UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

# DESARROLLO DE LA MAESTRÍA AJEDRECÍSTICA COMPUTACIONAL UTILIZANDO RECURSOS RESTRINGIDOS

JORGE EGGER MANCILLA

2003

UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

# DESARROLLO DE LA MAESTRÍA AJEDRECÍSTICA COMPUTACIONAL UTILIZANDO RECURSOS RESTRINGIDOS

JORGE EGGER MANCILLA

| COMISIÓN EXAMINADORA | | CALIFICACIONES | | |
|---|---|---|---|---|
| | | NOTA (n$^o$) | (Letras) | FIRMA |
| PROFESOR GUÍA<br>SR. CLAUDIO GUTIERREZ | : | ..... | ..... | ..... |
| PROFESOR CO-GUÍA<br>SR. ALEJANDRO BASSI | : | ..... | ..... | ..... |
| PROFESOR INTEGRANTE<br>SR. FELIPE CSAZAR | : | ..... | ..... | ..... |
| NOTA FINAL EXAMEN DE TÍTULO | : | ..... | ..... | ..... |

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN COMPUTACIÓN

SANTIAGO DE CHILE
JULIO 2003

# Índice general

# Capítulo 1

# Introducción

El presente documento corresponde a la memoria para la obtención del título de Ingeniero Civil en Computación. El tema desarrollado en este trabajo se denomina "Desarrollo de la Maestría Ajedrecística Computacional utilizando recursos restringidos".

La idea es el realizar un estudio acerca de la capacidad ajedrecística de una máquina con hardware fijo, complementándolo con un estudio comparativo de aquellos parámetros que intervienen en su fuerza de juego. Este trabajo involucra también el estudio del desarrollo computacional observado a nivel de Hardware y Software en la confección de las máquinas que juegan ajedrez, particularmente desde el punto de vista ajedrecístico, así como la realización de una investigación final acerca de competencias o torneos de ajedrez en las cuales estas máquinas han participado.

El documento presenta los aspectos iniciales en la preparación de este tema de memoria, la motivación y justificación que llevaron al autor a la elección de este tópico, presentación de objetivos generales y específicos buscados en la memoria de título, revisión bibliográfica inicial y el desarrollo del trabajo propiamente tal.

## 1.1. Antecedentes

Desde hace siglos, la posibilidad de crear una máquina que jugase al ajedrez ha sido un estímulo de la inventiva y creatividad de muchos hombres. Los primeros indicios de este objetivo datan del año 1769, cuando la primera máquina llamada "El Turco" efectuaba sus movimientos ante la corte de emperadores de Austria dejándolos asombrados por su dominio del juego de las 64 casillas. Pasarían desde entonces más de 70 años para demostrar que el famoso "autómata" no era más que una farsa al estar controlado en su interior por humanos.

A pesar de que esta historia puede asociarse sólo a una fallida fantasía, marca en sí el inicio de una larga y tediosa carrera por lograr que máquinas desarrollen la habilidad de "pensar" como un ser humano, habiéndose tomado al ajedrez como un medio de expresión y testeo del trabajo realizado por los investigadores en este tema.

Durante la segunda mitad del siglo XX el interés y esfuerzo enfocado en el desarrollo de máquinas que jugasen al ajedrez creció de una manera impresionante. Mucho tuvo que ver el apoyo obtenido por los avances logrados en otras ciencias como la física, matemáticas, electrónica y otras con no más de algunos decenios de años de antigüedad como la cibernética y la informática. Muy asociada a estas dos últimas ciencias entra la Inteligencia Artificial, tema que pasó por grandes altos y bajos durante los últimos 40 años del siglo XX, pero que impulsó aun más la carrera por lograr que una máquina jugase mejor al ajedrez que cualquier ser humano.

¿Que es lo que se buscaba con esto? Mas allá de sólo lograr demostrar al mundo que una máquina podía vencer al hombre en una actividad en que el mejor pensamiento, capacidad de cálculo y habilidad de razonar eran las bases de la victoria, se buscaba mejorar el desarrollo de la "inteligencia" en las máquinas, basándose en el ajedrez como elemento de test de avances en este desarrollo. Mucho hay escrito acerca de este tema, dado que apasionó e interesó a gran cantidad de investigadores sobretodo en el tema de la Inteligencia Artificial. A pesar de esta gran disposición de información, no mucho se ha realizado en función de analizar comparativamente el real avance a nivel de Hardware y Software realizado durante los últimos años y cuánto significaba en términos concretos (nivel de rating por ejemplo) cada avance que podía realizarse sobre el programa que jugaba ajedrez.

## 1.2. Justificación y Motivación del Tema

Muchos ajedrecistas e Investigadores asumen la victoria de DeepBlue sobre Kasparov en 1997 como la derrota del hombre a manos de la "fuerza bruta", en el sentido de que la máquina habría estado sobredimensionada en su capacidad de Hardware (velocidad de procesador, memoria, capacidad de disco, etc.) por sobre los avances en software (algoritmos, métodos de cálculo, etc.). ¿Cómo demostrar esto de una manera científica? ¿La victoria de Deep Blue fue causal de su impresionante capacidad de Hardware o bien de todos los avances reflejados en el software que este "monstruo computacional" disponía?

Estas últimas preguntas apuntan a lo que se buscaría con el desarrollo de esta memoria, investigar los puntos críticos en donde se demuestra la capacidad ajedrecística en las máquinas que juegan al ajedrez, analizando desde un punto de vista ajedrecístico el software desarrollado, su enfoque y avance que ha tenido en el tiempo. Relacionado con esto por supuesto debe haber un análisis de la capacidad de hardware/software y su relación con el nivel de la máquina.

¿Es posible responder cuanto "juega" una máquina de acuerdo a su capacidad de hardware y/o software? Esto podría medirse en forma concreta en términos del rating utilizado en el Ajedrez (ELO), pero esto ya supone inconvenientes ligados a las inexactitudes que podría tener este sistema. ¿Existirá entonces un buen y representativo sistema para medir la capacidad de una máquina? ¿Es el sistema ELO suficiente o bien deberíamos utilizar parámetros adicionales para medir esta capacidad?.

Las competencias entre máquinas de ajedrez han sido el lugar de encuentro de distintos investigadores para poder probar y testear en forma concreta el real nivel de sus maquinas. Luego de cada competencia muchas conclusiones son obtenidas respecto de las fortalezas y debilidades de

cada programa. Ahora bien, ¿qué tan exhaustivas han sido estas competencias en el sentido de saber diferenciar la capacidad de Hardware y Software de una maquina?

Las pautas establecidas en la mayoría de las competencias han distado de ser "ordenadas" en términos de limitar los niveles de Hardware y Software con tal de medir una capacidad particular dentro de la competencia. Relacionando esto con nuestras preguntas anteriores, un campeonato de ajedrez entre computadoras en el cual el hardware esté limitado (fijo y similar para todos los participantes) y el software sea modificable a partir de una base común podría ser una excelente oportunidad para investigar la diferenciación de nivel en base al esfuerzo en desarrollo de software. Una propuesta de competencia en base a estos parámetros es otro tema que ha estado probablemente lejos de investigación y puede ser el inicio de un evento de bastante interés a nivel ajedrecístico, de programadores e investigadores, con tal de buscar mejoras en algoritmos de inteligencia artificial. La realización de una competencia de este tipo en nuestro país es un proyecto aún sin propuesta, lo cual seguramente se deba a las limitancias señaladas anteriormente, si bien tendría la motivación de marcar un hito en nuestra historia computacional local, aparte de incentivar a gran número de estudiantes a probar sus capacidades en el desarrollo de algoritmos para este fin.

Finalmente, como jugador de ajedrez, el autor de esta memoria ha tenido una experiencia de más de 15 años participando en competencias de carácter Internacional. Campeón nacional en 3 oportunidades y representante chileno en las olimpiadas mundiales en 5 ocasiones, el interés por realizar un estudio relacionado con el "deporte ciencia" va unido a un buen nivel y entendimiento del juego, pretendiendo analizar los enfoques dados en los trabajos realizados en el pasado para mejorar el nivel de las máquinas, estudiando la correctitud y relevancia de los métodos aplicados por investigadores que seguramente no ostentaban un nivel de jugador profesional.

# Capítulo 2

# Objetivos

## 2.1. Objetivos Generales

- Estudiar un sistema de medición de la capacidad ajedrecística de una máquina dado parámetros fijos de Hardware.
- Realizar un estudio comparativo de los parámetros relevantes que intervienen en la medición de la capacidad ajedrecística de una máquina respecto de su software y hardware.
- En base a lo anterior, proponer las bases para competencias de ajedrez entre máquinas con software y hardware común para todos los competidores donde las diferencias estarán dadas en la modificación adecuada de un conjunto de parámetros.

## 2.2. Objetivos Específicos

- Analizar históricamente las propuestas realizadas por distintos investigadores en la realización de software para jugar al ajedrez.
- Enjuiciar desde un punto de vista ajedrecístico los avances históricos realizados en software para máquinas que jugasen al ajedrez.
- Analizar desde un contexto histórico y técnico las distintas competencias ajedrecísticas realizadas entre computadoras y hombres v/s computadoras
- Proponer la realización de una competencia entre computadoras en Chile, basado en el estudio realizado en esta memoria.
- Estudiar los aportes realizados por jugadores de ajedrez de nivel mundial en el desarrollo de máquinas para jugar ajedrez.
- Analizar la situación actual (interés y recursos involucrados) en el desarrollo de máquinas que juegan al ajedrez.

- Estudiar las investigaciones y trabajos realizados en Chile respecto de máquinas que jueguen ajedrez.
- Analizar partidas de las máquinas desarrolladas verificando sus fortalezas y debilidades ajedrecísticas, mejoras y avances respecto de versiones anteriores.

# Capítulo 3

# Metodología Propuesta

El primer punto por el cual debe partir este trabajo debe ser una exhaustiva revisión bibliográfica del tema "ajedrez y computadoras", esto pues la cantidad de trabajos y estudios realizados en los últimos 50 años respecto del tema es realmente grande. Cabe destacar que a pesar de haber mucha bibliografía, no existe una buena clasificación de esta de acuerdo al tema puntual que uno desee investigar referente a ajedrez y computadoras, lo cual involucrará algún esfuerzo extra en diferenciar lo que será útil para el propósito de nuestro trabajo.

La siguiente etapa en el desarrollo de este trabajo consistirá en analizar desde un punto de vista ajedrecístico el aporte en nuevas ideas de desarrollo de algoritmos entregado por los principales investigadores del tema y ver cuán fundamentales han sido estos aportes en los avances de la capacidad de juego de las máquinas desde que fueron presentados hasta la actualidad.

Esto último tiene como fin establecer criterios de análisis en la composición del software de una máquina que juega ajedrez, o bien dicho de otra manera, poder enjuiciar a un programa de ajedrez por aquellas implementaciones que presenta y aquellas las cuales carece, debiendo definir mediante algún método comparativo la importancia de estas distintas implementaciones. Justamente esto último sea probablemente una de las ambiciones mas difíciles de lograr en este trabajo, lograr determinar parámetros de medición de la importancia de ciertas implementaciones y desarrollos presentes en un programa que juega ajedrez.

Finalizado esto se propone realizar una propuesta de competencia entre máquinas que juegan ajedrez pero con requisitos que permitan medir la capacidad de software de las maquinas en competencia, vale decir, bajo un hardware fijo y con un código fuente base el cual puede ser modificado por los competidores.

# Capítulo 4

# Revisión Bibliográfica

A pesar de la corta vida del tema, existe muchísima bibliografía dedicada a ajedrez y computadoras, tanta que la propia revisión de ella resultó ser un tema que involucró bastante esfuerzo.

Desde el punto de vista computacional, mucha de esta literatura se ha enfocado en describir el "cómo hacen" las computadoras para jugar al ajedrez, citando las principales herramientas utilizadas por las máquinas para calcular y evaluar movimientos. Existen muchísimas publicaciones computacionales relacionadas con el tema, notándose un interés especial por su desarrollo.

Por otro lado, desde el punto de vista ajedrecístico no ha existido un interés tan "masivo" en el tema, puesto que las publicaciones de ajedrez que citan el trabajo realizado por computadores se han enfocado principalmente en el análisis de los movimientos y cálculos realizados por las máquinas, muchas veces en una forma quizás algo irónica respecto de la incapacidad de ellas de evaluar correctamente ciertas posiciones en base a parámetros estratégicos.

Las principales publicaciones mundiales de ajedrez (New In Chess, U.S. Chess, Europe Echecs) dan reportes de los campeonatos jugados (principalmente humanos v/s máquinas) sin analizar los aspectos relevantes que logran la fuerza en el juego del programa, lo cual no las hace un buen punto de referencia para esta memoria.

A continuación presentamos un resumen del principal material bibliográfico revisado y utilizado para esta memoria. Como mencionamos, hay muchísimas publicaciones existentes acerca del tema, por lo cual la tarea de clasificar y describir el material ha resultado bastante mas difícil de lo que se presumía en un principio, pero será un excelente punto de referencia para este trabajo además de un buen punto de partida para futuros estudios relacionados con el tema.

El orden de presentación de esta revisión es histórico, abarcando desde la primeras publicaciones e intentos a principios de siglo hasta las investigaciones actuales.

## 4.1. Publicaciones Previas a 1950

### 4.1.1. Primeras Ideas : Charles Babbage y Konrad Suze

Charles Babbage (1792-1871) fue un brillante científico matemático quien mantuvo cierto interés hacia el ajedrez pero no como un jugador de gran fuerza. Fue cerca de 1840 cuando Babbage se interesó en la posibilidad de crear una máquina que lograra jugar un juego de alto nivel intelectual. En su texto "The Life of a Philosopher" presenta sus primeras ideas acerca de "como" debería actuar la máquina frente a un juego de inteligencia: "*Frente a la pregunta de si es esencial el razonamiento humano para jugar juegos de inteligencia (como el tic-tac-toe, ajedrez, damas, etc.) muchos afirman con total seguridad de que esto es prácticamente indiscutible. De no ser así, cualquier autómata podría jugar a estos juegos. Aquellos con cierto apego a las ciencias exactas dieron cabida a que una máquina podría realizar este tipo de actividades, pero un número importante considera esto como imposible considerando de partida el numero de combinaciones de movimientos que involucran estos juegos.*
*En la primera parte de mi estudio muy pronto llegué a la conclusión de que cualquier juego de inteligencia es susceptible de ser practicado por un autómata. A cualquier movimiento que haga el autómata, una respuesta vendrá de su adversario. Ahora el estado alterado del tablero es uno de muchas posiciones de su adversario en la cual se supone el autómata será capaz de actuar. Luego la pregunta se reduce a realizar el mejor movimiento bajo cualquier combinación de posiciones posibles del adversario. Entonces, el autómata debe considerar preguntas de la siguiente naturaleza*:

1. *¿Es la posición de mi adversario consistente con las reglas del juego?*
2. *Si es así, ¿ha perdido el juego el autómata?*
3. *Si no, ¿ha ganado el autómata el juego?*
4. *Si no, puede ganar a la próxima movida? Si es así, hacer la movida.*
5. *Si no, ¿puede su adversario, moviendo, ganar el juego?*
6. *Si es así, el autómata debe prevenir esta movida.*
7. *Si el adversario no puede ganar el juego, el autómata debe examinar cómo realizar una movida como esa siguiendo una secuencia de dos movimientos. Si no encuentra solución, entonces seguir una secuencia de tres o más movimientos.*

*En consecuencia el problema de hacer que un autómata juegue en un juego de inteligencia depende de la posibilidad de que la maquina pueda representar el esquema de combinaciones relativas al juego.*"

En base a esto uno puede suponer que Babbage estimó, a mediados del siglo XIX, un esquema de búsqueda en profundidad que puede ser utilizado para automatizar el proceso analítico de una partida de ajedrez.

Por otro lado, el pionero de las ciencias de la computación en Alemania, Konrad Zuse, escribió acerca de la posibilidad de crear un programa de ajedrez en su libro "Der Plankalkuel", el

cual fue publicado en 1945. Zuse describió cómo un generador de movimientos legales puede ser programado, pero no consideró en su trabajo técnicas de evaluación o búsqueda. Desarrolló una computadora, la "Zuse-1", pero ésta no fue utilizada para jugar ajedrez.

### 4.1.2. Trabajos De Claude Shannon y Allan Turing

El 9 de Marzo de 1949 Claude E. Shannon, un investigador científico de los laboratorios Bell de New Jersey, presentó un paper en una convención en Nueva York. Su paper se llamaba "Programming a Computer for Playing Chess" [68] y su enorme significancia recae en que muchas de las ideas originales de Shannon son aún utilizadas en los programas de ajedrez de la actualidad. Shannon no hizo mención a cierta importancia en las computadoras que jugasen ajedrez, pero si relacionó que la investigación de este tema tendría como consecuencia el progreso en otras áreas de solución automática de problemas.

Shannon propuso varias consideraciones que deben ser tomadas en cuenta en la función de evaluación de posiciones de un programa de ajedrez:

- Ventaja Material
- Estructura de peones
  - Peones Aislados y retrasados.
  - Control relativo del centro.
  - Peones en color opuesto al alfil propio.
  - Peones pasados
- Posición de las piezas.
  - Caballos avanzados, protegidas o atacadas.
  - Torres en columnas abiertas o semiabiertas.
  - Torres en séptima fila.
  - Torres dobladas.
- Posibilidades de ataque.
  - Piezas que protegen a otras piezas.
  - Ataques sobre otras piezas.
  - Ataques sobre casillas adyacentes al rey enemigo.
  - Clavadas.
- Movilidad, medida por el numero de movimientos legales posibles.

Shannon describió dos tipos de estrategias para el juego de ajedrez por máquinas. La mas primitiva, llamada "tipo-A", consistía en la búsqueda en profundidad con un límite a lo largo de

cada “rama” del árbol, para luego utilizar el algoritmo mini-max. La estrategia “tipo-B” introdujo un método de búsqueda selectiva en el cual ciertas líneas eran más profundizadas que otras.

El mismo Shannon acotó que la estrategia “tipo-A‘” presenta al menos tres desventajas. En una posición de medio juego hay cerca de 30 posibles movimientos. Luego de una movida de cada bando las posibilidades crecen a 1000 nodos terminales los cuales deben ser evaluados. Luego de tres movimientos la proporción crece a $10^9$ nodos terminales, lo cual implica que a una tasa de evaluación de 1 por cada microsegundo el programa tardaría al menos 16 minutos para cada movimiento.

Segundo, un jugador de ajedrez examina algunas variantes con profundidad de pocos niveles y otras con cerca de 20 niveles de profundidad. Esto es lo que se llamo “búsqueda selectiva” en la estrategia “tipo-B”. La tercera desventaja es el problema derivado de las posiciones “inestables”, aquellas en donde hay cambios de piezas o bien secuencias de jaques.

Referido a esto último, Shannon introdujo el concepto de “estabilidad”, la cual aplicó a su segundo tipo de estrategia, proponiendo lo siguiente:

1. Examinar variantes forzadas lo más profundamente posible, evaluando sólo posiciones estables.
2. Seleccionar mediante un proceso externo las variantes importantes de ser analizadas, evitando perder tiempo analizando variantes inútiles.

Estos dos criterios son claramente utilizados por jugadores humanos, siendo el segundo nada más que el algoritmo alfa-beta el cual fue utilizado en programas a partir de 1960.

En 1951 Alan Turing publicó los resultados de su trabajo en computadoras de ajedrez en la Universidad de Manchester [20]. Turing fue uno de los principales pioneros durante los primeros años de la computación y estuvo convencido de que los juegos constituían un modelo ideal de estudio de la inteligencia en maquinas y como esta debía conducirse. Esta idea es seguida en los dias de hoy por quienes trabajan en Inteligencia Artificial, pero en los tiempos de Turing la opinión era mirada con bastante escepticismo.

Las primeras ideas de Turing acerca de computadoras de ajedrez datan de 1944. Cuando discutía sus ideas con varios colegas de la universidad. Entre 1947 y 1948 el y D.G.Champernowne realizaron un analizador de movimientos denominado TUROCHAMP. Al mismo tiempo Donald Michie y S.Wylie desarrollaron otro analizador llamado MACHIAVELLI. Estos analizadores permitieron la creación de una computadora que jugaba ajedrez con profundidad de 1 movimiento. Estos simplemente calculaban los puntajes de las posiciones con profundidad 1 y entonces realizaban el movimiento con mayor score. Un match entre estos dos entes se programo pero nunca fue realizado. Solo 30 años mas tarde, MACHIAVELLI jugó un match contra otro analizador, SOMA, desarrollado por Maynar Smith. SOMA utilizaba una función de evaluación en el cual 3 parámetros eran de importancia. Material, mobilidad y cambio de valores.

Durante el transcurso de su investigación en computadoras de ajedrez Turing intento programar a TUROCHAMP y MACHIAVELLI en la computadora Ferranti Mark 1 en Manchester, pero nunca pudo completar su trabajo y fue incapaz de lograr que las maquinas jugasen en forma automática. Las conclusiones de estos trabajos están publicadas en [21]. La significancia del trabajo

de Turing radica en haber sido la primera persona que diseñó un programa que podía jugar ajedrez. Siendo sinceros, el primer juego grabado entre dos maquinas resulto ser una tediosa simulación realizada en las manos de Turing (pág.124).

Turing utilizó una función de evaluación bastante simple en la cual el material era el factor dominante. El tamaño del árbol de variantes era de 1 jugada, examinando todas los movimientos posibles en profundidades mayores, deteniendo el análisis cuando una posición "muerta" era abordada, esto es, una posición en donde no existían movimientos considerables.

Un movimiento considerable era:

- Capturar una pieza no defendida.
- Recapturar una pieza.
- Capturar una pieza defendida con otra de menor valor.
- Dar Jaque Mate.

De ser las evaluaciones de material iguales en los movimientos analizados, los factores posicionales decidían el mejor. En estos se incluía: Movilidad, Piezas protegidas, Movilidad del Rey, Protección del rey, Enroque, posición de peones y Amenazas de jaque o jaque mate.

Turing asumía las debilidades de sus programas señalando que muchas veces estos eran una representación de su propio nivel de juego. "Uno no puede programar una maquina para que juegue mejor de lo que uno juega"S.

Tanto Turing como Shannon proponen las primeras técnicas e ideas básicas para la construcción de maquinas que jueguen al ajedrez, si bien con un enfoque mucho mas teórico que práctico. Curiosamente, las ideas propuestas por estos insignes investigadores corresponden a la base utilizada en la realización de software para jugar al ajedrez, lo cual hace tener a estas referencias un valor histórico muy alto. Se mencionan las primeras ideas acerca de búsqueda en profundidad, funciones de evaluación y técnica de juegos. No se hace un análisis de medición ni comparación de niveles de fuerza para máquinas que jueguen al ajedrez.

## 4.2. Trabajos De La Segunda Mitad De 1900

### 4.2.1. Los Primeros Programas De Ajedrez

**El Programa Los Alamos**.
El primer trabajo documentado acerca de una máquina que jugaba al ajedrez fue realizado en el año 1956, y trataba acerca del trabajo realizado por cinco científicos (Jame Kister, Paul Stein, Stanislaw Ulam, William Walden y Mark Wells) en el Laboratirio Científico de Los Alamos en Nuevo México, famoso por sus trabajos en el "Proyecto Manhattan". El grupo Los Alamos menciono la existencia de un artículo publicado en Pravda, que se refería a un programa que había sido escrito para una

computadora BESM en Moscú, pero el articulo no daba una mención detallada acerca de su nivel de juego, sino que sólo mencionaba que un jugador de nivel débil había logrado vencer a la máquina.

El juego jugado por el programa Los Alamos no era realmente ajedrez, sino que una versión simplificada de este (sin Alfiles y en un tablero de 6x6) con la cual los programadores pretendían simular técnicas de búsqueda y evaluación pero con un universo mas limitado, con tal de extrapolar luego sus resultados al juego real. El programa corría en una computadora MANIAC cuya velocidad era de 11.000 operaciones por segundo, siendo capaz de realizar búsqueda de 4 jugadas (profundidad 2) en cerca de 12 minutos. El programa utilizaba una función de evaluación basada en material y movilidad.

Una de las versiones del programa fue enfrentada contra Martín Kruskal, una matemático de Princeton quien era también un fuerte jugador de ajedrez. Kruskal le dio a MANIAC la ventaja de la dama. La partidam tomo varias horas de juego y atrajo el interés local. Luego de cerca de 15 movimientos Kruskal no había logrado recuperar ningún material e incluso empezo a llamar a su oponente "el" en vez de "eso". A medida que el juego progresaba parecía que el Dr. Kruskal iba a perder, pero cerca del movimiento 19 el programa eligió una débil continuación y Kruskal logro recuperar su dama dando mate.

El programa fue luego enfrentado con una joven mujer quien habia aprendido los principios básicos del juego hacia pocos dias. El resultado fue una victoria para el programa. Ambas partidas se encuentran documentadas en la mayoría de las fuentes históricas acerca del tema.

**El programa de Bernstein**.
En junio de 1958 Alex Bernstein y Michael Roberts [37] publicaban un artículo en el "Scientific American" haciendo referencia a la dificultad y complejidad del juego del ajedrez, describiendo al público un programa que habían desarrollado junto con Timothy Arbuckle y Martín Belsky. Las ideas de estos científicos habían sido expresadas por Shannon ocho años atrás, pero el paper de Shannon fue puramente teórico mientras que el artículo de Bernstein/Roberts describió un programa que jugaba razonablemente en una computadora IBM 704.

El programa tenia mucha similitud con las máquinas que conocemos hoy en día. Poseía un analizador de movimientos legales y su función de evaluación utilizaba cuatro parámetros: Movilidad, casillas controladas, defensa del Rey y material. Buscaba con profundidad de 4 jugadas (2 movimientos) y con tal de no demorar demasiado el análisis de las jugadas el programa decidía primero entre siete posibilidades de acuerdo a un análisis previo con una rutina de selección. Esta rutina se basó en los siguientes criterios en estricto orden : criterios de jaque, ganancia de material, posibilidad de enroque, desarrollo, ocupación de casillas críticas, ocupación de columnas abiertas, movimientos de peones y movimientos de piezas. Con esto el programa lograba una profundidad de 2 movimientos. Examinaba 7 posiciones con profundidad 1; 49 con profundidad 2; 343 con profundidad 3 y 2401 con profundidad 4; un total de 2800 posiciones, algo bastante manejable para un computador de la época.

**Primeros programas Soviéticos** Si bien no tuvieron mucha publicidad en el oeste, hubo mucha investigación en computadoras de ajedrez en la Union Soviética durante mediados de 1950. En 1956 V.M. Kurochkin escribió un programa que solucionaba problemas de ajedrez. Su programa corría en una computadora Strela, y podía solucionar problemas de mate en 2 movimientos mas rápido que cualquier humano (2 a 4 minutos), pero problemas de 3 o 4 movimientos requerían 10

a 12 minutos para ser solucionados. Kurochkin comentaba que el problema principal era el análisis de todas las posibilidades por parte de la maquina, algo que los humanos saben descartar desde un principio.

Un año mas tarde el mismo Kurochkin escribió un programa que jugaba el final de Rey y dos Alfiles v/s Rey. El primer intento de crear un programa que jugase una partida completa fue hecho por G.Sehlibs, con un programa que lograba una profundidad de 3 jugadas. No hay referencias de partidas jugadas por este programa, ni tampoco opiniones acerca de su nivel de juego.

El primer programa bien documentado escrito en la Union Soviética hizo su debut el año 1961. Fue escrito en el Instituto Matemático Styeklov de la Academia de Ciencias de la URSS por un equipo que incluía al profesor M.Shura-Bara, I.Zadykhalio, E.Lublinsky y V.Smilga (candidato a maestro soviético de ajedrez). Una descripción de su trabajo fue considerada lo suficientemente importante para ser publicada en el boletín numero 8 del match por el campeonato mundial de 1961 entre Tal y Botvinnick.

El programa Stylekov ocupaba las siguientes características en su función de evaluación: Material, movilidad, piezas defendidas, estructura de peones, control del centro, clavadas, protección del Rey. El programa no poseía algoritmo de búsqueda por lo que su juego era tremendamente limitado. Tomaba cerca de 30 y 58 jugadas por movimiento.

**Newell, Shaw y Simon (El programa NSS)**

Alan Newell, John Shaw y Herbert Simon [65]comenzaron su trabajo en computadoras de ajedrez el año 1955 en el Instituto de Tecnología de Carnegie (ahora conocido como la universidad de Carnegie Mellon) en Pittsburg. Sus progreso en el desarrollo de una máquina que jugase al ajedrez fueron bastante lentos, debido a que el principal objetivo del grupo era realizar un programa que realizara demostraciones de teoremas en lógica.

El programa NSS fue construido en base a módulos, cada uno asociado con un objetivo particular de una partida de ajedrez. Objetivos típicos son defensa del Rey, balance de material, control del centro, desarrollo, ataque y promoción de peones. Cada objetivo tenia asociado un conjunto de procesos: un generador de movimientos, una evaluador estático de posiciones y un generador de análisis.

El generador de movimientos asociado con cada módulo proponía movimientos relevantes a ese módulo. Por ejemplo el módulo de control de centro propondría el movimiento 1.e4 al principio de la partida, mientras que el módulo de material propondría movimientos que ganasen o mantuviesen igual el balance de piezas capturadas.

El valor asignado a cada movimiento propuesto por cada módulo era obtenido de una serie de evaluaciones. El “score” asociado a una determinada posición era el resultado de las distintas componentes. Cada componente expresaba la aceptación o rechazo de alguna posibilidad en base a su propio objetivo. Un ejemplo de módulo era el de “Control Central”. Este módulo operaría amenos que no hubiese peones centrales para movilizar en la cuarta o quinta fila. La forma de operar de este módulo era la siguiente:

- Mover e4, o d4 (movimientos primarios)
- Prevenir los movimientos primarios del oponente.
- Preparar los movimientos primarios propios mediante :
  - Apoyo de piezas
  - Eliminación de piezas o peones adversarios que controlen el centro.

La evaluación estática de control de centro simplemente contaba el número de casillas dominadas en el centro y ponderaba por las posibilidades de realizar movimientos que ayudaran a su mayor control.

La importancia de considerar objetivos principales y modularizarlos tuvo la ventaja de dar mucha mayor flexibilidad al programa en la selección de movimientos dependiendo de la etapa del juego en que se encontraba. Esta "selección" de acuerdo a objetivos principales, la cual descartaba posibilidades que no eran el objetivo principal de la partida, fue el principio del uso del algoritmo alfa-beta, cuyo primer uso en programas de ajedrez fue justamente en el programa NSS.

**El programa de Anderson/Cody**

En el año 1959 un programa canadiense fue exhibido en la Universidad de Toronto. Fue escrito por Frank Anderson, un maestro internacional de ajedrez, y Bob Cody. El programa corría en una computadora IBM 605 y jugaba sólo finales de peones muy simples (el mas complejo era de 2 peones y rey v/s Rey) y los programadores centraron sus esfuerzos en desarrollar una estrategia perfecta par este tipo de finales. El programa jugó una exhibición de simultáneas contra 50 oponentes, cada uno de los cuales podía elegir su posición de partida. En cada partida el programa jugó casi perfectamente. Lamentablemente no hay suficiente documentación acerca del proyecto ni de la estrategia de programación del juego de final. Ni siquiera el mismo Anderson cuando fue consultado en 1970 acerca de su método recordaba la forma en que lo hizo.

**El programa Kotok/McCarthy**

En 1961 Alan Kotok escribió un programa de ajedrez para su disertación de titulo en el Instituto de tecnología de Massachussets [55]. Su programa fue escrito bajo la guía de John MacCarthy, uno de las figuras lideres en el mundo de la Inteligencia Artificial, quien en ese entonces era profesor del MIT.

El programa de Kotok realizaba una búsqueda en profundidad con limite variable. Profundizaba en posiciones hasta que estas fueran "estables" o bien sobrepasara su limite máximo de búsqueda. El generador de movimientos del programa era similar al utilizado por Newell , Shaw y Simon, mientras que la función de evaluación utilizaba 4 objetivos: Material, estructura de peones, control del centro y desarrollo.

El trabajo de Kotok se inició en el verano de 1961. Al momento de presentar su tesis en 1962 el programa había jugado cuatro partidas, utilizando un promedio de 10 minutos por movimiento. Su nivel de juego no era bueno, e incluso en alguna oportunidad realizó movimientos ilegales.

Luego de graduarse en el MIT el interés de Kotok en ajedrez y computadoras fue decreciendo, si bien su programa se mantuvo . Cuando McCarthy dejó el MIT para hacerse cargo del Laboratorio de Inteligencia Artificial en Stanford, tomó los trabajos de Kotok y realizó varias modificaciones. A fines de 1966 se realizó un match entre la nueva versión del programa, corriendo en una IBM 7090, y un programa desarrollado en una computadora soviética M20 en el Instituto de Física Experimental y Teórica (ITEP) en Moscú. El programa ITEP fue escrito por Vladimir Arlazarov, George Adelson-Velsky, Alexander Bitman (maestro soviético), Alexander Ushkov y Alexander Zhivotovsky. Utilizaba una función de evaluación basada en: Estructura de Peones (Peones centrales, cadenas de peones, peones aislados y doblados, peones pasados), Movilidad, Defensa del Rey, Balance de material.

El programa ITEP utilizaba una estrategia "tipo-A" utilizando una profundidad de 3 jugadas. El programa americano utilizó una estrategia "tipo-B"logrando profundidad de 4 jugadas. El programa ruso ganó dos partidas mientras que las otras dos fueron tablas (pág. 125). Las anotaciones de estas partidas fueron publicadas en un artículo de Arlazarov y Bitman que apareció en Febrero de 1968 en la revista soviética "Shakhmaty v CCCP" .

Este match hizo mucho por el desarrollo de computadoras de ajedrez dado que estimuló este tópico en los Estados Unidos. Incluso cuando el match estaba siendo realizado, un nuevo programa fue desarrollado en el MIT, y por los siguientes ocho años se vivió una explosión en el interés por el tema. Las competencias entre computadoras empezaron a ser cada vez más frecuentes y algunos científicos empezaron a creer en la posibilidad de crear una maquina con nivel de maestro.

**El programa Greenblatt**

A principios de 1966 un programa de ajedrez fue desarrollado en una computadora PDP-6 en el laboratorio de Inteligencia Artificial del MIT. Este fue escrito por Richard Greenblat [46], entonces estudiante, con la asistencia de Donald Eastlake participando en un torneo de ajedrez local, perdiendo cuatro partidas y empatando una, logrando un rating de la USCF de 1243. En marzo de 1967 jugó otro torneo, ganando una partida y perdiendo cuatro. Su performance resultó ser de 1360. Un mes más tarde logró vencer en dos partidas cayendo en otras dos, pero logrando una performance de 1640. El programa fue bautizado como MACHACK VI y fue hecho miembro honorario de la Federación de Ajedrez de Estados Unidos.

El programa de Greenblatt poseía varias facilidades interactivas para localizar errores y mejorar su performance. Estas ayudas incluían facilidades de visualización en pantalla de evaluaciones seleccionadas, análisis de nodos terminales, factores que causaban que una movida fuese elegida, variante principal de análisis del programa, y estadísticas de tiempo de generación de cálculo y posiciones analizadas. Utilizando esta característica y jugando cientos de partidas contra adversarios humanos Greenblat fue capaz de realizar un programa que era rápido, eficiente y relativamente libre de errores de programación.

El generador de movimientos de Greenblatt poseía tres funciones básicas. Seleccionaba los movimientos que consideraba lógicos, de acuerdo a su orden de mérito los ubicaba con tal de optimizar el uso del algoritmo alfa-beta, para luego calcular valores posicionales y de desarrollo que decidirían que movimiento realizar. La mejor cualidad del programa era su conocimiento ajedrecístico, el cual estaba representado en las cerca de cincuenta heurísticas utilizadas en computar los movimientos posibles, las cuales estaban destinadas a distintas etapas del juego.

La función de evaluación del programa consideraba balance de material, capacidad de acción de las piezas, estructura de peones, defensa del Rey y control central. El árbol de búsqueda del programa se basaba en una estrategia Shannon tipo-B.

Una novedad introducida en este programa fue el uso de un pequeño libro de Aperturas. Este "libro" fue compilado por dos estudiantes del MIT, Larry Kaufmann y Alan Baisley, ambos jugadores de ajedrez.

**Trabajos de Botvinnick**

Mikhail Botvinnick fue campeón mundial de ajedrez por cerca de 12 años. Su primera victoria fue en el añ0 1948, perdiendo el título en forma definitiva frente a Petrosian en 1963. Aparte de ser un notable maestro de ajedrez Botvinnick fue también Ingeniero Eléctrico, manteniendo muchísimo interés en el tema de cómo desarrollar el proceso de análisis ajedrecístico en una máquina. Sus ideas resultaron ser bastante originales, la mayoría de las cuales fueron publicadas en [2] publicado en 1970. En este texto Botvinnick busca como mejorar, desde el punto de vista ajedrecístico, la capacidad de la máquina para evaluar posiciones y aumentar su profundidad de búsqueda. El autor propone varias interesantes ideas acerca de cómo hacer que un programa analice en profundidad tal como lo hace un jugador experto. Botvinnick propuso un algoritmo que permitía a la computadora analizar un árbol de variantes en la misma forma en que él lo realizaba. Sus publicaciones describieron como su programa PIONEER lograba solucionar problemas muy complejos, examinando sólo un número limitado de variantes y posiciones. Desafortunadamente el periodo de mayor interés de Botvinnick por el tema coincidió con una época en donde en la Unión Soviética el acceso a computadoras era tremendamente limitado. De haber tenido mayor apoyo en esto no cabe duda que sus aportes hubiesen sido mucho mayores.

### 4.2.2. Programas Contemporáneos

A partir de 1970 el desarrollo de programas computacionales se intensificó masivamente. Las competencias mundiales organizadas por la ACM así como los recursos destinados a investigación en el tema cooperaron mucho con su desarrollo en los Estados Unidos.

**CHESS 3.0**

El primer programa ganador de la competencia de la ACM fue CHESS 3.0, desarrollado en la Universidad de Northwest, Illinois. El programa repitió el suceso en los dos años siguientes. En el campeonato mundial de computadoras realizado en Estocolmo en 1974 obtuvo el segundo puesto detrás del programa soviético KAISSA, para luego ser quinto en el siguiente torneo de la ACM. El programa original fue escrito por Larry Atkin, Keith Gorlen y David Slate [69], siendo mejorado cada año de acuerdo a la experiencia obtenida en competencias.

El programa realizaba una búsqueda selectiva de movimientos de acuerdo a 15 rutinas modularizadas. Su capacidad de rapidez de análisis radicaba en la selección previa hecha de los movimientos posibles. CHESS 3,0 y sus descendientes tuvieron grandes resultados frente a otras maquinas. En los 3 primeros eventos de la ACM ganó las 10 partidas que jugó. En el 4to torneo de la ACM logro 3,5 pts en 4 juegos, siendo sólo derrotado en 1974, ya como CHESS 4.0. Mas tarde el mismo

año CHESS 4.0 pierde nuevamente otro juego frente a la computadora RIBBIT, desarrollada en la Universidad de Waterloo, siendo el primer torneo de la ACM en que el programa CHESS no logró ganar.

**TECH**

El programa "Technology" fue escrito por James Gillogly mientras se graduaba en la Universidad de Carnegie Mellon [45]. Su nombre se deriva de la filosofía básica desarrollada en el trabajo de Gollogly, el deseaba escribir un programa basado enteramente en tecnología (computadoras rápidas) y no tanto en heurísticas ajedrecísticas. El objetivo era simplemente generar todos los movimientos legales posibles a una profundidad fija y evaluar las posiciones terminales solo materialmente. Una idea así no puede jugar un ajedrez de buen nivel, pero la idea de Gillogly no era crear un juego de nivel maestro, sino que tener un programa el cual pudiese ser utilizado en la medición de fuerza de otros programas.

Los experimentos de Gillogly no resultaron crear un programa muy útil, dado que el estándar de juego del programa resultaba muy lento para cualquier nivel de profundidad de juego. El programa obtenía frecuentemente posiciones malas e incluso cometía errores tácticos en posiciones no estables. Dado esto Gillogly decidió dedicar cierto tiempo al desarrollo de heurísticas ajedrecísticas en su programa.

Una de las novedades de TECH fue el uso del tiempo de su rival en sus análisis. Si su oponente realizaba la movida predecida TECH era capaz en muchos casos de responder en forma instantánea. La mayor fuerza del programa estaba en su modulo de táctica (búsqueda tipo fuerza bruta). Todas las movidas eran analizadas con profundidad 5 hasta lograr posiciones estables. Incluso utilizando el algoritmo alfa-beta, la búsqueda generaba cerca de 500.000 nodos terminales. El análisis era logrado debido a la simplicidad de la función de evaluación del programa.

Cabe considerar que el desarrollo de heurísticas de TECH diferenció entre las distintas etapas del juego. La más importante de la Apertura era la de control central, con la cual lograba posiciones muy razonables al principio de cada partida.

**KAISSA**

En 1971 un grupo de programadores del Instituto de Control Científico comenzaron a reescribir el programa ITEP, utilizado en 1967 para el match con Stanford. El grupo estaba formado por la base de científicos que colaboraron con el proyecto ITEP mas Mikhail Donskoy, quien asumió el rol de líder en la creación de este programa [1]. Al año siguiente de estar listo, fue utilizado en dos partidas por correspondencia frente a los lectores del diario "Komsomolkaya Pravda". Todos los domingos durante 1972 el diario publicaba los movimientos de KAISSA en cada una de las partidas que se jugaban. Los lectores enviaban sus sugerencias al diario las cuales eran luego computadas. En cada partida la movida con mayor votación era la realizada frente a la maquina. KAISSA empató una partida y perdió la otra (pág. 127. El año previo Spassky había sido invitado para la misma prueba, empatando una partida y ganando la otra, lo cual hacia considerar que el resultado del programa no era para nada malo.

KAISSA poseía una profundidad de 7 jugadas, con análisis variable en variantes que involucraban capturas o movimientos forzados. Su función de evaluación era bastante compleja pues

involucraba muchos parámetros. Utilizó además una variación del algoritmo alfa-beta.

Una innovación interesante fue la mantención de movimientos de amenaza en cada movida. Si existía la posibilidad de crear una amenaza seria con un movimiento, pero que ahora no era posible, esa posibilidad se guardaba para ver si era posible en el movimiento siguiente. Este concepto se asimiló a la idea de amenaza latente en ajedrez. El único inconveniente en este algoritmo de "analogías" es tomar la decisión de "cuando" la posición ha cambiado lo suficiente como para seguir considerando una movida de amenaza.

El mayor suceso de KAISSA fue la obtención del campeonato mundial de computadoras el año 1974.

**Actividad a mediados de 1970**

En el año 1975 las actividades de ajedrez computacional en Europa empezaron a ser más coordinadas con la organización del primer campeonato alemán de computadoras de ajedrez. Este torneo fue realizado en Dortmund y logró atraer a gran cantidad de programadores alemanes. Los esfuerzos de los participantes no lograron llegar al nivel de KAISSA o del programa de la Universidad de Nothwest, pero hubo algunos hechos que merecieron consideración.

El programa ORWELL III, que ocupo la cuarta posición, fue creado por el matemático y artista Thomas Nitsche. Años mas tarde Nitsche crearía el primer computador MEPHISTO que inició su camino hacia la fama y fortuna en la compañía "Hegener & Gliser" en Munich.

La primera ocasión en que jugadores humanos empezaron a considerar a las máquinas como serios rivales fue en 1976, en el Torneo de Ajedrez Paul Masson realizado en Sarasota, California. Participaron 756 jugadores, entre ellos, CHESS 4.5, en la sección "B" para jugadores con rating USCF 1600-1800. El resultado de CHESS 4.5 no dejó de impresionar a nadie, pues logró un 100 % de score, convirtiéndose en la primera máquina jugadora de clase "A". CHESS 4.5 logró luego ganar el torneo de la ACM en 1976, con lo cual recuperó su reputación como el programa más fuerte del mundo.

Al año siguiente el mundo estaría nuevamente asombrado ante CHESS 4.5, puesto que lograba la victoria en el Open de Minnesota, un torneo a seis rondas en donde el programa perdió una sola partida frente a un jugador de mayor clasificación. Esta victoria clasificaba al programa para participar en la sección "cerrada" del torneo el siguiente fin de semana, donde sólo los mejores jugadores de Minnesota tomaban participación. Los jugadores humanos tomaron el asunto muy en serio, y por primera vez en la historia las partidas jugadas por una computadora fueron parte de la preparación teórica de jugadores humanos. CHESS 4.5 logró sólo una victoria y una tabla en sus cinco encuentros, pero dejó su marca en una competencia de ajedrez.

