais ilustres enxadristas que o mundo já viu, nascido em 7 de setembro de 1726. Criador da Defesa Philidor. Também foi um excelente músico.

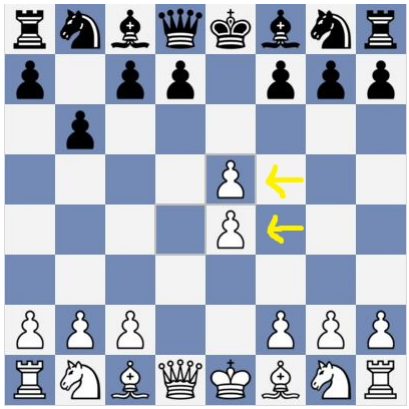

Figura 2.8: Peões dobrados

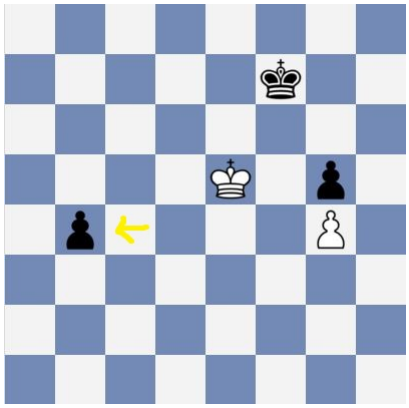

Figura 2.9: Peão passado

peão avançado pode reverter um jogo perdido. O peão passado é aquele que não possui peão inimigo nem na sua coluna nem na adjacente, ou seja, possui caminho livre para a promoção. Existem três tipos de peões passados: peão passado distante, peão passado protegido e peões passados unidos.

2.3.4 O peão passado distante

É um peão passado nas laterais do tabuleiro, é muito valioso porque faz com que o inimigo traga suas peças para sua ala afim de impedir sua promoção deixando o outro lado desprotegido.

2.3.5 O peão passado protegido

É um peão passado que possui a proteção de outro peão. A força deste peão está no fato de que ele não pode ser capturado pelo rei inimigo. Esta é uma grande vantagem quando se esta no fim do jogo.

2.3.6 O peão passado unido

São os peões passados que estão em colunas adjacentes. A melhor formação é quando estão na mesma horizontal.

2.3.7 O peão isolado

É aquele peão que não pode ser protegido por outros peões por não existirem peões companheiros nas colunas vizinhas. É uma desvantagem possuí-lo pois é um fácil alvo de ataques e a casa à sua frente se torna um ponto forte para o adversário.

2.3.8 O peão atrasado

É o peão situado atrés de seus companheiros. É uma debilidade séria pois a casa da sua frente se torna posto para a colocação de peças adversárias. O peão atrasado é explorado da mesma forma do peão isolado.

Figura 2.10: Peão atrasado

2.3.9 O peão atrasado ou isolado em uma coluna semi-aberta

Uma coluna é chamada aberta quando não há peões nelas, semi-aberta quando há somente um peão. O peão atrasado ou isolado em uma coluna semi-aberta é uma grande debilidade pois pode ser atacado frontalmente pela torre ou rainha.

Capítulo 3

O Programa Crafty

Este capítulo faz uma breve descrição da engine que será utilizada para testar as heurísticas, o Crafty.

3.1 História

O Crafty é um motor de xadrez de código aberto desenvolvido por Dr. Robert M. Hyatt. Crafty é constantemente melhorado por uma pequena equipe de desenvolvimento, incluindo Dr. Hyatt. Crafty é "descendente" do Cray Blitz, o computador campeão mundial de 1983 a 1989. O Cray Blitz, por sua vez, foi derivado do Blitz, um programa que foi feito quando Robert Hyatt ainda era aluno da graduaçao. Blitz fez seu primeiro movimento em 1968 e seu desenvolvimento continuou até 1980, quando a Cray Research resolveu patrocinar o projeto devido à crescente publicidade do xadrez computacional. Ele ganhou vários eventos da Association for Computer Machinary(ACM) e os prêmios mais expressivos foram dois campeonatos mundiais de xadrez computacional, o primeiro em 1983 em Nova Yorque e o segundo em 1986, na cidade de Cologne, Alemanha.