No fue sorpresa que para el año siguiente CHESS 4.6 lograra nuevamente el titulo del campeonato ACM, seguido por KAISSA. Uno de los espectadores de ese evento fue el ex campeón mundial Mikhail Botvinnik, quien expreso en la siguiente frase una profecía que pocos entonces creyeron : "Los maestros de ajedrez acostumbraban venir torneos entre computadoras para reír. Ahora, vienen para mirar. Pronto vendrán a aprender".

**La apuesta de Levy**

David Levy es uno de los mayores expertos en el mundo acerca de ajedrez y computadoras. Maestro Internacional de ajedrez, ha escrito muchas publicaciones relacionadas con el tema [11, 12, 13, 14, 15]. En 1978 jugó un match a seis partidas frente al computador "CHESS 4.7". El motivo del match era una apuesta realizada 10 años atrás con otros científicos en que Levy aseguraba no perdería un match con una computadora durante los siguientes 10 años.

Hasta 1977 parecía que Levy ganaría su apuesta sin mayor esfuerzo, puesto que los programas si bien participaban en competencias no lograban desarrollar un nivel de maestro. Pero entonces CHESS 4.5 comenzó a lograr buenos resultados, por lo cual no se tardo en organizar un match entre Levy y este programa. El primer enfrentamiento fue un match a dos juegos realizado en abril de 1977 en la Universidad de Carneige Mellon. Este match fue fácilmente ganado por Levy, pero aun restando ocho meses para cumplir el plazo de la apuesta una nueva versión del programa CHESS fue realizada, "CHESS 4.7", contra la cual el MI inglés debía enfrentar su reto final.

El match comenzó en Agosto de 1978 en el Centro de Exhibiciones Canadienses en Toronto. El programa estaba representado por David Slate y el match fue pactado al mejor de seis encuentros. La primera partida de este match resultó ser un shock emocional para todo el mundo, puesto que CHESS 4.7 logró una posición ganadora en la apertura, pero luego teniendo la ventaja no supo rematar terminando la partida en tablas (pág. 128). Levy aprendió la lección de esta partida y adopto una estrategia de juego "anti-computadora", la cual le reportó dos victorias seguidas (pág. 129). La cuarta partida del match fue un hito histórico en el encuentro, pues Levy buscando jugar con un estilo mas arriesgado termino perdiendo su primera partida del match, siendo la primera ocasión en que una máquina derrotaba a un jugador de nivel de maestro (pág. 129).

Queriendo asegurar el resultado del match Levy volvió a la estrategia inicial de espera con lo cual ganó las siguientes partidas del match y por ende la apuesta. Según palabras del propio Levy luego de este match la posibilidad de ver a una máquina derrotando a un gran maestro de ajedrez era algo que podríamos ver en un futuro no lejano.

**La carrera en la década de 1980**

El equipo desarrollador de la Universidad de Northwest (Slate y Atkin) había logrado a fines de los 70 crear una máquina con gran nivel de juego. Luego de su separación en 1979, Atkin concentró sus esfuerzos en escribir programas de juego para microcomputadoras. Slate escribió un nuevo programa llamado NUCHESS, el cual finalizó en segundo puesto en el ACM de 1981 (detrás de BELLE), primero en 1982, 4to en el Campeonato mundial de 1983 y sexto en 1984. Si bien aun permanecía en las "grandes ligas" Slate estaba siendo eclipsado por el equipo de BELLE.

El programa BELLE fue escrito por Ken Thompson y Joe Condon [7], logrando como primer resultado importante el campeonato ACM de 1978. Thompson había tomado parte en distintos proyectos de programas de ajedrez, pero no había logrado demostrar superioridad sobre el equipo de Northwest. El y Condon mencionaban que "El real avance de las computadoras de ajedrez se verá con el progreso que podamos obtener en Hardware". La conclusión a la que llegaron era la necesidad de crear hardware especialmente diseñado para este tipo de máquinas, con tal de reemplazar tareas normalmente realizadas por software.

En el torneo realizado en Toronto tomaron participación con una máquina de 25 chips finalizando en cuarto lugar. Luego, en Washington consiguieron nuevo hardware, una maquina de 325

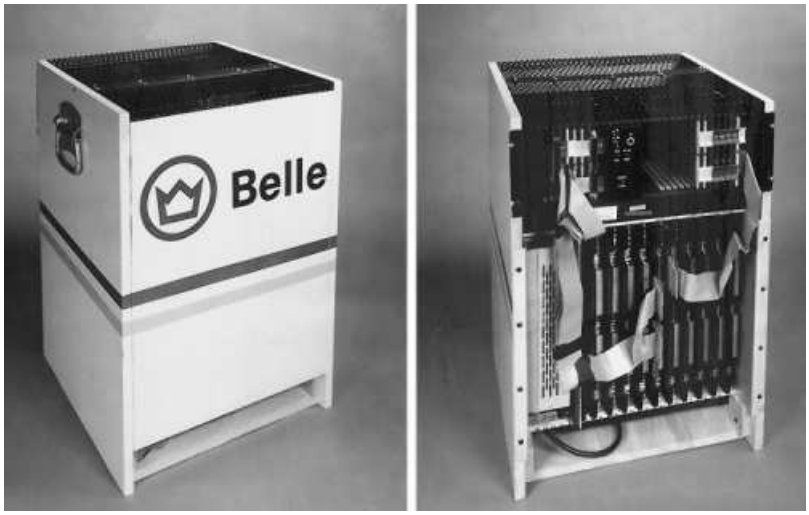



Figura 4.1: El computador BELLE

chips, con la cual vencieron a CHESS 4.7 y marcaron el fin de una era de dominio del equipo de Northwest.

En 1980, durante el desarrollo del tercer Campeonato Mundial de Computadoras jugado en Linz, Austria, BELLE había crecido a una máquina de 1700 chips, ganando con mucha facilidad el torneo.

BELL marcó entonces una nueva forma de desarrollo de máquinas para jugar al ajedrez. Los investigadores de la época creyeron que sólo iba a ser una cuestión de tiempo y de avance en hardware el lograr que una máquina logre la profundidad de búsqueda del nivel de un campeón mundial de ajedrez. Berliner entendiendo que no solo un buen cálculo bastaba para tener nivel de gran maestro, argumentó la importancia de incluir "conocimiento ajedrecístico" en las máquinas, ya sea mediante heurísticas especiales o bien bases de datos, lo cual se vio reflejado en su propio programa, HITECH.

Siguiendo el suceso de 1980 BELL logró los campeonatos de Los Angeles 1981y Dallas 1982. En 1983 luego de contabilizar sus resultados contra jugadores humanos, BELL logró el hito mundial de convertirse en la primera máquina con nivel de maestro, lo cual le valió un premio de U$5.000 por parte de la fundación Fredkin. Irónicamente Thompson y Condon recibieron el cheque durante el campeonato mundial de computadores de 1983, torneo que fue ganado por CRAYBLITZ.

CRAYBLITZ fue creado por el científico norteamericano Robert Hyatt[50, 51] e hizo su aparición en escena el año 1976, con el nombre de BLITZ, en el torneo de la ACM en Houston logrando el segundo lugar. En 1980 el nombre del programa fue cambiado a CRAYBLITZ para reflejar el poder de hardware que tenia incorporado. En 1981 CRAYBLITZ fue el segundo programa que logró ganar un torneo de estado, al vencer en el torneo de Mississipi.

Mientras la década del 70 fue marcada por el dominio de un solo equipo de programación, la siguiente década produjo una "encarnizada" lucha entre CRAYBLITZ y los tres "motores aje-

drecísticos" BELLE, HITECH y DEEP THOUGHT (originalmente llamado CHIPTEST). Estos obtuvieron todos los títulos importantes durante la década, seguidos de cerca por otros programas como CHESS CHALLENGER y MEPHISTO.

?'Cual fue el motivo de este crecimiento tan notorio en la fuerza de los programas? De acuerdo a las conclusiones sacadas por los propios científicos, cuatro fueron las razones principales:

- Primero, la disponibilidad de nuevo Hardware contribuyó a mejorar el poder de las máquinas. Mientras CRAYBLITZ comenzó su vida en un simple monoprecesador Cray-1, en el torneo mundial de 1989 en Edmonton ya corría bajo un multiprocesador Cray-XMP.
- Combinado con este crecimiento en hardware vino un notorio abaratamiento de costos de producción de circuitos integrados para aplicaciones específicas, con lo cual diseñar un circuito específico para jugar ajedrez era algo ya factible. Esto permitió que el ex-campeón mundial de ajedrez por correspondencia Hans Berliner, diseñara un multiprocesador llamado HITECH, con el cual logró grandes resultados contra rivales humanos. Trabajando junto a él en la Universidad de Carnegie Mellon un estudiante llamado Feng-Hsung Hsu creó un aún mas poderoso chip, el cual implementó en el programa CHIPTEST, el cual luego superó a HITECH.
- La tercera razón para el crecimiento "desmedido" de la actividad en los 80 fue simplemente la realidad que muchos más científicos aprendieron cómo escribir programas de ajedrez. La literatura abundante en papers, publicaciones y libros especializados en el tema fue una fuente de conocimiento disponible para gran cantidad de personas.
- Quizás la razón principal de la explosión ajedrecística fue la capacidad de adquirir máquinas de mayor poder para realizar pruebas y desarrollo de software. La facilidad de acceso a maquinaria mas especializada cooperó a que programas de ajedrez estuviesen a disposición de millones de personas, logrando masificar el conocimiento y desarrollo de este tópico.

En 1984 CRAYBLITZ confirmó su superioridad al vencer en el torneo de la ACM realizado en san Francisco, pero al año siguiente HITECH de Berliner se quedó con el torneo realizado en Denver.

En 1986 se organizó el Campeonato Mundial de Computadoras en Colonia, Alemania, produciendo gran expectación por la participación de CRAYBLITZ y HITECH. A pesar de tener un pésimo comienzo CRAYBLTZ logró empatar el primer puesto con HITECH, BEBE y SUNPHOENIX, ganando por desempate. Habiendo ganado este titulo en Colonia, CRAYBLITZ no participó en el campeonato de la ACM realizado en Dallas, el cual fue ganado por BELLE. En ese mismo torneo uno de los programas que logró el 50 % de puntos fue CHIPTEST.

Al siguiente año también en Dallas CHIPTEST dio la sorpresa al vencer a CRAYBLITZ y dejarlo en segundo lugar. Hsu mas tarde predijo que con la tecnología existente su programa podría lograr calcular 1 millón de posiciones por segundo, y que una profundidad de 14 jugadas sería pronto una realidad. Si bien CRAYBLIT era el campeón mundial en 1988 era claro que CHIPTEST era el programa mas fuerte del mundo. Ese año Hsu removió todo el código originario de HITECH en el cual basó su programa y lo renombró como DEEP THOUGHT. Bajo este nombre el programa de Hsu ganó reputación mundial, al ganar el 18vo torneo de la ACM en Orlando, Florida.

Luego de este suceso DEEP THOUGHT asombró al mundo del ajedrez al ganar el torneo de maestros de California compartiendo el primer lugar con el GM Tony Miles, y superando entre otros a los GM Larsen, Tal y Browne. Esta fue la primera ocasión en que una maquina vencía a un gran maestro en una partida, al ganar DEEPTOUGHT su partida contra Larsen (pág. 131).

Siguiendo su suceso en Long Beach, DEEP THOUGHT ganó el campeonato Mundial de Edmonton en 1989 con score perfecto. Su partida contra HITECH fue crucial, pues a pesar de que el programa de Berliner logró una gran posición, no fue capaz de realizar el remate, perdiendo en el final. El mismo año en el campeonato de la ACM DEEPTHOUGTH perdió en la ultima ronda contra el programa MEPHISTO, por lo cual debió jugar un desempate contra HITECH el cual ganó en forma convincente. A fines de ese mismo año David Levy organizó un nuevo match, pero esta vez el resultado fue dramático para los humanos, con un 4-0 a favor de DEEP THOUGHT. Ese mismo año los logros del programa lo llevan a jugar un match a dos partidas contra el campeón mundial Gary Kasparov, el cual lo derrota en ambas mostrando una diferencia abismante en el nivel de juego (pág. 132).

En 1993 Hsu, junto a Joe Hoane y Murray Campbell 4.2fueron contratados por la IBM, empresa que apadrinó el proyecto de crear una máquina que venciera el campeón mundial. La nueva versión de la creación de estos tres investigadores se denominó "DEEP BLUE". Este programa el cual utilizaba el mejor nivel de hardware de la época y cuyo desarrollo lo hacía mostrar una gran diferencia respecto de los otros programas. En 1996 se juega el primer match a 6 partidas entre esta nueva versión y Gary Kasparov, venciendo el campeón mundial por 4:2 con sendas victorias al final del match. El año que transcurrió entre este match y el match de revancha fue fundamental en la mejora de la circuitería de DEEP BLUE y la adaptación conocimiento ajedrecístico a su función de evaluación. El resultado fue la victoria del programa en 1997 por la cuenta de 3,5:2,5, con lo cual el fin de una carrera se había cumplido.

### 4.2.3. Programas Actuales

#### Programas de libre disposición

Programa Computacional Crafty

Este es quizás el más popular y conocido programa de ajedrez de código fuente abierto. Su creador es el Dr. Robert Hyatt, quien actualmente trabaja en la Universidad de Alabama en Birmingham.

El programa de ajedrez "Crafty" fue desarrollado en 1983 y actualmente es parte de muchas de las principales herramientas de análisis de partidas de ajedrez (ChessBase, Fritz, HIARCS, entre varios otros). La página del autor de este software es [78] , mientras que los códigos fuentes, historia y documentación de Crafty se encuentran en [77].

La particularidad de Crafty radica en la aplicación de prácticamente todas las herramientas de desarrollo de software para computadoras de ajedrez en su código fuente. El estudio de este código ayuda a comprender la aplicación de algoritmos en este tipo de aplicaciones, además de permitir, dada su cualidad de "open source", la posibilidad de probar ideas propias sobre el.

Figura 4.2: El equipo de Deep Blue en la sala de juego antes del inicio del match de 1997. Iz a Der. Jerry Brody, Feng-hsiung Hsu, Joe Hoane, Joel Benjamin y C.J.Tan. Sentado: Murray Campbell

Como dato interesante Robert Hyatt menciona su disponibilidad para consultas a través de ICC ( http://www.chessclub.com ), club de ajedrez virtual el cual frecuenta con el apodo de "hyatt". Su creación, Crafty, también esta disponible en este club virtual bajo el apodo de "Crafty". La información disponible en el sistema acerca del software es la siguiente:

```
1: Crafty v19.2 (4 cpus)\\
2: Utiliza bases de datos de finales para 3/4/5/6 piezas, 50+
gigas de tamaño. \\
3: Procesador: Intel SC450NX quad xeon 700mhz, 512mb, 130 gigas.\\
4: Horas en línea: 53929;  porcentaje de "vida" en línea: 79%
5:Crafty esta disponible gratuitamente en código fuente y
ejecutable. Quienes no posean una maquina Unix pueden obtenerlo
como usuario anónimo en ftp.cis.uab.edu/pub/hyatt.
```

Muchas referencias acerca de Hyatt y Crafty pueden encontrarse también en el grupo de news `rec.game.chess.comp`, lo cual hace de este proyecto una fuente muy valiosa de información practica y actualizada acerca de programación y aplicación de algoritmos para desarrollo de maquinas que juegan ajedrez.

Arasan. Freeware por Jon Dart. Arasan es un programa Windows disponible en dos versiones: 16 y 32 bits. El tamaño del archivo comprimido es cerca de 1 MB. Su código fuente esta disponible (C++). Hardware mínimo: procesador 386 y 4 MB RAM. Link: `http://www.arasanchess.org`

Bringer. Programa gratuito escrito por Gerrit Reubold. Plataforma Windows. Juega relativamente bien y posee una interfaz amigable. Link: http://www.reubold.onlinehome.de/ .

Chenard. Freeware para DOS, Windows y Linux/Unix realizado por Don Cross. No apto para competir contra programas fuertes. Requiere poco espacio en disco. Link: http://www.intersrv.com/~dcross/chenard.html

Comet. Freeware para DOS por U. Türke. Un fuerte programa que obtuvo una buena performance en reciente el Campeonato Mundial de Computadoras. Interfaz no muy amigable (requiere mucho ingreso de comandos vía teclado). Link: `http://members.aol.com/utuerke/comet/`

GnuChess. Programa Freeware disponible como parte del proyecto GNU de la FSF. El programa está escrito en lenguaje C sobre la base de contribuciones de distintos programadores del mundo y su actual versión es la 5.0. La sección de FAQ del proyecto GNU-chess posee rica información acerca de las contribuciones realizadas al programa. Respecto a su nivel de juego la única referencia concreta que existe es su rating de 2200 puntos jugando en el servidor de ajedrez FICS. Link : `http://www.gnu.org/software/chess`

Gromit. Freeware por Frank Schneider. Windows 3.*/95. Un programa relativamente fuerte. Gromit puede utilizar bases de datos de finales y tiene código abierto para la edición de su función de evaluación. Link : `http://home.t-online.de/home/hobblefrank/`

Rebel Decade

Freeware para DOS. Una versión más débil que su homólogo comercial, si bien no es del todo malo. Link : `http://www.rebel.nl/edsoft.htm (676 KB)`

Aparte de esta buena descripción y conglomerado de programas, se encuentran bastantes utilitarios, como programas para la realización de torneos, calculadores de Rating, programas para administración de partidas por correspondencia, lectores de partidas, protectores de pantalla, etc.

**Programas Comerciales**

Fritz. Este programa pertenece a la prestigiosa firma alemana de programas computacionales de ajedrez ChessBase. Su sitio web ubicado en [84] posee una gran cantidad de información comercial relacionada con este programa, así como publicaciones de eventos y otros productos relacionados con el deporte ciencia, tales como Bases de datos de partidas, catálogos de aperturas, utilitarios para educación, etc.

Este programa se encuentra en su versión comercial 8.0. Su fuerza radica en su nivel de juego, bases de datos incorporadas y módulos de análisis, lo cual lo hacen una buena herramienta de entrenamiento para jugadores de todo nivel. Comercialmente ha tenido muchísimo éxito luego de traspasarse la versión 4.0 de este software al idioma inglés. La próxima versión del software se denominará DeepFritz, cuya principal caracter'istica es la contar con soporte para procesamiento paralelo. Su valor comercial es de algo más de U$85.

Otros programas comerciales ofrecidos por la firma son HIARCS 8.0, DeepJunior 7.0, SHREDDER 6, ChessTiger 14.0 y NIMZO 8.0, además de Bases de datos de aperturas, finales y medio

juegos.

Uno de los principales valores del sitio, para nuestros propósitos, es la organización de competencias y match entre los programas ofrecidos tanto con otras aplicaciones como contra destacados grandes maestros. Existe una sección especial de cobertura para el reciente match entre Kramnik y Deep Fritz, empatado 4 a 4, y también para el próximo match entre Kasparov y Deep Junior, a jugarse en Enero del 2003. Ambos enfrentamientos se encuentran bajo la sección "eventos" del sitio.

Rebel. El software Rebel fue creado por programadores holandeses el año 1995, bajo el alero de la empresa Schreder BV. Gozó de bastante popularidad hace aproximadamente una década, pero la comercialización de ChessBase con su producto Fritz lo dejo bastante atrás en ventas por lo cual el proyecto perdió en fuerza de investigación y popularidad. El programa puede obtenerse desde el sitio `http://www.rebel.nl` el cual posee bastante información acerca del desarrollo del proyecto, datos históricos, eventos (Rebel vs Yusupov, Junio 1997 y Rebel vs Anand Julio 1988 como los más destacados) y una sección en donde se puede obtener una versión no comercial del software. El gran problema del sitio es su desactualización.

Chessgenius. Ofrecido en el sitio web `http://www.chessgenius.com`. Este sitio web posee bastantes aplicaciones comerciales, la mayoría de ellas escritas por Richard Lang, famoso por obtener varias victorias en campeonatos mundiales de computadoras de ajedrez. Existen aplicaciones para palms, pocketPC y computadoras personales. No existe posibilidad de revisión de código ni datos acerca de la programación de las aplicaciones, lo cual no la hace una página interesante para este trabajo.

## 4.3. Organizaciones y Publicaciones Periódicas

### Organización Internacional de Computadoras de Ajedrez

Un buen punto de partida en búsqueda de información es por supuesto la Organización Internacional de Computadoras de Ajedrez (International Chess Computer Association, ICCA) cuyo sitio en Internet está actualmente alojado en [72] bajo el nombre de "International Computers Games Association" (ICGA) el cual contiene información acerca de eventos de computadoras, un Journal, y algunos papers de libre disposición. El sitio está principalmente abocado a las competencias entre máquinas de ajedrez e intercambio de información técnica acerca de ellas conteniendo también interesante información histórica.

La organización fue fundada el ano 1977 (con el nombre de International Chess Computer Association) por un grupo de programadores de computadoras de ajedrez con el fin de organizar competencias entre computadoras así como facilitar el intercambio de información técnica entre sus miembros a través de un journal. Uno de los objetivos centrales de esta organización es el reflejar la contribución de los juegos de computadora a la Inteligencia Artificial así como a la experiencia humana en competencias.

En su sección editorial los directores de la organización comentan lo difícil que ha sido para ellos el periodo 1999-2002 dada las dificultades que han tenido en la obtención de apoyo económico

para la organización así como para sus eventos. La razón de esto es la precaria situación económica mundial así como la carencia de interés en el tema luego del match Kasparov-DeepBlue. La asociación ha sufrido también una baja considerable en su número de miembros inscritos. Estas palabras resultan ser bastante lamentables, tomando en cuenta que los temas tratados en el Journal así como el nivel de los colaboradores es realmente bueno.

A pesar de esto, la organización mantiene en este momento un acuerdo de auspicio para el Campeonato Mundial de Computadoras 2003 a realizarse en la ciudad de Graz (22 al 30 de Noviembre), y ya se está gestionando la realización del campeonato del año 2004. También se logro un acuerdo de 10 anos de apoyo financiero por parte de la Universidad de Maastrich, con el fin de apoyar los proyectos particulares de la organización. El actual presidente de ICGA es el conocido autor y Maestro Internacional de ajedrez David Levy.

### Asociación de Maquinaria Computacional, ACM

En el año 1970 la Asociación de Maquinaria Computacional (Association of Computing Machinery, ACM) organizó el primer torneo de ajedrez entre computadoras en el mundo. El torneo fue realizado en Nueva York, en las dependencias del New York Hilton, como parte de la conferencia anual de la ACM. El ganador de ese torneo fue un programa llamado CHESS 3.0, desarrollado en la Universidad de Northwestern, Evanston, Illinois.

El torneo de la ACM de 1970 fue un suceso total, lo cual hizo que se decidiera repetir la experiencia para el año siguiente, siendo desde entonces una competencia regular y que curiosamente atrae a gran cantidad de observadores a las conferencias de la ACM.

La ICGA posee una sección completa dedicada a los campeonatos mundiales de ajedrez por computadoras [72] la cual contiene rica información de resultados e información técnica de las maquinas participantes.

## 4.4. Publicaciones de los Últimos Años

Actualmente, la principal fuente de material relacionado con computadoras que juegan ajedrez es el journal de la ICCA (International Chess Computer Asociation). Sus publicaciones datan de 1977, inicialmente como el "ICCA Newsletter" y luego, desde 1983, como el "ICCA journal". El profesor Jaap van der Herik y el maestro internacional David Levy han sido sus editores durante las dos ultimas décadas.

"Advances in Computer Chess" ha publicado 5 volúmenes en 1977 [22], 1980 [23], 1982 [24], 1988 [25] y 1989 [26]. Los volúmenes 1 y 2 fueron publicados por la universidad de Edimburgo, los volúmenes 3 y 4 por "Pergamon Press" y el último volumen por "Elsiever Science Publishers B.V" (Holanda).

Otras publicaciones donde se hace referencia al tema son la serie "Machine Intelligence", publicada en 1967 por la Universidad de Oxford, "Artificial Intelligence", publicada mensualmente

por Elsevier y la "IEEE Transactions on Pattern Analysis and Machine Intelligence" publicado mensualmente por la IEEE.

Como trabajos personales no podemos dejar de nombrar las distintas publicaciones realizadas por el Maestro Internacional de Ajedrez británico David Levy. Entre algunas destacan "Chess and Computers" publicada en 1976 [9]; "More Chess and Computers, The Microcomputer Revolution, the Challenge Match : BELLE" presentada en 1980 [10] la cual trata acerca del proyecto BELLE realizado en la Universidad de Carnege Melon, USA; "All about Chess and Computers" 1982 [11]; "Chess computer Handbook" 1984 [12].;"Computer Chess Compendium" 1989; "How Computers play Chess" 1991 [14] y "All About Chess and Computers : Containing the Complete Works, Chess and Computers" 1993 [15]. Las 3 últimas publicaciones mencionadas fueron realizadas junto con Monty Newborn. Otros textos de investigadores que son citables son "Chess Skill in Man and Machine" de Peter W. Frey, 1977 [7], el cual recopila bastante información acerca de métodos de construcción de funciones de evaluación y parámetros a incluir en ellas; "The Machine Plays Chess" por Alex Bell, 1978 trata de explicar el como las máquinas logran jugar al ajedrez con técnicas y métodos de juegos.

Existen por supuesto algunos trabajos breves de otros destacados jugadores (Max Euwe "Computers and Chess", 1970, David Bronstein, 1977) pero distan de tener un análisis más profundo en lo técnico, dedicándose demasiado al tema de "cómo juegan los programas" mas que el "cómo mejorar el juego" de los programas.

Para los fines de esta memoria se encontraron tremendamente interesante los textos"A test for comparision of human and computer performance in chess" de Bratko y Kopec, y "Computer Chess strength" de K.Thompson [70] , publicado en "Advances in computer chess" volumen 3 en 1982. Ambas publicaciones hacen referencia a la comparación de nivel de fuerzas en computadoras y cómo presentar una forma de medición de fuerza de ellas, tema tremendamente relacionado con esta memoria.

### Kasparov - DeepBlue

Bastante hay de literatura acerca del match entre Kasparov y Deep Blue. Mucha de ella focalizada en el aspecto ajedrecístico del match dejando de lado el aspecto tecnológico involucrado. El sitio web oficial de este match, `http://chess.ibm.com` está totalmente desactualizado, y su información se enfoca en aspectos principalmente publicitarios.

Un sitio interesante que recopiló bastantes publicaciones relacionadas con el match (con reportes y comentarios de revistas especializadas y diarios de circulación masiva) es la página `http://www.rci.rutgers.edu/~cfs/472_html/Intro/ChessContents.html` , que sirve bastante como referencia histórica acerca de este evento.

Una primera buena publicación acerca de este tópico fue realizada por el investigador Monty Newborn en 1998 [17], el cual comenta aspectos técnicos del match pero enfocado desde un punto de vista computacional. Se agrega un compendio histórico de ajedrez y computadoras.

La última publicación de material relacionada con este match fue hecha por el norteamericano

Feng Hsiung-Hsu en su texto “Behind Deep Blue” [8] publicado el año pasado. El texto está enfocado a presentar la historia detrás del match desde un punto de vista no muy técnico. A pesar de esto, es una referencia importante para conocer la fuerza real del “monstruo” de IBM, la cual no podemos dejar de citar. Respecto del autor, existe una entrevista bastante interesante que se realizó en el “Internet Chess Club” (ICC) el 13 de octubre de 2002, la cual se encuentra en el link `http://www.chessclub.com/events/crazybird1.html`

### 4.4.1. Publicaciones en Español

En español, uno de los textos más referenciados es “Ajedrez y Computadoras”, de Pachmann y Kuhnmund [18], el cual es bastante bueno como referencia histórica pero adolece de una mayor profundidad en el análisis computacional y está ya bastante obsoleto para el año en que fue realizado. Presenta muchos ejemplos de partidas de los principales jugadores de ajedrez frente a programas computacionales, siendo interesantes algunos análisis técnicos respecto a la evaluación de la partida por parte de los programas en ciertas situaciones. A pesar de esto último, sus características como libro apuntan más a un público ajedrecista que computacional.

## 4.5. Algunas conclusiones acerca de la bibliografía

Como se mencionaba inicialmente, a pesar de existir muchas referencias literarias (sitios web, papers y publicaciones) no existe un estudio focalizado en la medición de fuerza de las máquinas a partir de ciertos parámetros específicos. Muchos mencionan brevemente el ELO como un sistema de medición válido, pero tal sistema si bien es generalmente aceptado ya ha sido cuestionado en variadas ocasiones por los propios ajedrecistas profesionales y los mismos programadores quienes no ven en este sistema de rating un parámetro que refleje el real nivel de fuerza de sus creaciones.

Se capta que el principal interés en la redacción de artículos y libros de este tema se centró en las técnicas de búsqueda y evaluación relacionadas directamente con la inteligencia artificial y teoría de juegos ( búsqueda en profundidad, algoritmo min-max, técnicas de poda alfa-beta, etc ) las cuales tienen en el ajedrez una fuente de testeo y evaluación directa. La “carrera” generada entre investigadores por lograr una máquina que superase al campeón mundial de ajedrez fue un aliciente para la masificación del tema e inversión de recursos de tiempo y dinero en él.

Lamentablemente el match Kasparov - DeepBlue marca de todas maneras un hito en la literatura y publicación de ajedrez y computadoras. El alcance del “objetivo” hizo disminuir el interés masivo en este tópico quedando en cierta forma pasado de moda. A pesar de enfocarse en temas de desarrollo, muy de interés para este trabajo serán los estudios iniciales acerca del tema (Shannon [68] y Turing [20] ), pues marcaron la pauta para lo que fueron los desarrollos subsiguientes. Las colaboraciones de jugadores de ajedrez son bastante escasas, muchos fueron mas bien tomados como asesores o bien colaboradores en el tema técnico, pero pocos de buen nivel se dedicaron a investigar el problema en profundidad.

# Capítulo 5

# Análisis de Avances Históricos en Ajedrez y Computadoras

En los últimos 50 años de desarrollo de máquinas que juegan al ajedrez los avances lograron ser realmente notables. Los nuevos diseños y capacidades de Hardware sumado a las nuevas técnicas desarrolladas en el software han hecho que la comparación entre una máquina actual y una de hace 3 décadas sea prácticamente algo totalmente fuera de proporción.

Cuando en 1949 Claude Shannon planteó en su famoso paper [68] la forma de cómo desarrollar una máquina que jugase al ajedrez tomó en cuenta 3 principales aspectos : Representación del juego, cálculo de variantes y evaluación de posiciones. Curiosamente, este paper fue la base del desarrollo a futuro en este tema, presentándose interesantes avances y nuevos problemas en el desarrollo de cada una de estos componentes del autómata ajedrecístico.

## 5.1. Representación del Juego - Tablero, Piezas y Movimientos

El primer problema al que se enfrentaron los creadores de máquinas que jugasen al ajedrez fue el cómo representar de una manera lo más sencilla y económica posible el tablero y las piezas. En un inicio la capacidad de hardware era demasiado limitada por lo que la eficiencia en la representación del juego era un factor importante a considerar en el desarrollo. Como si fuera poco, en una posición promedio existen cerca de 30 o más movimientos legales, algunos buenos y otros simplemente perdedores. Para humanos entrenados es fácil caracterizar un numero limitado y pequeño de alternativas "buenas" descartando rápidamente el resto, pero para un computador esto está lejos de algo directo de realizar.

4, 2, 3, 5, 6, 3, 2, 4,
1, 1, 1, 1, 1, 1, 1, 1,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
-1,-1,-1,-1,-1,-1,-1,-1,
-4,-2,-3,-5,-6,-3,-2,-4,
+1

Cuadro 5.1: Esquema de la posición inicial de las piezas según la representación de Shannon

### 5.1.1. La Propuesta de Shannon

Shannon propuso en [68] una de las primeras ideas de representación del juego mediante números y operaciones sobre ellos.

Según su propuesta, una casilla puede ser representada por un número que puede tomar 13 valores distintos: puede estar vacía (0) o bien ocupada por una de las seis piezas blancas (P=1, C=2, A=3, T=4, D=5, R=6) o bien por una de las seis piezas negras (P=-1, C=-2, A=-3, T=-4, D=-5, R=-6), vale decir, el estado de una casilla queda especificado por un entero entre -6 y +6. Las 64 casillas del tablero pueden ser numeradas de acuerdo a un sistema de coordenadas como el representado en la figura 5.1.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | 71 | 72 | 73 | 74 | 75 | 76 | 77 |
| 60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 |
| 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 |
| 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
| 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 |
| 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 |
| 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 |

**Codigo para Piezas :**

| | P | C | A | T | D | R |
|---|---|---|---|---|---|---|
| Blanks | 1 | 2 | 3 | 4 | 5 | 6 |
| Negras | -1 | -2 | -3 | -4 | -5 | -6 |

**Codigo para Movimientos:**

(Casilla origen , Casilla destino , Nueva pieza (caso de promocion ))

ejemplo :
P4R => (14,34,0)
P8AD = D => (63,73,5)

Figura 5.1: Representación del Juego en una máquina según Shannon

Un total de 256 bits (dígitos binarios) serían suficientes para utilizar esta representación. Un número extra, al cual Shannon denominó $\lambda$ tendría los valores -1 ó +1 de acuerdo al turno de jugar de las Negras o Blancas respectivamente. Algunos parámetros extras deben ser agregados para efectos de movimientos como el enroque y capturas al paso.

Bajo esta representación la posición inicial de las piezas queda dada por la tabla 5.1

Notar que el +1 al final señala el turno de movimiento de las blancas.

Generación de Movimientos

Un movimiento (aparte del enroque y la coronación) puede especificarse mediante los valores de las casillas de origen y destino de la pieza que se mueve. Una casilla es un numero entre 64 posibilidades, para cuya representación 6 dígitos binarios son suficientes lo cual nos da un total de 12 para el movimiento.Así por ejemplo la representación del movimiento inicial *1.e4* queda como 14;34. Para representar coronaciones de peón tres dígitos binarios son necesarios para especificar la pieza que corona. El enroque puede ser descrito mediante el movimiento del Rey (la única forma en que el rey puede moverse 2 casillas). Por lo tanto, una movida es representada por el vector $(a,b,c)$donde $a$ y $b$ son las casillas y $c$ la pieza que llega al tablero en caso de coronación.

Con esto la representación del juego es entregada a 8 subrutinas $T_0, T_1, ..., T_7$ cuyas funciones básicas son :
$T_0$ : Realiza el movimiento *(a,b,c)* en la posición P obteniendo una posición resultante.
$T_1$ : Realiza una lista de todos los movimientos posibles de peón en la posición P.
$T_2, .., T_6$ : Similar a $T_1$ pero para las otras piezas : Caballo, Alfil, Torre, Dama y Rey.
$T_7$ : Realiza una lista de todos los movimientos posibles en una posición.

Con esto la realización de un movimiento *(a,b,c)* sería estructurado de la siguiente manera :

1. La casilla correspondiente al número $a$ es ubicada en la posición de memoria.
2. El número presente en la casilla $(x)$ es extraído y reemplazado por 0 (vacío).
   - *a*) Si $x = 1$ y la primera coordenada de $a$ es 6 (peón blanco coronando) o si $x = -1$ y la primera coordenada de $a$ es 1 (peón negro coronando) el número $c$ es ubicado en la casilla $b$ (reemplazando lo que allí había.
   - *b*) Si $x = 6$ y $a - b = 2$ (enroque corto del blanco) un 0 es ubicado en la casilla 04 y 07 y un 6 y 4 en las casillas 06 y 05 respectivamente. Similarmente para los casos de $x = 6$, $b - a = 2$ (enroque largo del blanco) y $x = -6$, $a - b = \pm 2$ (enroque negro).
   - *c*) En cualquier otro caso, $x$ es ubicado en $b$.
3. El signo de $\lambda$ es cambiado.

Para cada tipo de pieza existe una subrutina encargada de generar sus movimientos. Un ejemplo típico es el programa para los movimientos del Alfil $T_3$. Sean $(x, y)$ las coordenadas de la casilla ocupada por el Alfil.

1. Construir $(x + 1, y + 1)$ y leer el contenido $u$ de esta casilla en la posición P.

2. Si $u = 0$ (casilla vacía) listar el movimiento $(x, y), (x+1, y+1)$ y seguir con $(x+2, y+2)$ en vex de $(x + 1, y + 1)$.
   Si $\lambda u$ es positivo (pieza propia en casilla)continuar con 3.
   Si $\lambda u$ es negativo (pieza del oponente en la casilla) listar el movimiento y continuar con 3. Si la casilla no existe continuar con 3

3. Construir $(x + 1, y - 1)$ y realizar el mismo proceso.

4. Similarmente con $(x - 1, y + 1)$.

5. Similarmente con $(x - 1, y - 1)$.

Mediante este programa es posible construir una lista de movimientos posibles de un Alfil dada una posición P. Programas similares listarían los movimientos de las otras piezas. Hay considerables opciones de simplificar estos programas. Por ejemplo, el programa de movimientos de Dama $T_5$ puede ser una combinación de los programas de Alfil y Torre, $T_3$ y $T_4$ respectivamente.

Utilizando los programas generadores de movimientos $T_1..T_6$ y un programa controlador $T_7$ la máquina puede construir una lista de *todas* los movimientos posibles en una posición P. El programa $T_7$ se estructuraría de la siguiente forma:

1. Iniciar en la casilla $(0, 0)$ y extraer su contenido $x$.

2. Si $\lambda.x$ es positivo iniciar el programa correspondiente $T_x$ y luego retornar a 1 sumando 1 al numero de la casilla.

3. Testear la legalidad de los movimientos generados y descartar aquellos que sean ilegales. Esto se logra realizando cada uno de los movimientos en la posición P (mediante $T_0$) y examinando si el rey queda en jaque.

Con los programas $T_0...T_7$ la máquina puede jugar ajedrez legal realizando una selección aleatoria de alguno de los movimientos obtenidos. El nivel de juego de este tipo de estrategia sería muy deficiente. Shannon mismo jugó un par de juegos contra una máquina con esta estrategia siendo capaz de vencer por jaque mate en 4 o 5 jugadas.

### 5.1.2. Representaciones en programas contemporáneos

Ya en la década del 70 la utilización de memoria era aún un gran problema para los programadores de la época. La idea planteada por Shannon si bien era intuitivamente sencilla no combinaba bien con las capacidades de hardware de las máquinas de entonces.

Como un intento de mejorar esta técnica algunos programas (por ejemplo SARGON [19]) extendieron el arreglo de 64 bytes con 2 filas y columnas las cuales contenían valores centinelas que indicaban casillas ilegales. Este truco aceleraba la generación de movimientos.

Por ejemplo un Alfil generará movimientos avanzando una casilla en diagonal hasta llegar a una casilla ilegal, entonces se detendrá. Con esto se evitaban las complejas manipulaciones de memoria para asegurar que una pieza no ocuparía un area de memoria no asociada al tablero. La segunda fila de casillas ilegales era requerida para los movimientos de caballo. Por ejemplo, un caballo en una esquina puede saltar dos columnas luego del limite del tablero, por lo cual una fila de casillas ilegales se hacía insuficiente.

El programa MYCHESS trató de mejorar este proceso representando el tablero en sólo 32 bytes, cada uno asociado a una pieza (i.e. el rey blanco, el peon "g" del blanco, etc.) y contenía el numero de casilla en donde cada pieza se ubicaba, o bien un valor nulo si la pieza había sido capturada. esta técnica tenía un serio problema : era imposible coronar un peón en una pieza que aún no había sido capturada. Versiones posteriores de este programa arreglaron este problema.

### 5.1.3. El avance de los BitBoards

A partir del surgimiento de computadoras de palabras de 64 bits, formas más elaboradas de representar el tablero fueron producidas. Aparentemente propuestas en la Unión Soviética cerca de los años 60 un bit-board (o mapa de bits) es una palabra de 64 bits la cual contiene información relativa a algún aspecto particular de la posición actual. Por ejemplo un bitboard puede contener"el conjunto de casillas ocupadas por peones negros" o "el conjunto de casillas a las cuales una dama desde e3 puede mover" o bien "el conjunto de piezas blancas actualmente atacadas por piezas negras".

Los bitboards son muy versátiles y permiten un procesamiento muy rápido al basarse en operaciones lógicas de 1 ciclo en la computadora.

Cada mapa de bits representa un atributo de la posición, y la intersección de dos mapas de bits define una combinación de ambos atributos. Dado que las computadoras pueden intersectar de manera lógica dos palabras de datos muy rápidamente, esta es una forma muy eficiente de almacenar y utilizar información. La eficiencia se incrementa así como los tamaños de palabras de la computadora se aproxima al tamaño del mapa de bits.

Algunos de los bitmaps que la mayoría de las computadoras utiliza son :

- 64 mapas los cuales representan las casillas atacadas por cualquier pieza (si hay alguna) que ocupa una casilla en particular.
- 64 mapas que representan, inversamente al anterior, las casillas desde las cuales hay piezas atacando una casilla en particular.
- 2 mapas que representan las casillas atacadas por cada bando (blancas y negras).
- 12 mapas cada uno de los cuales representa las casillas ocupadas por cada tipo de pieza de cada bando (por ejemplo, Caballos blancos).

Muchos atributos de la posición demasiado complicados de representar por bitmaps son representados en form de arreglos. Algunos de estos arreglos representan las pieza sen el tablero combinando la información de varios bitmaps. La información es guardada en más de una forma para su conveniente manipulación, utilizando arreglos especiales para traducir de una forma a otra.

Un arreglo en particular especial es el que mantiene la lista de movimientos. Para definir un movimiento el programa debe saber algo mas que la casilla de origen y la de destino de la jugada, por lo que este arreglo contiene información como que captura involucra, que tipo de pieza es capturada, si la movida afecta la posibilidad de enrocar, si involucra un jaque, jaque mate, ahogado, si un peón corona y como la movida ha sido buscada.

La forma básica de como generar movimientos con mapas de bits se describe a continuación:

1. Utilizar el mapa de bits para todas las piezas por color. Si encontramos un 1 una pieza de ese color está ubicada en la casilla correspondiente y sus movimientos deben ser generados. Si no, ir al siguiente bit.
2. Determinar si la pieza en cuestión es un peón. Esto se realiza mediante la operación lógica AND entre el bitmap de la posición actual y el de la ubicación de los peones.
3. Determinar las casillas a las cuales la pieza en cuestión puede mover legalmente. Si es un peón el programa inicia la operación con el bitmap de destinos de peones y casillas de captura, sino inicia con el bitmap de casillas atacadas desde la casilla considerada. El bitmap para ubicaciones de piezas de igual color es complementado o invertido con tal de entregar un mapa de las casillas no ocupadas por piezas del mismo color. La intersección de este bitmap con el de los destinos de peones o de ataque entrega el bitmap de casillas a las cuales la pieza puede mover. Estas manipulaciones de bitmaps son muy rápidas para las computadoras.

Debido a la complicación que significaba el considerar si una movida legal deja al rey propio en jaque muchos programas ignoraron este hecho en esta etapa con tal de verificarlo durante la búsqueda en el árbol de variantes.

### 5.1.4. Generación de Movimientos

Históricamente han existido 3 estrategias principales para la generación de movimientos:

- Generación Selectiva : Examinar el tablero y obtener una serie de movimientos “buenos” descartando el resto.
- Generación Incremental : Generar algunos movimientos, esperando que alguno de ellos será lo suficientemente bueno o malo tal que la búsqueda a lo largo de esa línea de juego pueda ser terminada antes de generar las otras.

- Generación Completa : Generar todos los posibles movimientos, esperando que la tabla de transposición contendrá información suficientemente relevante para hacer la búsqueda lo más eficiente posible.

La generación selectiva de movimientos (y su técnica de búsqueda asociada, denominada búsqueda selectiva) han prácticamente desaparecido desde mediados de 1970. Las otras dos técnicas representan en realidad dos caras de la misma moneda, cambiando esfuerzo en la generación de movimientos por sobre la búsqueda.

**Los dias iniciales: Búsqueda Selectiva**

En su paper de 1949 Shannon describió dos formas de construir un algoritmo de un programa que jugase al ajedrez:

- Buscar todos los movimientos posibles tanto propios como del adversario hasta una profundidad límite, la que llamó "estrategia A".
- Examinar solamente los "mejores" movimientos tanto propios como del adversario obtenidos a partir de un análisis detallado de la posición. Este método fue denominada como "estrategia B".

A primera vista la segunda alternativa aparentaba tener un mayor éxito. Después de todo es la forma en la que los humanos juegan y pareciese lógico asumir que el buscar pocas alternativas en cada movimiento será más rápido que buscar todos los movimientos posibles. Desafortunadamente, los resultados prácticos fueron en contra de lo que decía la teoría: los programas que utilizaron búsqueda selectiva no lograron jugar a un nivel avanzado. Como mejor resultado lograban un nivel de jugador de nivel medio de club, cometiendo a menudo errores casi infantiles en los peores momentos. EL vencer a un Gran Maestro de Ajedrez era algo definitivamente lejano a sus capacidades.