3.2 Estrutura

O Crafty pode ser compilado e executado de um terminal *Windows*, *Linux* ou *Macintosh*. Pode também ser executado através de uma interface gráfica como *Winboard* ou *Xboard*. É *freeware* e está disponível em www.cis.uab.edu/hyatt/crafty. Crafty utiliza a abordagem clássica do *bitmap* para representação do tabuleiro porém utiliza um método chamado rotação de *bitmap* para aumentar sua performance. Esse programa pesquisa em média 2.400.000 posições por segundo em uma máquina dual xeon de 2,8 Ghz. O Crafty joga no ICC regularmente e possui os *ratings* de 3286(*bullet*), 3388(*blitz*) e 2792(*standard*) (HYATT, 2014). As partidas no modo *bullet* são jogadas em 1 a 2 minutos, *blitz* é jogado entre 3 e 5 minutos e o *standard* em 2 horas (SCIMIA, 2014).

3.3 Heurísticas

O Crafty possui diversas heurísticas e funções de avaliação, dentre elas temos análise de material, avaliação de peões que considera a posição, peões passados, avaliação de outras peças considerando a mobilidade, segurança do rei considerando o escudo de peões. Possui avaliação de cavalos, bispos.

Capítulo 4

Percurso Metodológico

Este capítulo descreve os passos executados durante os experimentos.

4.1 Tentativas Iniciais

Os estudos se iniciaram com a *engine* TSCP por ser um programa de código aberto de relativa simplicidade. As primeiras tentativas foram focadas em incentivar o programa a valorizar as casas centrais do tabuleiro, tentando ocupá-las com os peões e movendo os bispos e cavalos para agirem sobre elas. Foi verificado que colocar as *engines* para jogar sem a consulta de um livro de aberturas não contribuía para o experimento pois as partidas tendiam a seguir os mesmos movimentos. Com a adição dos livros de abertura as partidas se tornaram bastante diversificadas e se pôde analisar vários aspectos. O TSCP original jogando contra a sua versão com a tabela nova obteve 19.5 pontos de um total de 40.

0	0	0	0	0	0	0	0
8	10	15	20	20	15	10	8
6	8	12	16	16	12	8	6
3	6	9	12	12	9	6	3
2	2	9	12	12	9	2	2
1	2	3	-10	-10	3	2	1
0	0	0	-40	-40	0	0	0
0	0	0	0	0	0	0	0

Tabela 4.1: Tabela de peões TSCP

O passo seguinte foi criar uma função de avaliação que levasse em conta a existência de peões passados ligados. Nos testes realizados o TSCP original ganhou 23 dos 40 pontos disputados. Após a análise de outras *engines* surgiu a idéia de valorizar a distância entre peões passados, ou seja, dois peões passados com 4 colunas de distância valeriam mais que um par de peões distantes entre si 2 colunas. Esta função se mostrou bastante prejudicial ao nível de jogo do TSCP pois de 44 pontos disputados só conseguiu 9 contra sua versão original. Analisando outra *engine*, desta vez a Rebel, surgiu a idéia de utilizar penalidades variáveis para os peões isolados, os novos valores levariam em consideração a posição no tabuleiro e a fase do jogo. Com isso seriam adicionadas 2 tabelas, para as duas fases de jogo, meio e final. O resultado foi 70.25 pontos para o TSCP original contra 29.5 do TSCP com as duas novas tabelas. Após este teste ficou evidente que o nível de jogo do TSCP era elementar demais, e que, embora se melhorasse a heurística, o prejuízo na capacidade de cálculo inviabilizava a continuação dos estudos.