El tiempo y análisis utilizado en el proceso de selección de los mejores movimientos no compensaba la reducción del árbol de variantes de análisis, por lo que muchos programas implementados con la estrategia B de Shannon fueron transformados en programas de estrategia A. Uno de los casos más emblemáticos es el del programa CHESS 4.5, cuyos primeros prototipos utilizaban la estrategia B. De acuerdo a sus autores, los algoritmos de selección utilizados eran demasiado complejos y no cumplían a cabalidad con seleccionar el mejor movimiento en una posición dada.

El problema radicaba también en que el "generador de mejores movimientos" para ser bueno debe en realidad ser perfecto. Supongamos que un programa es equipado con una función que busca las 5 mejores movidas en una posición y que la objetivamente mejor movida entre esas 5 se encuentra el 95 % de las ocasiones (esto es un supuesto muy optimista). Esto significa que la probabilidad de que el generador contenga la mejor alternativa todas las veces durante una partida de 40 jugadas es menor que un 13 %.

A mediados de 1970 el legendario equipo de la universidad de Northwestern y creador de "CHESS 4.5" (Slate & Atkin) decidieron olvidar el problema del generador de mejores movimientos especificando que el tiempo ganado en evitar costosos análisis durante la generación de movimientos era suficiente para cubrir el costo de tener una búsqueda completa (examinar todos los movimientos). Este descubrimiento prácticamente enterró la búsqueda selectiva.

Un ejemplo extremo de de algoritmo de búsqueda selectiva fue desarrollado en la Unión Soviética en la década de los 70 bajo la tutela del ex campeón mundial Mikhail Botvinnick.

Botvinnick era un convencido de que la única forma de lograr programar una máquina que jugase al ajedrez a nivel del mejor jugador del mundo era imitando la forma de jugar de los Grandes Maestros, es decir, examinar una pequeña cantidad de movimientos pero con gran profundidad y detalle. Su programa se aproximó a identificar e implementar el desarrollo de estrategias de alto nivel y patrones que jugadores de nivel mundial utilizarían durante una partida.

### 5.1.5. Generando todos los movimientos

Una vez que la generación selectiva de movimientos fue descartada, el interés se volcó en la estrategia de generar todos los movimientos posibles, denominada búsqueda completa.

La vía mas expedita de implementar esto era encontrar todos los movimientos legales en una posición, ordenarlos de acuerdo a algún criterio con tal de mejorar la búsqueda y luego buscar de un movimiento a la vez hasta que todos hayan sido examinados o bien ocurra algún evento que termine la búsqueda.

Los primeros programas, por ejemplo Sargon, realizaban esto observando de a una casilla a la vez, buscando piezas del bando que jugaba y computando posibles movidas de destino. Existiendo escasa memoria el costo de tiempo de CPU para computar los movimientos resultaba ser un mal necesario.

En estos dias las nuevas técnicas de uso de estructuras de datos como Tablas de transposición han mejorado el tiempo de ejecución de los programas. Si incluso uno de los movimientos ha sido ya buscado con anterioridad, y su evaluación (obtenida de la tabla) es tal que genera un corte en la búsqueda entonces no habrá necesidad de buscar más movimientos. Obviamente, mientras más grande la tabla de transposiciones y mientras mayor sea la probabilidad de una transposición mayor será la ganancia promedio en tiempo.

### 5.1.6. De vuelta a los inicios:Una movida a la vez

Programas de ajedrez más sofisticados creados en la década del 70, como "CHESS 4.5" han adoptado la estrategia opuesta: generar algunos movimientos, realizar una búsqueda

sobre ellos y si un corte es alcanzado entonces no habrá necesidad de buscar en el resto de los movimientos.

Una combinación de diferentes factores volvieron a hacer esta técnica muy popular:

- La búsqueda no requiere demasiada memoria. Los programas de la década de 1970 combinaban este proceso con pequeñas tablas de transposición y algunas otras técnicas lo que limitaba las deficiencias propias de buscar sobre todas las posibilidades.
- La generación de movimientos en particularmente complicada en el juego del ajedrez a diferencia de otros juegos, con enroques, capturas al paso y diferentes reglas de movimiento para cada pieza.
- A menudo una refutación es una captura (involucra ganancia de material). Dado que usualmente hay pocas capturas en una posición y dado que generar capturas en forma separada es relativamente fácil (ver punto de los bitboards), computar capturas es a menudo suficiente para lograr un corte en el análisis de variantes.
- Las capturas son uno de los pocos movimientos analizados durante la búsqueda en posiciones "estables", por lo cual generarlas resulta doblemente útil.

Muchos programas generan primero las capturas, ordenándolas de acuerdo a aquellas que logran una mayor ganancia de material y buscando un corte en las variantes. Algunos métodos que han sido desarrollados con tal de acelerar la generación de movimientos de captura involucrando bitboards.

- CHESS 4.5 mantenía dos conjuntos de 64 bitboards, con uno por cada casilla del tablero. Uno contenía las casillas atacadas por cada pieza que estaba ubicada en un casilla dada (de existir alguna pieza en esa casilla). La otra era la transpuesta, conteniendo todas las casillas ocupadas por piezas que atacan esa casilla. Luego, si el programa busca movimientos que logran la captura de la Dama negra entonces busca su casilla de ubicación en el bitboard de ubicación de las piezas, usa los otros bitboards para encontrar las casillas desde donde es atacada la posición de la dama y genera solo movimientos para las piezas ubicadas en esas casillas.
- El mantenimiento de estos "bitboards de ataque" luego de cada movimiento requiere bastantes operaciones en cada estructura mantenida en memoria, pero una herramienta denominada "bitboards rotados" (rotated bitboards) puede acelerar el trabajo en forma significante.

### 5.1.7. Generando Movimientos a nivel de Hardware

El avance a nivel de software que significó la utilización de bitboards tuvo su complemento con la creación de hardware dedicado para esta función específica.

| Programa | Año | Posiciones/segundo |
|---|---|---|
| BELLE | | |
| 1er prototipo | 1976 | 200 |
| 2do prototipo | 1978 | 5.000 |
| 3er prototipo | 1980 | 120.000 |
| CHEOPS | 1977 | 150.000 |
| BEBE | 1980 | 40.000 |
| HITECH | 1985 | 200.000 |
| DEEP THOUGHT | 1986 | 700.000 |
| DEEPBLUE | 1996 | 2.000.000 |

Cuadro 5.2: Evolución de la cantidad de posiciones analizadas por segundo en programas de ajedrez

El primer hardware de propósito especial para ajedrez fue desarrollado en 1976. Dado que la mayor parte del tiempo consumido en los programas de ajedrez era la generación de movimientos era inevitable el hecho de que este proceso pasaría a nivel de hardware. La idea era dar a un generador de movimientos a nivel de hardware una posición y obtener una lista de movimientos (en la misma forma en que un hardware de multiplicación recibe dos números y entrega un producto). Luego de la generación de movimientos el programa podía concentrarse en la búsqueda de movimientos y su evaluación.

En 1976 Joe Condon y Ken Thompson desarrollaron un prototipo de hardware generador de movimientos para la computadora BELLE.Los resultados a nivel de velocidad de evaluación de posiciones por segundo fueron impresionantemente mejores lo cual hizo que a partir de entonces las mejores computadoras de ajedrez incorporaran a nivel de hardware este proceso.

El proceso se mejoró incorporando ordenamiento de los movimientos (lo cual tiene incidencia directa en la búsqueda en profundidad). este ordenamiento era factible de realizarse por consideraciones como capturas, jaques, movidas decisivas, etc.

Ya en 1980 el tercer prototipo de BELLE contenía un generador de movimientos provisto de 64 circuitos transmisores y 64 circuitos receptores correspondientes a las 64 casillas del tablero siendo capaz de generar un ordenamiento basado en la captura de la pieza rival de mayor valor.

En 1985 un grupo de estudiantes de la universidad de Carniege Mellon bajo la supervisión de Hans Berliner desarrollaron un generador de movimientos a nivel de hardware basado en 64 chips con la idea de mejorar aun más la velocidad de evaluación de posiciones de la máquina. Su resultado llegó a la sorprendente cifra de 200.000 posiciones por segundo.

Al año siguiente otro grupo de la misma universidad siguiendo las ideas de BELLE y de HITECH desarrollaron el más fuerte programa de esa década. Inicialmente llamado CHIPTEST en 1988 tomó el conocido nombre de DEEP THOUGHT. Funcionó en una máquina SUN 3 con un generador de movimientos a nivel de hardware basado en tecnología VLSI (Very Large Scale Integration).

El programa alcanzó la espectacular cifra de 700.000 evaluaciones por segundo. De acuerdo a Feng Hsu, uno de sus creadores, la ventaja inicial de este programa estuvo en la optimización del diseño a nivel de hardware en la máquina, minimizando la cantidad de chips y transistores, lo cual aumentó considerablemente la velocidad de procesamiento del generador de movimientos. El generador de movimientos de Hsu se basó en un arreglo combinatorial de 8x8 - un elemento por cada casilla del tablero de ajedrez. Operaciones lógicas entre estos elementos describen los movimientos legales para cada pieza, permitiendo al arreglo generar el conjunto entero de movimientos posibles a partir de una posición dada.

Los avances finales en este tema vinieron ligados a la utilización de procesamiento en paralelo. Varios procesadores eran coordinados por un procesador central donde cada uno generaba movimientos correspondientes a una línea de análisis en particular. Las versiones posteriores de DEEP THOUGHT, DEEP THOUGHT II y DeepBlue utilizaron esta técnica con mejoras notorias en la capacidad de análisis de posiciones en las máquinas, logrando cifras que superaban el millón de posiciones por segundo.

### 5.1.8. Algunas conclusiones al respecto

El problema de generar movimientos resultó ser una de las tareas en donde mayor tiempo consumían los programas de ajedrez.

La primera idea propuesta por Shannon resultó ser la base que muchos programas utilizaron durante casi 20 años, en donde no hubo avances significativos a nivel de ideas de cómo mejorar el proceso de generar movidas en forma “inteligente”.

A principio de los 70 la introducción de los mapas de bits puede considerarse como el gran avance a nivel de software para la generación de movimientos. Estos combinaron la eficiencia de las operaciones lógicas de 1 solo ciclo de las computadoras con la mayor capacidad de almacenamiento en memoria que ya se disponía en esa época.

El nuevo paso para mejorar aún mas este proceso fue crear hardware dedicado para esta tarea. Los primeros intentos se realizaron en 1976 con bastante éxito, puesto que la tarea se realizaba con una mejora de rapidez asombrosa superándose en ordenes de magnitud la capacidad de análisis de posiciones por segundo en las máquinas.

Desde entonces los avances en el tema han estado ligados a mejorar la circuitería propia del hardware generador de movimientos. En este sentido la “reducción” en transistores y chips lograda por el equipo de CHIPTEST significó el duplicar la capacidad de análisis de posiciones por segundo. Las nuevas tecnologías de chips apoyadas por simulaciones optimizadas lograron llegar a lo que en la actualidad es la mayor velocidad teórica que se ha logrado obtener en una computadora que juega ajedrez, 2.000.000 de posiciones por segundo en DeepBlue durante su match con Gary Kasparov.

## 5.2. Técnicas de Búsqueda

El progreso de los programas de ajedrez desde un nivel de principiantes a grandes maestros en un periodo de 30 años se ha debido principalmente al desarrollo y refinamiento de las técnicas de búsqueda adaptadas en particular a las capacidades de los computadores, sumado a los rápidos avances en hardware.

Para una computadora un problema lejos de ser trivial es establecer cuales de todos los movimientos que dispone son "buenos" y "malos". La mejor forma de discriminar entre estos es mirar las consecuencias que cada uno de ellos involucra, es decir, la serie de movimientos futuros, digamos que 4 por lado y mirar los resultados. Para asegurar también el realizar la menor cantidad de errores posibles asumimos que nuestro rival siempre realiza el mejor movimiento.

Este es el principio básico detrás del algoritmo de **Minimax** el cual es la base de todas las técnicas de búsqueda de los programas de ajedrez.

### 5.2.1. El Árbol de Ajedrez y el Algoritmo de MiniMax

Cuando el juego de ajedrez se inicia las blancas eligen uno entre 20 movimientos posibles. Siguiendo el movimiento de las blancas, las negras tienen 20 opciones de respuesta cualquiera haya sido la elección de las blancas.

De acuerdo a esto, 400 posiciones distintas pueden surgir solo de la primera jugada de una partida de ajedrez. Luego de 2 jugadas el número de posiciones posibles crece sobre las 20.000, llegando a una cifra astronómica luego de varias jugadas más. El *árbol de ajedrez* posee más posiciones que la cantidad de átomos presentes en la via láctea. De acuerdo al reglamento del juego una partida es tablas (empate) si transcurren 50 jugadas sin que se haya realizado algún movimiento de peón y sin que se haya producido algún cambio de pieza, lo cual según algunos cálculos una partida de ajedrez puede durar como máximo 3150 jugadas [6], y consecuentemente el árbol de ajedrez puede tener una cantidad de posiciones limitada a esta cantidad de jugadas.

Si fuese posible examinar el árbol por completo, buscando todas las líneas de juego y sus conclusiones sería posible determinar cual es el mejor movimiento inicial. Sin embargo, en la practica el árbol es demasiado extenso para considerar este mecanismo. Incluso en posiciones de medio juego determinar el mejor movimiento buscando en el árbol de variantes a partir de una avanzada posición de esta etapa del juego resulta imposible. Lo mejor que puede realizarse es buscar en un sector limitado del árbol de variantes, esperando obtener suficiente información con tal de decidir correctamente cual es el mejor movimiento.

En el inicio de 1970 la búsqueda en árboles de variantes alcanzaba la cantidad aproximada de 200 posiciones por segundo. Hoy (año 2003), DEEP BLUE busca en 2.000.000 de posiciones por segundo. Los mejores programas logran examinar todas las secuencias de

movimientos con una profundidad de 8 a 10 movidas en el árbol (4 a 5 jugadas por bando).

Líneas de juego cruciales como jaques y capturas tienen una profundidad de búsqueda aun mayor. En la jerga de programas de ajedrez, una *búsqueda de fuerza bruta de 6 movimientos* significa una búsqueda de todos los movimientos posibles para cada bando hasta una profundidad de 6 niveles (3 jugadas por bando) y con mayor profundidad en líneas altamente tácticas.

Para un árbol de variantes dado, el algoritmo minimax entrega una regla para decidir qué movimiento realizar frente a una posición dada. El algoritmo comienza calculando el valor numérico o puntaje para la posición al final de cada variante. Estas posiciones son denominadas *Posiciones Terminales* y su puntaje es calculado por una *Función de Evaluación*. La función de evaluación intenta medir cuan buena es la posición para el primer jugador. Un puntaje positivo significa que la computadora tiene ventaja, un puntaje negativo significa que su oponente tiene ventaja.

El algoritmo entiende que el objetivo del primer jugador es llevar al juego a una posición en la cual maximice su puntaje, mientras que el objetivo del rival es el opuesto, lograr una posición que sea minima en puntaje. Esto es equivalente a plantear que en niveles similares del árbol de variantes el algoritmo Minimax asignará a cada posición no terminal un puntaje igual al máximo puntaje de alguna de las posiciones sucesoras. En otras palabras, la computadora tomará el movimiento que le reporta el mayor beneficio para el movimiento siguiente. En los niveles impares donde juega el oponente el algoritmo asignará a cada posición no terminal un puntaje igual al mínimo de las posiciones sucesoras. Dado que los puntajes de las posiciones sucesoras de una posición no terminal deben ser conocidos con tal de dar un puntaje a esa posición los puntajes de las posiciones terminales deben ser determinados primero y luego los puntajes de las posiciones restantes hasta llegar a la posición raíz.

En juegos de cálculo extenso como el ajedrez los programas pueden examinar sólo una parte del árbol de variantes sin llegar hasta el final del juego. En estos casos los programas buscan hasta un cierto limite de nodos, estimando en ellos las posibilidades de ganar de cada bando mediante la función de Evaluación.

Existen varios algoritmos que ayudan a mejorar la eficiencia del Minimax. Uno de ellos es la poda Alfa-Beta, bajo la cual el programa debe buscar sólo aproximadamente la raíz cuadrada del numero de nodos que debería buscar sin esta implementación. Otras implementaciones incluyen las tablas de transposición las cuales guardan la información de posiciones ya examinadas. Ambas técnicas son analizadas en puntos siguientes.

Desafortunadamente el algoritmo minimax es de un orden $O(bn)$ donde $b$ (factor de anchura) es el numero de movimientos legales disponibles en promedio en cualquier momento y $n$ (profundidad) es el numero de movimientos que se calculan, donde un movimiento es una movida de un bando. Este numero crece en forma exponencial a medida que la profundidad aumenta, por lo cual una considerable cantidad de trabajo ha sido hecha con tal de desarrollar algoritmos que minimicen el esfuerzo realizado en buscar movimientos a una determinada profundidad.

### 5.2.2. Alfabeta: Haciendo Minimax Factible

Un estudio mas cuidadoso del algoritmo Minimax concluye que existen muchas variantes en al árbol de búsqueda las cuales no necesitan ser examinadas debido a que no afectan la decision del movimiento ya elegido para realizar. La idea bajo este algoritmo es el desechar rápidamente alternativas que se ven inferiores a una alternativa ya analizada, convirtiendo el algoritmo Minimax en una búsqueda sobre un árbol muy simplificado.

La idea fue propuesta en 1950 por John McCarthy quien en ese entonces era profesor del MIT. Hart y Edwards, también del MIT, publicaron en 1961 un paper en el cual describían el algoritmo mientras que A.L.Brudno describió el algoritmo en la literatura soviética en 1963.

Esencialmente, el algoritmo alfa-beta corresponde al algoritmo de Minimax pero con la capacidad de descartar tempranamente variantes que son inferiores a la preseleccionada.

En la secuencia de variantes mostrada en la figura 9.2, utilizando el algoritmo alfa-beta sólo 3 de los 4 nodos terminales necesitarán ser examinados. La búsqueda determina primero la puntuación para la posición **D**. Luego de asignar su valor (0) éste es asignado a la posición de origen **B**. Este score establece un "techo" para la evaluación siguiente. La evaluación de la posición **E** con un score de +2 significa que para el adversario en la posición B el movimiento 1...Txd5 es mejor que 1...Axc3, por lo cual la posición B mantiene el valor 0, que es el mejor valor al que el primer jugador puede optar luego de 1.Axc3 .

La posición base puede tener ahora un score provisional de 0. Este será el valor de tope, sin importar los valores de las posiciones terminales luego de 1.Txd5. Al analizar la posición **F** un valor de -4 es asignado a ella, por lo cual no será necesario analizar las siguientes posiciones terminales, puesto que de todas maneras conviene seguir por la rama de la posición **B**. En otras palabras, el movimiento 1...Dxa5 refuta inmediatamente 1.Txd5, por lo cual esta continuación es descartada en preferencia de 1.Axc3, con lo que nos "ahorramos" el analizar otra rama luego de la posición **C**.

Más generalmente, el algoritmo alfa-beta dice que un movimiento y en la posición Y refuta al movimiento predecesor x en la posición X bajo un mismo nivel si el score provisional de cualquier posición al mismo nivel en la continuación desde la posición base X tiene un score mayor o igual al score mantenido asignado a la posición Y por el movimiento y.

En 1969 James Slagle, uno de los mejores jugadores ciegos de ajedrez de los Estados Unidos y un distinguido científico computacional, junto con J.Dixon demostraron que para un árbol de variantes uniforme, con igual cantidad de movimientos en cada posición (digamos, con anchura **B**) y cuyas posiciones terminales se encuentran todas a una igual profundidad la cual denominaremos **d**, el número de posiciones terminales computadas por el algoritmo alfa-beta es de al menos $2B^{d/2} - 1$ para un $d$ par y $B^{\frac{d+1}{2}} + B^{\frac{d-1}{2}} - 1$ para un $d$ impar.

En un partida normal de ajedrez una posición posee cerca de 36 movimientos posibles. Realizando el modelamiento con un árbol de anchura 36 encontramos que para una profundidad de 2 movimientos al menos $2x36^1 = 71$ posiciones terminales son examinadas. Para el

mismo árbol uniforme pero con profundidad 3 al menos $36^2 + 36 - 1 = 1331$ posiciones son examinadas y al menos $2x36^5 - 1 = 3359431$ posiciones para un árbol con profundidad $d = 10$ movimientos. Esencialmente el árbol de búsqueda se incrementa en tamaño en un factor de 6 (la raíz de 36)por cada nivel de profundidad al que se avanza. Esto implica que para buscar en un nivel extra de profundidad la computadora debe procesar 6 veces más rápido o tomar una cantidad de tiempo 6 veces mayor.

**Ordenando los movimientos**

La eficiencia en la búsqueda bajo Minimax depende del orden de los movimientos en que se realiza esta operación. Las ventajas y desventajas relacionadas con un "buen" orden de los movimientos no son triviales: Un buen orden, definido como uno que causará el mayo número de cortes, resultará en un árbol de búsqueda de un tamaño aproximado a la raíz cuadrada del tamaño total del árbol asociado al peor orden de jugadas posible.

Desafortunadamente, ordenar los movimientos de la mejor forma implica encontrar los mejores y buscar primero sobre estos, lo cual es una tarea bastante difícil de lograr. Por ejemplo, el orden podría iniciarse con capturas, coronaciones de peón (las cuales cambian dramáticamente el balance de material) o jaques (los cuales a menudo permiten pocas respuestas legales), siguiendo con movimientos que causaron recientes cortes en otras variantes a la misma profundidad (denominadas movidas-asesinas, killer-moves)y entonces observar el resto de los movimientos. Esta es la justificación para la utilización del algoritmo de poda alfa-beta, así como el uso de tablas históricas.

En 1975 Donald Knuth y R.E.Moore demostraron que para un árbol uniforme de profundidad 2 en el cual las posiciones terminales son asignadas con un score aleatorio, de las $B^2$ posiciones terminales en promedio $\frac{B^2}{\log B - 0{,}923}$ son evaluadas. Dos años mas tarde Newborn demostró que para los mismos árboles uniformes de profundidad 2 y factor de anchura $B$ si el score de cada posición terminal corresponde a la suma de los scores aleatorios de posiciones previas a partir de la posición raíz entonces menos cantidad de evaluaciones son realizadas. Este modelo de dependencia de las ramas del árbol para asignar evaluaciones a las posiciones terminales pareciese modelar al juego de una mejor forma que los primeros modelos. En ajedrez el score de una posición terminal está dominado por el material, y éste es muy dependiente de las capturas realizadas en los movimientos precedentes. Newborn demostró que para este modelo en promedio $\frac{2B}{logB}$ posiciones son evaluadas.

Un programa que utiliza el algoritmo alfa-beta puede buscar en un árbol con el doble de profundidad que uno que no lo implementa. En búsquedas con árboles de ajedrez reales el número de posiciones terminales es bastante menor que el número predecido por los modelos aleatorios.

En 1986 P. Bettadapur demostró que las capturas deben ser ordenadas por aquellas que capturan la pieza de mayor valor del rival hasta las de menor valor obteniendo mejores resultados en la búsqueda. Asimismo, la captura de la ultima pieza que movió el rival es también un buen tipo de ordenamiento.

Nótese que esta técnica no tiene relación con búsqueda selectiva puesto que todos los movimientos serán examinados en alguna instancia, siendo examinados al final aquellos que aparentan ser "malos".

Una observación final, dentro de los movimientos generados algunos de estos resultan ser ilegales debido a que dejan al rey en jaque. El problema de validar la legalidad de los movimientos durante su generación es demasiado costoso comparado con evaluar como corte las "capturas de Rey", es decir, si luego de cierto movimiento es posible realizar la captura del Rey entonces dicho movimiento es ilegal y se produce un corte en la variante. Por su puesto que si el corte se produce antes de examinar el movimiento ilegal este nunca es revisado.

### 5.2.3. La necesidad de Buscar

Un programa fuera de serie seria capaz de observar la posición en el tablero y determinar quien tiene ventaja, por cuánto y que tipo de plan debe seguirse en la partida con tal de consolidar esa ventaja. Desafortunadamente, en ajedrez existen demasiados patrones de discernimiento, demasiadas reglas y demasiadas excepciones, que hacen que incluso los mejores programas no sean buenos en este tipo de estrategia, si bien son muy buenos en calcular y computar rápido. Por esto, en vez de intentar encontrar buenos movimientos sólo mirando el tablero los programas de ajedrez usan su "fuerza bruta" para jugar, buscando todos los movimientos posibles tanto propios como del rival hasta que ocurre algún acontecimiento que corte la búsqueda.

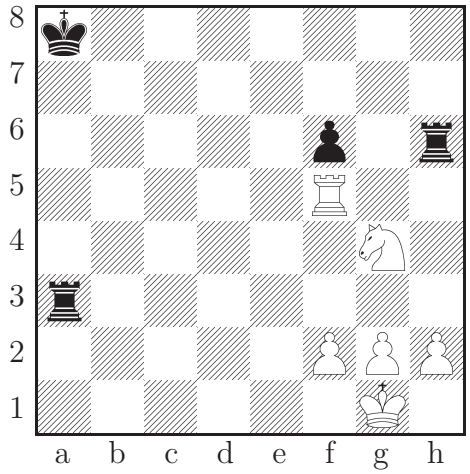
Las búsqueda sen profundidad son un buen método para enseñar a la máquina a resolver complicadas situaciones tácticas. Por ejemplo, considerar un ataque doble de caballo. Encontrar una forma de representación lógica de este "evento" requeriría algún esfuerzo, mas aún si debe determinarse si el caballo está o no protegido.

Sin embargo, un programa con profundidad de búsqueda de 3 movimientos será capaz de evaluar el ataque doble por sí mismo. Intentará situar el caballo en la casilla donde origina el ataque doble, revisará todas las respuestas del rival contra este ataque y luego capturará una de las piezas atacadas cambiando el equilibrio material de la posición. Por su puesto dado que una estrategia de búsqueda completa busca todo, nunca perderá una oportunidad de obtener ventaja material. Si existe una combinación de 5 movimientos que genera un jaque mate o bien captura la dama rival la maquina la verá tan rápido como sea su sistema de búsqueda.

### 5.2.4. Movimientos Asesinos (killer moves)

La heurística de "movimientos asesinos" es utilizada en la mayoría de los programas de ajedrez con el objetivo de identificar rápidamente buenos movimientos. Jim Gillogly [45] utilizó esta técnica en su programa TECH en 1972. Esencialmente, cuando es encontrado un movimiento que refuta cierta variante en el nivel de profundidad $k$ dentro del árbol de búsqueda, este es almacenado en la lista de movimientos "killer" (Existe una lista de

movimientos asesinos por cada nivel de profundidad de búsqueda en el árbol). Luego cuando se generan nuevos movimientos en el árbol de variantes la lista de movimientos asesinos es consultada en cada nivel y si un movimiento es encontrado entonces es ordenado con prioridad en la lista de movimientos a analizar.




La figura ilustra el funcionamiento de esta heurística. Si es el turno de mover de las blancas estas intentarían el movimiento 1.♘xh6 debido a que captura la torre por nada. Luego de examinar las replicas del negro encontrara que este movimiento es refutado por la respuesta 1... ♖a1 mate . Entonces, cuando el programa examine nuevos movimientos para el blanco el primer movimiento negro que tomara como respuesta sera 1... Ta1 debido a que es un movimiento legal que genero un corte en una variante anterior y existen chances de que lo haga en la actual variante. La heurística le enseña al programa que el movimiento 1...Ta1 es una seria respuesta a considerar luego de cada jugada del blanco.

Varias estrategias existen para guardar los movimientos killer. Lo mas simple es mantener una lista bastante corta de a lomas dos movimientos de profundidad. Solo las mas recientes refutaciones son mantenidas en el arreglo. Alternativamente cada refutación en el arreglo puede mantener un contador de cuantas veces ha causado cortes en los análisis. Cuando un nuevo movimiento killer es agregado este reemplaza al ultimo movimiento guardado en el arreglo. Utilizando esta técnica el numero de cortes es mayor y la eficiencia del algoritmo alfa-beta aumenta. En lugar de la lista de movimientos killer el programa PHOENIX de Jonathan Schaeffer utilizaba dos arreglos de 64x64 los cuales mantenían todos los movimientos de cada bando que causaban una refutación o corte. El arreglo con la continuación principal también podia ser considerado una buena fuente de movimientos killer, tal como acotaran Selim Akl y Newborn en 1977.

### 5.2.5. El concepto de Posición Estable

Un requisito para realizar la correcta evaluación de la posición en un nodo terminal es el que ésta no sufra grandes cambios en el movimiento siguiente del rival. Estos cambios

se refieren a movimientos que alteren considerablemente el equilibrio de material o bien que definan ventajas para un bando.

Por ejemplo, sería poco astuto evaluar una posición en donde se realiza un cambio de damas considerando sólo la captura de la primera dama y no la respuesta de recaptura del rival. Dado esto, el concepto de "posición estable" se reconoce como aquella posición en la cual no existen movimientos que alteren considerablemente su evaluación. Las búsquedas realizadas por prácticamente todos los programas contemporáneos consideran esta característica, modificando la profundidad de búsqueda de acuerdo a si la posición terminal es o no es estable.

### 5.2.6. Técnicas de poda y Búsqueda de profundidad variable en posiciones Estables

Los primeros programas de ajedrez utilizaron la búsqueda selectiva (forward pruning) para reducir el factor de anchura presente en cada posición en el árbol de búsqueda. Basados en consideraciones de tiempo los programas buscaron aplicar técnicas inteligentes para encontrar lo movimientos relevantes de una posición con tal de filtrar las alternativas consideradas inferiores. Algunas heurísticas fueron utilizadas para dar prioridad o descartar ciertas clases de movimientos. Estas heurísticas simulaban reglas tales como "Incluir al menos un movimiento para cada pieza" o bien "Incluya todos los movimientos de captura" eran utilizadas para el primer fin (inclusión) mientras que reglas como "No movilice la Dama tempranamente" o "No ubique al caballo en los bordes del tablero" son utilizadas como descartadoras. Sin embargo, las heurísticas de búsqueda selectiva eran independiente de estas técnicas y los programas a menudo seleccionaban movimientos incorrectos. Gradualmente, en la década de los 70 las heurísticas descartadoras de movimientos fueron desapareciendo de los programas más exitosos.

Si bien la búsqueda selectiva resultó ser infructuosa, los mejores programas utilizaron búsqueda con profundidad variable. Estas se basaban en una de profundidad base, la cual una vez lograda era evaluada con tal de seguir a niveles más profundos. Movimientos como jaques, capturas que definían posiciones materialmente inestables eran seleccionados para seguir los cálculos con tal de llegar a posiciones sin este tipo de eventos y cuya evaluación fuese más precisa. HITECH por ejemplo extendía la profundidad en un nivel por sobre la profundidad base para cada recaptura que equilibrase el nivel material. BEBE realizaba lo mismo para avances de peón a la sexta, séptima u octava filas. En su paper publicado en [69] David Slate y Larry Atkin comentaban que la típica búsqueda de su programa CHESS 4.9 contenía prácticamente la mitad de las posiciones de búsqueda a una profundidad por sobre la profundidad base, logrando resultados dramáticamente positivos.

### 5.2.7. Extensiones Singulares

En 1987 los programadores de DeepThougth agregaron a su programa la heurística de Extensiones Singulares, describiéndola de la siguiente forma en la publicación de Diciembre de 1988 del ICCA [30].

" La heurística utiliza una búsqueda de fuerza bruta modificada la cual entrega mediciones del nivel *forzado* de un movimiento (es decir, de cuan "obligatorio" es el movimiento en cierta posición).Un movimiento es declarado como "singular" si retorna un valor mucho mejor que todas las otras alternativas analizadas.

Luego, cuando encontramos un movimiento singular que afecta la continuación de la búsqueda si su valor cambia, la posición resultante de este movimiento es analizada con profundidad de 1 movimiento extra. En el ajedrez, dependiendo del criterio utilizado en seleccionar los tipos de movimientos singulares la sobrecarga puede variar desde un 5 % a 20 % para posiciones estables. Para posiciones ricas en posibilidades tácticas la sobrecarga puede crecer al nivel de doblar la cantidad de nodos buscados a una profundidad dada, pero dado que las extensiones permiten al programa analizar los resultados de variantes tácticas tempranamente, la sobrecarga se compensa con la rápida evaluación de variantes claves.

### 5.2.8. Búsqueda con Profundidad Iterativa

Un problema no menor el cual los programadores debieron enfrentar fue el cómo setear en forma correcta el parámetro de profundidad de búsqueda antes de iniciarla. Si éste era demasiado bajo entonces el programa realizaría movimientos innecesariamente rápidos, no tomando ventaja del tiempo restante. Si el parámetro era demasiado grande el programa tomaría un tiempo excesivo en movimientos cuyo descubrimiento no debería tomar más de tres minutos. De hecho, en los primeros programas la adecuación a un tiempo limitado era un gran problema si éstos poseían un parámetro de profundidad elevado.

La búsqueda con profundidad iterativa empezó a popularizarse a mediados de la década del 70, y prácticamente solucionó el problema de seteo del parámetro de profundidad. Slate y Atkin introdujeron esta solución en 1975 en su programa CHESS "4.5" bajo la supervisión de Peter Frey, profesor de psicología de la Universidad de Northwestern. En los dos años siguientes la técnica pasó a ser parte de cada programa de ajedrez. En los años siguientes la técnica encontró utilidad en otros problemas relacionados con la Inteligencia Artificial, en particular con la demostración automatizada de teoremas.

Con la búsqueda de profundidad iterativa mas allá de realizar una sola búsqueda a una profundidad dada, una secuencia de búsquedas con profundidad incremental es realizada comenzando con profundidad uno, luego dos, y continuando hasta que el tiempo de reflexión máximo se completa. Cada iteración encuentra variantes principales las cuales tienen prioridad de análisis en las iteraciones siguientes. En cada iteración se almacenan posiciones en la tabla de hash para su uso en iteraciones siguientes. El resultado global de esto es nua mejora

en la eficiencia del algoritmo alfa-beta, el cual compensa la pérdida de tiempo requerida para la realización de nuevas búsquedas. Más importante aún es el hecho de que la búsqueda con profundidad iterativa permite terminar la búsqueda en cualquier momento sin consecuencias nefastas. Lo peor que podría ocurrir es que cuando el programa está en la $n$-ésima iteración y debe jugar, éste ya posee la mejor evaluación en la iteración $n-1$. Detenerse en el medio de una búsqueda perderá la posibilidad de encontrar un mejor movimiento en esa iteración sólo si éste está bajo el orden dado al movimiento actualmente analizado. Esto ocurre cuando el mejor movimiento no es evaluado como tal en la penúltima evaluación.

### 5.2.9. Búsqueda de Ventana

A fines de los 70 la búsqueda de ventana empezó a ser utilizada en los programas de ajedrez en conjunto con la búsqueda con profundidad iterativa. Su uso incrementó la eficiencia del proceso de búsqueda al lograr obtener una mayor cantidad de "cortes" en las variantes analizadas. La búsqueda de ventana está basada en la idea de que en una misma variante es muy probable que el score a encontrar en la iteración actual será probablemente el mismo que el encontrado en la iteración anterior. Un "score esperado" (RS) es supuesto al principio de la nueva iteración, usualmente el score guardado en la raíz del árbol en la iteración previa, y una búsqueda de ventana es realizada en torno a ese score. Entonces durante la búsqueda una serie de "cortes" ocurren en posiciones cuyos scores no están dentro del rango de la ventana. Luego durante la búsqueda ocurren una serie de "cortes" en variantes cuyo score no cae dentro del rango de la ventana.

El "tamaño" de la ventana es típicamente de 2 peones (2P). Al principio de cada iteración el score de la raíz es inicializado en $RS - P$ y cuando la búsqueda obtenga una evaluación en la nueva posición su score tendrá el valor de $RS + P$, donde $P$ corresponde al valor de un peón. Estos valores corresponden a $-\infty$ y $+\infty$ cuando la búsqueda de ventana no es utilizada. Se dice que la ventana es inicializada en los valores $< RS - P, RS + P >$. La búsqueda entonces procede como es usual. Si un movimiento posee una evaluación que está dentro de los rangos de la ventana cuando la iteración termina, la próxima iteración comienza con una ventana de ancho 2P pero centrada en el nuevo valor obtenido. Si ningún valor cae dentro del rango de la ventana se dice que la búsqueda ha fallado "alto" si el score es igual o mayor a $RS + P$ y ha fallado "bajo" si el score es menor o igual a $RS - P$. Si la búsqueda falla la iteración debe ser repetida para encontrar el real valor de la raíz y corregir la continuación principal.

En el caso de una falla "alta" la ventana debe setearse a $< RS+P, +\infty >$ para asegurar que la segunda evaluación no fallará. Similarmente en el caso de una falla "baja" la ventana debe setearse en los valores $< -\infty, RS - P >$. Mientras más limitado el ancho de la ventana la primera evaluación terminará antes, pero con mayor chance de fallar. Cuando es utilizada la búsqueda de ventana, cada´iteración de búsqueda debe considerarse como consistente de dos etapas, la segunda innecesaria si la primera es exitosa.

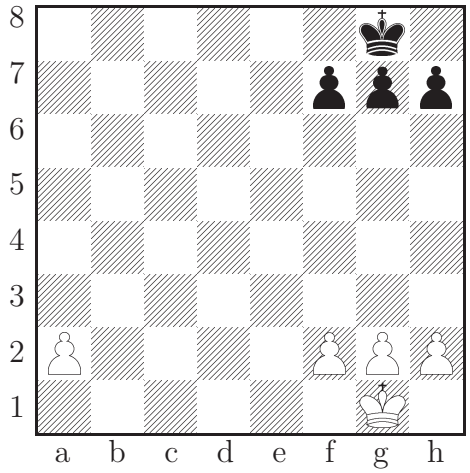
Tipos más sofisticados de búsqueda de ventana que permiten setear la ventana y realizar búsquedas sobre sub árboles de variantes en todas las posiciones del árbol principal son

utilizadas por algunos programas. Estos procedimientos recursivos fueron descritos en papers de J.Pearl en 1980 y de A.Reinfeld en 1985.

### 5.2.10. El Efecto Horizonte

Uno de los mayores dolores de cabeza de los programadores de maquinas de ajedrez en relación a la búsqueda es el llamado "efecto horizonte", descrito en principio por Hans Berliner.

Supongamos que un programa busca con profundidad de 8 movimientos, y descubre que su oponente le capturará la dama en el movimiento 6. Para evitar esto entregará material tal que la captura de dama es evitada hasta el movimiento 10, que el computador no verá debido a que su profundidad es de 8 movimientos. Desde el punto de vista del programa la dama está salvada puesto que su captura no es visible, pero ya ha perdido un alfil y la captura de dama reaparecerá en los próximos movimientos. Se deduce que el encontrar una posición en donde un programa pueda razonar correctamente acerca del valor relativo de las fuerzas (estáticas y dinámicas) no es una tarea trivial de lograr y que el buscar líneas de juego a una misma profundidad puede ser muy desventajoso. Numerosas técnicas se han desarrollado para evitar el efecto horizonte: búsqueda de "Posiciones Estables" y las "Extensiones Singulares" desarrolladas en Deep Blue son algunas de las más populares.




*Ejemplo del "Efecto Horizonte". El programa (con negras) de no tener profundidad mayor a 6 movimientos no se percatará del avance del peón "a" del blanco.*

## 5.3. Función de Evaluación

Un programa de ajedrez debe tener alguna forma de evaluar si en una posición dada tiene ventaja o bien ha perdido la partida. Esta evaluación depende fuertemente de las reglas

del juego: mientras que el "Balance de Material" (numero y valor de las piezas en el tablero) es un factor dominante en ajedrez, debido a que poseer ventaja incluso de un peón puede decidir la partida, en otros juegos no lo es.

Desarrollar una buena función de evaluación es una tarea difícil y muchas veces frustrante. Shannon en [68] propuso una serie de parámetros a medir dentro de la función de evaluación los cuales tomaban en cuenta consideraciones materiales y también posicionales. Él mencionaba que estos factores podían tener una valoración muy distinta dependiendo de la etapa del juego en que nos encontremos (Apertura, Medio Juego o Final). *El valor relativo de cada factor está abierto a un considerable debate y debe ser determinado en procedimientos experimentales. Deben por su puesto también existir una serie de otros factores a considerar.*

Esta propuesta resultó ser la base de los primeros programas de ajedrez, que si bien incorporaban esta capacidad no lograban aún llegar a un nivel de maestro puesto que no estaban aún aptos para considerar patrones más allá del concepto material.

### 5.3.1. La Necesidad de la Función de Evaluación

Para guiar la selección de movimientos los primeros programas de fuerza bruta utilizaron una función de evaluación estática de la posición. Esta estaba basada en la premisa de que es posible examinar una posición y calcular un score que representa su valor relativo a otras posiciones. Si esto es cierto, entonces la función de evaluación se reduciría a la simple tarea de buscar en el arreglo de posiciones posibles con tal de encontrar aquella con el mejor score.

Desde un punto de vista teórico esta premisa es razonablemente cierta. Sin límites de tiempo, recursos de hardware ilimitados y asistencia de grandes maestros de ajedrez un equipo de buenos programadores podrían presumiblemente escribir una muy detallada función de evaluación que consideraría factores que un jugador humano toma en cuenta al jugar al ajedrez. Con la ayuda de un reconocedor de patrones y una gran tabla de hash de posiciones un evaluador retornaría un valor de posición muy preciso para la mayoría de las posiciones.

Desafortunadamente, esta función de evaluación sería tan grande y compleja que requeriría mucho tiempo y poder computacional para ejecutarse en un tiempo razonable. Un programa buscando a una profundidad de 8 movimientos como BELLE o CRAY BLITZ sólo podían disponer de 5 a 10 microsegundos para la evaluación de cada posición. Para búsquedas de 5 movimientos los programas deben evaluar posiciones en uno a 10 milisegundos.

La función de evaluación debe a la vez ser muy simple y fácil de ejecutar, y debe funcionar en base a operaciones que la computadora puede manipular con facilidad en vez de conceptos que tendrían sentido bajo el punto de vista del maestro de ajedrez. La experiencia ha demostrado que el dar a un programa demasiado "conocimiento ajedrecístico" es una tarea difícil y peligrosa. Si la información es incompleta se pueden tener efectos contrarios a lo deseado. El programa puede utilizar esta información en un momento y forma no adecuados. Si el código es exhaustivo en la manipulación de estas excepciones entonces puede influir en alterar la velocidad de funcionamiento del programa lo cual afectará su capacidad de

búsqueda.

Las funciones de evaluación utilizadas en el mundo real no pretenden evaluar las posiciones con suficiente detalle. El objetivo es dar al programa suficiente información con tal que pueda usarla en su búsqueda dentro del árbol de variantes con tal de descubrir el efecto de factores dinámicos los cuales la función no puede evaluar, tales como el resultado de combinaciones. La función de evaluación debe sin embargo ser capaz de entregar al programa información acerca de aspectos estáticos de la posición tales como estructura de peones, piezas sin defensa, movilidad, etc. Esta característica logra efectos a largo plazo en el juego los cuales la búsqueda muchas veces no puede descubrir.

Por todas estas razones la función de evaluación de un programa no puede ser considerada como una medida precisa del valor de la posición, si bien es una herramienta indispensable. Los programas de fuerza bruta, por ejemplo, no dependen de la función de evaluación como su "última palabra" en muchas de las posiciones las cuales evalúan. Durante la búsqueda en el árbol de variantes el programa confirmará la evaluación de la posición con la búsqueda sobre el siguiente movimiento.

La misión de la función de evaluación es entonces bastante simplificada. No es su labor el evaluar en forma precisa factores dinámicos como el resultado de combinaciones, pues estos son dejados al motor de búsqueda. Tampoco debe hacer mediciones posicionales precisas puesto que los límites de tiempo que el programa dispone para realizar su movimiento le impiden perder tiempo en esta consideración. Por lo tanto, la misión de la función de evaluación es el cooperar con el árbol de búsqueda en el determinar en forma aproximada el valor de una posición observando qué puede ocurrir en el juego futuro. Para lograr esto en forma efectiva la función de evaluación debe ser capaz de medir el equilibrio material (cantidad de piezas para cada bando) y generar un razonable análisis de la situación estática de la posición el cual la búsqueda en profundidad es incapaz de determinar.

Para mejorar la eficiencia en el árbol de búsqueda y proveer de alguna guía (si bien imperfecta) en los muchos casos en que consideraciones posicionales evolucionarán a demasiado largo plazo tal que puedan ser detectadas en el árbol de búsqueda, la función de evaluación contiene heurísticas de análisis dinámico de la posición. Este es el mayor desafío en la construcción de funciones de evaluación. Cuanta información puede obtener el programa con el tiempo disponible? Como regla general la respuesta es "no lo suficiente". En este caso un código claro puede hacer una notable diferencia en la fuerza de juego del programa.

### 5.3.2. La estructura de una función de evaluación

La función de evaluación de un típico programa de ajedrez consiste en un número de parámetros de medición (cada uno de ellos diseñado para rápido cálculo computacional) los cuales son adicionados con tal de generar un score de la posición. Cabe notar que ésta no es la única forma de evaluar posiciones. Algunos programas utilizan funciones de evaluación no lineales y las posiciones son evaluadas por medio del efecto de un movimiento propuesto en una posición dada [33].

**Equilibrio Material**

El patrón más importante en la función de evaluación es la medición de material. Debido a que un programa no posee avanzadas herramientas para medir las consideraciones posicionales del juego, especialmente aquellas de influencia de largo plazo, es ideal el saber si una ventaja posicional justifica un sacrificio de material. Para minimizar este problema el equilibrio material tiene un efecto dominante en la función de evaluación de los programas. Esto ha tenido una serie de efectos no deseados. Los programas son demasiado "avaros" y no pueden realizar sacrificios o resistir aquellos realizados por el oponente a menos que un resultado decisivo ocurra en su búsqueda en profundidad. También dentro de la búsqueda se asume que no se realizarán sacrificios de material, lo cual altera la correcta evaluación de varias continuaciones.

Por otro lado un programa de fuerza bruta gana bastante del hecho de que su función de evaluación valorice altamente el material. Como dijo el famoso jugador francés Tartakower en una oportunidad "Los mejores sacrificios son los del oponente" el programa explotará en forma infalible cualquier error cometido en el cálculo de un sacrificio de material, obteniendo la ventaja. Muchas partidas de ajedrez se deciden por algún error u omisión táctica pro parte de uno de los bandos, por lo cual esta característica tiene un beneficio especial en estos casos. Si la función de evaluación del programa está adaptada para favorecer posiciones dinámicas com muchas posibilidades tácticas y si puede lograr suficiente profundidad en la búsqueda entonces el programa no tendrá igual en el cálculo de variantes tácticas frente a su oponente.

El equilibrio de material es una simple expresión la cual el programa puede evaluar muy rápidamente. Cuando el algoritmo alfa-beta trabaja en forma apropiada el equilibro de material es el único factor a considerar en la mayoría de las posiciones examinadas por el árbol de variantes, dado que bajo estándares humanos sería absurdo considerar posiciones con un desequilibrio de material excesivo.

Una típica expresión de equilibrio material es una combinación algebraica de la diferencia de material, la cantidad total de material en el tablero y el número de peones presentes en el bando con ventaja. [68]. Los últimos parámetros son utilizados para modificar la diferencia de material tal que el bando con ventaja gana puntuación por realizar cambios de piezas y a la vez conservando los peones lo cual es en general una buena estrategia para el final de partida.esto por su puesto puede traer problemas en otras situaciones, puesto que en ajedrez existen muchas excepciones para cada regla. Los scores generalmente asignados a cada pieza son relativos al valor del peón, con los valores:

♘ : 3 peones ♖ : 5 peones
♗ : 3 peones ♕ : 9 peones

El Rey en algunos programas no tiene un valor asignado si bien en otros se asigna un valor infinito con tal de cambiar drásticamente la evaluación material en caso de su captura. El valor de un peón es un valor arbitrario que generalmente va entre 50 y 100 puntos.

La simple contabilización de los puntos en ajedrez sería una estrategia básica para un jugador iniciado, pero para jugadores que logran un nivel de maestro este concepto es reemplazado por una evaluación más sofisticada del valor de cada pieza en una situación particular durante la partida. El maestro realiza esto en forma prácticamente instintiva basado en la experiencia adquirida en cientos de partidas jugadas. Por ejemplo, el maestro sabe que si bien el Alfil es algo mejor que el Caballo existen ciertos tipos de posiciones en las cuales un Alfil no cumple un rol mayor que un peón, o incluso situaciones en las cuales un Caballo puede ser más valioso que una Torre. Estos conceptos por supuesto resultaban demasiado sofisticados para los programas de la época.

Debido al factor dominante del material en la evaluación de la posición, un programa genera un score de prueba basado en este concepto y aplicando ciertas correcciones (generalmente menores que el valor de un peón) las cuales representan el máximo efecto posible de los factores posicionales. El score de prueba es entonces examinado por la rutina de búsqueda para verificar si produce un corte. Si el corte se produce el programa ha encontrado una refutación basada en factores materiales y no necesita realizar una evaluación posicional de la situación. Esta facilidad de la función de evaluación salva mucho tiempo de análisis dado que cerca del 50 y 90 por ciento de las posiciones actualmente evaluadas pueden ser eliminadas de consideraciones sólo materiales.

Asumiendo que esto no ocurra, el programa genera entonces un valor para el factor posicional de la evaluación. Este corresponde a la suma del número de términos que tienen diferente peso dependiendo de su importancia. En general los factores relacionados con material y consideraciones tácticas poseen un peso mayor que aquellos que representan conceptos mas abstractos.

**Evaluación Posicional**

La función de evaluación debe tomar en cuentas consideraciones de largo plazo tanto a favor como en contra dentro de la evaluación de una posición, es decir, efectos que puedan persistir sobre un número de movimientos mayor que los que él programa pueda calcular. Estos efectos se basan muchas veces en consideraciones estáticas sobre la posición, es decir, consideraciones basadas en la ubicación de las piezas y peones y sus posibilidades de evolucionar dentro de la partida. Estas consideraciones son las denominadas consideraciones posicionales.

Los primeros programas de ajedrez fallaban rotundamente en este tipo de consideraciones cometiendo errores similares a los novatos en el juego. El problema era que para dar a la máquina una fuerte evaluación posicional era necesario el entregarle una serie de parámetros relacionados con "conocimiento ajedrecístico", algo que los grandes maestros de ajedrez obtienen principalmente en base a su experiencia. La duda era entonces cuanto "peso" dar a los factores posicionales por sobre el conocimiento ajedrecístico a ocupar en la función de evaluación. Los primeros programas debieron dar importancia a la capacidad de búsqueda y limitarse a una función d evaluación la cual considerara factores principalmente materiales. Con el paso de los años y la mayor capacidad de las máquinas, el incluir factores posicionales

dentro de la función de evaluación ya no iría en contra de la capacidad de cálculo del programa, por lo cual empezó a ser una tarea para los programadores. Los primeros intentos no fueron fáciles, puesto que para esto los creadores de máquinas de ajedrez necesitaron del asesoramiento de expertos para incluir el conocimiento en sus programas. Aún así , las consideraciones posicionales dentro del ajedrez resultan ser tantas y con tantas excepciones lo cual hizo que la tarea de dar un conocimiento posicional a los programas fuese a ratos extremadamente frustrante.

De acuerdo a Botvinnik en [3] un factor posicional que logra un efecto positivo en una posición dada puede perfectamente ser negativo en otra. Por ejemplo, los peones doblados son buenos o malos ? La respuesta depende de la situación. En algunas ocasiones los peones doblados son un objetivo de ataque mientras que en otras pueden no ser una debilidad y defender casillas críticas. Lo mismo puede ser dicho para otros factores. Su propuesta de análisis posicional se basaba más bien en un modelo matemático basado en el control de casillas y consideraciones materiales.

- **Pieza sin defensa**
  Esta evaluación está basada en el concepto de que una pieza sin defensa es una desventaja debido a que al menos en algún momento deberá perderse un tiempo en defenderla o retirarla de alguna amenaza rival. Si más de una pieza se encuentra sin defensa esto da lugar a amenazas de ataques dobles de parte del rival. Esta consideración puede aproximarse a una consideración dinámica que el árbol de búsqueda debería encontrar, pero en muchos programas es tomada en cuenta como una consideración estática debido a los factores antes mencionados, los cuales son difíciles de evaluar

- **Movilidad**
  La cantidad de casillas disponibles para cada pieza entrega un factor de "libertad de movimiento" disponible para ellas. En ajedrez es muy importante el que las piezas tengan libre acción y a la vez entorpecer la acción de las piezas rivales. La evaluación de movilidad medirá simplemente la cantidad de movimientos legales disponibles para todas las piezas de cada bando considerando la diferencia para cada pieza en la cantidad de casillas atacadas por Blancas y Negras. Esta característica de movilidad hace que el programa priorice el desarrollo en la apertura de sus piezas menores ubicándolas en posiciones favorables y reservando el desarrollo de Torres y Dama para más tarde.

- **Estructura de Peones**
  Las relaciones entre los peones y su ubicación son muchas veces la base de una evaluación posicional. Las estrategias posicionales desarrolladas por los jugadores humanos están a menudo basadas en el avance de peones y otros planes enfocados a alterar la estructura de peones de manera favorable. Con la práctica se ha observado que ciertos tipos de estructuras de peones, por ejemplo peones doblados, aislados o bien retrasados, son debilidades posicionales muy serias debido a que pueden ser un objetivo de ataque para el rival. Si el peón débil no se pierde usualmente significa la pérdida de movilidad de alguna pieza. este tipo de situación es probablemente uno de los factores decisivos en partidas entre jugadores humanos de buen nivel.

  Un programa evalúa estructuras de peones asignando bonos o penas para algunas relaciones entre peones. El programa penaliza las situaciones con peones doblados, retrasa-

dos o aislados, mientras que favorece a los peones pasados o aquellos que quitan movilidad a las piezas enemigas.

- **Scores asignados a cada pieza**
  Una serie de evaluaciones es realizada a cada pieza en función de su ubicación en el tablero, capacidad de movimiento y etapa del juego. Por ejemplo para el caso del Rey en la Apertura y Medio juego necesita ser protegido. La protección del rey depende del material presente (por ejemplo si existen o no Damas) y de la estructura de peones que lo protege de ataques rivales. En el Final la evaluación del Rey toma otro carácter pues allí es más importante que esté en el centro del tablero, ataque peones rivales y a la vez apoye el avance de los propios.

- **Scores asignados en el final de partida**

  Durante el final algunos cambios deben ser realizados en la función e evaluación con tal de poder asignar valores razonables. Los cambios al Rey ya fueron mencionados. En el caso de las torres, por ejemplo, se valoriza el que éstas estén detrás de los peones pasados o bien limiten la acción del Rey rival, los caballos reciben un bono por el bloqueo de peones rivales, etc.

  Un factor particularmente importante en el final de juego es el agregar la "Regla del Cuadrado" para los peones que buscan coronar, la cual fue implementada en varios programas de los 70. Esta simple regla implementada en CRAY BLITZ logró que subiese su nivel de juego en aproximadamente 200 puntos de Elo [51].

## 5.4. Otros avances desarrollados

### 5.4.1. Tablas de Transposición

En ajedrez existe a menudo más de una forma de obtener una misma posición mediante diferentes secuencias de movimientos. Por ejemplo la posición resultante de la secuencia 1.e4 e5 2.f4 d6 es la mis que en la secuencia 1.f4 d6 2.e4 e5. Obtener posiciones idénticas con una secuencia distinta de movimientos se denomina transposición.

Ahora, si nuestro programa ha tomado considerable esfuerzo en buscar y evaluar la posición resultante de 1.e4 e5 2.f4 d6 sería muy útil si fuese capaz de recordar los resultados de esa posición con tal de que sea innecesario el evaluar la posición resultante de la secuencia 1.f4 d6 2.e4 e5. Desde que el programa de Richard Greenblat MacHack VI [46] desarrolló esta técnica prácticamente todos los programas la han incorporado.

Este método presenta varias ventajas, incluyendo :

- Velocidad: En situaciones donde hay muchas posibles transposiciones (por ejemplo, en la etapa del final) la tabla de transposiciones contiene con seguridad un 90 % de las posiciones consultadas.

- Libre Profundidad: Supongamos que se necesita buscar cierta posición a una profundidad dada, digamos, 4 movimientos (es decir, 2 jugadas por cada bando).Si la tabla de transposiciones contiene resultados de 6 movimientos para la posición inicial no sólo se evita la búsqueda sino que se obtienen mejores resultados que los que se hubiesen obtenido de haber realizado la evaluación.
- Versatilidad : Todo programa de ajedrez posee un "libro de aperturas", es decir, una lista de posiciones conocidas en la práctica del ajedrez y los mejores movimientos seleccionados en ellas.

Dado que el modus operandi del libro de aperturas es idéntico al de las tablas de transposición (buscar la posición y entregar los resultados en caso de que haya se encuentre en la tabla)se puede inicializar la tabla con el contenido del libro de aperturas tal que si se llega a una posición desconocida en el libro la tabla de transposición puede contener información acerca de ella.

El único problema real relativo a las tablas de transposición es su "voracidad" en términos de memoria. Para ser realmente útil la tabla debe contener varios miles de registros, un millón o más es un buen número. Con 16 bytes por entrada esto puede ser un problema en sistemas carentes de memoria.

El programa CHESS 4.5 empleaba Tablas de Hash para mantener los resultados de computaciones de alto costo que raramente cambiaban en valor o alternancia entre un pequeño numero de posibilidades :

Estructura de peones. Indexando sólo la posición de los peones esta tabla requiere poco espacio y dado que hay pocos movimientos de peones esta cambia muy poco con decir que el 99 %de las posiciones resultan en hit de esta tabla.

Esto puede que no sea muy útil en estos dias en que gozamos de muchos ciclos de CPU, pero la lección es muy valiosa : mediadas de preprocesamiento pueden salvar mucho calculo computacional al costo de poca memoria.

### 5.4.2. Generación de Llaves de Hash para posiciones

Las tablas de trasposición descritas anteriormente son usualmente implementadas como diccionarios de hash con algún orden, lo cual lleva a la pregunta ?' Cómo generar llaves de Hash para posiciones en forma rápida y eficiente ? El siguiente esquema fue descrito por Zobrits en 1970:

- Generar 12x64 N-bit numeros aleatorios (donde la tabla de transposición posee $2^N$ entradas) y guardarlos en la tabla. Cada número aleatorio es asociado con una pieza o una casilla. Una casilla desocupada se representa con una palabra vacía.
- Iniciar con una llave de hash nula.

- Escanear el tablero. Al encontrar una pieza, XOR su numero aleatorio con la actual llave de Hash. Repetir hasta que el tablero completo sea examinado.

Un efecto interesante de este esquema es que el valor del hash es fácil de actualizar luego de cada movimiento sin tener que re-escanear el tablero completo. Las tablas de Hash están diseñadas para mantener tanto espacio memoria como se permita.El número de entradas es usualmente una potencia de 2. Tamaños típicos van desde 4000 posiciones hasta 4 millones. Cada entrada mantiene la información acerca de una posición. CRAY BLITZ utilizaba 6 millones de palabras en su tabla de hash con 64 bits por palabra. BEBE utilizaba una tabla con palabras de 96 bits. Podia mantener hasta 256K de posiciones con cada posición en una palabra.

Una excelente descripción de estas herramientas se encuentra en el texto de Levy y Newborn, "How Computers Play Chess", 1991 [14].

### 5.4.3. Tablas Históricas

La "heurística histórica" es una descendiente de la técnica de las "movidas decisivas" (killer moves). Una tabla histórica debe mantenerse con tal de notar qué movimientos han tenido resultados interesantes en evaluaciones pasadas y deberían ser intentadas en la posición actual. La tabla es un simple arreglo de enteros de 64x64. Cuando el algoritmo de búsqueda decide que un movimiento es una fuerte amenaza le consultará a la tabla histórica con tal de aumentar su valor. Los valores presentes en la tabla serán utilizados para el ordenamiento de las movidas y asegurar que las movidas "históricamente fuertes" sean analizadas primero.

### 5.4.4. Bases de Datos

En 1966 cuando el programa soviético del Instituto de Física Teórica enfrentó al programa de Kotov/McCarthy las partidas fueron declaradas tablas cuando entraron en la etapa del Final. De esta manera ambos programas evitaron entrar en una etapa en donde hubiesen dado un triste espectáculo en virtud de su nivel de juego en esta fase del ajedrez. Los programadores entendían que conceptos relativos a esta etapa del juego eran totalmente desconocidos por sus programas, en parte no sólo porque las características de juego no estaban incluidas en la función de evaluación de los programas, si no que además en esta etapa del juego muchas posiciones requieren de profundidades de búsqueda pro sobre las capacidades de las más rápidas computadoras de aquellos años.

#### Bases de datos de Finales

Los primeros conceptos relativos a esta etapa del juego fueron incluidos en la función de evaluación de los programas. Aún así, existían conceptos tales como el Zugzwang o bien

la oposición los cuales resultaban tremendamente difíciles de programar en la evaluación. En 1977 el programa PEASANT escrito por Newborn desarrollaba cualquier final entre Reyes y peones. El programa demostró que mediante un algoritmo de fuerza bruta y una función de evaluación lo suficientemente adaptada el programa era capaz de realizar búsquedas desde 4 a 13 niveles en 2 minutos, resolviendo posiciones de avanzada dificultad, si bien presentó ciertas dificultades.

La introducción de Tablas de Transposición tuvo un positivo efecto en el nivel de los programas en esta etapa del juego. HITECH y CRAY BLITZ fueron sometidos a tests similares a los del programa PEASANT obteniendo mejores resultados y en un tiempo menor. Los programadores se habían concentrado en realizar programas con mejores funciones de evaluación y búsquedas más eficientes, siendo de gran apoyo los avances en hardware que dispusieron. Con computadoras mas rápidas y de mayor capacidad de memoria ciertos finales lograron ser totalmente analizados por computadora. Grandes bases de datos que permitían un juego perfecto fueron desarrolladas para estos finales, a pesar de que las bases de datos no entregaban ningún tipo de conocimiento ajedrecístico acerca de cómo jugar esta etapa del juego.

En Diciembre de 1985 H.J.J. Nefkens de la Universidad de Delft describía en el ICCA Journal cómo un computador de 64K de memoria podía construir una base de datos para el final de ♔♕v/s ♔. Primero, el rey débil podía ser ubicado en cualquiera de las 64 casillas del tablero, si bien por simetría sólo 10 casillas deberían ser consideradas. Luego de ubicado el rey débil existen al menos $64x64$ formas de ubicar las 2 piezas restantes. Por lo tanto, el final de ♔♕v/s ♔debe contener $10x64x64 = 40{,}960$ posiciones. Para cada posición la base de datos debe saber cómo gana el bando fuerte y en cuantos movimientos. La base de datos es creada mediante un proceso denominado "Análisis Retrógrado" que consta de los siguientes pasos:

1. Cada una de las 40.960 entradas es incluida en un arreglo. Se analiza la legalidad de cada una de las posiciones (aquellas donde ambos reyes ocupan la misma casilla o bien una adyacente al del otro son ilegales).

2. Luego, todas las posiciones de mate en una se determinan observando aquellas posiciones en las cuales el rey débil se encuentra en jaque, se observa si éste posee un escape. Si no existe escape (jaque mate) entonces se buscan todas las posiciones legales en donde el bando fuerte puede mover y obtener la posición inicial.

3. partiendo con $n = 1$ recursivamente determinar todas las posiciones que son mate en $(n + 1)$ movimientos desde las posiciones que son mate en $n$.

Con esta técnica se ha demostrado que por ejemplo en el peor de los casos el mate del final ♔♕v/s ♔se logra en 10 jugadas. En otros casos como Rey y Torre vs Rey el mate se logra en 16 jugadas. Construir bases de datos para finales con peones es posible considerando los finales elementales mostrados con anterioridad. En particular los análisis de este tipo de finales no terminan necesariamente en mate o tablas sino que más bien se transforman en otro final al coronar uno de los peones.

Ken Thompson [71] ha sido el líder en el desarrollo de bases de datos que permiten juego perfecto en algunos finales. Su más reciente trabajo han estado relacionados con los finales de 5 piezas, incluyendo en particular el ♔♗♗v/s ♔♘, el ♔♕♙v/s ♔♕y el ♔♖♙v/s ♔♖. En 1977 Thompson se presentó en el Campeonato Mundial de Computadoras de Toronto con una base de datos que jugaba en forma perfecta el final de ♔♕v/s ♔♖. La impresión por este desarrollo fue tal que un match entre el campeón norteamericano Walter Browne y BELLE fue realizado para demostrar la perfección de la base de datos. Browne tenía 2 horas y media para realizar 50 movimientos mientras que BELL respondía en forma casi automática. En la primera partida la máquina logró las tablas con la torre, mientras que en la segunda el jugador americano luego de gran preparación logró ganar el final.

Los trabajos de Thompson produjeron a principios de los 80 un shock en el mundo del ajedrez al demostrar que ciertos finales requerían un número mayor de 50 movimientos sin avances de peón y sin cambios para ganar (el final de ♔♖♗v/s ♔♖). El descubrimiento hizo cambiar la regla de los 50 movimientos respecto de este final.

Otros notables que han trabajado en este tópico han sido Vladimir Arlazarov y Aron Furter quienes a fines de 1970 presentaron una base de datos del final de Rey, Torre y peón vs Rey y Torre, así como la de Rey, Dama y peón vs Rey y Dama.

Actualmente con prácticamente todos los finales de 5 piezas resueltos la atención está centrada en los finales de 6 piezas. Cada pieza adicional incrementa el número de posiciones en la base de datos en un factor cercano a 64. Existen 256 finales de seis piezas con 4 piezas blancas y 2 negras y otros 256 con 3 piezas por bando y aún otros 256 con 2 piezas blancas y 4 negras. Muchos de estos 768 finales no son de interés debido a la diferencia de material pero hay muchos que sí lo son como ♔♖♗v/s ♔♘♘, ♔♕♘v/s ♔♕♘♖♖, ♔♕♗v/s ♔♖♘,etc. para un resumen de los resultados obtenidos en estos finales ver la tabla 7.2

**Bases de Datos de Aperturas**

Así como las bases de datos de finales resultaron ser un gran avance en el nivel de juego de los programas en esta etapa, la etapa inicial del juego conocida como Apertura tiene la característica de ser bastante medible en su cantidad de movimientos, pudiendo éstos estar tabulados en alguna base de datos.

Los primeros libros de aperturas correspondieron simplemente a tablas de trasposición en donde se mantenía información acerca de movimientos y posiciones a las cuales el programa podía llegar en sus primeros movimientos. La idea de estos libros es el incrementarlos con información en el tiempo, ya sea con nuevas variantes o bien conclusiones obtenidas por los mismos programas.

Existen datos interesantes acerca de algunos programas que incluyen en sus libros de aperturas partidas de jugadores en particular con tal de adaptar el estilo de juego de la máquina. Feng Hsu en su texto “Behind DeepBlue” [8] menciona su preferencia por mantener en el libro de aperturas de la computadoras partidas de Anatoli Karpov, la cuales

tenían aperturas que se adaptaban al estilo de juego de la máquina. De allí, explica Feng, la costumbre del computador de jugar por ejemplo la defensa caro-kann con las negras.

Con el aumento en capacidad de memoria y disco de las máquinas, los libros de aperturas han llegado a ser gigantes, manteniendo actualmente análisis hasta sobre la jugada 20 y siendo siembre factibles de ser mejorados durante el tiempo.

### 5.4.5. Uso eficiente del tiempo de reflexión

En cada movimiento un programa de ajedrez debe tomar una importante decisión: ?'cuanto tiempo asignar al análisis de la posición? Posiciones de fácil determinación requieren un menor tiempo que aquellas que son de difícil decisión o cálculo. Por supuesto, decidir que es una posición de "fácil determinación" y una de "difícil determinación" es un problema, así como el determinar qué es "poco" y "mucho" tiempo!. Afortunadamente se han desarrollado buenos algoritmos para tomar estas decisiones. Robert Hyatt describe como CRAY BLITZ administra su tiempo en su paper publicado en 1984 [51].

Las partidas son generalmente jugadas bajo dos tipos de control de tiempo. Lo más convencional son 2 horas para las primeras 40 jugadas y luego 1 hora para cada 20. Esto entrega un promedio de 3 minutos de reflexión por jugada. AL principio de cada movimiento el programa determina el número de movimientos para llegar al control de tiempo, digamos $N$, y el tiempo restante para el control $T$. El cociente $N/T$ entrega el tiempo base $B$ permitido para ese movimiento. Un tiempo máximo es definido para el movimiento, típicamente $4B$ si bien esto debe ser ajustado en las cercanías del control. El programa inicia entonces su búsqueda de profundidad iterativa. Luego de buscar en un tiempo $B/2$ el programa decide cuánto más buscar tomando en consideración los scores obtenidos para cada variante. Si éstos indican que la computadora tiene un movimiento fácil como una recaptura o promoción la búsqueda finaliza. Esto salva una magnitud de tiempo $B/2$ para el siguiente movimiento. Si el programa decide continuar entonces lo hace hasta llegar al limite de tiempo $B$, donde nuevamente se realiza la decisión de cuánto más buscar. En este momento, la máquina se detiene si el score de la variante principal no ha sido cambiado en la búsqueda. Si ha cambiado, indicando problemas en el horizonte, entonces el programa continúa hasta el límite de tiempo $2B$. Al llegar ahora a ese límite el programa debe nuevamente decidir cuánto más buscar. En este momento el termina la búsqueda a menos que se encuentre en una situación muy inferior. La búsqueda siempre termina al llegar al límite máximo $4B$.

Los últimos encuentros entre humanos y computadoras han impuesto un límite de tiempo fijo para toda la partida. Por ejemplo los matches entre Kasparov y DeepBlue se jugaron al límite de 1 hora y 30 minutos para a partida. Esto hace que el algoritmo descrito anteriormente sea más complejo dado que no es claro cuánto durará la partida. Algunos programas asumen que el partido durará otros 30 movimientos con lo cual imponen $N = 30$. Las computadoras están programadas para "pensar" durante el tiempo de su oponente. Apuestan a que éste realizará el movimiento predecido en la variante principal y calculan la futura réplica. Si la réplica del oponente es la esperada la máquina puede salvar tiempo respondiendo en forma inmediata o bien continuar la búsqueda. Si no, simplemente reinician la búsqueda. Los

mejores programas adivinan el movimiento de su oponente con una probabilidad del 50 % con lo cual logran una ganancia de tiempo del 50 %. Esto es equivalente a correr un programa en una máquina con 50 % de mayor capacidad.

# Capítulo 6

# Comparación de Avances en Función de Hardware y Software

Las primeras máquinas que lograron jugar al ajedrez realizaron funciones mínimas tales como resolver problemas de jaque mate en dos movimientos o bien mediante reglas de posición resolver finales de mates elementales. Estas máquinas funcionaban mediante componentes mecánicos, los cuales mediante sistemas de imanes o ranuras lograban descifrar la situación de la posición en el tablero y generar el movimiento adecuado.

El paso del tiempo ha dado lugar al desarrollo computacional en máquinas electrónicas de alta velocidad y hardware sofisticado, cuyo progreso ha sido meteórico en comparación a su corta existencia. Técnicas refinadas de uso de algoritmos y maquinaria más especializada capaz de realizar miles de operaciones en pocos segundos dieron nacimiento a máquinas de ajedrez que ya podían jugar al nivel de candidatos a maestro. Este avance dividido tanto en software como en Hardware es lo que analizamos en el presente capítulo, comparando eficiencia, efectividad y avances logrados en el tiempo.

## 6.1. Avances de Hardware

En los últimos años el avance del Hardware aplicado a computadoras que juegan ajedrez ha tenido un enorme impacto en el desarrollo de esta área. Estos avances incluyen procesadores de mayor velocidad, mayor capacidad de memoria, mayor largo de "palabras de bits" y circuitería de mayor confiabilidad, diseño de hardware específico para ajedrez y multiprocesadores.

### 6.1.1. Procesadores de mayor velocidad

La velocidad de unidades de procesamiento se ha incrementado dramáticamente debido principalmente a la evolución de la tecnología de semiconductores. Las computadoras han progresado desde ejecutar aproximadamente 10.000 instrucciones por segundo ,en la época de los primeros programas,a aproximadamente 1.000.000.000 en la actualidad. El aumento de velocidad ha sido del orden de magnitud de 100.000 veces. Imaginemos comparativamente si la velocidad de transporte hubiese aumentado en esa magnitud durante el mismo periodo de tiempo. Los aviones volarían a 100.000.000 millas por hora; un viaje al sol tomaría cerca de 1 hora de vuelo y un viaje ala estrella más cercana Alpha Centauri, la cual está a 4,5 años luz de distancia, tomaría cerca de sólo 70 años.

En los años venideros puede esperarse un continuo crecimiento en la velocidad de los procesadores siendo razonable esperar que en los próximos 10 años el incremento será nuevamente cercano a las 100 veces. Ya a inicios del año 2000 las computadoras son 10.000.000 de veces más rápidas que las utilizadas en los primeros programas de ajedrez. Un viaje a Alpha Centauri tomaría ahora menos de 1 año!. Cada incremento en 6 veces la velocidad de la máquina le otorga la posibilidad de realizar una búsqueda en aproximadamente 1 nivel extra de profundidad. Un incremento de 100 veces otorga aproximadamente 2,5 niveles extras de profundidad. Con DEEP THOUGHT realizando búsquedas en una profundidad cercana a los 9 o 10 niveles en el año 1989 no fue sorpresa el hecho de que DeepBlue superara el nivel de profundidad de 12 movimientos en 1997.

El efecto del incremento en la velocidad del hardware en la performance de los programas de ajedrez ha sido estudiado durante varios años. La historia ha demostrado que los ratings Elo de las máquinas han aumentado cerca de 200 puntos por cada nivel adicional de búsqueda logrado. Esto se muestra en la tabla, donde las profundidades de búsqueda, años y ratings han sido levemente redondeados con tal de demostrar el efecto de los “200 puntos”. Lo que la tabla no muestra es que muchas otras mejoras han sido realizadas a los programas con tal de lograr búsquedas con mayor profundidad, y que correr un programa en una máquina 6 veces más veloz no aumenta directamente la fuerza del programa en 200 puntos de rating Elo.

A fines de los 70, Thompson hizo jugar a varias versiones de BELLE con tal de medir la performance como función sólo de la velocidad con tal de medir los efectos de otras mejoras. El midió la performance como función de la profundidad de búsqueda, pero velocidad y profundidad son parámetros íntimamente relacionados. En sus experimentos, Thompson varió la profundidad de búsqueda desde 3 a 9 niveles. Sus resultados demostraron que para niveles de rating entre 1400 y 2000 existían 200 puntos de diferencia por cada nivel extra de profundidad. Sin embargo, para ratings por sobre los 2000 puntos y con búsquedas sobre los 7 movimientos, existía un decremento en los 200 puntos teóricos de diferencia. Un estudio posterior de Newborn, el cual consideró búsquedas entre 3 y 13 movimientos llegó a la misma observación, pero adicionalmente demostró que los ratings incrementan a medida que la búsqueda también aumenta.

| Profundidad de Búsqueda | Año | Programa | Rating |
|---|---|---|---|
| 5 | 1972 | CHESS 3.5 | 1600 |
| 6 | 1975 | CHESS 4.0 | 1800 |
| 7 | 1978 | CHESS 4.7 | 2000 |
| 8 | 1980 | BELLE | 2200 |
| 9 | 1986 | HITECH | 2400 |
| 10 | 1989 | DEEP THOUGHT | 2600 |
| 12 | 1997 | DEEP BLUE | 2800 |

Cuadro 6.1: Evolución de los ratings de programas de acuerdo a su capacidad de búsqueda

### 6.1.2. Capacidad de Memoria

Memorias de mayor capacidad son el resultado de la rápida evolución en tecnología de semiconductores. Los primeros programas de ajedrez corrían en máquinas que utilizaban memorias de base magnética. A inicios de 1970 aparecen las memorias realizadas en base a semiconductores utilizadas en la serie de computadoras IBM 370. Así como la velocidad de los computadores se incrementó en un factor de aproximadamente 100.000, la capacidad de memoria creció en una proporción similar. Este hecho es particularmente importante en programas que utilizan tablas de transposición. A medida que aumenta la velocidad de la computadora memorias de capacidad proporcionalmente mayor son necesarias para mantener la cantidad extra de posiciones que son buscadas.

Así como se espera tener mayores incrementos en la capacidad de procesadores en los próximos años, no es un abuso decir que la capacidad de memoria continuará creciendo de manera impresionante. Memorias de mayor capacidad podrán ser utilizadas por programas con tablas de hash de mayor envergadura, las cuales mantendrán la información en forma permanente.

### 6.1.3. Tamaño de Palabras

El tamaño de palabras de bits ha logrado crecimientos bastante importantes sobretodo a nivel de pequeñas computadoras. Los procesadores Intel 80386 y 80486 están basados en palabras de 32 bits. En el otro extremo se utilizaron palabras de largo 64 bits, diseñadas especialmente para máquinas de ajedrez dado que el tablero, para buena fortuna de los programas ajedrecistas, posee exactamente 64 casillas. Palabras de largo aún mayor significaría que una mayor cantidad de información puede trasladarse dentro de la computadora en un ciclo de reloj pero su implementación es algo que involucra mucho costo. No es probable el que en el futuro las computadoras usen palabras de tamaño mayor a los 64 bits, pero se puede predecir que tendrán mayor capacidad de flexibilidad en la manipulación de palabras de rango 1 a 64 bits.

### 6.1.4. Computadoras de menor tamaño

Computadoras que en alguna ocasión ocupaban una pieza completa ahora ocupan sólo un escritorio siendo considerablemente más poderosas que las anteriores. Las unidades de proceso de las primeras computadoras de ajedrez ocupaban varias cabinas de tubos de vacío ocupando grandes volúmenes. Los microprocesadores de hoy en día ocupan menos de una pulgada siendo mucho mas poderosos.

### 6.1.5. Hardware Dedicado

El hardware de propósito especial para máquinas que jugasen ajedrez apareció por vez primera a mediados de los 70 iniciando un rápido proceso de sofisticación y mejoras. El primero de estos sistemas fue diseñado utilizando componentes que podrían ser clasificados como circuitos integrados de mediana escala. Cada chip estaba constituido por 10 a 100 componentes lógicas. Los circuitos más recientes han sido diseñados con tecnología VLSI. Uno de los ejemplos más populares fue el diseño de la computadora Deep Thought, la cual tanto en su generador de movimientos y función de evaluación poseían componentes a nivel de Hardware.

El suceso del hardware dedicado para ajedrez ha estimulado su utilización en otras problemas de Inteligencia Artificial dependientes de búsqueda.

El primer hardware de ajedrez fue desarrollado en el año 1976. El primer objetivo de su diseño fue atacar la generación de movimientos, proceso que involucraba mayor cantidad de tiempo a los programas de ajedrez. La idea era entregar al hardware una posición y que éste retornase una lista con movimientos en la misma manera en la cual un multiplicador recibe dos argumentos y retorna un producto. Posterior a la generación de movimientos el trabajo realizado por un programa en la función de evaluación es el siguiente en mayor consumo de tiempo y un número importante de evaluadores a nivel de hardware fue construido.

En 1976 Joe Condon y Ken Thompson [42] desarrollaron un prototipo de generador de movimientos a nivel de hardware constituido por 25 chips para el programa de ajedrez BELLE. Este programa presentaba un nivel de búsqueda de 200 posiciones por segundo.

En 1978 un segundo prototipo de BELLE con un hardware constituido de 325 chips lograba analizar 5000 posiciones por segundo. El hardware cumplía las funciones de generador de movimientos, función de evaluación y mantenedor de las tablas de transposición.

Dos años más tarde el tercer prototipo de BELL estaba constituido por 1700 chips logrando un nivel de búsqueda de 120.000 posiciones por segundo. El generador de movimientos presentaba 64 circuitos transmisores y 64 circuitos receptores, cada uno correspondiente a las 64 casillas del tablero siendo capaz también de dar un orden al listado de movimientos generado (fundamental para el algoritmo de poda alfa-beta). La función de evaluación fue también diseñada con un conjunto de 64 circuitos. Utilizaba 8 ciclos para evaluar “clavadas, descu-

biertos, ataques, defensas y aspectos relativos al control de casillas". Una segunda parte del hardware de evaluación utilizaba otros 8 ciclos para analizar la estructura de peones. Este módulo detectaba peones pasados, bloqueados, retrasados, aislados y columnas abiertas o semi abiertas. También existía un evaluador estático de la posición, el cual consideraba el material como factor principal, si bien factores como resguardo del Rey también eran tomados en consideración. Como factor importante, el prototipo incluía una tabla de transposición de 128K. Esta version de BELLE obtuvo varios logros, entre otros el campeonato mundial de Linz en 1983 y el título de Maestro de la Federación de Ajedrez de Estados Unidos.

En 1985 un grupo de estudiantes de la Universidad de Carnegie Mellon bajo la supervisión de Hans Berliner desarrollaron un generador de movimientos y circuitería de propósito especial para evaluar posiciones a gran velocidad. Eveling y Palay desarrollaron la mayor parte del circuito. Su programa, HITECH logró realizar búsquedas de 200.000 posiciones por segundo convirtiéndose en el más fuerte de esos años. Una de las fortalezas de su diseño radicaba en que parámetros extras podían ser añadidos a la función de evaluación siendo computados en paralelo con los ya existentes lo cual no disminuía la velocidad de cálculo. En otras palabras, era factible añadir inteligencia a la función de evaluación sin pérdida de velocidad de cálculo.

En 1986 un segundo grupo de Carnegie Mellon siguiendo las ideas de BELLE y en cierto grado de HITECH desarrollaron un programa de mayor fortaleza utilizando tecnología VLSI. El sistema "Deep Thought" desarrollado por Feng-hsiung Hsu junto con Thomas Anantharaman y Murray Campbell logró ganar los campeonatos ACM de 1987 y 1988 con una capacidad de búsqueda de 700.000 posiciones por segundo.

Luego en 1993 Feng-hsiung Hsu mejoró el desarrollo a nivel de Hardware de su programa creando una nueva versión aún más poderosa denominada "Deep Blue" la cual hasta el día de hoy es reconocida como la más fuerte máquina de ajedrez cuyo nivel de búsqueda llegó a la cifra de 2.000.000 de posiciones por segundo. La fortaleza en el diseño de hardware dedicado de Deep Blue se basó en el diseño de la circuitería de transistores, según relata el mismo Feng en [8]

### 6.1.6. Sistemas Multiprocesadores

A inicios de 1981 los sistemas de multiprocesadores fueron introducidos en máquinas que jugaban al ajedrez. El objetivo inicial era lograr un aumento de la velocidad de proceso en un factor de $N$ veces con $N$ procesadores. El problema principal de este tipo de desarrollo fue el cómo dividir el árbol de búsqueda de variantes con tal de mantener a todos los procesadores ocupados en todo el tiempo de búsqueda evitando además búsqueda redundante.

Otro problema era la realización de debugging en este tipo de programas. Los procesadores funcionan en forma asíncrona y eventos que ocurren primero en un procesador y luego en otro pueden suceder en otro orden en otra simulación.

El programa OSTRICH fue el primero en participar en una competencia oficial de com-

putadoras utilizando esta tecnología. Cinco computadoras serie Data General 16-bit Nova estaban conectadas mediante un paquete de comunicación de alta velocidad. En años subsiguientes 8 computadoras DG fueron utilizadas logrando un incremento de la velocidad de cálculo en un orden 5. Otros programas famosos que utilizaron multiproceso fueron Cray-Blitz (1983 y 1984, procesador CRAY X-MP, 2 y 4 procesadores), CHESS CHALLENGER (1986, 20 microprocesadores 8086) y Deep Thought (1989, 3 procesadores de tecnología SUN y VLSI). DeepBlue para su match contra Kasparov utilizó un total de 2 frames RS6000 SP albergando 15 computadoras RS/6000 cada uno. Cada nodo contenía dos circuitos de ajedrez con 8 procesadores cada uno, lo que otorga en total 8x2x15x2 = 480 procesadores en el sistema completo.

Otra aplicación interesante de las máquinas multiprocesadores fue realizada por Thompson y Stiller a mediados de los 70, quienes desarrollaron mediante este tipo de hardware las bases de datos de finales para computadoras de ajedrez [71].

El procedimiento de sincronización y búsqueda con multiprocesadores se encuentra bastante detallado en [14]

## 6.2. Avances en Software

En términos de herramientas para el desarrollo de programas que jugasen al ajedrez en el día de hoy disponemos de un mejor y más eficiente software para editar, compilar, armar, enlazar y debugear programas.

Es algo difícil el cuantificar cuánto mejor es el software actual respecto al de antaño, puesto que no se han realizado análisis a partir de un parámetro de software dado, si bien bajo cualquier factor de medición se podría asegurar que éste no ha mejorado en los niveles que lo ha hecho el hardware. Algo sorprendente en este campo es la carencia del desarrollo de un lenguaje de programación de propósito especial para ajedrez.

A pesar de esto, podemos mencionar algunas mejoras realizadas en la programación de los principales módulos de los programas que juegan ajedrez, vale decir, generador de movimientos, función de evaluación y motor de búsqueda.

### 6.2.1. Generador de Movimientos

La forma inicial de programar un generador de movimientos fue el generar mediante fórmulas matemáticas los movimientos legales de cada pieza sobre el tablero, obteniendo todas las posibilidades con tal de entregárselas como una lista al software de búsqueda. Esta propuesta fue mencionada por vez primera en el paper de Shannon [68] y se aplicó a prácticamente todos los programas de la época.

La idea inicial fue el que el programa generara sólo los mejores movimientos con tal

de reducir drásticamente el árbol de variantes (estrategia “B”, según la nomenclatura dada por Shannon) pero los resultados distaron de ser positivos puesto que el problema principal relacionado con este proceso fue que en las búsquedas en profundidad esta forma de generar los movimientos tomaba un tiempo excesivo, lo cual hacía muy lento el proceso global, motivo por el cual se buscaron otras formas de programar la generación de movidas en base a operaciones que la computadora pudiese realizar más rápidamente.

Contrariamente a lo que podría creerse, la generación de movimientos es una de las tareas de mayor consumo de tiempo en un programa de ajedrez dado que deben realizarse una gran cantidad de operaciones para poder obtener todas las posibilidades. La limitada capacidad de hardware hacían que este proceso fuese muy costoso en tiempo de procesador a pesar de que las primeras funciones de software generador de movimientos fuesen eficientes.

Sólo hasta principios de 1970 (gracias a la presencia de hardware y ambientes de desarrollo de mayor capacidad) se utilizó la técnica de los mapas de bits (bit-boards)la cual significó un gran avance en este proceso del juego de la máquina dado que se redujo la complejidad de operaciones a aquellas que son básicas para la máquina. A pesar de este avance, era clara la necesidad de implementar fuera del software esta función del programa, dado que la necesidad de hacer búsquedas más rápidas y profundas se basaba en un generador de alta velocidad, cosa que era muy difícil lograr a nivel de hardware.

Las mejoras en esta función del programa vinieron principalmente del lado del desarrollo de hardware específico para la generación de movimientos. En 1977 el programa “Belle” fue el primero en utilizar circuitos digitales para la generación de movimientos logrando aumentar su velocidad de búsqueda de 200 a 160.000 posiciones por segundo. El generador utilizado en Belle sirvió como punto de partida para máquinas más poderosas. El computador que derrotó a Kasparov en 1997, DeepBlue, tenía 30 procesadores IBM RS-6000 SP acoplados a 480 chips. Esta máquina fue capaz de lograr velocidades computacionales de 200 millones de posiciones por segundo.

Las característica principal de estos generadores a nivel de Hardware es el poder caracterizar a las casillas de origen y destino mediante transmisores y receptores respectivamente, para luego generar mediante un árbol de prioridades los movimientos ordenados de acuerdo a criterios de capturas, jaques, etc. [38]. La principal ventaja entre el generador de movimientos de DeepBlue y Bell es que el primero solucionó el problema de generar en primer orden los movimientos de jaque.

En la actualidad la utilización de mapas de bits es prácticamente universal en todos los programas de ajedrez. Las mejoras se ven principalmente en los tipos de mapas generados de acuerdo al tipo de movimientos buscados (reglas de mapas de bits). El principal desarrollo en este tema es a nivel de hardware en donde los avances se han visto en el orden de jugadas entregado en la generación de los movimientos. En el último año se han desarrollado también tarjetas de hardware específicas para implementación en computadoras personales (Field Programable Gate Arrays) las cuales han sido utilizadas en forma experimental.

### 6.2.2. Técnicas de Búsqueda

Las técnicas de búsqueda en el ajedrez han sido probablemente la parte más desarrollada en términos de investigación para la mejora de algoritmos dentro del proceso del juego. Es acá en donde el programa realiza el mayor esfuerzo en recolectar información suficiente para decidir por un movimiento.