0	0	0	0	0	0	0	0
10	12	16	20	20	16	12	10
10	12	16	20	20	16	12	10
10	12	16	20	20	16	12	10
6	8	10	16	16	10	8	6
4	6	8	10	10	8	6	4
2	4	4	10	10	4	4	2
0	0	0	0	0	0	0	0

Tabela 4.2: Tabela de penalidades para peões isolados na abertura e meio jogo

0	0	0	0	0	0	0	0
0	2	4	2	2	4	2	0
2	4	6	10	10	6	4	2
4	6	10	16	16	10	6	4
6	8	12	16	16	12	8	6
8	10	14	20	20	14	10	8
10	12	16	20	20	16	12	10
0	0	0	0	0	0	0	0

Tabela 4.3: Tabela de penalidades para peões isolados no fim do jogo

4.2 Experimentos

O próximo programa a ser utilizado foi o Crafty por ser uma *engine* mais madura e que possuía um nível de jogo bastante alto. Os jogos foram feitos utilizando a interface gráfica Arena 3.0. Cada partida era jogada com 5 minutos para cada particpante, afim de que o tempo reduzido levasse os programas a não fazer uma busca muito profunda na árvore de jogo e as alterações nas heurísticas se tornassem mais aparentes no resultado. Inicialmente se tentou aprimorar a tabela de meio jogo de peões, se baseando na tabela do Stockfish, outra poderosa *engine*. Os resultados inicais foram 14 a 6 para o Crafty original.

0	0	0	0	0	0	0	0
-20	-6	4	14	14	4	-6	-20
-20	-6	9	11	11	9	-6	-20
-20	-6	17	34	34	17	-6	-20
-20	-6	17	54	54	17	-6	-20
-20	-6	9	34	34	9	-6	-20
-20	-6	4	14	14	4	-6	-20
0	0	0	0	0	0	0	0

Tabela 4.4: Tabela 1 de peões para a abertura e meio jogo

Com alguns ajustes na tabela consegiu-se obter 12 a 8. As alterações seguintes fizeram o novo Crafty, que passou a ser chamado de UnebV3, conseguir um 11.5 a 8.5 contra a sua

versão original.

A partir deste momento buscou-se uma outra *engine* para ajudar na comparação dos resultados. O Toga2 1.4 foi o escolhido por estar uma posição acima do Crafty no ranking das engines. Após 100 partidas realizadas o Toga 2 1.4 fez 61,5 pontos contra 38,5 do Crafty. O UnebV3 contra o Toga marcou 15 contra 25, fazendo praticamente a mesma pontuação do Crafty.

Após esse resultado foi aplicada uma tabela para o fim de jogo, O UnebV3 só com a tabela de finais consegiu 53 pontos de 100 contra o Crafty e 45 contra o Toga 2 1.4. Com as duas tabelas o Uneb praticamente empatou com o Crafty 49.5 a 50.5 e perdeu de 35 a 65 contra o Toga 2 1.4.

0	0	0	0	0	0	0	0
66	80	66	66	66	66	80	66
30	40	20	30	30	20	40	30
15	20	6	16	16	6	20	15
7	9	4	13	13	4	9	7
8	10	5	10	10	5	10	8
0	0	0	-12	-12	0	0	0
0	0	0	0	0	0	0	0

Tabela 4.5: Tabela V3 de peões para o fim de jogo

4.3 Alteração da Tabela de Finais

A tabela original do Crafty para a avaliação de peões nos finais apresenta algumas características insatisfatórias:

1. O aumento do valor relativo dos peões cresce abruptamente quando eles passam da sexta para a sétima fileira. Contudo, praticamente não há estímulo para que avancem, antes disso. Porém, na ausência de inconvenientes táticos definidos, os peões, nos finais, precisam ser avançados, ainda que cautelosamente.

2. Os peões das colunas "d"e "e"entre a terceira e a sexta linha, são avaliados muito mais positivamente do que os das demais colunas. Na prática, porém, o avanço dos peões nas colunas laterais "a", "b", "g"e "h"tende a ser mais perigoso do que o avanço dos peões centrais, porque as peças adversárias terão dificuldades em bloqueá-los (principalmente se existirem peões (possivelmente passados, ou "passáveis", nas duas alas).

3. Exceto no caso das colunas "d"e "e", praticamente não há estímulo para que os peões avancem da segunda para a terceira fileira. Porém, muitos finais os peões na segunda linha são um estorvo, porque são fáceis de atacar, difíceis de defender e atrapalham a movimentação das peças amigas, em especial as torres, que devem ficar atrás dos peões, e não diante deles.