Los algoritmos desarrollados para este fin han sido variados. La primera propuesta, presentada por Shannon, fue la búsqueda por fuerza bruta a profundidad fija, examinando todas las continuaciones posibles hasta cierto límite de movimientos. Esta búsqueda se realizaría mediante el algoritmo Minimax.

Hasta principios de los 70 los programas de ajedrez seguían basándose en los modelos de Shannon para realizar su proceso de búsqueda. Hasta ese momento el diseño de programas de ajedrez se focalizaba principalmente en el orden de los movimientos con tal de lograr la mejor "poda" de variantes: jaques, capturas, "movidas asesinas", amenazas y avances de peones pasados. Categorizando los movimientos en distintos grupos mediante amenazas tácticas (de corto plazo) o estratégicas (de largo plazo) se podía asegurar el que las variantes "forzadas" serían analizadas primero. Esta técnica lograba generar cortes o reducciones en el árbol de búsqueda para aquellas variantes que se clasificaban como innecesarias de analizar.

También durante la década de 1970 la noción de búsqueda iterativa fue refinada y sometida a prueba. Con un enfoque de completitud y uniformidad las búsquedas eran generalmente hechas a una profundidad fija (en forma iterativa dependiendo del tiempo de reflexión restante). El uso del tiempo para controlar la búsqueda tomó también un carácter de suma importancia, dado que ofrecía la flexibilidad de confirmar si el movimiento encontrado en la anterior iteración era efectivamente el mejor. Esto hizo que de alguna manera la búsqueda selectiva (estrategia "B" de Shannon) cayera en desuso, debido a la dificultad de su implementación en las computadoras.

Resultó que la seguridad de buscar en todos los movimientos posibles otorgó mejores resultados que el descarte de variantes basado en su baja relevancia. La búsqueda selectiva vio prácticamente su muerte en esta década, si bien fue la estrategia líder de los 60 con el programa MackHack, el cual estaba implementado con un selector de movimientos plausibles [46]. A finales de la década los beneficios de la búsqueda de posiciones estables con profundidad variable fue clara, y esto llevó a la preponderancia de los programas de estrategia tipo A los cuales utilizaron búsquedas basadas en distintas etapas.

Con el aumento en velocidad de los procesadores pudieron realizarse búsquedas más profundas causando a principio de los 80 un interés por realizarlas en etapas. La idea fue realizar una primera búsqueda a profundidad fija de varios movimientos seguida de una búsqueda selectiva y terminando con una búsqueda de posiciones estables a partir de movimientos como capturas o respuestas a jaques. Este modelo de búsqueda demostró ser a la vez bastante robusto. de todas maneras, este tipo de búsqueda en etapas no reflejaba adecuadamente la noción intuitiva de que las extensiones en la profundidad de búsqueda deben ser selectivas en vez de uniformes, aplicándose en variantes forzadas. Era claro que la extensión de un

movimiento debía realizarse cuando uno de los bandos se encontraba en jaque. Un idea similar era extender la búsqueda cuando un jugador tuviera sólo un movimiento legal posible, pero lo que realmente se deseaba buscar era una extensión en situaciones en que cualquiera de los dos jugadores tuviese sólo un movimiento "sensible" (por ejemplo, una captura). Esto implicaba algún tipo de pre-poda con tal de detener la expansión de identificación de malos movimientos. Dos ideas de pre-poda bastante trabajadas variantes fueron los cortes a raz y en vano ("razoring" and "futility" cutoffs). Por ejemplo, si justo antes del horizonte de búsqueda el score está sobre el límite beta del minimax para el bando que juega, cortar inmediatamente (corte al raz). Alternativamente, si el movimiento actual está bajo el límite alfa y los factores posicionales no tienen el potencial para aumentarlo, descontinuar la variante (corte en vano).

Métodos de búsqueda variable como estos fueron refinados para asegurar el control de las extensiones, sobretodo en las búsquedas sin límite (por ejemplo, jaques perpetuos) y en cambios drásticos de equilibro material (por ejemplo en promociones de peón). otro factor primera preocupación para los programadores de la época fue el cómo luchar contra el efecto horizonte. Como conclusión era clara la necesidad de algún tipo de búsqueda selectiva de profundidad variable.

En los 90 ya estaban bien entendidos los criterios para extender en forma automática la búsqueda, con lo cual el foco de atención se centralizó nuevamente en la búsqueda selectiva con pre-poda (forward pruning), una idea que había sido repetidamente intentada en décadas pasadas pero con resultados contradictorios. Una generalización de las podas de corte de navaja (razoring) y de valor nulo(futility) fue el uso del "'movimiento de paso" en la búsqueda de posiciones estables [32]. La esencia de la tercera etapa de búsqueda es considerar sólo movimientos de captura, algunos jaques y movimientos tácticos que alteran considerablemente el equilibrio material. El uso de un movimiento de paso (que en el fondo significa permitir al adversario mover dos veces) asegura que una amenaza del rival puede ser encontrada más rápidamente. Técnicas de movimiento de paso fueron la pre-poda utilizada en inicios de los 90. La implementación de esta pre-poda con mayor suceso fue desarrollada por Goetsch y Campbell [44].

En el tiempo en que los métodos de "movimiento de paso" (null-move) eran descritos se comenzó a trabajar en otras formas de variar la profundidad de búsqueda. Era claro ya que las respuestas a jaques no debían contar como un movimiento que adelanta un paso al horizonte de búsqueda. para esto, una extensión automática puede realizarse por cada movimiento forzado pero con respuesta única. A principios de los 90 la noción de "Extensiones Singulares" fue introducida e implementada [30]. Movimientos que son sustancialmente mejores otros de su mismo nivel de búsqueda son analizados con un nivel más de profundidad con tal de reducir el riesgo del efecto horizonte.

En el pasado la búsqueda selectiva fue un método de alto riesgo pero con gran potencial de desarrollo. En la actualidad los métodos de búsqueda variable son un área de desarrollo activo. Recientemente algunos métodos de poda existentes fueron mejorados por Heinz [49], quien los generalizó con tal de realizar podas en niveles por sobre el horizonte mientras que Marsland [58] realizó estudios sobre las búsquedas en nodos factibles de ser podados. Ambos métodos lograron mejoras en el nivel de juego y están siendo empleados por varios de los más

fuertes programas de la actualidad.

Con el incremento en espacio de memoria, renovado interés tuvo el desarrollo de algoritmos de búsqueda desechados con anterioridad como el SSS* de Stockman y el B* de Berliner [34] y métodos combinados como el DUAL* [57] los cuales eran computacionalmente eficientes pero lentos de ejecutar.

### 6.2.3. Función de Evaluación

Las primeras propuestas de función de evaluación estaban orientadas a tener una evaluación estática de la posición basada fundamentalmente en el concepto de material. Rápidamente se captó que esta evaluación no era suficiente, considerando que en el ajedrez existen factores estructurales los cuales afectarán a largo plazo el curso de la partida, por lo cual es muy probable que el programa no encuentre las consecuencias de esta situación en su búsqueda en profundidad. Estos aspectos, denominados "Posicionales" debieron incluirse en la función de evaluación con tal de dar al programa una mejor capacidad de dar un puntaje a la posición resultante de acuerdo a sus parámetros dentro de la función de evaluación.

Los primeros programas incluyeron algunos parámetros posicionales básicos, los cuales tenían un peso importante en la evaluación de la posición pero no igualable al peso que poseía el factor de material. El problema entonces de dar un adecuado "peso" a cada parámetro de la función de evaluación, y en qué momento de la partida darlo a uno u otro parámetro constituía un problema denominado carencia de "conocimiento ajedrecístico" cuya solución estaba en captar las impresiones de los jugadores humanos "expertos" y asimilarlas en el programa.

La inclusión de un mayor "conocimiento ajedrecístico" en los programas fue logrado mediante el trabajo conjunto entre programadores y maestros de ajedrez. Desafortunadamente, costó bastante el poder combinar de manera eficiente una idea de función de evaluación lo más completa posible con la mayor cantidad de parámetros de medición, pero a la vez muy rápida en su capacidad de cálculo.

Nuevamente acá los avances en hardware permitieron crear funciones de evaluación más poderosas y cuya utilización no significaba un alto costo de CPU. El problema entonces se redujo a cómo dar pesos adecuados a cada parámetro de la función con tal de que la combinación de éstos entregue una evaluación al menos similar a la que concluiría un maestro de ajedrez.

A este respecto resulta destacable el trabajo realizado por Hsu,Campbell y Anantharaman [31], quienes testearon su programa DeepBlue con cerca de 900 posiciones particulares de partidas de Grandes Maestros. Durante el análisis por parte de la máquina de estas posiciones, los programadores iban ajustando manualmente los pesos de cada parámetro de la función de evaluación con tal de que el programa encontrara el movimiento seleccionado. Este ajuste de "pesos" resultó ser un notorio avance en la performance de la función de evaluación de este programa. Respecto de esto una cita de Anantharaman en el ICCA Journal

describiendo esta metodología :

"*La función de evaluación del software de Deep Thought es ajustada contra una base de datos de partidas de maestros de ajedrez. La diferencia de evaluación entre el movimiento elegido por el programa y el maestro es minimizada. La performance del programa de ajedrez con la función de evaluación ajustada fue medida en forma experimental. El programa jugó varios matchs de 500 partidas con diferentes ajustes contra un programa fijo. Los resultados mostraron que un 98 % de la performance de la función de evaluación puede ser mejorada en aproximadamente una semana ajustándola a partir de la información de partidas de la base de datos*".

Este método lineal de ajuste de los pesos de parámetros de la función de evaluación fue combinado con extensivas sesiones de partidas entre programas y Grandes Maestros, en donde los experimentados humanos entregaban sus observaciones acerca de las características de juego de la máquina.

Hsu relata en [8] acerca de la necesidad de realizar sesiones de entrenamiento entre DeepBlue y el GM Joel Benjamin, en donde el campeón norteamericano aportó con excelentes observaciones acerca del comportamiento de la evaluación de la máquina, qué cosas valoraba por sobre otras y que ajustes podían realizarse con tal de mejorar su "concepto" ajedrecístico.

Un excelente ejemplo de esta técnica de refinamiento se vio en la partida 2 del match entre DeepBlue y Kasparov en 1997 (ver anexo de partidas). En posiciones en donde se producían "tensiones de peones" (peones que pueden ser cambiados despejando columnas) la máquina tendía a realizar el cambio de peones con tal de ocupar luego la columna abierta generada por ese cambio con las torres. Esta actitud del programa debía mejorarse puesto que en vez de cambiar peones y ocupar la columna es por lo general mucho mejor primero ocupar la columna "candidata a abrirse" con las Torres y luego proceder a los cambios de peones. El movimiento 24 de DeepBlue en la citada partida fue consecuencia de esta observación.

En la actualidad las mejoras introducidas en la función de evaluación de los programas apuntan a los trabajos desarrollados con redes neuronales y algoritmos genéticos los cuales apuntan a la utilización de métodos de aprendizaje por parte del programa [52]. La apuesta es bastante interesante pues toma en consideración una de las capacidades básicas de los humanos en la mejora de su nivel de juego, el hecho de adquirir conocimientos nuevos frente a experiencia. Las técnicas apuntan a mejorar el nivel de conocimiento ajedrecístico de los programas.

## 6.3. Conclusiones del capítulo

En los 60 con una capacidad muy limitada de Hardware los intentos de implementar programas de estrategia "B" no lograron un nivel de juego mínimamente aceptable. Esto influyó que en los 70 se descartase la idea de implementar los programas en base a búsqueda selectiva tomando un cambio hacia la búsqueda de fuerza bruta.

Los 80 fueron un periodo de tremendos cambios tecnológicos con el precio de pequeñas computadoras cayendo por debajo del precio de los automóviles. Durante este periodo la velocidad de los procesadores se duplicaba cada dos años, y las computadoras de uso personal se hicieron comunes. Las capacidades de memoria crecieron también en forma considerable así como la capacidad de disco para mantener información. Esto trajo como consecuencia algunos hechos interesantes:

Primero, los programas de ajedrez migraron rápidamente de computadoras de de propósito general (mainframes) a máquinas de uso personal dedicadas a una aplicación a la vez. Segundo, los mecanismos de orden de movimientos continuaron siendo importantes con lo cual se realizaron nuevos avances respecto de ellos. Por ejemplo, el uso de "movimientos asesinos" fue cambiado por el uso de heurísticas históricas - tablas de 64x64 las cuales mantienen información de cuan a menudo ciertos movimientos han causado cortes de variantes [67]. Finalmente, la capacidad adicional de memoria y mayores espacios de dirección hicieron posible el incremento de la capacidad de las tablas de transposición con lo cual fueron utilizadas para más funciones que el sólo mantener la información de búsquedas en sub-árboles de variantes.

A ser verdad, las tablas de transposición comenzaron a ser el mecanismo más popular de orden de movimientos, guiando a la búsqueda desde una iteración a otra a través de la mejor opción encontrada por este método. Por este medio en algunas ocasiones se evita la generación de movimientos (por causa de algún corte) o aplazada hasta que sea claro que algún otro movimiento debe ser examinado.

A pesar del creciente poder computacional logrado en la década de los 80 los mejores programas estaban lejos aún del nivel de un Gran Maestro de ajedrez en un juego a ritmo de torneo, si bien ya lograban dar sorpresas en juego rápido. El fin de la década vio el nacimiento de computadores con hardware dedicado y el aumento de la confianza en el uso de tablas de transposición no sólo para apoyar la búsqueda iterativa sino que también para acelerar la búsqueda con horizonte de posiciones estables. Este periodo vio también mucho desarrollo en la producción de bases de datos de finales de 3,4 y 5 piezas, y la revisión de algunos finales de partidas que se creían tablas por la regla de los 50 movimientos [71]. El inicio de la década de los 90 contaba entonces con todas las herramientas para que las computadoras derrotaran a los Grandes Maestros.

Las continuas mejoras en hardware, la creación de máquinas dedicadas y las mejoras en técnicas de búsqueda junto con aplicación de conocimiento en la función de evaluación terminaron por lograr lo que por muchos años se esperó con ansiedad, lograr derrotar a Grandes Maestros en condiciones de torneo.

En la actualidad es claro que los avances de hardware seguirán sorprendiéndonos año a año, mientras que en temas de software aún hay campo abierto para avances específicos en temas de podas en búsquedas avanzadas y mejoras en la inclusión de conocimiento en la función de evaluación, para lo cual se están trabajando con técnicas de aprendizaje. A pesar de esto, cabe preguntarse si los avances en Hardware fueron en cierta medida la causa de una ansiosa espera por parte de los investigadores para dar mayor atención a todas las técnicas mejorables desde este punto de vista más que del software.

**Fuerza Bruta o Inteligencia?**

Las técnicas de "fuerza bruta" fueron el método por excelencia ocupado durante la década de los 80 por los programadores, quienes dejaron muy de lado mejoras a otros aspectos del programa (como por ejemplo, la función de de evaluación) como una forma de "asegurar" el recorrido por la mayor cantidad de nodos del árbol. Mucho se habló de cómo apoyar esta búsqueda por medios como selectividad de movimientos o bien podas de variantes, pero poco se habló de cómo hacer entender a la máquina inmediatamente, sin tener que profundizar en variantes, el por qué un movimiento es peor respecto de otro y así descartarlo inmediatamente, algo que los humanos establecen de una sola ojeada muchas veces sin la necesidad de calcular.

Esta idea trató de ser el primer tipo de implementación en los años 60, siendo mencionada por Botvinick en [2] como *la única y mejor manera en la cual las máquinas pueden tener éxito en ajedrez, simulando el razocinio de los grandes maestros.* A pesar de la buena intención de esta medida, su fracaso radicó en la dificultad de implementar una selectividad previa de movimientos y la insuficiente capacidad de los programas de evaluar con precisión una posición, errando muy a menudo en las aseveraciones y cometiendo errores de principiante. Ante este escenario tan desalentador resulta casi lógico entender que luego de las notables mejoras en máquinas desarrolladas bajo la estrategia de fuerza bruta ésta fuese la forma de desarrollo de programas durante la siguiente década.

Podríamos entonces plantear el que los programadores prefirieron sacrificar el hacer a las máquinas más "inteligentes" por aprovechar la capacidad de cálculo de ellas? Esto es cierto en el sentido de que se desecharon técnicas propias de la mentalidad humana al jugar ajedrez, pero por otro lado se prefirió potenciar la mayor capacidad de los computadores de entonces en el rápido calculo de miles de operaciones lo cual trajo mejores resultados en el juego de los programas.

En cierta medida la velocidad empezó a ser un parámetro de mayor consideración que la inteligencia y sapiencia ajedrecística de la máquina, y la diferencia entre una y otra se daba en base a esta mayor capacidad de ver movimientos futuros. Resultó curioso entonces el fenómeno ocurrido ya a inicios de los 80 en donde si bien existían máquinas capaces de realizar cálculos en profundidad de hasta 6 movimientos éstas aún estaban lejos de la posibilidad de derrotar a un gran maestro, puesto que carecían del conocimiento ajedrecístico suficiente para poder determinar con precisión la influencia a largo plazo de factores posicionales del juego. Este fue entonces el siguiente objetivo de los programadores. Poder combinar la ya buena capacidad de cálculo con una función de evaluación lo suficientemente "buena" la cual pudiese discriminar de acuerdo a parámetros posicionales.

La tarea resultó bastante más difícil de lo que se creía y sólo a principios de los 90 aparecieron máquinas capaces de combinar ambas capacidades y competir a nivel de Grandes Maestros. Un ejemplo interesante es la evolución de Deep Blue y algunos hitos que marcaron su desarrollo. En 1987 luego de la estrepitosa derrota de Deep Thought (versión previa de DeepBlue) frente a Kasparov, sus creadores mencionaban la carencia aún de capacidad de hardware para poder realizar un cálculo que superara la capacidad de su rival humano,y que esto fue el principal motivo de la estrepitosa derrota en ese match. Ya como Deep Blue y luego de la derrota por 4-2 en 1996, Feng-Hsiung Hsu expresa en [8] acerca de las conclusiones luego del match:

*"Nuestra primera conclusión fue de que la velocidad de búsqueda de DeepBlue parecía ser ya la adecuada. Garry estaba venciendo a DeepBlue posicionalmente pero no tácticamente. Esto nos llevó hacia nuestra segunda conclusión, Deep Blue poseía un inadecuado nivel de conocimiento ajedrecístico comparado al que poseía el campeón del mundo".*

Un muy buen ejemplo de esta "ineptitud posicional" de DeepBlue fue la partida 6 del match (Partida 32) en donde la máquina no evaluó adecuadamente el encierro de un alfil y torre por el resto de la partida, lo cual a largo plazo fue un problema insolucionable. Los trabajos realizados en DeepBlue durante el año siguiente apuntaron a mejorar la función de evaluación del programa mediante el trabajo realizado con Grandes Maestros.

El resultado del año de trabajo entre un match fue una versión de DeepBlue la cual poseía gran cantidad de Hardware dedicado y una función de evaluación con mas de 80 parámetros modificables en "peso" y optimizados mediante el trabajo con una base de datos de partidas de grandes maestros. La gigantesca capacidad de hardware de esta máquina hizo contar con los suficientes medios para realizar cálculos y evaluaciones en tiempos ínfimos y con resultados más que asombrosos.

Por supuesto el caso de DeepBlue es algo totalmente extremo dentro del desarrollo histórico de las máquinas que juegan ajedrez, pero que nos lleva a cuestionarnos acerca de cómo se relacionan las limitancias en la capacidad de hardware de una máquina con su correspondiente implementación de software, o más bien preguntarnos qué tan buenos programas de ajedrez se pueden realizar en este momento bajo límites de memoria y velocidad de procesamiento.

Esto no es comparable a repetir la situación de 20 años atrás, cuando se desarrollaba justamente bajo una tecnología bastante más limitada, puesto que el desarrollo de software también ha evolucionado así como otras perspectivas de cómo darle fuerza ajedrecística a los programas (función de evaluación!). Con esto podríamos definir y comparar los aspectos importantes a la hora de "confeccionar" un programa de ajedrez tanto desde el punto de vista de software como ajedrecístico, tomando en cuenta características propias del hardware bajo el cual el programa va a funcionar. Este es el punto de inicio de la discusión del siguiente capítulo.

# Capítulo 7

# Propuesta de Parámetros de Análisis de Fuerza de Juego en Computadoras de Ajedrez

Hasta el momento hemos podido hacer una recapitulación de todos los conceptos, avances e hitos referidos a este apasionante tema del ajedrez a nivel de computadoras. Una carrera marcada por tendencias que cambiaron en el tiempo combinada con fracasos y éxitos de investigadores ha sido una característica de este proceso iniciado por las ideas de Shannon acerca de cómo una máquina debería jugar al ajedrez hasta la aparición de Deep Blue, un monstruo computacional en todo sentido.

A pesar de que el principal objetivo de esta carrera ya fue cumplido hace un par de años (derrotar al campeón mundial de ajedrez) la inquietud que asalta a muchos es cuantificar cuán "inteligente" era realmente el programa que derrotó al campeón mundial y cual fue la base de esta victoria. Fue Deep Blue un sistema que llegó a "pensar" como lo hacen los ajedrecistas humanos o más bien un "monigote" capaz de calcular millones de posiciones y movimientos pero sin tener mayor comprensión de lo que realmente está ocurriendo en el juego?. Probablemente la respuesta más acertada a esta interrogante es que una combinación de ambas ideas fue el resultado de un Deep Blue impresionante en su capacidad de cálculo y a la vez sorprendente en varios movimientos que hacían pensar que quien estaba jugando era un humano. Por supuesto la creación de esta super máquina se basó en una implementación de Hardware fuera de lo común (2 frames albergando cada uno a 15 RS-6000 SP) la cual pareciese muy difícil de que se repita nuevamente dado que el incentivo de una nueva victoria sobre el campeón mundial no es tan deseado como lo fue por vez primera.

Ahora bien, sólo observando la particular configuración de hardware bajo la cual funcionó Deep Blue uno podría asegurar que no existe aún sobre el planeta otra máquina que se le pueda comparar en nivel de juego dadas sus capacidades gigantes de procesamiento, pero cabe preguntarse, si Deep Blue no hubiese contado con tal nivel de Hardware, habría sido capaz de obtener los mismos resultados? Habría logrado derrotar a Kasparov si hu-

biese funcionado en un computador común y corriente? Probablemente no, dirían muchos creyendo que la real fuerza de esta máquina estaba en su nivel de "fierro", o quizás si, dirían otros, argumentándose en el excelente desarrollo de software y conocimiento ajedrecístico que poseía el programa, pero ahora surge otra interrogante y es que "cuan" bueno era el programa en hardware y en software ?, Cuánto hubiese bajado su nivel si no hubiese contado con la tecnología de hardware de ese momento?. Cómo hubiésemos podido medir la capacidad de software de Deep Blue frente a otros programas de la época pero sin su capacidad de hardware?. Esto es en el fondo un problema de comparación entre capacidades particulares de programas de ajedrez, y es acá en donde deseamos detenernos para realizar un análisis comparativo o mejor aún, proponer herramientas de medición de nivel de una máquina por sobre otra considerando no sólo niveles de hardware bajo el cual funcionan.

En la carrera por obtener la máquina que derrotase al mejor ajedrecista humano la necesidad imperiosa de lograr mayores velocidades de procesamiento, mejoras en los métodos de búsqueda, aplicación de mayor conocimiento, etc. dejó un poco de lado la investigación comparativa de qué es lo que realmente diferencia a una máquina ajedrecista de otra. A la hora de comparar dos programas, qué es lo que debo observar para sacar conclusiones acertadas ?! Una primera respuesta sería el nivel de hardware de la máquina. es muy probable que un programa en un hardware de mayor capacidad sea mucho mejor que otra en uno de menor envergadura. Una mayor velocidad de procesamiento y capacidad de memoria harían una diferencia notable entre un programa y otro, pero a partir de esto surgen la preguntas de cuánto más capacidad tendría un programa en un hardware avanzado que otro en uno más limitado?. Si suponemos que ambos hardware son similares, qué diferenciación podríamos hacer en este caso? Quizás una forma entretenida de solucionar esta disyuntiva sería poner a jugar ambos programas y ver los resultados luego de varias partidas, pero lo que nos gustaría en realidad sería el encontrar parámetros de medición que ayuden a medir la capacidad de los programas bajo condiciones similares de software, en el fondo, medir las capacidades de los programas pero poniendo límites a la diferenciación en hardware. Este es el tema que queremos atacar en este capítulo, buscar parámetros de medición de la capacidad de un programa en función de su hardware y software.

## 7.1. Software

El primer análisis de parámetros de comparación en la fuerza ajedrecística de un programa de ajedrez lo haremos en función del software. Es en este aspecto en donde radica la parte "inteligente" de la máquina, aquella que no depende del nivel de tecnología en "fierro" sino que de la capacidad de procesar reglas y utilizarlas para asimilar el razocinio humano.

A la hora de clasificar las herramientas de software utilizadas por los programas de ajedrez, nos encontramos con una característica común en los programas de ajedrez, digamos, una especie de "estándar" de construcción de programas el cual no ha variado en todos los años de desarrollo el cual está basado en la herramienta de generación de movimientos legales (generador de movimientos), el motor de búsqueda de posiciones dentro del árbol de variantes del juego (Algoritmo de Búsqueda) y la función que determina el score de la posición analizada

(Función de Evaluación). Esta combinación de herramientas es la base del software de un programa de ajedrez, y a partir de ellas haremos una análisis comparativo de la fuerza de las máquinas.

### 7.1.1. Búsqueda

La herramienta de búsqueda de un programa de ajedrez tiene la misión de realizar un recorrido ordenado por el árbol de variantes a partir de una posición inicial (nodo raíz). Como ya hemos discutido en capítulos anteriores, la extensión del árbol de variantes es tal que sería imposible creer que un programa de computadora sería capaz de recorrerlo por completo. A pesar de esto, la capacidad de cálculo y almacenamiento de los computadores hacen que superen ampliamente al jugador "humano" en esta característica del juego del ajedrez (cálculo de variantes en profundidad) y a partir de esto es que los programadores han tratado de darle fuerza a los programas con búsquedas lo más profundas posible dentro del árbol de variantes.

A pesar de las mejoras que se han ido realizando sobre los años al motor de búsqueda, sobretodo en torno a la selectividad de las variantes a profundizar con tal de dar mayor velocidad (y profundidad) a la búsqueda, los jugadores humanos de buen nivel poseen la capacidad de discriminar rápidamente cuáles variantes analizar y cuáles no, característica que fue muy difícil de implementar en programas de ajedrez, tanto así que se optó por darle cabida al desarrollo de la búsqueda extensiva sobre el árbol de variantes, utilizada por los llamados programas de fuerza bruta, con tal de aprovechar la mayor capacidad y velocidad de los programas lo cual les aseguraba al menos no cometer errores de nivel de iniciado en el juego (por ejemplo capturas de material o bien ataques de mate).

Los programas superaban entonces con creces a los humanos en el cálculo concreto de variantes hasta cierto nivel de profundidad, y a medida que iban mejorando el desarrollo de algoritmos y técnicas de poda la capacidad de "ver un movimiento más allá" iba creciendo. A nivel de jugadores humanos el tener un nivel mayor de profundidad de búsqueda equivale a calcular un movimiento extra en una secuencia de jugadas. Es decir, un programa que posee un nivel de profundidad extra por sobre su rival tendrá la capacidad de analizar en todas las variantes y durante todos los cálculos realizados durante cada jugada un movimiento por sobre su rival. Esto no pareciese ser mucho como para pensar en una diferencia importante entre un programa con profundidad $B$ y otro con $B + 1$, pero si llevamos esta diferencia al juego humano estamos hablando de una categoría de jugador por sobre otra, vale decir, un jugador que siempre ve un movimiento más que su rival y que en resumen tiene mayor información de lo que va a acontecer en cada secuencia de movimientos.

Respecto de esto se han realizado distintos experimentos entre programas con idéntica capacidad de hardware, la misma función de evaluación y motores de búsqueda similares, pero con diferencias en el nivel de profundidad. Uno de los experimentos más renombrados es el realizado en el programa "Belle" por Thompson en 1983. Se definieron 6 programas distintos, P4, P5, ... ,P9. P4 busca con profundidad de 4 movimientos y a partir de acá realiza una búsqueda de posición estable. P5, P6, ... son programas similares que buscan con mayor

| Lug | Prog | P9 | P8 | P7 | P6 | P5 | P4 | PTS | perf |
|---|---|---|---|---|---|---|---|---|---|
| 1 | P9 | - | 14.5 | 16 | 18.5 | 20 | 20 | 89.0 | 2328 |
| 2 | P8 | 5.5 | - | 15 | 18.5 | 19.5 | 20 | 78.5 | 2208 |
| 3 | P7 | 4 | 5 | - | 16 | 17 | 20 | 62.0 | 2031 |
| 4 | P6 | 1.5 | 1.5 | 4 | - | 16.5 | 19 | 42.5 | 1826 |
| 5 | P5 | 0 | 0.5 | 3 | 3.5 | - | 15 | 22.0 | 1570 |
| 6 | P4 | 0 | 0 | 0 | 0.5 | 5 | - | 5.5 | 1235 |

Cuadro 7.1: Resultados del Experimento de Thompson en programas con distinto nivel de profundidad de búsqueda.

profundidad en forma progresiva.

Los programas compitieron en un torneo todos contra todos (round-robin) a 20 rondas (cada programa jugaba 20 partidas contra cada otro de la competencia). Cada programa jugó con distintos colores a partir de 10 posiciones seleccionadas en forma aleatoria. Las posiciones fueron seleccionadas de la Enciclopedia de Aperturas de Ajedrez a partir del movimiento 10 y con una evaluación de posición igualada o bien con compensación. Cerca de 300 partidas fueron jugadas. Lamentablemente no se tiene referencia del tiempo de juego para las partidas. La performance de rating de cada programa fue determinada (en forma arbitraria) calculando el rating necesitado por un programa para tener como expectativa los puntos obtenidos. Los ratings fueron entonces iterados hasta que convergieron para cada programa. Los resultados se aprecian en la tabla 7.1

Resulta sorprendente ver la regularidad de comportamiento de los programas diferenciados por niveles de profundidad. Existe una correlación prácticamente lineal entre el performance de rating del programa y su límite de profundidad. El problema del test radica en que desafortunadamente a pesar de realizarse 20 partidas entra cada oponente y con porcentajes de victoria cercanos al 75 % no se puede argumentar con total confianza que un programa es mejor que otro en general. De acuerdo Mysliwietz en [61] para establecer una diferencia de rating de 70 puntos son necesarias al menos 500 partidas mientras que para diferencias de 40-50 puntos cerca de 100 partidas son necesarias.

Newborn en [64] ha sugerido que si uno duplica la velocidad de un programa de fuerza bruta este mejora su rating en 100 puntos. Si asumimos una profundidad de factor 5,5 entonces un movimiento de ventaja otorga 250 puntos. En el rango linear de P4 a P6 (el rango observado por Newborn en programas reales) esta relación se mantiene. En el rango extendido de este experimento (P7-P9) el incremento de rating no es similar al predecido. Algunas conclusiones respecto de este experimento realizado por los investigadores de Belle:

1. Los ratings obtenidos a partir del experimento son similares a los ratings que Belle obtiene contra jugadores humanos en cada nivel de profundidad.

2. Los programas no poseen un factor de. Esto significa que si P9 cree estar abajo en un score de una fracción de peón intentará empatar con P4. El resultado muestra que P9 tiene un mayor rating contra P8, luego contra P7, etc.

3. La función de evaluación para cada programa en el experimento fue la misma. En la evolución de un programa la función de evaluación debería evolucionar a la fuerza del programa. La función de evaluación de Belle posee elementos de un jugador de 1600 pts. sabe mucho acerca de protección del Rey y estructuras de peones, mientras que poco acerca de debilidad de casillas y colaboración de piezas. Claramente P8 y P9 se beneficiarían a partir de una función de evaluación más sofisticada mientras que en P5 o P6 ésta perdería su capacidad dada el bajo nivel de profundidad.

Otros tests de este mismo estilo realizados en computadoras fueron realizados por Szabo en 1988, Berliner en 1990, Mysliwietz en 1995 y Jughanns en 1997. La descripción de estas experiencias se encuentra en [48]

En términos de búsqueda podemos argumentar que cualquier mejora en el algoritmo de recorrido sobre el árbol de variantes (podas o algoritmos de selectividad, por ejemplo) se verá reflejado en una búsqueda de mayor velocidad, o mejor dicho, en la capacidad de lograr mayor profundidad en igual tiempo que otro programa que no posea la mejora en su algoritmo de búsqueda. Esto apunta a demostrar que cualquier mejora en el proceso de búsqueda se verá reflejado en un aumento en el límite de profundidad de ella.

### 7.1.2. Función de Evaluación

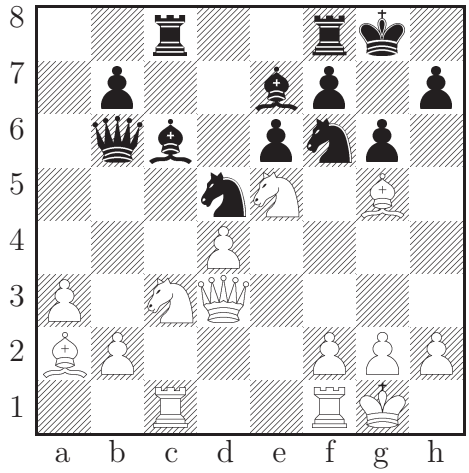
La función de evaluación cumple la misión de asignar un valor numérico a las posiciones analizadas por el programa de ajedrez en la decisión de por qué continuación seguir en el árbol de variantes. Comúnmente sólo son evaluadas con la función las posiciones terminales dentro de la búsqueda, procediéndose luego mediante mini-max a determinar la mejor continuación que podría seguir el programa considerando las mejores respuestas de su oponente.

Realizando una comparación con el ajedrez humano, la función de evaluación se asemeja al proceso de decisión que realiza un jugador de carne y hueso respecto de cuan favorable le es una posición. Este punto es bastante interesante puesto que podemos atribuir a la función de evaluación toda la característica "inteligente" del programa dado que debe contener todo el aspecto racional del jugador de ajedrez al decidir el valor de una posición.

#### La percepción humana en el ajedrez

Varios trabajos se han realizado acerca del tipo de análisis que realiza un humano al observar una posición en su mente y "evaluarla". Charness en [7] menciona algunos trabajos realizados por el psicólogo De Groot enfocados a investigar el proceso de razocinio utilizado por un ajedrecista real. La intención de De Groot era el determinar por qué unos jugadores son mejores que otros y qué factores incurrían en esto. Uno de sus experimentos fue el obtener descripciones verbales de lo que pensaban varios jugadores de buen nivel al presentárseles una posición nueva para ellos, solicitándoseles seleccionar un movimiento describiendo las razones que lo llevan a esa selección. La experiencia fue realizada en jugadores de nivel mundial como

Alekhine, Keres y Euwe así como en jugadores de nivel de aficionado. Sus resultados fueron sorprendentes en el sentido de desmitificar el supuesto del gran cálculo de variantes realizado por los jugadores de gran nivel.



*El mejor movimiento para el blanco es 1.♗xd5!*

Al exponer a los jugadores a una misma posición de ejemplo en donde debían seleccionar el mejor movimiento (el cual era ya conocido por el experimentador) tanto grandes maestros como jugadores de nivel experto calculaban un promedio de 7 jugadas hacia adelante. La gran diferencia entre ambos estuvo en que todos los grandes maestros mencionaron considerar en algún momento el mejor movimiento mientras que sólo 2 de los 5 expertos la mencionaron en alguna ocasión en sus análisis. Esto demostraba que había "algo" en la posición que atraía la atención de los grandes maestros a elegir el movimiento correcto, pero no la atención de los expertos.

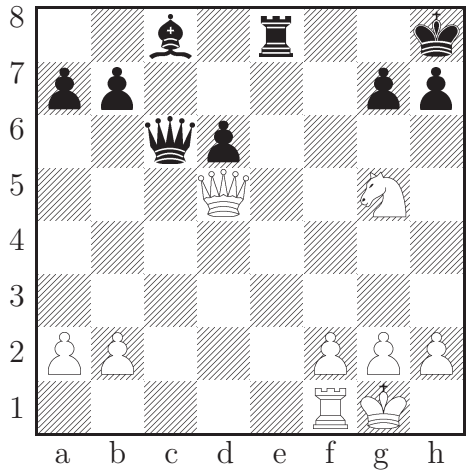
Un argumento expresado por De Groot fue que ambos tipos de jugadores *percibían* la posición de manera muy distinta. El rol de la percepción quedó mucho mejor definido con otro experimento realizado por De Groot en donde presentó a distintos jugadores una posición no familiar por sólo algunos segundos. Luego los jugadores debían reconstruir la posición que habían observado. Los jugadores de mayor nivel realizaron esto mencionando la posición de cada una de las piezas mientras que los de menor nivel debieron utilizar un tablero para ubicarlas. Los resultados del experimento fueron sorprendentes. Los grandes maestros mencionaron correctamente la posición de las piezas con un 93 % de efectividad. Los expertos respondieron con un 72 % de piezas correctas mientras que los jugadores de primera categoría obtuvieron un 52 % de efectividad. Esta experiencia fue repetida por Chase y Simon [40]con similares resultados en posiciones consideradas como típicas, pero con posiciones en donde las piezas fueron ubicadas en forma aleatoria el resultado fue sorprendente con todos los jugadores bajando la cantidad de piezas correctamente ubicadas a un 20 %.

Chase y Simon concluyeron que el nivel superior de un maestro de ajedrez no se debía a su extraordinaria memoria visual, sino que más bien a una habilidad específica para el juego

del ajedrez. Estos estudios en la capacidad de análisis de los jugadores sugieren que no es el proceso de pensamiento sino que más bien el nivel de percepción de lo que ocurre en la posición lo que diferencia a los jugadores de un nivel u otro. Una analogía a la diferencia entre un maestro y un aficionado es el caso de un niño aprendiendo a leer versus un adulto que es ya un habitual lector. Cuando un niño mira una página llena de letras debe esforzarse por unirlas y leerlas como palabras. El adulto, sin embargo, rápidamente mira la página como una serie de palabras y posiblemente frases. Ambos miran la misma página pero producen diferentes códigos de descripción de ella, basados en el tamaño de los patrones utilizados para su lectura.

Todas estas experiencias hacen preguntarse si acaso el secreto de los grandes jugadores es el memorizar cerca de 50.000 patrones de ajedrez y utilizarlos de acuerdo a la etapa del juego en que están presentes?. Si un jugador novato hiciese esto sería probablemente capaz de responder tan rápidamente como los grandes maestros en los experimentos de De Groot y Chase, pero su nivel de juego no mejoraría lo suficiente. Entonces, cómo es posible enlazar el vasto conocimiento ajedrecístico de patrones con fuerza ajedrecística ?

Chase y Simon sugerían que la correlación entre capacidad de percepción y fuerza ajedrecística puede obtenerse asumiendo que muchos patrones de ajedrez están directamente asociados con movimientos "buenos". Esto es, cuando un maestro mira una posición de ajedrez su percepción de configuraciones familiares genera ciertos "procesos" los cuales pueden modelarse como : Si existe una condición $X$ entonces realizar la acción $Y$. Ajedrecísticamente hablando esto sería como "si existe el patrón $X$ entonces considere el movimiento (o plan) $Y$", o más concretamente, si existe una columna abierta entonces considere mover una torre hacia ella. UN ejemplo más complejo surge de la posición de la siguiente figura:




*Ejemplo del Mate Philidor.*

La mayoría de los jugadores de fuerte nivel reconocerían el mate que viene luego de la secuencia 1.♘f7+ ♔g8 2.♘h6+ ♔h8 (2...♔f8 3.♕f7 mate) 3.♕g8!+ ♖xg8 4.♘f7 mate . Si variamos elementos no esenciales de la posición (por ejemplo, moviendo el peón negro de a7 a a6) el mate no cambia. Otros cambios, sin embargo, hacen una gran diferencia, por

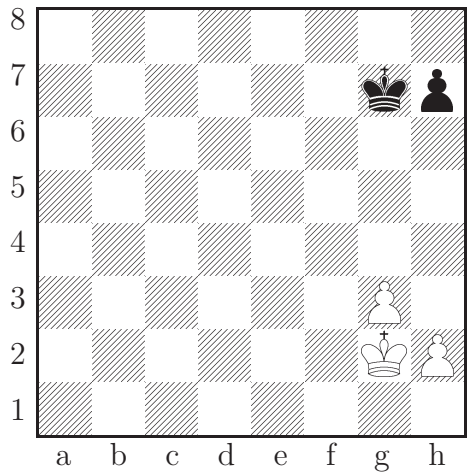
ejemplo mover el peón negro de h7 a h6 o intercambiar de posición la dama y torre negras. Los humanos son probablemente "sensibles" a los factores críticos de la posición. En el caso del reconocimiento del jaque mate los patrones de percepción sean probablemente : La dama dominando la diagonal a2-g8; caballo capaz de dar un jaque en f7 al siguiente movimiento y Rey negro acorralado en h8. probablemente estos patrones son suficientes para encontrar el plan a seguir : Ejecutar el mate conocido como "mate de Philidor". Existen muchos ejemplos similares de cómo ciertos patrones en el ajedrez son parte del repertorio de jugadores de buen nivel ajedrecístico.

Esto nos lleva a una explicación de porqué en la posición "A" los grandes maestros consideraron el movimiento correcto, mientras que sólo algunos expertos lo hicieron. Probablemente existían algunos patrones dentro de la posición los cuales al ser reconocidos generaban el movimiento correcto. Sólo los jugadores que lograban reconocer estos patrones y asociarlos correctamente con la acción a ejecutar generaban el movimiento correcto para su posterior evaluación.

**Evaluación Humana e Intuición**

Por supuesto que la sola evaluación estática de una posición después de un movimiento no es suficiente para juzgarla en forma correcta. En esencia el problema de seleccionar un movimiento entre varios para obtener una posición se reduce a elegir un movimiento que pueda ser correctamente evaluado como superior a otros. El jugador debe decidir "este movimiento me conduce a una posición mejor que los otros?". La decisión es muy simple en el caso de un movimiento que da jaque mate, pero en la mayoría de las posiciones el proceso de evaluación es complejo.

El punto de buscar con mayor profundidad es el encontrar una posición la cual puede ser evaluada en forma confiable. La regla de detención en la búsqueda es aparentemente el llegar a una posición la cual puede ser reconocida como "estable" y evaluarla como buena o mala. Los jugadores de mayor fuerza (aquellos con mayor cantidad de patrones en memoria) deberían reconocer posiciones terminales con mayor velocidad que los jugadores aficionados. Es posible que los patrones de ajedrez alojados en memoria estén asociados no sólo con la información acerca de movimientos plausibles, sino que también con información evaluativa. Por ejemplo, la mayoría de los jugadores de buen nivel evaluarían la posición de la figura como ganada para las blancas.



*Final de partida teóricamente ganado para las blancas.*

La base de esta evaluación podría estar fundamentada en el hecho de que en un final de peones un peón extra es victoria casi segura. Ahora bien, si agregáramos a esta misma posición un par de caballos (uno para cada bando) o un par de alfiles o bien un par de torres los mismos jugadores evaluarían la posición como de tablas. En este caso, el reconocimiento viene del hecho de que en este tipo de final con presencia de piezas y con peones en el mismo flanco no hay una victoria forzada. A la vez, para jugadores experimentados la evaluación de trasponer al final de peones de la figura luego de algunos cambios es directa dada la evaluación inicial del final de peones. Los jugadores sólo necesitan generar las posiciones del final como posiciones terminales sin necesidad de evaluar con mayor profundidad.

Aquel jugador que tenga la capacidad de evaluar con mejor precisión una posición como favorable, desfavorable o neutral tendrá mayores chances de vencer a un rival cuyas evaluaciones son menos precisas, particularmente con ambos jugadores realizando búsquedas al mismo nivel de profundidad. Según De Groot "En el análisis del juego es usual que dos jugadores realicen un cálculo similar de las variantes que siguen de una posición base. Por lo general la diferencia entre ambos jugadores es la capacidad de uno de captar movimientos mejores que su rival al poseer una mejor percepción del juego". Esto mismo puede extrapolarse a programas con diferentes funciones de evaluación pero con idénticos niveles de búsqueda en profundidad.

Muchos jugadores se han enfrentado en alguna oportunidad al problema de decidir entre cerrar una posición, abrirla o bien realizar un cambio de piezas pero han sido incapaces de tomar una decisión correcta dado que no pueden evaluar la posición resultante en forma adecuada.

Siguiendo esta línea de pensamiento es posible especular en cómo los Grandes Maestros realizan movimientos o bien sacrificios "intuitivos". Posiblemente lo que hay detrás de un movimiento "intuitivo" es el hecho de que el jugador ha reconocido ciertos patrones de la posición que le hacen creer que es del tipo donde un ataque casi siempre triunfará. El jugador

sabe que no necesita calcular todas las consecuencias del sacrificio puesto que está seguro que posteriormente encontrará el remate de la posición.

**Construyendo la función de Evaluación**

Al escribir una función de evaluación para un programa de ajedrez es esencial emplear instrucciones de uso eficiente dado que este componente del programa es utilizado en forma repetitiva (miles o cientos de miles de veces durante cada selección de movimiento). por esta razón sea preferible el no incluir excesivos términos en la función de evaluación dado que cada nuevo término significa un pequeño incremento en el tiempo requerido para evaluar cada posición terminal. Una buen función de evaluación es aquella que evalúa los aspectos críticos de la posición en cuestión y lo realiza de la manera más eficiente posible. Por cada nuevo término incluido en la función de evaluación uno debe evaluar si es que la información agregada compensa la cantidad de tiempo computacional que dicha evaluación requerirá.

La estructura de la función de evaluación es dependiente del tipo de procedimiento de búsqueda empleado. En un programa cuyo énfasis está en el evaluar un pequeño número de nodos terminales (por ejemplo, 500 nodos en el programa HITECH de Berliner) en forma minuciosa se requerirá una función de evaluación de alta complejidad y con evaluaciones lo más precisas posibles. En programa con gran cantidad de nodos terminales la función de evaluación debe ser rápida y simple. El programa TECH desarrollado por Gillogly en Carnegie-Mellon examinaba sólo el equilibrio material para cada posición terminal de entre las cerca de 500.000 examinadas en cada movimiento. En la década de los 70 no era claro qué procedimiento fuese el mejor, si bien los humanos utilizan claramente el primero.

Ahora bien, por lo que hemos descrito acerca del problema a resolver por la función de evaluación y las características del pensamiento humano, es muy factible el que cada función de evaluación de todos los programas de ajedrez tengan grandes diferencias y sólo se asimilen en sus patrones más básicos (por ejemplo, material). El problema de construir una función de evaluación es el que, a diferencia de la búsqueda, no existe un método estándar de construcción. Los patrones potenciales que podría manejar una función de evaluación son cerca de 50.000, los cuales toman distintos valores de acuerdo a la etapa del juego en que nos encontremos y además el contexto de la posición (por ejemplo, cerrada o abierta). Es claro que existen cierta base de patrones básicos en las funciones de evaluación que todo programa presenta : Equilibrio material, movilidad de piezas, casillas atacadas, seguridad del rey, dominio central, actividad de piezas, desarrollo, etc. Estos patrones fueron propuestos por Shannon en [68] y son factibles de encontrar en distinta literatura ajedrecística, pero si suponemos que nuestra función de evaluación ya contiene en su código el reconocimiento adecuado de estos patrones, cómo podemos seguir mejorándola?. Este fue un problema que ocurrió a varios programas los cuales en un momento debieron decidir por recurrir a la consulta a expertos con tal de obtener información relevante de cómo incrementar la cantidad de patrones de reconocimiento de la función de evaluación.

**Funciones de Evaluación "Buenas" y "Malas"**

Dos programas de similares características de hardware y motor de búsqueda, pero con distinta función de evaluación se diferenciarán en el hecho de que aquel que tenga una mejor sofisticación en su evaluación será capaz de avaluar en forma más exacta el valor de las posiciones del juego con lo cual podrá, por ejemplo, aseverar correctamente si una posición le es favorable y entrar en ella sabiendo que su rival puede pensar de manera incorrecta que está en una situación de igualdad. Diferencias entre funciones de evaluación significan por lo general una desventaja que se hará palpable a largo plazo.

Supongamos un ejemplo en que un programa 'A' evalúa correctamente que la mala estructura de peones de su rival será una desventaja permanente la cual se verá reflejada sobretodo en la etapa final del juego. Su rival 'B', con una función de evaluación de menor capacidad podrá suponer que la estructura de peones no es tan mala como para "castigarla" con menor puntaje, concluyendo que el final es con chances para ambos bandos. Sólo después de varios movimientos el programa 'B' se percatará (probablemente por búsqueda) que está en una situación en la cual no puede escapar de una posición inferior como consecuencia de su mala evaluación en varios movimientos atrás. Situaciones como estas eran causal de las frecuentes derrotas de los programas frente a jugadores humanos. El humano tenía una mayor capacidad de percepción de los patrones posicionales importantes a la hora de evaluar una situación, por lo que aprovechaban la incapacidad de los programas de "entender" factores posicionales los cuales se revelarían fatales en una etapa posterior del juego.

Cuando nos referimos a los patrones posicionales importantes estamos hablando de aquellos que son distinguibles por jugadores de gran nivel, y que como discutíamos al inicio de esta sección, los grandes jugadores los tienen mejor asimilados que los aficionados. Esto es en el fondo lo que muchos denominan como "Conocimiento Ajedrecístico". Esta diferencia en conocimiento entre una función y otra nos asegura una diferencia en nivel, pero, qué valor puede tener esa diferencia en nivel? A diferencia del caso del parámetro de profundidad de búsqueda no se han realizado gran cantidad de experiencias entre máquinas con distinta función de evaluación con tal de ver su diferencia de puntaje de "rating". En todo caso, podemos comparar esta diferenciación a la producida entre dos jugadores humanos de igual capacidad de cálculo pero distinto conocimiento ajedrecístico. Las diferencias acá no se verán por omisión de algún movimiento en largas variantes, si no que por algo mucho más sutil y oculto, lo cual puede ser como no un factor decisivo dentro del juego. En todo caso sería muy raro el que un programa con función de evaluación menos sofisticada derrotase sistemáticamente a otro con una función de mayor capacidad, más bien deberíamos suponer un resultado a favor del segundo pero por un estrecho margen.

**Estilos de juego**

La gran cantidad de patrones de conocimiento presentes en el juego del ajedrez ha hecho el que en un inicio las funciones de evaluación se remitieran a tomar en consideración sólo aquellos más básicos en función de no hacer el que la evaluación de una posición consumiese

demasiados recursos del sistema. Con el tiempo y los avances tecnológicos este problema fue superado y funciones de evaluación cada vez más complejas iban surgiendo.

Se requirió la colaboración de jugadores expertos con tal de poder traspasar a los programas un mayor conocimiento ajedrecístico, pero surgió una observación bastante curiosa respecto a la distinta importancia que jugadores le otorgan a ciertos patrones. Por ejemplo, jugadores gustosos de posiciones llenas de opciones de combinación y con posibilidades de ataques al rey evalúan muy favorablemente la creación de líneas abiertas para la evolución de las piezas hacia distintos sectores del tablero y le darán un peso mayor a este factor (potencial de ataque de la posición) a diferencia de un jugador quien gusta del juego de maniobras lentas, con un control absoluto de la situación y sin permitir ninguna chance de contrajuego de su rival (potencial de juego posicional de la posición). Ambos jugadores intentarán llevar el juego a una forma la cual más les acomode, probablemente por un asunto de psicología relacionada de acuerdo a sus personalidades y sus distinta forma de "percibir" el juego del ajedrez. Ahora bien, es factible preguntarse entonces si los programas tienen esta diferencia ? La respuesta es que esto es totalmente factible, dado que la diferencia en este caso se da en la importancia o "peso" dado a ciertos patrones por sobre otro. Por ejemplo un programa enfocado al ataque otorgará un mayor "peso" o importancia a obtener líneas abiertas sobre el enroque enemigo, mientras que un programa enfocado a la defensa priorizará el evitar que esto ocurra en vez de utilizar la misma estrategia sobre el enroque enemigo. Pero entonces, ¿ existe un estilo de juego mejor que otros ? La respuesta es que en realidad no, y esto es uno de los aspectos más atractivos del ajedrez, en donde el estilo de juego es un reflejo de la distinta importancia dada a los miles de patrones presentes en el juego. Esto hace que tengamos muchos "tipos de jugadores" de acuerdo a su estilo de juego, lo cual ocurre en forma similar en los programas de ajedrez!.

Para citar un ejemplo DeepBlue poseía una función de evaluación tremendamente compleja la cual estaba optimizada en base a cerca de 700 partidas y ejemplos de grandes maestros. Cada una de estas partidas era presentada al programa y mediante un método lineal de asignación de distintos pesos a los términos de la función se buscaba que el programa "encontrara" los movimientos correctos realizados por los grandes maestros en las partidas. Muchas de las partidas presentadas al programa correspondían a jugadores de carácter más técnico y posicional en vez de agresivo y de ataque. Esto según comenta Hsu en [8] puesto que el libro de aperturas del programa estaba realizado en base a partidas del ex campeón mundial Anatoly Karpov, lo cual se adecuaba al estilo buscado en el programa.

**Algunas conclusiones**

La dificultad en realizar una función de evaluación no está por lo general en su parte básica, sino que más bien en la parte que abarca aspectos más técnicos del juego los cuales son difíciles de percibir para el jugadores aficionados pero que en jugadores expertos están internalizados. En todo caso, la sola percepción de estos patrones no es suficiente para garantizar una diferencia en nivel ajedrecístico, debe existir también una capacidad de asociar acciones a ejecutar posteriores al reconocimiento del patrón en cuestión y que serán distintas respecto de la fase del juego y también respecto al valor que toman otras características de la partida.

Esta gran combinación de factores hace que la confección de funciones de evaluación de gran calidad sea una tarea bastante compleja. Otra observación importante es que la ventaja de tener una función de evaluación más sofisticada no es a priori tan "decisiva" como el hecho de tener un nivel de búsqueda en profundidad por sobre el rival dado que las diferencias en "percepción" se darán sólo si el patrón de diferencia se presenta durante el juego, e incluso si ocurre esto, puede que las diferencias de percepción sean mínimas y por esto no decisivas.

Estas últimas características hicieron que en un principio se diese una mayor importancia al desarrollo de técnicas de búsquedas por sobre la investigación en mejoras a la función de evaluación, si bien curiosamente a principios de los 90 cuando los programas ya poseían un nivel de búsqueda suficientemente bueno se volviera a investigar en cómo mejorar la función de evaluación con tal de dar mayor capacidad a los programas. Esto resultó ser decisivo para mejorar el conocimiento ajedrecístico de las máquinas y evitar errores de largo plazo los cuales eran la principal causa de las derrotas de programas frente a rivales humanos. Con la colaboración de ajedrecistas profesionales las funciones de evaluación mejoraron notablemente durante esos años, logrando sorprender muchas veces la capacidad de los programas de realizar movimientos "que sólo un humano podría realizar" dadas las características posicionales y perceptivas de éstos (ver partida 34 del Anexo B, DeepBlue - Kasparov, movimiento 37).

Comparativamente hablando resulta difícil la labor de determinar si una función de evaluación de un programa es mejor que otra y más aún "cuánto" mejor puede ser, dada la complejidad que comúnmente presentan las funciones de evaluación de buena calidad. Comparar funciones de evaluación sería algo parecido a comparar la visión del juego de dos jugadores humanos, distintos en términos de percepción y emociones. Por ejemplo, las "funciones de evaluación" de un Mikhail Tahl [1] y un Tigran Petrosian [2] son tremendamente efectivas y complejas, pero absolutamente diferentes!. El primero valorizará en una posición aspectos muy dinámicos como chances de ataques y líneas abiertas mientras que el segundo tomará aspectos más posicionales como dominio de casillas, ventaja de espacio, etc. Ambos tipos de percepción del juego lograron un reconocido nivel mundial, pero resulta interesante el notar sus tremendas diferencias.

En máquinas de idénticas características de hardware y con motores de búsqueda similares habría que realizar un ejercicio en donde primero se identifiquen los patrones de diferencia entre las funciones de evaluación de todos los programas y luego realizar varias partidas entre ellos revisando jugada a jugada los scores dados por las funciones a la posición en curso y a las que surgen de la búsqueda en profundidad, observando los resultados y determinando si las evaluaciones de posición fueron decisivas en el resultado final de la partida. También podríamos aislar un patrón de la función de evaluación de un programa (por ejemplo, la asignación de puntaje de castigo a cierto tipo de estructura de peones)agregándolo en la función de evaluación de un programa y omitiéndolo en otro, haciendo jugar a los programas y visualizar los resultados y los momentos en donde el factor determinó diferencias en las evaluaciones de los programas.

[1] Campeón mundial entre 1960 y 1961, destacó por su estilo tremendamente arriesgado y lleno de combinaciones imaginativas. Le denominaban "El Mago de Riga".

[2] Campeón mundial entre 1963 y 1969, destacó por un estilo muy conservador, logrando victorias en base a la acumulación progresiva de pequeñas ventajas. Le denominaban "La Boa".

Estos ejercicios serían bastante interesantes de realizar, pero con seguridad la gran cantidad de patrones presentes en el juego y el hecho de que no es seguro que en una partida se presenten y sean decisivos puede hacer que la experiencia de comparación sea una tarea interminable y de resultados no tan exactos. Por esto prácticamente no se ha mencionado "cuanta" diferencia en rating tendría una función de evaluación respecto de otra que le sea "comparable" en parámetros y complejidad, pero si se hace referencia a la importancia de considerar la mayor cantidad de patrones posibles en la función (darle mayor conocimiento ajedrecístico) y darles la adecuada medida de importancia respecto de la forma de juego que se crea acomoda más al programa.

### 7.1.3. Bases de Datos

Las bases de datos tanto de aperturas como de finales empezaron a hacerse presente en los programas de ajedrez a principios de los años 70, y tuvieron su desarrollo principal en el año 1975, con los trabajos desarrollados por Thompson [71].

La necesidad de incorporar bases de datos en los programas surge del hecho de dar a éstos una herramienta bajo la cual puedan reducir considerablemente el tiempo del proceso de búsqueda en profundidad al entregarles una base de datos de posiciones las cuales ya están previamente evaluadas. Los primeros programas de ajedrez que carecían de bases de datos de finales presentaban un nivel de juego muy bajo en esta etapa de la partida, tanto así que los programadores decidían terminar las partidas antes de entrar en esta etapa del juego (ver partidas 3 a 5 en Anexo B). El problema era que en la etapa del final de partida los eventos importantes requieren por lo general muchos movimientos para hacerse presentes, por lo cual los programas con su capacidad de profundidad limitada a algunos pocos movimientos eran incapaces de "ver" situaciones que se producirían en el futuro. Las funciones de evaluación tampoco estaban lo suficientemente desarrolladas como para cooperar en la selección de movimientos correctos en esta etapa del juego, lo cual se convirtió en un problema de difícil solución para los programadores de la década de los 60. Ya en la década de los 70 los programas poseían mejoras en los procesos de búsqueda y funciones de evaluación más completas, lo cual hacía que jugasen a un nivel "aceptable" esta etapa del juego, si bien se les reconocía como una debilidad ante la cual los rivales humanos se mostraban superiores. El problema seguía siendo el efecto de no poder llegar a evaluar posiciones de muchos movimientos a futuro, es decir, la búsqueda requería demasiadas iteraciones con tal de poder tomar una decisión correcta.

La llegada de las bases de datos constituyó una enorme mejora para los programas. Bastaba ahora que lograsen llegar a una posición dentro de las conocidas en la base de datos (probablemente almacenadas en una tabla de hash) con lo cual ya no se debía seguir profundizando puesto que se conocía evaluación de este final. El desarrollo de estas bases de datos sorprende hasta el día de hoy, con finales de 5 piezas ya totalmente resueltos por computadora (ver capítulo 5), es decir, para un programa con este tipo de base de datos un final de Rey, Torre y peón versus Rey y Torre está totalmente resuelto a partir de cualquier configuración legal de esas piezas sobre el tablero.

| Strohlem (1970) | | | | Thompson (1986) | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Piezas | | | Movimientos para Ganar | Piezas | | | Movimientos para Ganar | Piezas | Movimientos para Jaque Mate |
| ♔♕ | v/s | ♔ | 10 | ♔♗♗ | v/s | ♔♘ | 66 | ♔♖♖ v/s ♔♖ | 31 |
| ♔♖ | v/s | ♔ | 16 | ♔♕ | v/s | ♔♘♘ | 63 | ♔♕♖ v/s ♔♖ | 35 |
| ♔♖ | v/s | ♔♗ | 18 | ♔♕ | v/s | ♔♗♘ | 42 | ♔♕♘ v/s ♔♕ | 41 |
| ♔♖ | v/s | ♔♘ | 27 | ♔♕ | v/s | ♔♗♗ | 71 | ♔♕♗ v/s ♔♕ | 33 |
| ♔♕ | v/s | ♔♖ | 31 | ♔♖♘ | v/s | ♔♖ | 33 | ♔♕♖ v/s ♔♕ | 67 |
| | | | | ♔♖♗ | v/s | ♔♖ | 59 | ♔♕♕ v/s ♔♕ | 30 |

Cuadro 7.2: Resultados obtenidos en el análisis de finales de 5 o menos piezas

Las bases de datos de apertura cumplen con un fin similar, entregar al programa posiciones pre-evaluadas con tal de reducir su tiempo asignado a la búsqueda en profundidad. A diferencia de las bases de datos de finales éstas entregan referencias de los mejores primeros movimientos a partir de una posición raíz. estas fueron construidas principalmente en base a literatura ajedrecística y análisis previo de las propias computadoras.

El programa que tenga acceso a una base de datos tendrá la posibilidad de que al llegar a una posición conocida realizará un "corte" en su búsqueda en profundidad, asignando inmediatamente el valor que indica la base de datos para esa posición a la rama de la variante que se está calculando. En términos prácticos, esto significa que el programa podrá utilizar más tiempo para calcular más posiciones de otras ramas del árbol de variantes y probablemente lograr analizar uno o más niveles de profundidad. Por lo tanto, la comparación entre programas que poseen una base de datos incorporada (de finales o bien un libro de aperturas) se reduce a una comparación entre programas con distintos niveles de profundidad de búsqueda, lo cual como decíamos al principio de este capítulo significará cerca de 200 puntos de rating por cada nivel de profundidad de ventaja.

Ahora bien, un caso interesante de realizar comparaciones es el cómo comparar distintas bases de datos. Una primera comparación es por su tamaño o bien cantidad de posiciones incorporadas en ella. Mientras más posiciones incorporadas muchas más posibilidades de que el programa "encuentre" su posición actual en la base de datos. Otra comparación se refiere a la evaluación asignada en la base de datos a las posiciones presentes. Si las evaluaciones no son correctas la base de datos podría convertirse en un segundo antagonista del programa, al entregar resultados incorrectos acerca de una posición en la búsqueda. Es por esto que uno de los factores más importantes en las bases de datos es su correctitud en la evaluación de las variantes.

En el día de hoy las bases de datos poseen miles de posiciones la cuales se almacenan en las tablas de Hash de los programas o bien se cran rutinas que consulten la base de datos para cada iteración sobre una posición. El requerimiento principal para la incorporación de la base de datos al programa es una buena cantidad de memoria, esto para la confección de tablas de Hash lo suficientemente grandes.

## 7.2. Hardware

Muchos plantean que el principal avance en las computadoras de ajedrez se debe al notable incremento en las capacidades del Hardware. Esto es en gran medida cierto, si bien no se pueden desconocer los progresos hechos en el desarrollo de software, pero cabe plantearse cuan efectivos, medibles y hasta necesarios han sido estos avances en hardware para las computadoras.

En términos de efectividad muchas veces un avanzado hardware lograba disminuir o hasta omitir deficiencias en el software de los programas. Esto pudo resultar en ciertos casos bastante injusto para aquellos desarrolladores de gran capacidad en la programación del software, puesto que por ejemplo un procesador de mucha mayor velocidad puede superar una deficiencia en el proceso de búsqueda (por ejemplo, un mal algoritmo de poda de variantes), lo cual nos habla de un programa ineficiente en software pero cuya capacidad de hardware logra opacar esta diferencia. Vale decir, como saber si el programa sacaba el máximo provecho de su avance en Hardware si no poseo medidas de la efectividad en software? Perfectamente muchos programadores pudieron haber descansado en las ventajas de un buen hardware sin tener que optimizar los algoritmos ni códigos del software.

Cuan medibles son los avances en hardware? Acá hay buenas medidas de efectividad como la velocidad del procesador central (MegaHertz), la capacidad de memoria del computador (muy importante para las tablas de Hash), la cantidad de palabras de bits que maneja el hardware del programa y en ciertas ocasiones la capacidad de disco para almacenamiento de bases de datos. Estos parámetros marcaban la diferencia entre un programa y otro, si bien aquellos que contaban con buena tecnología durante los 70 aún no lograban un nivel de juego aceptable para competir con Grandes Maestros, motivo por el cual se buscaron otras formas de poder mejorar la capacidad de velocidad de procesamiento de la maquina, y una de ellas fue el Hardware de propósito especializado o conocido como programas de Hardware Dedicado. En el tema de efectividad resultará interesante el hacer una comparación de dispositivos en los cuales pueden implementarse programas de ajedrez y analizar la capacidad potencial del programa frente a limitancias de velocidad de procesamiento, memoria o bien capacidad de disco.

Ahora bien, en estos últimos años los avances en hardware nos siguen sorprendiendo, y los programas logran realizar búsquedas de hasta 30 movimientos lo cual les es suficiente para poder eventualmente derrotar a cualquier Gran Maestro de la elite mundial. El hardware de los últimos años se mostró como suficiente para lograr niveles de juegos notables, por lo cual el interés volvió a la mejora del software para mejorar a los programas. Por qué esta decisión y no mejor seguir progresando a nivel de hardware? A esto apuntamos al decir si los aumentos en la velocidad y memoria eran muchas veces necesarios. Un programa que funciona en forma óptima a una velocidad $v$ podrá funcionar de forma igualmente óptima a una velocidad $2v$, por lo cual los esfuerzos en aumento de la capacidad de velocidad pueden haber llegado a ser incluso innecesarios. Por supuesto no debe ser fácil el determinar el nivel óptimo de hardware de un programa, pero seguramente muchos desarrolladores notaron que el momento adecuado de volver al desarrollo de programas ajedrecísticamente inteligentes fue aquel en que los progresos en hardware no entregaban el valor agregado esperado.

### 7.2.1. Hardware Dedicado

Las computadoras de ajedrez han estado en forma consistente utilizando las últimas capacidades de la tecnología. Con el programa Cheops los 70 vieron la introducción de hardware de propósito especial en programas de ajedrez. Más tarde las Redes de Computadoras hicieron su aparición, en 1983 Ostrich utilizó un sistema Data General de ocho procesadores mientras que en 1985 CrayBlitz utilizó un procesador dual CrayX-MP. Otros programas que utilizaron hardware de propósito especial fueron Belle (Condon y Thompson, 1982), Bebe (Scherzer, 1990), Advance 3.0 y BCP (Welsh y Baczynskyj, 1985). Aparecieron también varios sistemas comerciales experimentales empleando chips de tecnología VLSI de alta performance.

Las implementaciones de Hardware dedicado apuntaron a acelerar los procesos de generación de movimientos, búsqueda en profundidad y también implementar parte de la función de evaluación en ellos, siempre buscando acelerar las velocidades en los procesamientos.

### 7.2.2. El problema del Tiempo

El tiempo es un factor necesario de considerar en la práctica de ajedrez de torneo. Los cálculos de movimientos, decisiones y análisis deben realizarse dentro del ritmo de juego considerado en la partida, lo cual hace del ajedrez una disciplina en que quien sepa tomar la mejor decisión en los límites de tiempo establecidos sea quien tenga las mayores chance de victoria.

El problema del factor tiempo fue el incentivo por el cual el enfoque principal del desarrollo de los programas fue el aumentar la velocidad de cada proceso involucrado en la decisión por un movimiento; generación de movimientos, búsqueda y evaluación. Dada la limitancia de tiempo y la gran cantidad de cálculos a realizar, mientras mayor velocidad de cálculo y mientras más ahorro de análisis de variantes entonces mayor cantidad de jugadas se podrán calcular y una mejor decisión respecto al movimiento a realzar podrá ser tomada. En el caso particular del ajedrez la “explosion” de cantidad de movimientos generados ya en pocos niveles de profundidad hacia imposible tener cálculos precisos, además, mientras más bajamos en profundidad más tiempo es requerido en calcular todos los movimientos del nivel en que nos encontramos, lo cual hará que en la búsqueda requiramos cada vez más tiempo para calcular las posibilidades de cada movimiento. Por ejemplo, si suponemos un programa que logra calcular 100 posiciones por segundo, a una tasa de 20 movimientos posibles por cada posición (una estimación promedio y bastante optimista) obtendríamos una gráfica de Profundidad v/s Tiempo como la de la figura 7.1.

Podríamos suponer que si se dispusiera de un tiempo infinito probablemente cualquier máquina podría descubrir el mejor movimiento a pesar de tener muy baja capacidad de procesamiento, pero acá hay un punto interesante a considerar y que es la relación de la capacidad de memoria con el proceso de búsqueda en profundidad. De nada serviría contar con mucho tiempo si carecemos de la memoria suficiente para poder mantener guardados los millones de nodos generados ya en un quinto nivel de profundidad. En este caso, la

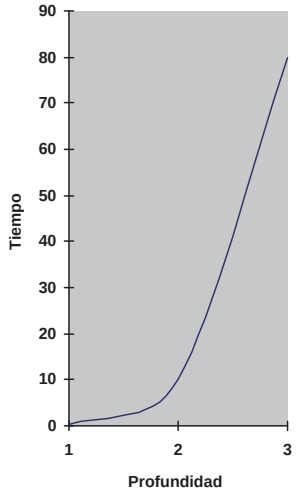



Figura 7.1: Gráfica de Profundidad v/s Tiempo para un programa de capacidad de cálculo de 100 posiciones por segundo y estimación de 20 movimientos por posición

búsqueda debería "ahorrar" la mayor cantidad de nodos mantenidos en memoria con tal de poder almacenar sólo la información relevante. Para este caso lo más apropiado sería realizar búsquedas iterativas que tengan en consideración la limitancia física de memoria, y a este respecto una idea interesante han sido las propuestas de algoritmos de búsqueda como el $A^*$ o más recientemente el $B^*$ [34] ambos con profundidades iterativas y capacidad de memoria limitada.

### 7.2.3. Generador de Movimientos

El principal avance a nivel de hardware relacionado con la generación de movimientos fue la utilización de palabras de mayor tamaño de bits, lo cual está directamente relacionado con la utilización de bitboards para generar jugadas de acuerdo a una regla prediseñada.

Los primeros computadores con cantidad limitada de palabras no podían utilizar la técnica de los mapas de bits, razón por la cual la generación de movimientos pasaba a ser un proceso bastante caro en cantidad de CPU. Los movimientos eran generados a través de fórmulas matemáticas las cuales al no utilizar el tipo de operación básica de los computadores (como es el caso de los bitboards) tomaban un tiempo considerable en ser generadas.

Máquinas de bajo tamaño en palabras de bits deberán hacer uso de su capacidad de procesamiento con tal de compensar la carencia de velocidad en el proceso de generar movimientos. El aumento en el tamaño de palabras de bits en los computadores logró hacer que los movimientos fueran generados directamente en Hardware. Con esto se aceleró considerablemente este proceso y los esfuerzos pasaron a mejorar la búsqueda y la capacidad de la función de evaluación.

La mayoría de las computadoras de hoy utilizan arquitectura de 32 bits (Pentium, Atlon, Cyrix). Esto implica que no pueden manipular en forma eficiente instrucciones de 64-bits. Incluso a pesar de que los últimos compiladores soportan variables de tamaño 64 bits (Delphi, Visual C++, gcc, Watcom, etc.) estos deberán dividir la variable en dos instrucciones de 32 bits antes de enviarlas al procesador para luego volver a juntarlas. Esto es un problema en performance. Las computadoras de 64 bits que podemos encontrar en el mercado no son de alto costo (por ejemplo, el modelo Dec Alpha 21164 tiene un valor cercano a los US 3000) pero no son de uso común como las de 32 bits y se espera que pasen algunos años antes de tenerlas como un estándar.

Por su puesto la velocidad de procesamiento juega acá un rol esencial(como prácticamente en todos los factores), puesto que de no ser suficiente se perdería la capacidad ganada en tiempo con los mapas de bits. De existir poca velocidad, por más rápida que sean las operaciones con bitboards, la lentitud de los otros procesos opacará la labor de esta etapa. El factor de memoria acá no juega un rol tan decisivo puesto que se necesita para mantener los movimientos disponibles a partir de una posición, los cuales promedian 30 en posiciones de medio juego.

### 7.2.4. Búsqueda

El proceso de búsqueda es probablemente uno de los más costosos dentro del proceso general. Este depende directamente de la capacidad de procesamiento del hardware así como la capacidad de memoria.

Máquinas con baja capacidad de procesamiento estarán muy limitadas en el cálculo de variantes en profundidad y el tiempo disponible será un factor de gran importancia. A pesar de esto, en el caso del ajedrez una buena capacidad de memoria podría eventualmente compensar la pérdida de velocidad al hacer posible el uso de tablas de trasposición mediante las cuales muchas variantes del árbol pueden ser desechadas dado su cálculo previo, con lo cual la búsqueda se aceleraría (se limita la cantidad de ramas a calcular). Además, una alta capacidad de memoria permite guardar gran cantidad de posiciones con sus respectivas evaluaciones, con lo cual la búsqueda no llegaría a estar limitada en la cantidad de nodos a procesar (puedo guardar mucha información).

Ahora bien, suponiendo el caso contrario, una gran capacidad de procesamiento y baja capacidad de memoria, se produce otro fenómeno de "compensación" de características. La gran capacidad de cálculo hará posible búsquedas de mucha profundidad, realizando cálculos y evaluaciones en forma muy acelerada. A pesar de esto, la baja memoria haría imposible mantener una tabla de Hash mínimamente aceptable, por lo cual muchas posiciones podrían llegar a analizarse más de una vez produciéndose una sobrecarga debido a las trasposiciones. Muchos dispositivos modernos gozan de esta característica (Palms, microcomputadores) y sus juegos de ajedrez no llegan a un nivel lo suficientemente avanzado puesto que a pesar de sus buenas capacidades de procesamiento la limitancia de memoria hará que se realicen cálculos similares más de una vez. En el ajedrez las transposiciones resultan ser bastante frecuentes por lo que la capacidad de reconocerlas disminuye considerablemente los cálculos

realizados en la búsqueda.

Sería interesante el poder analizar cuanto afecta la reducción de memoria la performance realizada por un programa en su búsqueda. Un experimento a proponer sería el manipular la capacidad de memoria de un programa y analizar los tiempos ocupados en búsquedas o bien cuantos niveles de profundidad alcanza en un tiempo dado. Un problema sería la dependencia de los resultados con la complejidad de la posición a la cual sería enfrentado el programa, puesto que de existir gran cantidad de trasposiciones presentes en ella la carencia de memoria tendrá mayor peso en la demora en el proceso de búsqueda.

**Búsqueda a nivel de Hardware**

La idea de llevar el proceso de búsqueda al hardware fue el aumentar en forma considerable la velocidad de procesamiento del análisis de posiciones en el árbol de variantes. La ventaja de utilizarlo radicaba en ahorrar sumas considerable de tiempos de cálculo los cuales a nivel de software resultaban demasiado caros.

Por lo general el uso que se le ha dado a este proceso en Hardware es una búsqueda completa hasta cierto nivel de profundidad para luego proceder a nivel de software con búsquedas más selectivas.

La búsqueda en hardware logra por lo general captar rápidamente todas las opciones tácticas de la posición (maniobras de corto plazo) las cuales al realizarse con el metodo de fuerza bruta son prácticamente imposibles de omitir.

**El caso de Deep Blue**

De acuerdo a los creadores de DeepBlue en [41], el programa de búsqueda realizaba una búsqueda de ventana nula a profundidad fija incluyendo una búsqueda de posición estable. Para lograr un equilibrio entre la velocidad del hardware de búsqueda y la eficiencia y complejidad de la búsqueda a nivel de software se limitó al chip de hardware a realizar sólo búsquedas de profundidad limitada. Los resultados fueron profundidades de 4 a 5 movimientos con estabilidad en posiciones de medio juego aumentando en algo en los finales. Los principales parámetros del hardware de búsqueda eran los siguientes :

1. Profundidad de búsqueda, el cual controla la profundidad de la búsqueda completa a nivel de hardware. Este era el parámetro principal para controlar el tamaño de la búsqueda.
2. Profundidad de búsquedas posteriores, con tal de detectar condiciones singulares, binarias o trinarias en la posición raíz del hardware de búsqueda.
3. Número de jaques adyacentes al rey permitidos por cada bando en la búsqueda de posiciones estables. Este parámetro controla la búsqueda global de posiciones estables.

4. Distintos Flags que permiten el aumento de un movimiento de profundidad luego de movimientos de peones a séptima fila o bien múltiples piezas sin defensa.

**Búsqueda en paralelo**

Otra aplicación de hardware bastante investigada en los últimos años es el procesamiento de la búsqueda mediante la coordinación de varios procesadores quienes se encargan de realizar la búsqueda en profundidad en ciertos sectores del árbol de variantes entregando sus resultados a un nodo madre para su posterior procesamiento. La principal dificultad con esta técnica radica en la coordinación de los procesadores en el sentido de que cada uno realice un único recorrido por alguna rama del árbol y además que cada procesador realice siempre un trabajo útil, sin quedar 'desocupado". El fin de realizar esta implementación es lograr una mayor velocidad de procesamiento y realizar búsquedas de mayor profundidad.

El trabajo mas exitoso de este tipo de implementación fue el realizado pro Hsu en DeepBlue. Esta supercomputadora estaba compuesta de una computadora RS/6000 SP de 30 nodos y 480 chips de ajedrez (16 chips por nodo).Lo nodos SP se comunicaban con otros utilizando el estándar de "Interface de Paso de Mensajes" (MPI). Cada chip de ajedrez se comunicaba con el nodo madre mediante un bus Micro Channel.

Hsu realizó algunos experimentos de comparación de la performance del sistema de procesamiento paralelo en base a tests para computadoras de ajedrez a los cuales fueron expuestos un sistema Deep Blue de 24 chips con otro de un solo chip. Los resultados variaron de acuerdo a la complejidad de cada problema. Para posiciones con muchas secuencias de movimientos forzados las velocidades promediaban 7 segundos de diferencia, con eficiencias de cerca del 30 %. Para posiciones más "estables" las velocidades promediaron 18 segundos, con eficiencias del 75 %. La naturaleza no determinística del experimento particularmente en posiciones tácticas, hizo difícil el realizar estas mediciones.

### 7.2.5. Función de Evaluación a nivel de Hardware

El ejemplo más emblemático de utilización de función de evaluación a nivel de hardware es el utilizado por DeepBlue en su versión que enfrentó al Ruso Gary Kasparov en 1997. Para cada posición a evaluar en la búsqueda, la función de evaluación realizaba una primera evaluación a nivel hardware tomando en consideración parámetros básicos de la posición (Material, movilidad, dominio central, etc) para luego devolver un "score inicial" al software.

De requerirse una evaluación más acabada de la posición (por ejemplo si es que presenta poca diferencia de score con otras posiciones terminales) el software se encargaba de realizarla. Las ventajas del método utilizado fueron la mejora de la velocidad de procesamiento del programa con un consecuente aumento en la capacidad de profundidad de búsqueda. Los patrones presentes en la función de evaluación de Deep Blue eran cerca de 8000. El trabajo principal obtenido en esta característica del programa fue realizado en el periodo entre los

| Posición | Programa | Características | Rating |
|---|---|---|---|
| 1 | Shredder 7.04 | UCI 256MB Athlon 1200 MHz | 2810 |
| 2 | Shredder 7.0 | 256MB Athlon 1200 MHz | 2770 |
| 3 | Fritz 8.0 | 256MB Athlon 1200 MHz | 2762 |
| 4 | Deep Fritz 7.0 | 256MB Athlon 1200 MHz | 2761 |
| 5 | Fritz 7.0 | 256MB Athlon 1200 MHz | 2742 |
| 6 | Shredder 6.0 Pad | UCI 256MB Athlon 1200 | 2724 |
| 7 | Shredder 6.0 | 256MB Athlon 1200 MHz | 2721 |
| 8 | Chess Tiger 15.0 | 256MB Athlon 1200 MHz | 2720 |
| 9 | Shredder 7.0 | UCI 128MB K6-2 450 MHz | 2717 |
| 10 | Chess Tiger 14.0 | CB 256MB Athlon 1200 | 2717 |

Cuadro 7.3: Los 10 mejores programas de ajedrez según la SSSF (Julio 2003)

matches de 1996 y 1997 frente a Kasparov. De acuerdo a palabras del propio Hsu, la versión de 1997 de DeepBlue poseía una abismante diferencia en términos de conocimiento ajedrecístico lo cual lo ubicaba por varios niveles de juego sobre la versión original del programa.

Mejores funciones de evaluaciones a este nivel otorgan la posibilidad de evaluar con mucha mayor precisión el real score de ella. La ventaja DeepBlue en esta implementación se vio principalmente en el hecho de no cometer graves errores posicionales, lo cual fue la tónica del primer match.

### 7.2.6. El futuro del Hardware

Es un problema a estas alturas la capacidad de hardware disponible para construir programas de ajedrez de gran nivel? Por qué aún no podemos tener a DeepBlue al alcance de la palma de la mano?

Esto suena como el próximo avance de la ciencia, pero la verdad es que aún estamos bastante lejos de que esto ocurra. La capacidad de procesamiento de DeepBlue se basaba en su sistema multiprocesador paralelo, gran capacidad de memoria y disco alojado en dos frames RS/6000, lo cual en estos momentos resulta inimaginable de instalar en dispositivos pequeños. El motivo de esto no es sólo la cantidad de chips necesarios en el dispositivo, sino que además los buses de interconexión que deberá disponer y los mecanismos de enfriamiento para soportar velocidades de cálculo de gran calibre.

A pesar de esto, con los adelantos existentes cualquier computador de capacidades estándares (procesador de 1GHz, memoria de 256 KB) sería útil para que un programa pudiese jugar a nivel de elite, lo cual nos lleva a la conclusión de que cualquier programador lo suficientemente perito podría construir un programa capaz de jugar ajedrez de alto rango. Un ejemplo interesante de evaluación de programas con sus capacidades de hardware adjuntas se muestra en la tabla 7.3 obtenida de [83] y actualizada a Junio de 2003. El puntaje de Elo considera partidos oficiales de los programas.

Llama la atención el hecho de que los programas poseen características de Hardware muy similares y que son las estándares de la época actual. El motivo seguramente es el hecho de que los programas tienen un fuerte carácter comercial, puesto que si estuviesen adecuados a un hardware específico no podrían estar a la libre disposición del mercado.

Los altos ratings Elo de los programas hacen ver la realidad de que el problema del Hardware ya no es una limitante para hacer máquinas que sean capaces de jugar a nivel de elite, vale decir, con un computador de capacidad convencional un buen programador puede en este momento construir un programa que juegue perfectamente al nivel de un maestro de ajedrez.

## 7.3. Métodos Comparativos utilizados en programas

A pesar de los métodos comparativos factibles de aplicar a programas que hemos mencionado en este capítulo, en el desarrollo de los programas ajedrecísticos han existido otros métodos comparativos aplicados los cuales han gozado de importante popularidad a la hora de determinar la fuerza de los programas. Estos son el rating Elo, puntaje asignado a los programas en función de sus resultados en competancias, y también los "tests de resolución de problemas" que consistían en un conjunto de posiciones de problemas de ajedrez los cuales eran presentados a los programas midiéndose los tiempos de respuestas en las soluciones a ellos.

### 7.3.1. El Rating Elo

Durante gran parte de la historia del desarrollo de máquinas que jugasen al ajedrez el principal método de comparación entre éstas fue el rating ajedrecístico que cada máquina poseía, conocido en el ambiente del ajedrez como "Elo" . El Elo de cada programa era obtenido de acuerdo a los resultados frente a jugadores humanos que estaban integrados a la lista de rating ya sea local o bien internacional. A medida que las máquinas fueron ingresando a la lista de rating las competencias entre ellas fueron agregando más programas al listado, llegándose a una situación en que prácticamente todos los programas poseían un rating de competencia.

Esta medida comparativa entre la fuerza de los programas fue una de las más utilizadas a la hora de definir qué programa era mejor que otro. Esto resulta en cierta forma curioso puesto que el rating se compone exclusivamente del resultado frente a otros rivales con rating, sin considerar parámetros propios de las computadoras como composición del software o hardware, vale decir que este tipo de medición era muy relativa al desempeño en competencias de los programas. El punto positivo del Elo era el que los programadores podían realizar un análisis de la evolución histórica del desempeño de sus programas además de medir sus resultados frente a otros programas teóricamente más fuertes o débiles. El rating resultó ser una medida de fácil comprensión para las personas y fue rápidamente propagándose como la

medida principal de comparación entre computadoras.

Hitos interesantes respecto de esta medida de fuerza pueden observarse en la figura 7.2, en donde se agregan los ratings de los campeones mundiales de cada periodo.

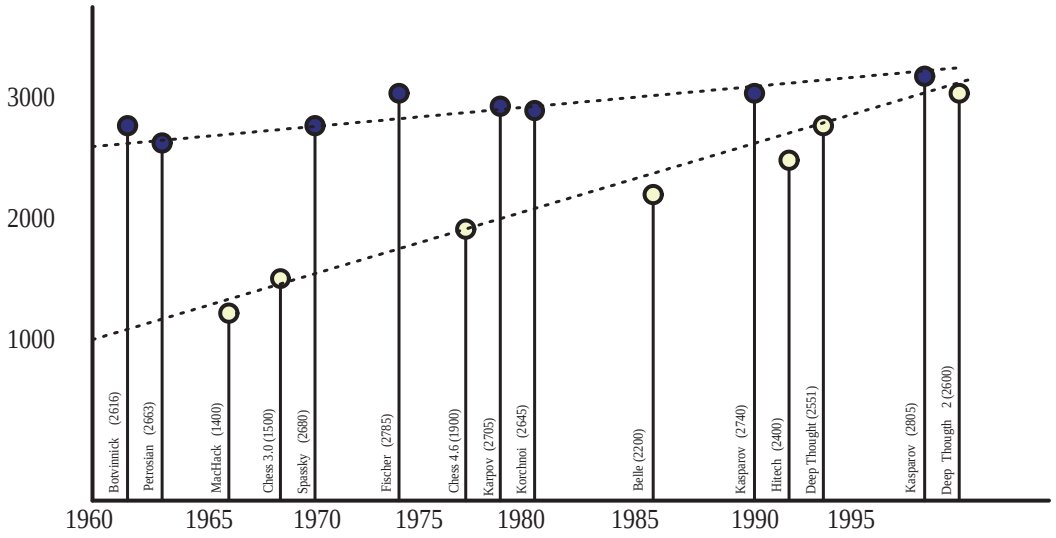
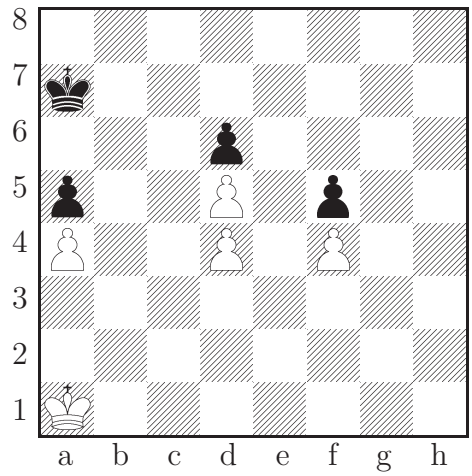



Figura 7.2: Evolución del Rating en los principales programas computacionales

### 7.3.2. Tests de ajedrez para computadoras

Otro método de bastante popularidad en el ambiente de los programadores de computadoras eran las competencias de resolución de problemas. Se seleccionaban distintos problemas de ajedrez, por lo general de muy difícil nivel de solución, y eran presentados a los programas, considerándose parámetro de medición de capacidad del programa el tiempo tomado en cada problema y el tiempo global en resolver todo el "set".

Por lo general cada problema tenía un puntaje distinto de solución respecto de su nivel de dificultad. Programas con mayores niveles de profundidad en búsqueda y hardware que les otorgaba mayor velocidad de procesamiento eran capaces de resolver con mayor facilidad este tipo de contiendas. Por su puesto los problemas presentados debían entonces ser de alta dificultad y con soluciones que implicaran varios movimientos. Algunos aspectos interesantes de los problemas eran que podían medir si el programa poseía ciertas características en su desarrollo de software, como por ejemplo, capacidad de profundidad, tablas de transposición, ciertos patrones reconocidos dentro de su función de evaluación, etc. A modo de explicación la siguiente posición es un típico ejemplo de estos problemas:



*Las blancas ganan mediante 1.♔b1!! ♔b7 2.♔c1!!*

Este tipo de problema puede ser solucionado sólo mediante el uso de Tablas de Transposición, también conocidas como Tablas de Hash. De otra manera, como menciona Newborn en [7], una búsqueda de 30 movimientos sería necesaria, y eso tomaría miles de horas de tiempo de CPU. Por otro lado, la tabla de hash logra descubrir la solución del problema en 6 movimientos, puesto que existen 130 diferentes tipos de posiciones obtenibles desde la inicial. Una vez que ellas son evaluadas como victorias o tablas la búsqueda procede en forma mucho más rápida.