A partir dessas considerações, pode-se compreender as mudanças feitas, resultando na tabela para avaliação de peões V3, usada nos testes aqui apresentados:

1. O estímulo para que os peões laterais avancem, a partir da quarta linha, foi bastante ampliado. Cabe observar, contudo que, exceto no caso das colunas "d"e "e", há um leve desestímulo para que avancem da terceira para a quarta fila. Isso porque, especialmente nos finais de torres, os peões costumam se posicionar melhor na terceira linha do que na quarta, exceto os peões centrais, que ficam melhores na quarta linha (frequentemente apoiados pelo rei).

2. O avanço dos peões laterais recebe avaliação melhor do que o avanço dos peões centrais, com o objetivo de criar dificuldades defensivas para o adversário luta contra os peões avançados.

3. O avanço dos peões da segunda para terceira fileira passa a ser premiado, a fim de que posicionem melhor para fins de livre movimentação das peças amigas e para possam ser defendidos com mais flexibilidade.

Vale ressaltar que a avaliação relativa dos peões, com base em tabelas fixas, é evidentemente muito genérica. Por isso, qualquer escolha de valores pode ser tornar imprópria em

situações específicas. O que importa, portanto, é traduzir, por meio da pontuação tabulada, princípios apropriados ao máximo possível de circunstâncias e, ao mesmo tempo, com pouca possibilidade de se tornarem demasiado "nocivos"para a avaliação posicional do engine, mesmo em casos muito peculiares.

4.4 Avaliação dos Resultados

Os resultados considerados foram os conseguidos somente com a tabela de finais, pois estes se mostraram mais positivos. O Uneb com a tabela de finais conseguiu 52 vitórias, 90 empates e 38 derrotas com o Crafty. O que equivale a um ganho de 27 pontos no *rating* Elo. No total foram 55,6% de vitórias das brancas e 55,2% de vitórias das negras . O Toga contra o Crafty obteve 42 vitórias , 39 empates e 19 derrotas, o que significa uma diferença de 81 pontos de *rating* Elo, foram 41% de vitóras das brancas e 33% das negras. Confrontando o Uneb com a tabela de finais o Toga venceu 34, empatou 42 e perdeu 24, foram 41% de vitórias das brancas e 33% das negras, a diferença do *rating* caiu para 35. Houve um ganho de 46 pontos de *rating* Elo.

Capítulo 5

Conclusão

Neste capítulo serão expostas as conclusões obtidas com o trabalho, algumas considerações e trabalhos futuros.

5.1 Considerações Finais

Apartir dos resultado dos testes realizados podemos afirmar que, os 27 pontos de rating Elo conseguidos contra o Crafty é a diferença entre o segundo(Aronian, Levon) e o sexto(Kramnik, Vladimir) colocados no rankign FIDE. Também é a diferença entre entre o sexagésimo primeiro(Korobov, Anton) e o centésimo segundo(Sadler, Mathew D) colocados. A melhora de 46 pontos jogando contra o Toga é a diferença entre o segundo(Aronian, Levon) e o décimo primeiro(Dominguez Perez, Leiner), também é a diferença entre quadragésimo sétimo(Kasimdzhanov, Rustam) e o centésimo segundo(Sadler, Mathew D). No rankging dos computadores, 27 pontos é a diferença entre entre o quinto(Critter 1.6a 64-bit 4CPU) e o oitavo(BlackMamba 2.0 64-bit 4CPU) e também entre o décimo(Hannibal 1.4b 64-bit 4CPU) e o décimo quinto(Hiarcs 14 4CPU). O Crafty com o acrésimo de 27 pontos no *rating* ganharia 3 posições, subindo da vigésima sétima para a vigésima quarta. Com os 46 pontos ganharia 4 posições subindo para a vigésima terceira.

Conseguiu-se melhorar o nível de jogo do programa apenas modificando suas heurísticas,

sem aumento de consumo de memória numa busca mais aprofundada. Durante os testes ficou evidente que uma abertura que é muito jogada pelo programa já caiu em desuso, a abertura Bird, com isso alguns resultados foram prejudicados justamente pela escolha equivocada das jogadas iniciais.