Uno de los tests más conocidos es la colección de problemas de Hyatt y Newborn, la cual utilizaron en 1997 para probar la capacidad de su programa CRAFTY. La colección abarca un total de de 347 posiciones las cuales están clasificadas de acuerdo a las etapas del juego: Test de Aperturas, Test de medio juego y tests de Finales.

Los problemas entregan una forma de medir la velocidad de procesamiento (búsqueda) del programa así como las capacidades incorporadas en su código fuente. Una estimación numérica de la fuerza de cada programa en este tipo de contiendas puede realizarse a partir de los tiempos de respuesta entregados, pero no deberían ser tomados como un valor demasiado exacto a nivel de capacidad de software de los programas dada la dependencia del resultado con la capacidad de hardware de ellos.

# Capítulo 8

# Competencias entre Computadoras: Análisis y Propuesta

Desde el año 1970 se han realizado un sin fin de competencias de carácter oficial entre computadoras de ajedrez. El creciente interés en el desarrollo e investigación de máquinas que jugasen ajedrez de buen nivel hacía necesario el realizar competencias en donde los investigadores pudiesen medir el nivel de sus avances y testear a nivel de torneo la capacidad de sus programas.

La principal organización encargada de realizar estas competencias en los años 70 fue la "American Computer Machinery" (ACM), la cual organizó en forma anual un torneo el cual formaba parte de las conferencias organizadas año a año. En un principio este torneo tomó el nombre de "Campeonato de los Estados Unidos de Computadoras de Ajedrez" pero luego con el pasar de los años y la participación de programas extranjeros fue denominado "Torneo de la ACM". En forma paralela y cada 4 años la "International Computer Games Association" (ICGA) [72] en conjunto con la ACM organizaron los Campeonatos Mundiales de Computadoras de Ajedrez, torneos en que participaban los mejores programas de todo el mundo y daba el carácter de campeón mundial de la especialidad.

Con el pasar de los años y la fama ganada por estos eventos no resultó sorprendente el que se organizaran competencias de jugadores humanos con computadoras participando en ellas. Estas llamaban la atención por la capacidad de los programas de poder hacer pasar algún "mal rato" a algún renombrado maestro, pero más allá de eso permitieron medir la fuerza de juego humano que lograban los programas con el paso del tiempo y cuán distantes estaban aún de poder competir con el Campeón Mundial.

Las competencias de computadoras tienen por su puesto la particular característica de que los participantes no son seres humanos, si no que programas de ajedrez funcionando en computadores. A diferencia de un humano, un programa de ajedrez se mantiene en todo momento en su puesto de juego, no fuma, no molesta al rival ni tampoco se sentirá interrumpido por él. Tampoco habrán problemas de aplicación de reglamento, como casos de "piezas tocadas" o bien partidas que se decidan por exceso en el límite de tiempo. En la sala de juego

se podrá conversar, fumar y hacer ruido puesto que ningún "jugador" será perturbado y mejor aún, la conducta de los jugadores será intachable. Los espectadores estarán rodeados de un excelente ambiente con comentarios en vivo acerca de las partidas jugándose, conversaciones abiertas acerca del futuro de las computadoras y por su puesto observando ese extraño espectáculo de máquinas realizando una actividad tan propia del intelecto humano.

Esto hace pensar de que la competencia será un placer para los árbitros a cargo de ella, pero en realidad se deben considerar factores que para las competencias humanas resultan obvios pero que acá toman relevancia. Por ejemplo, ¿Qué realizar en caso de un corte de energía? ¿Cuanto tiempo se debe dar a un programa en caso de que se "caiga" y tenga que partir nuevamente? ¿Existen las ofertas de tablas? ¿Quien las realiza? ¿Cómo debe ser la relación entre el asistente del programa y éste? Estas son preguntas no fáciles de responder en forma inmediata, y que toda base de competencia entre computadoras debe considerar.

Claro está que acá nos hemos referido hasta ahora a aspectos propios de la organización, pero la intención de este capítulo es primero analizar la forma en que han sido organizadas las competencias entre computadoras y proponer tipos de competencias en las cuales pueda valorarse en forma efectiva la capacidad de programación y adecuación a restricciones de hardware impuestas a los participantes.

## 8.1. Las primeras competencias

En los primeros años de organización de torneos entre computadoras, los programas funcionaban por lo general en sendas computadoras ubicadas en los lugares de origen de los participantes. Estas computadoras se conectaban al lugar del torneo mediante terminales o bien comunicación telefónica entre los operadores. Sólo los participantes que habían desarrollado sus programas en micro computadores podían en ciertas ocasiones llevar sus pequeñas máquinas al lugar de juego. Los torneos fueron realizados desde un principio mediante el sistema suizo, una forma bastante común de realizar los torneos de ajedrez con gran cantidad de participantes y limitado número de rondas la cual se basa en el enfrentamiento entre jugadores que llevan los mismos puntos.

Las reglas de tiempo de reflexión eran similares a las de competencias entre humanos, 2 horas para realizar los primeros 40 movimientos y luego 30 minutos por cada 10 movimientos realizados [1]. El director de torneo tenía el derecho de adjudicar alguna partida luego de cinco horas de juego. Si un equipo tenía dificultades técnicas (problemas del programa, fallas en la comunicación o bien problemas de hardware) el director puede permitir detener el reloj del programa por un máximo de 30 minutos con tal de que los asistentes solucionen los inconvenientes. Esto será permitido en un máximo de 3 oportunidades durante el transcurso de una partida, pero el tiempo *total* de las interrupciones no puede exceder los 30 minutos.

[1] Cada jugador debe realizar sus 40 primeros movimientos antes de que transcurran 2 horas de su tiempo de reflexión. Luego del movimiento 40 se agregan 30 minutos al tiempo restante y en el tiempo resultante se deben realizar los siguientes 10 movimientos

No se permitían ajustes manuales de los parámetros del programa durante el curso de las partidas. En caso de fallas, los parámetros del programa deben volver a sus valores originales, y la información propia de la partida como derecho de enroque, capturas al paso, etc. deben ser ingresadas al programa luego de una falla. Si el programa consulta por el tiempo restante en su reloj o en de su oponente debe proveerse esta información, sin embargo, es el programa quien debe iniciar el requerimiento de esta información.

El primer match entre computadoras fue realizado en 1966 y 1967 entre el programa soviético ITEP y el programa norteamericano Kotok/MacCarty desarrollado en la universidad de Stanford. Cuatro partidas fueron jugadas en forma simultánea vía comunicación con telégrafo. El match se inició en noviembre de 1966, continuando en 1967 y terminando con 2 victorias para el equipo soviético y dos empates (ver apéndice B). Luego en 1970 el primer gran torneo entre computadoras fue organizado en Nueva York, a cargo del profesor M. Newborn, jugándose entre el 31 de Agosto y 1 de Septiembre. En esa fecha ya existía un número interesante de programas desarrollados en Estados Unidos, existiendo un real interés por poder comparar sus reales niveles de fuerza. Uno de los principales objetivos de estos torneos era el desarrollar un formato que fuese interesante de observar y además que entregase medidas de comparación razonables. Respecto de esto Newborn cita en [16]:

*Las reglas bajo las cuales se realizaría el torneo fueron formuladas varias semanas antes de su realización. La comisión directiva estaba formada por Keith Goren, Dennis Cooper, Tony Marsland y el autor. Todas las reglas del torneo fueron de aceptación general del comité director, a excepción de una. La comisión estuvo de acuerdo en que todos los movimientos de cada programa debían realizarse en un límite de tiempo arbitrario, pero Marsland proponía que el tiempo total entregado a cada programa debería depender de la velocidad de la computadora. Computadoras más rápidas debían recibir una menor cantidad de tiempo que otras más lentas. El resto de nosotros creyó que sería imposible determinar una medida justa de handicap puesto que hay muchos factores aparte de la velocidad que hacían a una computadora "mejor" que otra. A pesar de que estábamos consientes de que no era la forma más justa de proceder respecto de las diferencias entre las computadoras, se decidió que todas recibirían una similar cantidad de tiempo de reflexión..*

En esta reflexión expresada por Newborn queremos centrar la discusión de este capítulo. Las capacidades de las computadoras dependerán directamente de su capacidad de hardware, pero en competencias entre ellas no es el ideal el que el primer puesto esté determinado por una diferencia entre el hardware del ganador y el del resto de los participantes. ¿Cómo limitar entonces las diferencias de Hardware? ¿Qué parámetros deben considerarse para establecer las limitancias? ¿Qué aspectos consideran limitar un parámetro u otro? Nuestra propuesta de bases tratará de responder estas consultas entregando información acerca de los aspectos considerados en la manipulación de un parámetro u otro.

## 8.2. Competencias contemporáneas

Desde la primera competencia oficial realizada en 1970 hasta la actualidad (2003) la ACM ha organizado mas de 20 torneos, se han jugado 10 campeonatos mundiales de ajedrez de Computadoras [79] y sin dudar una infinidad de campeonatos en que tanto humanos como computadoras son protagonistas. Uno de los torneos mas populares en que tanto humanos como computadoras participaron fue el realizado en forma anual entre 1985 y 1997 por la empresa holandesa AEGON. El campeonato gozó de bastante popularidad y fue cubierto en las principales publicaciones de ajedrez así como de ciencia en Inteligencia Artificial. Lamentablemente el desencanto del "boom de ajedrez y computadoras" ocurrido luego del match de Kasparov y DeepBlue en 1997 no permitió contar con los fondos suficientes para realizar una nueva edición en 1998.

En la actualidad la cantidad de campeonatos organizados son bastante menos que en la década de los 80. El principal problema son por un lado los costos de organización y a la vez el bajo interés y pasión que despierta el tema en comparación a años anteriores. La ICGA [72] ha realizado muchos esfuerzos para poder mantener su calendario de competencias, puesto que los intereses de auspiciar un evento de computadoras van más por el lado comercial o bien de incluir a un humano en el evento.

Las reglas de las competencias han cambiado muy poco respecto de las primeras realizadas en la década de 1970. Los programadores pueden disponer de cualquier clase de hardware para correr sus programas y las reglas están más enfocadas a aspectos relativos a conectividad y problemas a presentarse durante el transcurso de las partidas.

## 8.3. Competencias realizadas en Chile

Costaría creer que en nuestro país se hayan organizado competencias entre computadoras que jueguen al ajedrez. El estímulo que existe detrás de estas actividades está ligado por lo general a una buena difusión del ajedrez como deporte y a la vez a un buen nivel de investigación en computación dentro del país. Lamentablemente en estos dos campos Chile no han logrado aún una madurez deportiva ni científica que haga pensar en la realización periódica de actividades como ésta, si bien dentro del ambiente del ajedrez propiamente tal la participación de computadoras es observada con bastante entusiasmo.

En el año 1996 la institución educacional denominada "Instituto Tecnológico de Computación", ubicada en Santiago, propuso al "Club de Ajedrez Chile"(El principal club de ajedrez del país) realizar un match entre destacados jugadores nacionales y programas de ajedrez los cuales correrían en computadoras de la época. Cada "equipo" estaba integrado por 5 jugadores. Cada integrante de un equipo jugaba contra todos los integrantes del equipo rival, lo cual hacía un total de 5 partidas. Las partidas se jugaron al ritmo de 1 hora finish, y con computadores 386 los cuales eran los más modernos de ese año. Por el equipo "humano" participaron los jugadores Pedro Donoso y César Velásquez (ambos Maestros FIDE), Cristian

Michel y Jorge Egger (ambos Maestros Internacionales) y otro jugador cuya referencia no ha sido encontrada. Por el equipo de los programas jugaron una de las versiones de la época de "Crafty", el programa "M-Chess", "Sargon", "Genius 5" y otro programa cuya referencia no se ha encontrado. Los programas corrían bajo hardware de similares características y eran asistidos por académicos del instituto organizador. No hubo una especificación de bases adecuada al tipo de encuentro a realizar.

El resultado de la contienda fue una rotunda victoria para el equipo "humano" con el MF Pedro Donoso obteniendo la mejor puntuación (4,5 pts en 5 partidas). A pesar de esta derrota los programas provocaron más de algún desagrado a los jugadores. Cabe notar también que los programas no eran de lo mejor que existía en la época, y que el evento tuvo un fin más comercial que de investigación.

Otra participación interesante de programas computacionales en torneos nacionales fue la inclusión del programa "Fritz 4" en un torneo de ajedrez semi rápido realizado el año 1997 en el Mall Alto Las Condes y organizado por la Liga Nacional de Ajedrez de Chile. El programa participaba como un jugador más y corría en una computadora de marca "Acer" (firma que auspiciaba el evento) de la serie 386. Las reglas impuestas por los organizadores a la computadora fueron la exclusión de obtener cualquier premio y además el pago de un incentivo a los jugadores que empataran o la derrotasen. El resultado del programa en el torneo fue bastante por debajo de lo que se esperaba, no quedando entre las 10 primeras posiciones en un torneo que congregó a cerca de 100 jugadores de nivel semi-aficionado.

Participaciones aisladas de programas se han observados en los campeonatos de ajedrez rápido realizados en forma semanal en los clubes de ajedrez del país. En estos torneos los programas se mostraban tremendamente efectivos, obteniendo los primeros puestos en varias oportunidades. Una de estas participaciones fue del programa "Genius 6" en los torneos de ajedrez rápido del Club de Ajedrez Chile. Los únicos problemas del programa estaban ligados a sus propios apuros de tiempo y a la poca experiencia de sus operadores.

Cabe suponer que estas actividades tuvieron como común denominador el "boom" publicitario de los encuentros entre Kasparov y Deep Blue, evento que recorrió todo el mundo como noticia e incentivó la realización de este tipo de actividades. Lamentablemente la moda de computadoras en torneos ya no es tan fuerte como años atrás, lo cual ha hecho que prácticamente desaparezca esta costumbre, si bien en nuestro país nunca tuvo otro fin que fuese la recreación en vez de uno científico.

## 8.4. Parámetros a considerar

En el capítulo 7 se discutió acerca de una serie de parámetros que deben ser de consideración dentro de la fuerza demostrada por un programa de ajedrez tanto a nivel de Software como de Hardware. Los puntos que consideraremos en la propuesta de bases con sus respectivas observaciones de la influencia en la performance del programa son los mostrados en la tabla 8.1.

## 8.5. Propuesta de Bases

### BASES GENERALES COMPETENCIA DE COMPUTADORAS DE AJEDREZ

El torneo se regirá por las Leyes del Ajedrez establecidas por la Federación Internacional de Ajedrez en 1996, aplicándose las siguientes adaptaciones y extensiones para efectos de esta competencia.

1. **Adaptaciones**

   *a*) Acerca del artículo 4: Finalización de una Jugada.
   Cada computadora estará acompañada por un operador quien deberá realizar todas aquellas tareas que la computadora no pueda realizar por si misma. En particular, tan pronto como la computadora defina su próxima jugada a realizar éste debe ser realizado sobre el tablero de juego así como ingresar a la computadora el movimiento realizado por el rival. Una jugada estará finalizada cuando el operador la haya sobre el tablero. Si un error del operador significa una desventaja para su oponente, el árbitro deberá compensar el tiempo perdido por el oponente. Ambos relojes deberán ser ajustados si es necesario.

   *b*) Acerca del artículo 5: Finalización de la Partida.
   El operador tiene el derecho de abandonar una partida, ofrecer tablas o aceptar un ofrecimiento de tablas realizado a la computadora. Si una partida finaliza de alguna de las maneras mencionadas, el arbitro podrá decidir si el partido debe continuar. Reclamos de tablas basados en el artículo 9.2 (repetición de jugadas) o 9.3 (regla de las 50 jugadas) deben ser realizadas pro la propia computadora.

   *c*) Acerca del artículo 6: El reloj de Ajedrez.
   Si una computadora no puede operar el reloj de juego, esto deberá ser realizado por el operador. Mientras las computadoras se mantengan jugando en base a su libro de aperturas no habrá necesidad de utilizar el reloj.

   *d*) Acerca del artículo 7: Irregularidades.
   Si durante la partida se observa que una pieza es ubicada en una casilla incorrecta, la posición deberá ser restaurada hasta el momento antes de ocurrir la irregularidad, y la jugada indicada por la computadora debe ser hecha sobre el tablero. El

| Parámetro | Características | Cualidades a Medir |
|---|---|---|
| Capacidad de Memoria (RAM) | 64MB hasta 512MB | Limitar la capacidad de las tablas de Hashing. Optimizar el algoritmo de búsqueda en base a memoria limitada. |
| Velocidad de procesamiento | 200MHz a 1.8GHz | Optimización de los algoritmos de búsqueda. Mejora en otros procesos que simplifiquen el trabajo de la búsqueda. |
| Tamaño de palabras de bits | 32 o 64 | Medir diferencias en el proceso de generación de movimientos en base a implementaciones con mapas de bits. |
| Algoritmo de Búsqueda | Fijo | Medir la capacidad de implementación de otros aspectos del software como función de evaluación y generador de movimientos. |
| Algoritmo de Búsqueda | Modificable | Medir la capacidad de mejora del proceso de búsqueda en profundidad a partir de un código base. |
| Función de Evaluación Base | Fija | Medir la capacidad de mejora del proceso de búsqueda y comparar los resultados |
| Función de Evaluación Base | Modificable | Medir la capacidad de implementación de conocimiento ajedrecístico en el programa. |
| Código Fuente Base | Código Fijo | Medir la adaptabilidad y diferencias a nivel de hardware entre un programa y otro. |
| Código Fuente Base | Código Modificable | Medir la real capacidad de mejora de un programa a partir de una capacidad fija. |

Cuadro 8.1: Parámetros a considerar Competencias de Computadoras de Ajedrez

tiempo utilizado será reducido por el arbitro, en base al promedio de movimientos realizados y tiempo completo de juego. El árbitro podrá agregar tiempo extra al oponente en caso de necesidad de reconstrucción de las Tablas de Hash de su programa.

*e*) Acerca del artículo 8: Notación de la partida.
A pesar del hecho que las computadoras mantienen un registro actualizado de las jugadas, el operador deberá mantener un registro completo de ellas y facilitar este registro a la organización en caso de que lo requiera.

*f*) Acerca del artículo 12: La conducta de los jugadores.
Está prohibido distraer o interferir la operación del oponente en cualquier forma.

*g*) Acerca del artículo A: El aplazamiento de la Partida.
Si una partida necesita ser aplazada, el árbitro se asegurará que sólo él y el operador correspondiente conocen el movimiento sellado. El chequeará también que el movimiento correspondiente sea el guardado bajo sobre.

2. **Extensiones**

*a*) Con tal de administrar en forma correcta el tiempo de las partidas ante problemas de funcionamiento de las computadoras en cada juego se utilizará un reloj extra que será operado sólo por el árbitro.

*b*) Este reloj extra deberá ser utilizado en caso de ocurrir problemas de funcionamiento que no requieran ajuste de tiempo de juego. Los problemas de funcionamiento se dividirán en tres casos :

1) Ante fallas de energía las partidas deberán ser interrumpidas hasta que el juego pueda volver a continuar. Si es necesario podrá compensarse con tiempo extra con tal de recuperar las pérdidas en Tablas de Hash.
2) Ante una falla de la computadora de un participante (hardware, líneas de comunicación o software de sistema) se le dará al operador la oportunidad de resolver el problema y continuar el partido, o bien restablecer la comunicación.
3) Ante errores de programación el operador tendrá la oportunidad de resolver el problema y continuar la partida. Para esto el error del programa debe ser corregido (editar, compilar y linkear), y/o el programa deberá ser reinicializado con otros parámetros. Eventualmente una versión anterior del programa deberá ser reutilizada.
   El máximo de tiempo disponible para cada jugador para errores de tipo ii) o iii) será de 15 minutos. Luego de transcurrido este tiempo el reloj del jugador será reiniciado y el problema de funcionamiento deberá resolverse con el tiempo del jugador.

*c*) Requerimientos y/o comandos desde la computadora hacia el operador no podrán ser ambiguas y deberán ser atendidas por el operador en forma correcta y sin mayor pérdida de tiempo.

Los requerimientos serán limitados a: Ingresar tiempo utilizado o remanente. Ingresar resultados de otras partidas.

Los comandos estarán limitados a: Cambiar o conectar algún dispositivo (por ejemplo diskettes o CD-Roms).

El operador no deberá influir en la ejecución del programa. No está permitido cambiar los niveles de juego durante la partida salvo con permiso expreso del árbitro. Está permitido sin embargo ingresar el tiempo disponible para el programa.

*d*) El tiempo de juego será de 90 minutos para toda la partida. Esto implicará el uso forzado del artículo 10 de la FIDE (acerca del juego rápido a finish). No habrá compensaciones de tiempo por lentitud en líneas de comunicación.

*e*) Si un participante no está de acuerdo con alguna determinación del árbitro podrá apelar al comité de apelaciones. Este comité consistirá en tres personas más dos miembros auxiliares quienes actuarán en caso de que alguno de los miembros titulares esté involucrado en la apelación. El comité será designado antes del inicio de la competencia y decidirá por mayoría de votación.

## BASES ESPECIFICAS

1. Acerca del Hardware a Utilizar. Las limitancias de hardware a utilizar para los participantes serán las siguientes. Memoria : 256MB RAM Velocidad de Procesador : 1GHz Arquitectura : 32bits

   En caso de que algún participante no disponga de hardware para la competencia, la organización le facilitará una computadora con las características especificadas.

2. Categorías de participación.

   Habrá 3 distintos tipos de competencias, basadas en las siguientes categorías:

   *a*) Categoría Programación. Se entregará un código fuente base de un programa de ajedrez el cual podrá ser modificado pero sólo en la agregación de código fuente. El código base deberá mantenerse intacto. El hardware a utilizar en esta categoría será el especificado por la organización en el punto A.

   *b*) Categoría Algoritmo de Búsqueda. Se entregará un código fuente base consistente en la Función de Evaluación y el Generador de Movimientos de un programa de ajedrez. Los participantes deberán agregar a este código, sin alterarlo, el algoritmo de búsqueda del programa. El hardware a utilizar en esta categoría será el especificado por la organización en el punto A.

   *c*) Categoría Función de Evaluación. Se entregará un código fuente base consistente en el Algoritmo de Búsqueda y Generador de Movimientos de un programa de ajedrez. Los participantes deberán agregar a este código, sin alterarlo, la Función de evaluación. El hardware a utilizar en esta categoría será el especificado por la organización en el punto A.

   *d*) Categoría Parámetros Ajedrecísticos. Se entregará un código fuente base de un programa de ajedrez en cuya Función de Evaluación se identificarán cerca de 50 parámetros, cada uno asociado a un patrón ajedrecístico (por ejemplo, material, dominio central, desarrollo, etc). Los participantes dispondrán del código fuente 1 día antes del inicio de la competencia y sólo podrán alterar los valores asignados a cada parámetro con tal de modificar los valores entregados por la función de evaluación del programa. El hardware a utilizar en esta categoría será el especificado por la organización en el punto A.

*e*) Categoría Restricción de Memoria. Los participantes deberán correr sus programas bajo las condiciones especificadas en el punto A, pero con una memoria máxima de 64MB de RAM.

*f*) Categoría Restricción de Velocidad de Procesamiento. Los participantes deberán correr sus programas bajo las condiciones especificadas en el punto A, pero con una velocidad de procesamiento máxima de 100MHz.

# Capítulo 9

# Conclusiones

El desarrollo de máquinas que juegan al ajedrez ha abarcado el interés del hombre desde hace más de 100 años. La primera inventiva, denominada "El Turco", a pesar de ser una simple farsa fue el primer indicio de una carrera de varios años y esfuerzos de muchas horas de trabajo en mejorar la calidad y capacidad de los programas con el objetivo de poder derrotar al mejor jugador humano de la especialidad. Este interés estaba ligado a la relación que se podía establecer entre las soluciones encontradas al problema del ajedrez y su aplicación en otras áreas de la Inteligencia Artificial.

Las primeras ideas apuntaron a imitar la forma de pensar del jugador humano, con rápidas selecciones de movimientos y evaluaciones basadas en aspectos estratégicos. Lamentablemente, esta primera intención chocó contra los pésimos resultados obtenidos por los programas y las tremendas dificultades que significó el tratar de modelar la forma de jugar ajedrez de los humanos, motivo por el cual el fin de los 60 marcó el inicio de una nueva estrategia para enfrentar el problema, los programas de "fuerza bruta"

Los buenos resultados obtenidos por estos programas junto con los continuos avances en Hardware y desarrollo de Software crearon nuevas esperanzas a los de poder crear una máquina que derrotase al hombre, si bien la idea de "buscar un movimiento más" y el desarrollo enfocado a extender lo más posible la fuerza de cálculo (aspectos tácticos) dejó muy de lado el desarrollo ligado a la comprensión propia del ajedrez (aspectos estratégicos del juego).

En época muchos opinaron acerca del tiempo restante en en derrotar al campeón mundial, pensando en que los próximos adelantos en hardware serían suficiente para que en base a un cálculo concreto de muchas jugadas se pudiese superar la desventaja estratégica frente a los jugadores humanos. Los resultados a principio de los 80 resultaron ser bastante sorprendentes, pues si bien muchos programas ya eran capaces de desarrollar un nivel de juego aceptable la idea de derrotar al campeón mundial se veía aún muy lejana. A pesar de aventajar notablemente en cálculo a los humanos, los programas carecían de una comprensión estratégica suficiente, lo cual los hacía cometer errores que serían explotados muchos movimientos después.

El enfoque volvió entonces hacia el lado de la comprensión del juego. La función de evaluación de los programas tomó un carácter de neta importancia y el trabajo junto a los expertos empezó a ser bastante común. El año 1997 marca el clímax de esta larga historia con la derrota del campeón mundial Kasparov por el supercomputador DeepBlue. El objetivo fue cumplido y el interés en el tema bajó notoriamente. El proceso en sí estuvo demasiado enfocado a este objetivo, y dentro de su desarrollo existieron muchos aspectos que probablemente fueron desestimados en búsqueda del logro principal. Dentro de estos estaba la medición de la real capacidad de "inteligencia" de las máquinas que juegan al ajedrez, qué aspectos hacían a unas más fuertes que otras y qué herramientas podían utilizarse para medir estos aspectos. Más allá de un adelanto en la velocidad de procesamiento o bien la memoria del hardware en donde funcionase el programa, hay un desarrollo de software que no necesariamente fue comparado ni evaluado de adecuada manera. Se "descansó" en el hecho de que la evolución del hardware compensaría algunas posibles deficiencias de eficiencia y optimización del software. Esta idea quedó expresada en las competencias realizadas entre computadoras de ajedrez, en donde la medición de fuerza ajedrecística estaba reflejada por la obtención de puntos y resultado en una tabla de posiciones sin tomar en cuenta aspectos propios del hardware en donde funcionaba el programa ni la capacidad demostrada por los desarrolladores en el código fuente de ella.

Este trabajo buscó justificar esta hipótesis y proponer herramientas de medición de la capacidad ajedrecística de las máquinas, restringiendo y nivelando ciertas capacidades de su funcionamiento. La tarea fue bastante extensa puesto que debió realizarse un extenso trabajo de revisión y clasificación bibliográfica, determinar cuáles fueron los principales avances históricos realizados desde un punto de vista computacional y ajedrecístico y observar cómo influyen aspectos propios del funcionamiento de los programas en estos avances. Algunos aspectos del funcionamiento de los programas resultaron fáciles de medición, tales como los relacionados con el hardware y la capacidad de búsqueda en profundidad, pero otros ligados a aspectos propios de la comprensión del juego y sobretodo la forma de "ver y sentirlo" no podían ser estandarizados ni evaluados con parámetros similares. En este último aspecto se descubre una capacidad propia del ajedrez y la adaptabilidad humana hacia él en las distintas formas de percepción e intuición.

### El futuro

El principal problema del futuro desarrollo de programas que jueguen ajedrez es la carencia de entusiasmo por crear nuevos avances puesto que el principal objetivo de años pasados ya ha sido cumplido. Aún así, instituciones relacionadas con el tema como la ICGA mantienen su actividad y organizan aún competencias si bien ligadas ya otros juegos. Los avances científicos podrán estar ligados a extender la capacidad de los algoritmos de búsqueda en su selectividad y optimización, y también en poder mejorar la función de evaluación en base a estrategias de aprendizaje. Otra área de interés podrá ser la derivación de estrategias a partir de bases de datos de partidas de ajedrez. En este momento las bases de datos sólo entregan el movimiento óptimo en base a un análisis de neto cálculo para posiciones de 5 o menos piezas. Lo que se necesita son programas que a partir de estas bases de datos puedan deducir estrategias correctas de juego. Una solución a este problema resultaría en un gran

beneficio para la Inteligencia Artificial.

Probablemente en el futuro seguiremos viendo competencias entre computadoras, si bien no podemos asegurar que haya un aumento en el interés en el tema puesto que la capacidad de realizar programas de alto nivel está al alcance de un hardware estándar y un buen manual de programación. A pesar de esto, los avances en dispositivos de mínima capacidad podrán ser un interesante tema de avance para poder probar hardware en espacios físicos cada vez más reducidos.

# Bibliografía

## Libros de Ajedrez y Computadoras


[1] Adelson, Velsky , Arlazarov & Donskoy, "Algorithms for Games", 1988.

[2] Botvinnick, Mikhail , "Computers, chess and long range planning", Heildelberg Science Library, Vol 11, Springer-Verlag, 1970

[3] Botvinnick ,Mikhail , "Computers in Chess, Solving Inexact Search Problems", Symbolic Computation Series, Springer-Verlag, 1980

[4] Ebeling, C. , "All rigths moves: A VLSI Architecture for Chess", MIT Press, 1987

[5] Euwe, Max , "Computers and Chess", 1970

[6] Fabell, Bonsdorff y Riihimaa, "Ajedrez y Matemáticas" Colección de Ajedrez "Escaques", Número 48. Ediciones Martinez Roca.

[7] Frey, Peter , "Chess Skill in Man and Machine", Springer Verlag, 1977

[8] Hsu, Feng-Hsiung , "Behind Deep Blue", Princeton University Press, 2002

[9] Levy, David , "Chess and Computers", 1976

[10] Levy, David , "More Chess and Computers: The Microcomputer Revolution, the Challenge Match, BELLE : 1978-80", 1976

[11] Levy, David & Newborn, Monty , "All about Chess and Computers", 1982

[12] Levy, David & Newborn, Monty , "Chess computer Handbook", 1984

[13] Levy, David , "Computer Chess Compendium", 1989

[14] Levy, David & Newborn,Monty "How Computers play Chess", 1991

[15] Levy, David & Newborn,Monty , "All About Chess and Computers : Containing the Complete Works, Chess and Computers", 1993

[16] Newborn, Monty , "Computer Chess", New York, Academic Press, 1975

[17] Newborn, Monty , "Kasparov Versus Deep Blue : Computer Chess Comes of Age", Springer-Verlag, New York, 1996

[18] Pachmann & Kunhell "Ajedrez y Computadoras", Coleccion Escaques, 1979

[19] Spraklen, D., "SARGON III Computer Chess", 1984.

[20] Turing A.M., "Computing machinery and intelligence", 1950.

[21] Turing A.M., "Digital Computers applied to games", 1953.

## Papers y Publicaciones

[22] "Advances in Computer Chess", vol. 1 M.R.B. Clarke, ed., Edinburgh University Press, Edinburgh, 1977

[23] "Advances in Computer Chess", vol. 2 M.R.B. Clarke, ed., Edinburgh University Press, Edinburgh, 1980

[24] "Advances in Computer Chess", vol. 3 M.R.B. Clarke, ed., Edinburgh University Press, Edinburgh, 1982

[25] "Advances in Computer Chess", vol. 4 D.F. Beal,ed. Pergamon Press, Oxford, England. 1982

[26] "Advances in Computer Chess", vol. 5 D.F. Beal,ed., Pergamon Press, Oxford, England. 1989

[27] "'Advances in Computer Chess", vol. 6 D.F. Beal,ed., Pergamon Press, Oxford, England. 1991

[28] Aldeson-Velsky, Arlazarov, Bitman y Zhivotovsky, "Programming a computer to play chess", Russian Math Surveys, vol 25 1970.

[29] Arlazarov & Futre, "Computer analisis of a rook endgame", Machine Intelligence 9, University of Edinburgh, 1978.

[30] Anantharaman, Campbell y Hsu, "Singular Extensions:Adding selectivity to brute force searching". ICCA Journal, Vol. 11, No. 4, pp.135-143, 1988.

[31] Anantharaman, Campbell y Hsu, "Evaluation Tuning for Computer Chess: Linear Discriminant Methods". ICCA Journal, Vol 20, No 4 , Diciembre de 1997.

[32] Beal, D., "A Generalized Quiescence Search Algorithm", Artificial Intelligence, Vol 43, pp. 85-98, 1990.

[33] Berliner, H.J. "Chess as Problem Solving: the Development of a Tactics Analyzer", Ph.D. Thesis. Pittsburgh: Carniege-Mellon University, 1974.

[34] Berliner, H.J. "The B*-Tree Search Algorithm - A best first proof procedure", Artificial Intelligence, Vol 12, No 1 pp 23-40, 1979.

[35] Berliner, H.J. "Computer chess at Carnegie Melon University", Advances in Computer Chess 4, 1986.

[36] Berliner, H. y Ebeling C. "The SUPREM architecture: a new intelligent paradigm", Artificial Intelligence, 1986.

[37] Bernstein, De Roberts "A chess planning program for the IBM 704", Western Join Computer Conference, 1958.

[38] Boulé, M., Zilic, Z., "An FPGA Based Move Generator for the Game of Chess", Universidad de McGill, Montreal, Canadá, 2002.

[39] Bratko L & Kopec D ., "A test for comparision of human and computer performance in chess", Advances in Computer Chess 3, 1982.

[40] Chase W. , Simon H., "The mind's eye in chess", Visual Information Processing, New York Academic Press 1973.

[41] Campbell, Hoane y Hsu, "Deep Blue", Artificial Intelligence N 134, p 57-83, 2002.

[42] Condon & Thompson, "BELLE chess hardware", Advances in Computer Chess 3, 1982.

[43] Condon & Thompson, "BELLE", Chess skills in Man and Machine, 1983.

[44] Goetsch, G., Campbell, M., "Experimenting with the Null-Move Heuristic", Computers, Chess and Cognition (eds. Marsland and Schaeffer) pp. 158-168, 1990.

[45] Gillogly, J.J. "The Technology Chess Program", Artificial Intelligence, Vol 3, 1972.

[46] Greenblatt R. "The Greenblatt chess program", Proceedings of the Fall Joint Computing Conference, pp. 801-810, San Francisco 1967.

[47] Heinz, Ernst, "DarkThought Plays Chess", Institute for Program Structure and Data Organization, Universidad de Karlsruhe, 1997.

[48] Heinz, Ernst, "DarkThought goes Deep", Institute for Program Structure and Data Organization, Universidad de Karlsruhe, 1998.

[49] Heinz, Ernst, "Scalable Search in Computer Chess", Tesis de Ph.D., Universidad de Karlsruhe, Alemania, 1999.

[50] Hyatt, R.M., "Using time wisely", ICCA Journal vol 7 no 1 1984.

[51] Hyatt R.M., Gower & Nelson, "CRAY BLITZ", Advances in Computer Chess 4, 1985.

[52] Kendall, G., Whitwell, G., "An Evolutioanry Approach for the Tuning of a Chess Evaluation Function using Population Dynamics", Proceedings of the 2001 IEEE Congress on Evolutionary Computation, Seul 2001.

[53] Kister J. Stein, P., "Experiments in chess", J.Assoc Comput Match, vol 4 1957.

[54] Knuth D, Moore R. "An analisis of alpha-beta pruning", Artificial Intelligencem col 6, 1975.

[55] Kotok A. "A Chess planning program for the IBM 7090", AI Proyect 1962.

[56] Marsland, T.A., "Parallel game-tree search", IEEE 1985.

[57] Marsland, T.A., Reinefeld A., "Low Overhead Alternatives to SSS*", Artificial Intelligence, Vol 31, pp. 185-199, 1987.

[58] T.A. Marsland y Y. Björnsson, "Variable Depth Search", Advances in Computer Games, University of Maastricht p. 5-20, 2000.

[59] Michie D., "Brute force search in chess and science", ICCA Journal vol 12 no 3, 1989.

[60] Michie D & Bratko "Ideas in knowledge syntesis stemming from the KBBKN endgame", ICCAA journal vol 8 no 1, 1985.

[61] Mysliwietz, P. "Konstrucktion und Optimierinung von Bewertungsfunktionem beim Schach", Disertation (PhD. Thesis) Universidad de Paderborn, 1994.

[62] Nelson H.L. "Hash tables in CRAY BLITZ", ICCA Journal vol 8 no 1, 1985.

[63] Newborn M. "A parallel search chess program", Proc of 1985 ACM Annual Conf. 1985.

[64] Newborn M. "A hypotesis concerning the strengh of chess programs", ICCA Journal, vol 8 no 4, 1986.

[65] Newell, Shaw & Simon, "Chess planning programs and the problem complexity", IBM Journal if research Development vol 4 no 2, 1958.

[66] Rivest, Ronald, "Game Tree Searching by Min/Max Approximation", Artificial Inteligence vol 34 no 1 p 77-96, 1987

[67] Schaeffer, J., "The Historic Heuristic", ICCA Journal, Vol 6, No 3 pp.16-19, 1983.

[68] Shannon, Claude E, "Programming a computer for playing chess",1950

[69] Slate D.J., "CHESS 4.5 - The Northwestern University Chess Program", Chess Skills in Man and Machine, 1977.

[70] Thompson K., "Computer chess strength", Advances in computer chess 3, 1982.

[71] Thompson, K. Retrograde Analysis of Certain Endgames. ICCA Journal,Vol. 9, No. 3, pp. 131-139, 1986.

## Sitios Web

[72] Página web de la Asociación Internacional de Computadoras de Ajedrez, `http://www.cs.unimaas.nl/ICGA/`

[73] Página web de Paul Verhelst, "Computers Chess", `http://www.xs4all.nl/~verhelst/chess/programming.html`

[74] "Tim Mann Chess Pages", `http://www.tim-mann.org/chess.html`

[75] Dr A. N. Walker's Computer Chess pages, `http://www.maths.nott.ac.uk/personal/anw/G13GT1/compch.html`

[76] "Louis Kessler's Chess and Computer Chess Links",
http://www.lkessler.com/cclinks.shtml

[77] Página web del programa "Crafty",
http://www.limunltd.com/crafty/

[78] Página web de Robert Hyatt,
http://www.cis.uab.edu/info/faculty/hyatt/hyatt.html

[79] Campeonatos Mundiales de Ajedrez Computadoras,
http://www.mark-weeks.com/chess/wcc-comp.htm

[80] V Campeonato de programadores de Computadoras de Ajedrez
http://www.vrichey.de/cct5/cct5.html

[81] Página web del programa Dark Thought,
http://supertech.lcs.mit.edu/ heinz/dt/

[82] Página web de Louis Kessler,
http://www.lkessler.com/cclinks.shtml

[83] Asociación Sueca de Computadoras de Ajedrez,
http://w1.859.telia.com/ u85924109/ssdf/

[84] Chess Base,
http://www.chessbase.com

[85] Chess Programs & Utilities,
http://www.enpassant.dk/chess/softeng.html

# Apéndice A

# Resumen Histórico

1864 C. Babbage especula sobre cómo podría jugar al ajedrez su "Analytic Engine" (antecesor mecánico ideal de las computadoras modernas).

1944 J. von Neumman y O. Morgenstern aducen el ajedrez como caso modelo del teorema del minimax de su teoría de los juegos ("Theory of Games and Economic Behavior").

1945 K. Zuse publica un algoritnmo teórico para el juego del ajedrez.

1949 Claude A. Shannon desarrolla varias estrategias realizables, basadas en el teorema del minimax, y recomienda la programación de dichas estrategias sobre las computadoras entonces existentes [68]. Hasta la fecha, la mayoría de las aplicaciones de programación de ajedrez se fundan en este revolucionario artículo.

1951 A.M. Turing desarrolla una estrategia algorítmica y la pone en práctica (simulando sobre el papel) contra un oponente humano.

1955-58 Aparecen en EEUU los primeros programas de ajedrez: "Los Alamos Chess Program" (J.Kister y otros), "Bernstein Chess Program" (A.Bernstein y otros), y el programa "Carnegie Mellon" (A.Newell, J.Shaw y H.Simon).

1966-67 El programa de ajedrez soviético "ITEP" (Institute of Theoretical and Experimental Physics of Moscow de G.Adelson-Belski y W.Alasarov) derrota en un match a distancia por el score de 3:1 al programa norteamericano "Kotok/MacCarthy" (A.Kotok, A.McCarthy).

1967 El programa "MacHACK VI" es el primero en derrotar a un humano en el Torneo del Estado de Massachussets.

1968 Mikhail Botvinnick realiza una crítica fundamental de los programas existentes y anuncia sus propias investigaciones ("Algoritm Igri v Shajmati").

El maestro Internacional David Levy realiza una apuesta de U$ 3000 dolares en donde afirma que ninguna computadora podra vencerlo en un match de ajedrez en condiciones de torneo. La apuesta fue realizada originalmente con Jon McCarthy, un distinguido investogador en Inteligencia Artificial.

1970 Primer Campeonato Norteamericano de ajedrez de Computadoras, Nueva York. Vencedor "Chess 3.0" (Larry Atkin y David Slate, Northwestern University, Ilinois).

1970-73 Las series de "Chess 3.0" a "Chess 3.6" vencen en los sucesivos campeonatos de computadoras de EEUU.

1972 Los lectores del "Kosomolskaia Pravda" juegan dos partidas a votación contra el programa "Kaissa" (M.Donskoi y W.Alasarov).

1974 *5 al 8 de Agosto*. Primer Campeonato Mundial de Computadoras de Ajedrez realizado en Estocolmo, Suecia. Ganadores : "Kaissa", "Chess 4.0", "Chaos" y "Ribbit".

1975-77 Las series de CHESS 4.4 a CHESS 4.7 ganan los VI y VIII campeonatos de EEUU.

1977 *8 al 12 de Agosto*. Segundo Campeonato Mundial de Computadoras de Ajedrez realizado en Toronto, Canadá. Ganadores : "Chess 4.6", "Duchess" y "Kaissa".

1978 El programa "BELLE" gana el IX campeonato de EEUU. "Chess 4.7" clasificado en segundo lugar.

1980 *25 al 29 de Septiembre*. Tercer Campeonato Mundial de Computadoras de Ajedrez realizado en Linz. Ganadores : "BELLE", "Chaos" y "Duchess".

1983 IV Campeonato Mundial de Computadoras realizado en Nueva York. Ganadores : "Cray Blitz", "Bebe" y "Awit".

"BELLE" es la primera computadora que logra el nivel de Maestro Nacional de los EEUU.

1986 "BELLE" vence en el torneo de la ACM. En segundo lugar remata "Lachex" (Los Alamos Chess EXperiment).

V Campeonato Mundial de Computadoras de Ajedrez realizado en Colonia, Alemania. Ganadores : "Cray Blitz", "Sun Phoenix","Hitech" y "Bebe"

1987 Campeonato de la ACM en Dallas. Vencedor : "ChipTest" (F.Hsu, Carniege Mellon).

1988 *28 al 30 de Mayo*. Torneo open de Pittsburgh con participación de máquinas y humanos (organizado por la fundación Fredkin).DeepThougth 0.01(versión mejorada de ChipTest) ocupa el segundo puesto. ChipTest el quinto.

*24 al 27 de Noviembre*. Campeonato de la ACM en Orlando, Florida. Vencedor : "DeepThougth". Segundo lugar : "Fidelity Challenger".

*24 al 27 de Noviembre*. Software Toolworks Championship realizado en Long Beach, California. DeepThougth obtiene el segundo puesto detrás del GM Antony Miles, y sobre los GM Bent Larsen (a quien derrota en su encuentro personal) y el GM y ex-campeón mundial Mikhail Thal.

1989 La IBM contrata a los creadores de "DeepThougth" Feng-Hsiung Hsu, Joe Hoan y Murray Campbell con tal de apadrinar el proyecto de una computadora que derrote al campeón mundial de ajedrez.

VI Campeonato Mundial de Computadoras realizado en Edmonton, Canadá. Ganadores : "DeepThougth" (score perfecto), "Bebe" y "Cray Blitz".