5.2 Trabalhos Futuros

Primeiramente uma alteração que seria muito interessante o estudo é a escolha de um livro de aberturas melhor para o programa. Como já foi dito antes, o Crafty utiliza uma abertura que é muito prejudicial à sua estrutura de peões, por isso sua retirada deve melhorar o resultado em algumas partidas. Seguindo a linha de pensamento da melhora das heurísticas de peões outra possiblidade seria a utilização de tabelas diferentes para finais distintos de modo que suas pontuações reflitam de modo mais preciso as especificidades das posições avaliadas. Essa alteração deverá implicar em uma carga adicional de processamento, afim de que o algoritmo identifique o tipo de final.

Referências Bibliográficas

BITTENCOURT, G. *Inteligência Artificial*. Florianópolis, SC, BR: Editora da UFSC, 1998.

COCHLIN, D. *Kasparov versus Turing*. Manchester, UK: [s.n.], 2012. Disponível em: <http://www.manchester.ac.uk/discover/news/article/?id=8432>. Acesso em: 10 Maio 2014.

D'AGOSTINI, O. G. *Xadrez Básico*. [S.l.]: Ediouro, 1955.

FRAYN, C. *Computer Chess Programming Theory*. Santa Clara, CA: [s.n.], 2005. Disponível em: <www.frayn.net/beowulf/theory.html>. Acesso em: 10 Maio 2014.

HSU, F. siung et al. A grandmaster chess machine. *Scientific American*, October 1990.

HYATT, R. *Robert Hyatt*. 2014. Disponível em: <http://www.cis.uab.edu/info/faculty/hyatt/hyatt.html>. Acesso em: 22 Maio 2014.

JUNGHANNS, A.; SCHAEFFER, J. Search versus knowledge in game-playing programs revisited.

LEVY, D.; NEWBORN, M. *How Computers Play Chess*. Bronx NY, USA: Ishi Press International, 2009.

MCCARTHY, J. Ai as sport. *Science*, June 1997.

SCIMIA, E. *Speed Chess*. 2014. Disponível em: <http://chess.about.com/od/chessvariants/a/Speed-Chess.htm>. Acesso em: 22 Maio 2014.

SOLTIS, A. *Pawn Structure Chess*. New York, USA: McKay Chess Library, 1995.

Apêndice A

Compilando o Crafty 23.8 com o Dev C++

Uma vez que já se tenha os códigos fonte do Crafty e o Dev instalado, estes passos devem ser seguidos para que se possa fazer a compilação corretamente.

1. Crie um projeto do tipo C++ e exclua o arquivo main.c que foi criado.
2. Clique com o botão direito no projeto escolha ”Add to project”.Importe todos arquivos ”.c” e ”.h” do Crafty para o projeto criado, com exceção de crafty.c
3. Selecione Menu -> Project -> Project Options. Selecione a segunda aba ”Files”. Selecione todos os arquivos, exceto egtb.ccp, e desmarque a opção ”Compile file as C++”.
4. Selecione a aba ”Parameters” e adicione **-DNT_i386** nas colunas ”C Compiler” e ”C++ Compiler“.
5. O projeto já está configurado e já pode ser compilado e executado.

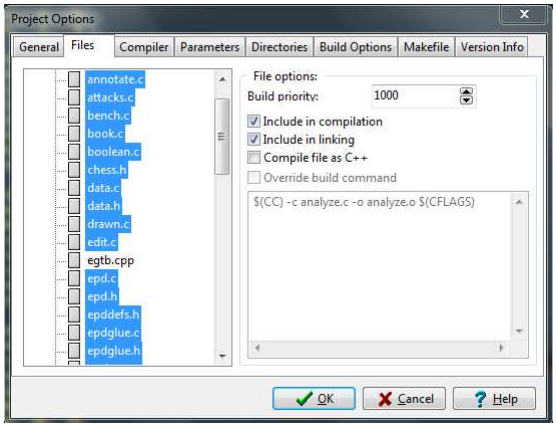

Figura A.1: Configurando a compilação dos arquivos

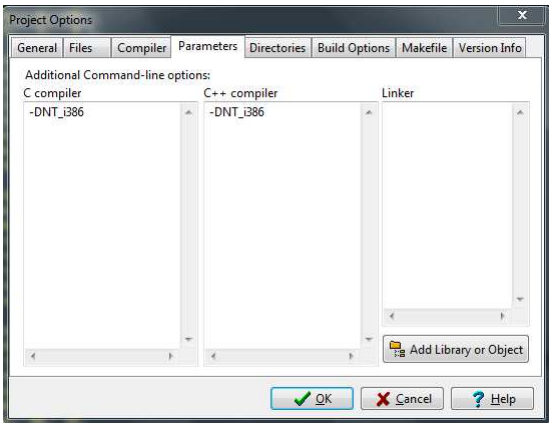

Figura A.2: Projeto configurado

Apêndice B

Configurando o Arena 3 para fazer torneios entre engines

Para configurar os torneios siga os passos a seguir

1. Baixe o Arena 3 que pode ser encontrado em *http://www.chess.com/download/view/arena-30*
2. Descompacte o arquivo e execute.
3. Selecione Engines -> Tournaments ou pressione a tecla F9.
4. No painel ”Available Engines” escolha os participantes. Mude o valor de ”Rounds” para escolher o número de partidas, em ”Level” escolha a duração.
5. Na aba ”Files” deixe marcado o campo ”Save Games in PGN format” e escolha o nome do arquivo em que serão salvas as partidas.
6. Clique em Start e o torneio iniciará.

Para instalar novas engines escolha Engines -> Install New Engines, e selecione o arquivo executável do programa. Será perguntado a interface gráfica utilizada e se deseja iniciar a engine, escolha ”não”.

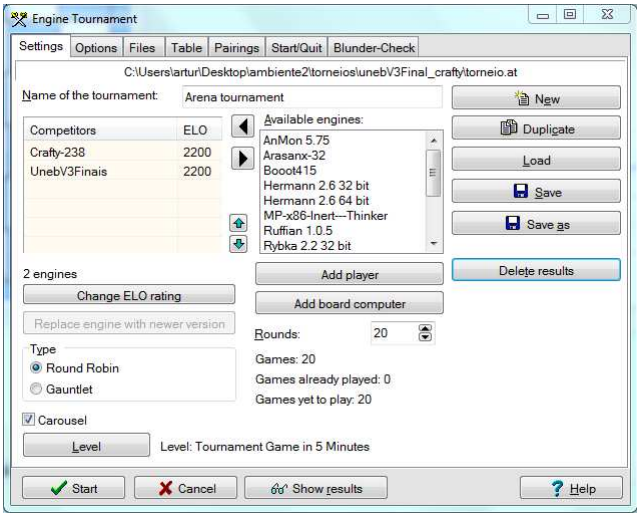

Figura B.1: Tela de configuração do Torneio

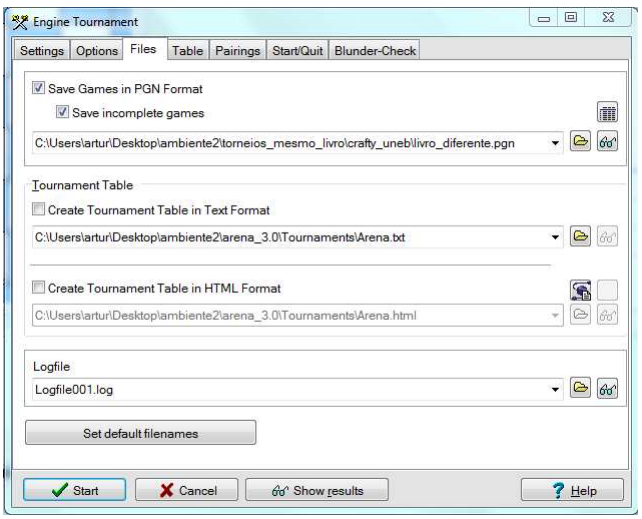

Figura B.2: Tela de arquivos salvos do torneio