"DeepThough" vence al GM Antony Miles en una partida de definición del torneo Software Tools de 1988

*22 y 23 de Octubre*. Match entre "DeepThougth" y Gary Kasparov. Vence Kasparov por 2-0.

*Diciembre*. Match entre "DeepThougth" y el MI escocés David Levy. "DeepThougth" vence por 4-0.

1990 *2 de Febrero*. Partida de exhibición entre "DeepThougth" y Anatoly Karpov. Vence Anatoly Karpov.

1991 *17 al 20 de Noviembre*. Campeonato de la ACM en Alburquerque. "DeepThougth II" vence con score perfecto.

1992 El programa Fritz 2 derrota al campeón mundial Garry Kasparov en una partida a 5 minutos por jugador realizada en Colonia, Alemania. Esta es la primera ocasión en que un programa derrota a un campeón mundial en ajedrez blitz.

VII Campeonato Mundial de Computadoras de Ajedrez realizado en Madrid, España. Ganadores : Chessmachine, Zugzwang y Cumulus 2.

1993 *24 al 28 de Febrero*. Encuentros de exhibición entre "DeepThougth II" y equipo nacional de Dinamarca.

Rediseño del Hardware de "DeepThougth II". Creación de una nueva versión llamada "DeepBlue".

1994 *Mayo*. Campeonato de la ACM en Cape May. "DeepThougth" vence con score de 4/5.

Gary Kasparov pierde una partida de ajedrez a 30 minutos por lado frente a programa ChessGenius 2.9 (90 MHz) en el Grand Prix de ajedrez rápido Intel realizado en Londres. Esta es la primera derrota de un campeón mundial frente a un programa en una partida de ajedrez no blitz.

1995 *Mayo*. VIII Campeonato Mundial de Computadoras de Ajedrez realizado en Hong Kong. Ganadores : "Fritz","Star Socrates" y "DeepBlue".

1996 *10 al 17 de Febrero*. Primer match entre Kasparov y "DeepBlue" realizado en Nueva York. Vence Kasparov por +3 -1 =2.

*10 al 17 de Abril*. Se realiza en La Haya, Holanda, el $11^o$ Torneo AEGON de Humanos v/s Computadoras. Entre los 50 jugadores humanos jugaron varios MAestros Internacionales y Grandes Maestros. La mayor parte de las 50 computadoras presentes corrian en un Pentium HP de 166 MH con 16 MB de Memoria RAM. el GM Yasser Seirawan vencio en el torneo con 6 victorias en 6 partidas. La mejor computadora fue "QUEST" con 4.5/6 y una performance de 2652. Las maquinas ganaron el evento con 162,5 puntos frente a 137,5 de los humanos.

1997 *Abril*. Match revancha entre Kasparov v/s "DeepBlue" realizado en Nueva York a un total de 6 partidas. Vence Deep Blue por +2 -1 =3.

1999 *Junio*. IX Campeonato Mundial de Computadoras de Ajedrez, Paderborn, Alemania. Ganadores : Shredder, Ferret, Cilkchess y Fritz.

2002 *Julio*. X Campeonato Mundial de Computadoras de Ajedrez, Maastricht, Holanda (realizado como parte de los VII Olimpiadas Mundiales de juegos de computadoras). Ganadores : "Shredder", "DeepJunior" y "Brutus".

*2 al 22 Octubre*. Match entre el GM ruso Vladimir Kramnik y Deep Fritz, realizado en Manama (Bahrein). Empate 4-4 (+2 -2 =4).

2003 *Enero*. V Campeonato de Programadores de Computadoras de Ajedrez [80]. Realizado en el Internet Chess Club.

*26 de Enero al 7 de Febrero* Match entre Gary Kasparov y DEEP Junior. Empate 3-3 con 1 victoria para cada uno.

# Apéndice B

# Selección de Partidas

Partida 1 : **Simulación de Turing — Humano**
1951

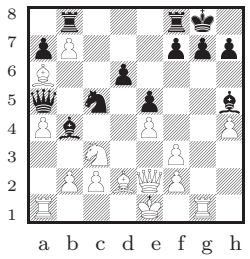
**1. e4 e5 2. ♘c3 ♘f6 3. d4 ♗b4 4. ♘f3 d6 5. ♗d2 ♘c6 6. d5 ♘d4 7. h4**

Este movimiento se explica dados los parámetros de la función de evaluación. Esta asigna un bonus por avanzar los peones, y entregar movilidad a las piezas. El movimiento h4 obtiene un bono de 0.6 por el avance de 2 filas del peon h y por incrementar la movilidad de la torre desde 2 (score 1.4) a 4 (score 2)

**7... ♗g4 8. a4** Evidentemente! **8... ♘xf3+ 9. gxf3 ♗h5 10. ♗b5+ c6?** Obviamente ♘d7 era mejor. **11. dxc6 O–O 12. cxb7 ♖b8 13. ♗a6 ♕a5 14. ♕e2 ♘d7**

El negro podia recuperar un peon con 14... ♗xf3 15. ♕xf3 ♕xa6

**15.♖g1 ♘c5**



**16. ♖g5**

Turing comentaba que el programa se enfrentaba a la perdida del peon "b" e intenta evitar la perdida material tan lejos como sea posible, pero luego de la retirada del alfil atacado el peon de "b7"caerá de todas maneras. Este movimiento es un ejemplo de lo que hoy se conoce como "Efecto Horizonte".

**16... ♗g6 17. ♗b5?**

17. ♗c4! ♖xb7 18. h5 h6? 19.♖xg6

**17... ♘xb7 18. O–O–O?**

18. ♗c4 intentando jugar 19.h5 da clara ventaja al blanco. El programa está mas atraído por el bono entregado por el enroque.

**18... ♘c5 19. ♗c6 ♖fc8 20. ♗d5 ♗xc3 21. ♗xc3 ♕xa4 22. ♔d2**

22. h5

**22... ♘e6 23. ♖g4?**

23. ♗xe6 Era lo correcto. Ahora el caballo negro puede transformarse en una pesadilla.

**23... ♘d4**

23... ♘f4! seguido de la captura del alfil blanco.

**24. ♕d3 ♘b5 25. ♗b3 ♕a6 26. ♗c4**
26.♖dg1

**26... ♗h5 27. ♖g3**

27. ♖g5!

**27...♕a4 28. ♗xb5 ♕xb5 29. ♕xd6**

Luego de 29. ♕xb5 ♖xb5 30. ♖dg1 g6 31. ♔e3 el blanco podía reagrupar sus torres

y aprovechar la ventaja de la pésima posición del alfil negro. **29... ♖d8**

El blanco había omitido la fuerza de este "profundo"movimiento . El programa solo busco a una profundidad de 2 movimientos por lo que cuando consideró 29.♕xd6 y no fue capaz de ver la posición en la que su dama es capturada (que estaba a profundidad 4)**.**

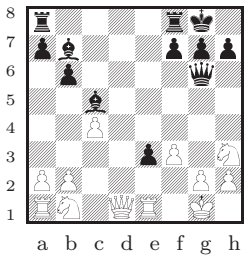
**0–1**

Partida 2 : **Programa de Bernstein — Humano**
1958

**1. e4 e5 2. ♗c4 b6 3. d3 ♘f6 4. ♗g5 ♗b7 5. ♗xf6 ♕xf6 6. ♘f3 c6 7. O–O d5 8. exd5 cxd5 9. ♗b5+ ♘c6 10. c4?**

El blanco ganaba un peon con 10. ♘xe5! pero luego de 11...♕xe5 el programa con su profundidad de 2 movimientos solo ve que el negro gana una pieza por un peon omitiendo la ganadora 12.♖e1.

**10... dxc4 11. ♗xc6+ ♕xc6 12. dxc4 e4 13. ♘g5 ♕g6 14. ♘h3 e3 15. f3 ♗c5 16. ♖e1 O–O**



**17. ♘c3??**

Este movimiento muestra una deficiencia en las rutinas de decision. ♘o hay ninguna rutina quem realice las preguntas ¿puedo dar jaque? o bien ¿puedo atacar una pieza enemiga? con lo cual el siguiente movimiento del negro no estaba en los considerados.

**17... e2+ 18. ♘f2 ♗xf3 19. g3 exd1=♕ 20. ♘cxd1 ♕c2 21. b3 ♖ad8 22. h4 ♖xd1 0–1**

Partida 3 : **Programa NSS — Simon**
Carnegie Mellon, 1960

La siguiente partida fue jugada por Simon contra el programa NSS el cual corría en un computador RAND JOHNIAC. Esta computadora operaba a la mitad de la velocidad de la IBM 704 utilizada por Bernstein. **1. d4 ♘f6 2. ♘c3 d5 3. ♕d3?! b6 4. e4 ♗b7 5. exd5 ♘xd5 6. ♘f3 e6 7. ♗e2 ♗e7 8. ♗e3 O–O 9. O–O ♘d7 10. ♖fe1 c5 11. ♖ad1 ♕c7 12. ♘xd5 ♗xd5 13. a4 (13. c4) 13... ♖ac8 14. ♕c3 ♗f6 15. ♗b5 ♗xf3 16. gxf3 ♖fd8 17. ♗xd7! ♕xd7 18. b3 cxd4 19. ♕d2 ♕c6 20. ♗f4 ♕xc2 21. ♕xc2 ♖xc2 22. ♖c1 ♖dc8 23. ♖cd1 ♖8c3 24. b4 ♖xf3 25. ♗g3 d3 26. ♖c1 ♗g5 27. ♖xc2 dxc2 28. ♗e5 c1=♕ 29. ♖xc1 ♗xc1 0–1**

Partida 4 : **ITEP (3 Movimientos) — Kotok/McCarthy**
Match Stanford - Moscú 1966, partida 1

El programa ITEP utilizaba una estrategia de Shannon tipo A . En dos de sus partidas jugo con profundidad de 3 movimientos mientras que en los otros de 5. El programa americano utilizaba una estrategia tipo B con profundidad de 4 movimientos. Las notas de las partidas fueron tomadas de un articulo de Arlazarov y Bitman aparecido en

Shakhmaty v CCCP en 1968

**1. e4 e5 2. ♘c3 ♘c6 3. ♘f3 ♗c5 4. ♗c4** 4.♘xe5! **4... ♘f6 5. O–O O–O 6. d3 d6 7. ♗e3 ♗g4 8. h3 ♗h5 9. ♗d5** La etapa del desarrollo ha concluido y el programa no sabe que hacer. Acá lo mas razonable era continuar con ♘a4 seguido de c3.

**9... ♗d4**

Pareciera que el programa americano sufre los mismos inconvenientes.

**10. g4 ♗xc3 11. bxc3 ♗g6 12. ♗g5 ♖e8 13. ♖b1 ♖b8 14. ♕e2**

En la variante 14. ♖xb7 ♖xb7 15. ♗xc6 el programa ha sobrepasado los 3 movimientos por lo que no valora el hecho de que recuperará el material y con un peon extra.

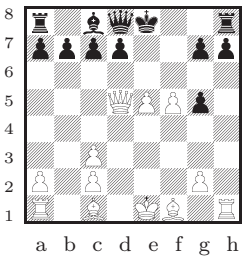
**14... ♔h8 15. d4 ♔g8 16. ♕c4 ♘a5 17. ♗xf6 ♕xf6 18. ♕d3 c6 19. dxe5 dxe5 20. ♗b3 ♖bd8 21. ♕e3 b6 22. ♖fd1 ♖d6 23. g5 ♕e7 24. ♖d3 ♖xd3 25. cxd3 ♖d8 26. ♖a1 ♕d6 27. d4 exd4 28. cxd4 ♘xb3 29. axb3 a5 30. ♖a4 ♕e6 31. ♘e5 ♕e8 32. f4 ♖d6 33. f5 ♗h5 34. ♘c4 ♖d8 35. ♘xb6 ♖b8 36. ♘c4 ♗d1 37. ♖a3 ♗c2 1/2–1/2**

Partida 5 : **Kotok/McCarthy — ITEP (3 Movimientos)**
Match Stanford - Moscú 1966, partida 2

**1. e4 ♘f6 2. e5 ♘d5 3. ♘f3 e6 4. ♗b5 a6 5. ♗a4 b5 6. ♗b3 ♗b4 7. ♘c3 ♘f4 8. O–O ♗b7 9. d4 ♗xc3 10. bxc3 ♘d5 11. ♗xd5 ♗xd5 12. ♗a3 d6 13. exd6 cxd6 14. ♖e1 ♘c6 15. ♖e3 O–O 16. ♕e2 ♗c4 17. ♕e1 ♕c7 18. ♗b4 a5 19. ♗a3 ♔h8 20. ♘g5 h6 21. ♘e4 ♖fd8 22. ♘xd6 ♖xd6 23. ♗xd6 ♕xd6 24. a3 ♘e7 25. ♖e5 ♘c6 26. ♖c5 e5 27. ♕e4 ♖a6 28. ♖d1 g6 29. ♖d2 g5 30. ♖d1 a4 31. ♖d2 f6 32. ♕e3 exd4 33. cxd4 ♘e7 34. ♕g3 ♕xg3 35. hxg3 ♘d5 36. ♖c8+ ♔h7 37. ♖f8 b4 38. axb4 ♘xb4 39. c3 ♘d5 40. ♖c8 1/2–1/2**

Partida 6 : **ITEP (5 Movimientos) — Kotok/McCarthy**
Match Stanford - Moscú 1966, partida 3

**1. e4 e5 2. ♘f3 ♘c6 3. ♘c3 ♗c5 4. ♘xe5! ♘xe5 5. d4 ♗d6 6. dxe5 ♗xe5 7. f4 ♗xc3+ 8. bxc3 ♘f6 9. e5 ♘e4 10. ♕d3 ♘c5 11. ♕d5 ♘e6 12. f5 ♘g5 13. h4 f6 14. hxg5 fxg5**



**15. ♖xh7! ♖f8 16. ♖xg7 c6 17. ♕d6 ♖xf5 18. ♖g8+ ♖f8 19. ♕xf8 1–0**

Partida 7 : **Kotok/McCarthy — ITEP (5 Movimientos)**
Match Stanford - Moscú 1966, partida 4

**1. e4 ♘f6 2. e5 ♘d5 3. ♘f3 ♘b4 4. ♗b5 c6 5. ♗a4 d6 6. d4 ♕a5 7. c4 ♘c2+ 8. ♔f1 ♘xa1 9. ♘c3 ♕b4 10. ♕e2 dxe5 11. dxe5 ♗e6 12. ♕d1 ♗xc4+ 13. ♘e2 b5 14. ♗c2 ♘xc2 15. ♕xc2 ♗xa2 16. ♘ed4 ♕c4+ 17. ♔g1 c5 18. ♕d2 cxd4 19. ♘xd4 e6 20. ♘f3 ♘c6 21. ♕g5 ♖d8 22. ♗d2 ♕c1+ 23. ♗e1 ♕xb2 24. ♕f4 ♗d5**

**25. ♕g3 ♕e2 26. ♗c3 b4 27. ♗e1 ♗xf3 28. gxf3 ♕xe1+ 29. ♔g2 ♕xe5 30. ♕h4 a5 31. ♖c1 ♘d4 32. ♖f1 ♘xf3! 33. ♕h3 (33. ♔xf3 ♖d4) 33... ♖d3 34. ♕g3 ♘e1+ 35. ♖xe1 ♖xg3+ 36. ♔f1 ♕b5+ 37. ♖e2 ♖a3 38. ♔e1 ♖a1+ 39. ♔d2 ♕d5+ 40. ♔e3 ♖a3+ 41. ♔f4 ♕f5 0–1**

Partida 8 : **Lectores del Ural Worker — Programa Soviético**
URSS 1968

**1. e4 ♘c6 2. d4 d5 3. ♘c3 dxe4 4. d5 ♘e5 5. ♗f4 ♘g6 6. ♗g3 f5?! 7. ♗b5+ ♗d7 8. ♘h3! c6! 9. ♗c4 ♕b6 10. ♕d2 ♕c5 11. dxc6 ♗xc6 12. ♗e6! ♘h6 13. O–O–O ♘e5 14. ♘g5 ♘hg4 15. f3 g6 16. fxg4 ♗g7 17. ♗xe5 ♕xe5 18. ♕d8+ ♖xd8 19. ♗f7+ 1–0**

Partida 9 : **KAISSA — Lectores del Komsomolskaya Pravda**
URSS 1972

**1. e4 c5 2. ♘c3 ♘c6 3. ♘f3 d6 4. ♗b5 ♗d7 5. O–O g6 6. d4 cxd4 7. ♗xc6 dxc3 8. ♗xb7 ♖b8 9. ♗d5 ♗g7 10. b3 ♘f6 11. ♗e3 ♕c7 12. ♕d4 a5 13. ♗c4 O–O 14. ♖ae1 ♗c6 15. e5 ♗xf3 16. exd6 exd6 17. gxf3 ♘h5 18. ♕d3 ♗e5 19. ♗d4 ♔g7 20. ♖e3 f6 21. ♖fe1 ♘f4 22. ♕xc3 ♖bc8 23. a4 ♕d7 24. ♗xe5 fxe5 25. ♔h1 ♕h3 26. ♖g1 ♘d5 27. ♕xa5 ♖c5 28. ♕a7+ ♖c7 29. ♕a5 ♖c5 30. ♕a7+ ♖f7 31. ♕xc5 dxc5 32. ♗xd5 ♖f4 33. ♖xe5 ♖xf3 34. ♗xf3 ♕xf3+ 35. ♖g2 ♕d1+ 1/2–1/2**

Partida 10 : **Lectores del Komsomolskaya Pravda — KAISSA**
URSS 1972

**1. b3 e5 2. ♗b2 ♘c6 3. c4 f6 4. ♘c3 ♗b4 5. ♘d5 ♘ge7 6. a3 ♗d6 7. g3 O–O 8. ♗g2 ♘g6 9. e3 f5 10. ♘e2 ♖e8 11. ♕c2 e4 12. d3 exd3 13. ♕xd3 ♖f8 14. f4 ♗e7 15. h4 h6 16. h5 ♘h8 17. e4 d6 18. O–O–O ♖f7 19. ♘xe7+ ♕xe7 20. ♘c3 ♗e6 21. ♘d5 ♕d7 22. ♘e3 fxe4 23. ♗xe4 ♘e7 24. ♗xb7 ♖b8 25. ♗e4 ♘f5 26. ♘d5 a5 27. g4 ♘e7 28. ♘xe7+ ♖xe7 29. g5 hxg5 30. f5 ♘f7 31. fxe6 ♕xe6 32. ♗d5 ♕e3+ 33. ♕xe3 ♖xe3 34. ♖df1 1–0**

Partida 11 : **Robert Fischer — Programa Schneider**
Campeonato Alemán de Computadoras 1975

**1. ♘c3 d5 2. d4 ♗g4 3. f3 ♗f5 4. e4 dxe4 5. fxe4 ♗d7 6. ♘f3 ♘c6 7. e5 e6 8. ♗g5 ♗e7 9. ♕d2 g6 10. ♗d3 b6 11. ♗xe7 ♘gxe7 12. O–O–O O–O 13. ♕h6 ♘f5 14. ♗xf5 gxf5 15. ♘g5 ♕xg5+ 16. ♕xg5+ ♔h8 17. g4 fxg4 18. ♕xg4 f5 19. ♕h4 f4 20. ♘e4 f3 21. ♘g5 ♖f7 22. ♘xf7+ ♔g8 23. ♕f6 f2 24. ♘h6 1–0**

Partida 12 : **MACHACK/CHEOPS — Levy, David**
MIT, 23 de Agosto de 1978

**1. e4 c5 2. ♘f3 d6 3. d4 cxd4 4. ♘xd4 ♘f6 5. ♘c3 g6 6. f4 ♗g7 7. e5 ♘h5 8. ♗b5+ ♗d7 9. e6 fxe6 10. ♘xe6 ♗xc3+ 11. bxc3 ♕c8 12. ♕d4 ♘f6 13. ♕c4 ♘c6 14. ♘d4 ♘xd4 15. cxd4 ♕xc4 16. ♗xc4 ♗f5 17. ♗b5+ ♔f7 18. ♗c4+ d5 19. ♗d3**

**♖hc8 20. O–O ♖c7 21. ♖b1 ♖ac8 22. ♗e3 ♘e4 23. ♖f3 ♘d6 24. ♖b2 b6 25. a4 ♗xd3 26. cxd3 ♖c3 27. ♖h3 h5 28. ♗d2 ♖c2 29. ♖xc2 ♖xc2 30. ♗e1 ♘f5 31. a5 bxa5 32. ♗xa5 ♘xd4 33. ♖e3 ♖a2 34. ♗c7 a5 35. ♖e1 a4 36. ♗e5 ♘c6 37. ♗h8 a3 38. ♖d1 ♖c2 39. ♗a1 a2 40. h3 ♘a5 41. d4 ♘b3 42. f5 ♖c1 43. ♖xc1 ♘xc1 0–1**

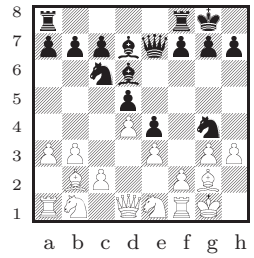
Partida 13 : **Levy, David — CHESS 4.7**
Toronto (m/1), Agosto de 1978
Comentarios de David Levy

**1. g3 d5 2. ♗g2 e5 3. d3 ♘f6 4. ♘f3 ♘c6 5. O–O ♗d7 6. b3 ♗c5 7. ♗b2 ♕e7 8. a3 e4 9. ♘e1 O–O 10. d4 ♗d6 11. e3**

Enfrentando a una fuerte programa de computadora intenté jugar mas bien contra mi oponente que contra la posición. Mi plan era crear una situación en la cual "nada estuviese ocurriendo", con tal de luego expandir mi flanco de dama. Desafortunadamente el programa había aprendido como atacar en el flanco de rey.

**11... ♘g4 12. h3??**

12.c4 debió ser jugado, pero entonces incluso 12...♕g5 es fuerte. Omití completamente el movimiento 13 del negro.



**12... ♘xe3 !!**

El programa respondió en forma automática, indicando que esperaba la jugada anterior y que calculó su respuesta mientras yo estaba pensando!

**13. fxe3 ♕g5 14. g4**

Asumiendo que había sido absolutamente superado, pensé que mi única esperanza era sacrificar la calidad con tal de cambiar las damas.

**14... ♕xe3+ 15. ♖f2 ♗g3 16. ♕e2 ♕xf2+**

Por supuesto habría sido mejor capturar con el alfil y mantener las damas debido a lo expuesto de mi rey, pero el programa asume que debe realizar cambios dada su ventaja en material. **17. ♕xf2 ♗xf2+ 18. ♔xf2 f5 19. gxf5 ♘e7 20. c4 ♖xf5+ 21. ♔g1 c6 22. ♘c3 ♖h5 23. ♔h2 ♖f8 24. ♘d1 ♘g6 25. ♖c1 ♗xh3!**

Yo había previsto esto pero nada podía prevenirlo.

**26. ♗xh3 ♖f1 27. ♘g2 ♖f3 28. cxd5 ♖hxh3+ 29. ♔g1 cxd5 30. ♖c8+ ♘f8 31. ♗c3 ♖d3 32. ♘de3 ♖hxe3 33. ♘xe3 ♖xe3 34. ♗b4**

Mi primera amenaza en todo el partido, si bien no me sentía aún muy feliz con tres peones de menos.

**34... ♖f3 35. ♖d8 h6**

Sin duda que el programa analizó que luego de 35... ♖f5 36.♗xf8 ♖xf8 37.♖xd5 los peones "e" o "b" caerían pro lo que no analizó más allá de este punto.

**36. ♖xd5 ♖xb3 37. ♖d8 ♖f3 38. ♖a8 g5 39. d5 h5 40. d6 ♔g7 41. ♖xa7 ♖f7 42. ♖a5**

Depronto la posición no está *totalmente* perdida, si no que sólo *algo* perdida.

**42... ♔f6 43. ♗c3+ ♔g6 44. ♖e5 ♖f3 45. ♗b4 ♖f4 46. ♖e7 ♖f7 47. ♖xe4 ♖d7 48. ♖e7 h4 49. ♔g2 g4 50. ♔h2?!**

50.♗c5 habría prevenido 50... b6.

**50... b6 51. ♔g2 ♖d8**

Auxilio! el negro está empezando a desenvolver sus piezas.

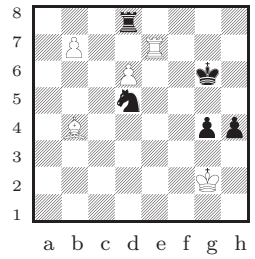
**52. a4 ♘d7 53. a5 ♘f6**

Esperaba 53...bxa6 54.♗xa5 ♖a8 con la idea de que 55.♗c3 mantendría la posición.

**54. axb6! ♘d5 55. b7!**

En este momento la computadora

cayó enferma y debió llamarse a los doctores. Veinticinco minutos más tarde, con el programa teniendo aún mucho tiempo de reflexión, jugó



**55... ♘xe7!**

Una decisión brillante. 55...♘xb4 no gana para el negro debido a 56.♖e4 ♘c6 57. ♖xg4+ ♔h5 58. ♖c4. A la vez 55... ♘f4+ 56. ♔h1 se ve bien para el blanco. CHESS 4.7 había encontrado la única forma en la cual el negro lograba las tablas!

**56. dxe7 ♖h8!** Si 56...♖e8 57.♗a5 gana. **57. ♗d6 ♔f6 58. b8=♕ ♖xb8 59. ♗xb8 ♔xe7 60. ♗f4 ♔f6 61. ♗d2 ♔g6 62. ♗e1 ♔g5 63. ♗f2 ♔h5 64. ♗e1**

David Slate me ofreció las tablas en nombre del programa. La partida tuvo carácter histórico. Era la primera ocasión en que un programa empataba con un maestro internacional en condiciones de torneo. Antes del match David Slate tenía algunas dudas acerca de si su programa estaba preparado para jugar contra mi, pero ta partida las disipó. **1/2–1/2**

Partida 14 : **CHESS 4.7 — Levy, David**
Toronto (m/2), 1978

**1. ♘c3 c5 2. e4 ♘c6 3. f4 a6 4. ♘f3 g6 5. d4 cxd4 6. ♘xd4 ♗g7 7. ♗e3 d6 8. ♘xc6 bxc6 9. ♗e2 ♖b8 10. ♕c1 ♕a5 11. ♗d2 ♕b6 12. ♘a4 ♕a7 13. ♘c3 ♗d4 14. ♘d1 ♘f6 15. c3 ♗b6 16. ♕c2 ♘g4 17. ♕a4 O–O 18. ♗xg4 ♗xg4 19. ♕xc6 ♗xd1 20. ♔xd1 ♗e3 21. b3 ♗xd2 22. ♔xd2 ♖bc8 23. ♕a4 ♕f2+ 24. ♔d3 ♕xg2 25. ♕d4 ♕f3+ 26. ♔c2 ♕e2+ 27. ♔c1 e5 28. fxe5 dxe5 29. ♕xe5 ♖fe8 30. ♕g3 ♖xe4 31. ♕h3 ♖d8 32. ♕f1 ♕d2+ 33. ♔b1 ♖e2 34. ♕xe2 ♕xe2 35. ♖e1 ♕xe1+ 36. ♔b2 ♖d2+ 37. ♔a3 ♕xa1 0–1**

Partida 15 : **Levy, David — CHESS 4.7**
Toronto (m/3), 1978

**1. c4 ♘f6 2. a3 ♘c6 3. ♘c3 d5 4. cxd5 ♘xd5 5. d3 ♘xc3 6. bxc3 e5 7. g3 ♗e7 8. ♗g2 ♕d6 9. ♘f3 ♗e6 10. O–O O–O 11. ♕a4 ♕c5 12. ♗d2 b5 13. ♕c2 f6 14. ♖fb1 ♖ad8 15. ♕b2 ♖b8 16. ♗e3 ♕d6 17. ♘d2 ♗d5 18. ♗xd5+ ♕xd5 19. ♕b3 ♕xb3 20. ♘xb3 f5 21. ♗c5 ♗d6 22. ♖b2 ♔h8 23. ♖ab1 a6 24. ♗xd6 cxd6 25. ♘d2 f4 26. ♔g2 fxg3 27. hxg3 ♖bd8 28. a4 ♘a7 29. ♘e4 bxa4 30. ♖b6 d5 31. ♘c5 ♘b5 32. ♘xa4 ♖a8 33. c4 dxc4 34. dxc4 ♘d4 35. e3 ♘f3 36. c5 ♘g5 37. c6 ♘e4 38. c7 ♖xf2+ 39. ♔g1 ♖ff8 40. ♖b8 h5 41. ♖xa8 ♖xa8 42. ♖b8+ 1–0**

Partida 16 : **CHESS 4.7 — Levy, David**
Toronto (m/4), 1978

**1. e4 e5 2. ♘f3 f5 3. exf5 e4 4. ♘e5 ♘f6 5. ♘g4 d5 6. ♘xf6+ ♕xf6 7. ♕h5+ ♕f7 8. ♕xf7+ ♔xf7 9. ♘c3 c6 10. d3 exd3 11. ♗xd3 ♘d7 12. ♗f4 ♘c5 13. g4 ♘xd3+ 14. cxd3 ♗c5 15. O–O h5 16. ♘a4 ♗d4 17. ♗e3 ♗e5 18. d4 ♗d6 19. h3 b6 20. ♖fe1 ♗d7 21. ♘c3 hxg4 22. hxg4 ♖h4 23. f3 ♖ah8 24. ♔f1 ♗g3! 25. ♖e2 ♗c8 26. ♔g2 ♗d6 27. ♗g1 ♖h3 28. ♖ae1 ♖g3+ 29. ♔f2 ♖hh3 30. ♖e3 ♗a6 31. ♘e2 ♗xe2**

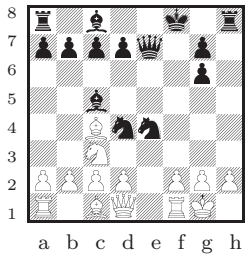
**32. ♖1xe2 c5 33. f4 ♖xe3 34. ♖xe3 ♖h4 35. ♔g3 ♖h1 36. ♗f2 ♖d1 37. ♖a3 cxd4 38. ♖xa7+ ♔f8? 39. ♖d7 ♖d3+ ?? 40. ♔g2 ♗c5 41. ♖xd5 ♖d2 42. b4 ♗xb4 43. ♖d8+ ♔f7 44. ♖d7+ ♔f8 45. ♖xd4 ♖b2 46. ♔f3 ♗c5 47. ♖d8+ ♔e7 48. ♗h4+ ♔f7 49. g5 g6 50. ♖d7+ ♔f8 51. fxg6 ♖xa2 52. f5 ♖a3+ 53. ♔g4 ♖a4+ 54. ♔h5 ♖d4 55. ♖c7 ♗e7 1–0**

Partida 17 : **Levy, David — CHESS 4.7**
Toronto (m/5), 1978

**1. c4 ♘f6 2. a3 c6 3. d3 d5 4. ♕c2 dxc4 5. ♕xc4 e5 6. ♘f3 ♗d6 7. g3 ♗e6 8. ♕c2 ♘bd7 9. ♗g2 O–O 10. O–O ♕b6 11. ♘bd2 ♕c5 12. ♕b1 h6 13. b4 ♕b5 14. ♕c2 ♘b6 15. ♗b2 a5 16. a4 ♕a6 17. bxa5 ♕xa5 18. ♗c3 ♕c5 19. ♖fc1 ♘bd7 20. a5 ♕a7 21. ♕b2 ♘g4 22. ♘e4 ♗c7 23. h3 f5 24. hxg4 fxe4 25. dxe4 ♗xg4 26. ♗e1 ♘c5 27. ♖cb1 ♖ae8 28. ♗d2 ♖f7 29. ♗e3 ♗d6 30. ♕c2 ♗xf3 31. ♗xf3 ♖a8 32. ♖c1 b6 33. ♔g2 ♕b7 34. axb6 ♖xa1 35. ♖xa1 ♘e6 36. ♖a7 ♕c8 37. ♕a2 ♖f6 38. ♖a8 ♗b8 39. ♗g4 ♔f7 40. ♕a7+ ♗xa7 41. ♖xc8 ♗xb6 42. ♗xb6 1–0**

Partida 18 : **BLITZ 6.5 — BELLE**
ACM NACC Washington 1978

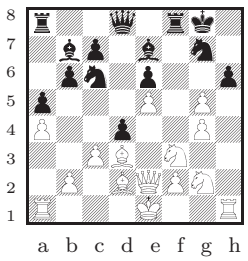
**1. e4 e5 2. ♘f3 ♘c6 3. ♘c3 ♘f6 4. ♗b5 ♘d4 5. ♗c4 ♗c5 6. ♘xe5 ♕e7 7. ♗xf7+ ♔f8 8. ♘g6+ hxg6 9. ♗c4 ♘xe4 10. O–O**



**10... ♖xh2!! 11. ♔xh2 ♕h4+ 12. ♔g1 ♘g3 13. ♕h5 gxh5 14. fxg3+ ♘f3 0–1**

Partida 19 : **BELLE — CHESS 4.7**
ACM NACC Washington 1978

**1. e4 ♘c6 2. d4 d5 3. ♘c3 e6 4. ♘f3 ♗b4 5. e5 ♘ge7 6. ♗d2 ♘f5 7. ♘e2 ♗e7 8. c3 O–O 9. ♘f4 f6 10. ♗d3 fxe5 11. dxe5 g5 12. g4 ♘g7 13. ♘g2 b6 14. ♕e2 ♗b7 15. ♖g1 a5 16. a4 ♔h8 17. h3 ♔g8 18. ♖h1 h6 19. h4 d4 20. hxg5**

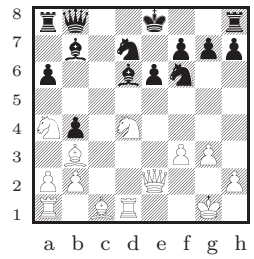


**20... ♘b4 21. gxh6 ?? (21. cxb4 ♗xf3 22. ♗h7+ ♔h8 23. ♖xh6 ♗xe2 24. ♗b1+ ♔g8 25. ♗h7+ ♔f7 26. ♗g6+ con Tablas) 21... ♘xd3+ ??** 21... dxc3 22. ♗h7+ ♔xh7 23. hxg7+ ♔xg7 24. ♗h6+ ♔g8 25. ♖d1 ♗xf3 26. ♖xd8 ♖axd8 27. ♗xf8 ♗xe2 **22. ♕xd3 dxc3 23. ♕g6 cxd2+ 24. ♘xd2 ♖f7 25. hxg7 ♖xg7 26. ♕xe6+ ♖f7 27. ♕h6 ♖g7 28. ♕h8+ ♔f7 29. e6+ ♔xe6 30. ♕xg7 ♗xg2 31. ♖h6+ ♔d7 32. O–O–O ♗d5 33. ♘e4 ♔c8 34. ♖h8 ♗xe4 35. ♖dxd8+ ♗xd8 36. ♕e7 ♔b7**

**37. ♕xe4+ ♔a7 38. ♖g8 ♖b8 39. g5 ♗e7 40. ♖xb8 ♗xg5+ 41. f4 ♗xf4+ 42. ♕xf4 ♔xb8 43. ♔d2 ♔b7 44. ♔d3 ♔c8 45. b4 axb4 46. ♕xb4 ♔d7 47. ♕b5+ ♔d8 48. ♔e4 1–0**

Partida 20 : **CHAOS — NUCHESS**
3rd WCCC, Linz 1980

**1. d4 d5 2. c4 dxc4 3. ♘f3 ♘f6 4. e3 e6 5. ♗xc4 c5 6. ♕e2 a6 7. O–O b5 8. ♗b3 ♗b7 9. ♖d1 ♘bd7 10. ♘c3 ♗d6 11. e4 cxd4 12. ♘xd4 ♕b8 13. g3 b4 14. ♘a4 ♗xe4 15. f3 ♗b7**



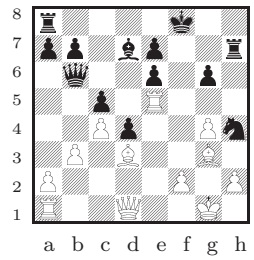
15... e5! 16. ♘e6 fxe6 17. fxe4 ♗c5+ 18. ♘xc5 ♘xc5 19. ♕c4 ♕b5 **16. ♘xe6!! fxe6 17. ♕xe6+ ♔d8**

17... ♗e7 18. ♖e1 ♕d8 19. ♗f4 ♔f8 20. ♖ad1 CHAOS–CHESS 4.0, Estocolmo WCCC 1974

**18. ♖xd6 ♕c7 19. ♗e3 ♖e8 20. ♗b6 ♖xe6 21. ♗xe6 ♗xf3 22. ♖c1 ♖a7 23. ♖xc7 ♖xc7 24. ♗xc7+ ♔xc7 25. ♖xa6 ♗c6 26. ♖a7+ ♔b8 27. ♖a5 ♔c7 28. b3 ♔d6 29. ♗xd7 ♗xd7 30. ♘b6 ♔c6 31. ♘xd7 ♘xd7 32. ♔f2 ♘c5 33. ♔e3 ♔b6 34. ♖a8 ♘e6 35. ♔e4 ♔c5 36. ♖a5+ ♔b6 37. ♖e5 ♘c5+ 38. ♔d4 ♘b7 39. ♖e7 ♘d6 40. ♔e5 ♘b5 41. ♖xg7 h5 42. ♖f7 ♘c3 43. ♖f2 ♔c5 44. ♖d2 ♘b5 1–0**

Partida 21 : **Deep Thougth — HITECH**
19th NACC, Orlando 1988

**1. e4 ♘f6 2. e5 ♘d5 3. d4 d6 4. ♘f3 ♘c6 5. c4 ♘b6 6. e6 fxe6 7. ♘g5 g6 8. ♗d3 ♘xd4 9. ♘xh7 ♘f5 10. ♘xf8 ♔xf8 11. O–O c5 12. b3 d5 13. ♘d2 ♕d6 14. ♘f3 ♘d7 15. ♖e1 d4 16. ♘e5! ♘xe5 17. ♗f4 ♖h7 18. ♖xe5 ♕b6 19. g4 ♘h4 20. ♗g3 ♗d7**



**21. ♖h5!! gxh5 22. ♗xh7 e5 23. ♗xh4 ♗xg4 24. ♕d3 ♖c8 25. ♖e1 ♕e6 26. f3 ♗h3 27. ♕g6 ♕xg6+ 28. ♗xg6 ♖c6 29. ♗xh5 ♖e6 30. ♗g3 ♖a6 31. a4 d3 32. ♖xe5 ♖d6 33. ♖e1 ♖b6 34. ♗f4 a5 35. ♗e3 ♖xb3 36. ♗xc5 d2 37. ♗xe7+ ♔g7 38. ♖d1 ♖e3 39. ♗h4 ♖a3 40. ♗e8 ♖xf3 41. ♗g5 (41. ♖xd2?? ♖f1) 41... ♖f8 42. ♗b5 ♔g6 43. ♗e3 ♖f3 44. ♗xd2 ♖d3 45. c5 ♖d5 46. c6 bxc6 47. ♗xc6 ♖d6 48. ♗f3 ♖d4 49. ♗xa5 ♖xa4 50. ♖d6+ ♔f5 51. ♗c3 ♖a2 52. ♖h6 ♗g4 53. ♗d5 ♖c2 54. ♖c6 ♖e2 55. h4 ♔f4 56. ♖c4+ ♔g3 57. ♗a5 1–0**

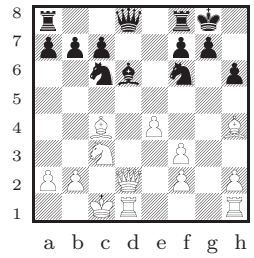
Partida 22 : **Bent Larsen — Deep Thougth**
Long Beach, 1989

**1. c4 e5 2. g3 ♘f6 3. ♗g2 c6 4. ♘f3 e4 5. ♘d4 d5 6. cxd5 ♕xd5 7. ♘c2 ♕h5 8. h4 ♗f5 9. ♘e3 ♗c5 10. ♕b3 b6 11. ♕a4 O–O 12. ♘c3 b5 13. ♕c2 ♗xe3 14. dxe3 ♖e8 15. a4 b4 16. ♘b1 ♘bd7 17. ♘d2 ♖e6 18. b3 ♖d8 19. ♗b2 ♗g6 20. ♘c4 ♘d5 21. O–O–O ♘7f6 22. ♗h3 ♗f5 23. ♗xf5 ♕xf5 24. f3 h5 25. ♗d4 ♖d7 26.**

♔b2 ♖c7 27. g4 hxg4 28. ♖hg1 c5 29. fxg4 ♘xg4 30. ♗xg7 ♖g6 31. ♕d2 ♖d7 32. ♖xg4 ♖xg4 33. ♘e5 ♘xe3 34. ♕xd7 ♘xd1+ 35. ♕xd1 ♖g3 36. ♕d6 ♔xg7 37. ♘d7 ♖e3 38. ♕h2 ♔h7 39. ♘f8+ ♔h8 40. h5 ♕d5 41. ♘g6+ fxg6 42. hxg6+ ♔g7 43. ♕h7+ ♔f6 0–1

Partida 23 : **Antony Miles — Deep Thougth**
Long Beach, 1989

1. d4 d5 2. c4 dxc4 3. e4 ♘f6 4. ♘c3 e5 5. ♘f3 exd4 6. ♕xd4 ♗d6 7. ♗xc4 O–O 8. ♗g5 ♘c6 9. ♕d2 h6 10. ♗h4 ♗g4 11. O–O–O ♗xf3 12. gxf3



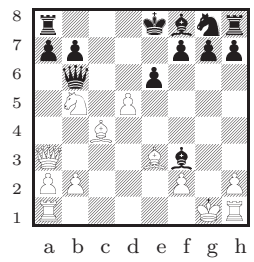
12... ♘xe4 13. ♗xd8 ♘xd2 14. ♗xc7 ♗xh2!! 15. ♗xh2 ♘xc4 16. ♖d7 b6 17. f4 ♖ad8 18. ♖hd1 ♖fe8 19. b3 ♖xd7 20. ♖xd7 ♖e1+ 21. ♘d1 ♘4a5 22. ♔d2 ♖h1 23. ♗g3 h5 24. f5 h4 25. ♗f4 ♘b4 26. a3 ♘bc6 27. ♖d3 ♘d4 28. b4 ♘ac6 29. f6 gxf6 30. ♘c3 ♘e6 31. ♘d5 ♔g7 32. ♗d6 ♘g5 33. ♔e2 h3 34. ♖d1 ♖xd1 35. ♔xd1 ♘e5 36. ♘e3 h2 37. ♘f5+ ♔g6 38. ♘g3 ♘e4! 39. ♘h1 ♘xd6 0–1

Partida 24 : **Deep Thougth — Kasparov**
New York (m/1), 1989

1. e4 c5 2. c3 e6 3. d4 d5 4. exd5 exd5 5. ♘f3 ♗d6 6. ♗e3 c4 7. b3 cxb3 8. axb3 ♘e7 9. ♘a3 ♘bc6 10. ♘b5 ♗b8 11. ♗d3 ♗f5 12. c4 O–O 13. ♖a4 ♕d7 14. ♘c3 ♗c7 15. ♗xf5 ♕xf5 16. ♘h4 ♕d7 17. O–O ♖ad8 18. ♖e1 ♖fe8 19. c5 ♗a5 20. ♕d3 a6 21. h3 ♗xc3 22. ♕xc3 ♘f5 23. ♘xf5 ♕xf5 24. ♖a2 ♖e6 25. ♖ae2 ♖de8 26. ♕d2 f6 27. ♕c3 h5 28. b4 ♖8e7 29. ♔h1 g5 30. ♔g1 g4 31. h4 ♖e4 32. ♕b2 ♘a7 33. ♕d2 ♖4e6 34. ♕c1 ♘b5 35. ♕d2 ♘a3 36. ♕d1 ♔f7 37. ♕b3 ♘c4 38. ♔h2 ♖e4 39. g3 ♕f3 40. b5 a5 41. c6 f5 42. cxb7 ♖xb7 43. ♔g1 f4 44. gxf4 g3 45. ♕d1 ♖be7 46. b6 gxf2+ 47. ♖xf2 ♕xd1 48. ♖xd1 ♖xe3 49. ♖g2 ♘xb6 50. ♖g5 a4 51. ♖xh5 a3 52. ♖d2 ♖e2 0–1

Partida 25 : **Kasparov — Deep Thougth**
New York (m/2), 1989

1. d4 d5 2. c4 dxc4 3. e4 ♘c6 4. ♘f3 ♗g4 5. d5 ♘e5 6. ♘c3 c6 7. ♗f4 ♘g6 8. ♗e3 cxd5 9. exd5 ♘e5 10. ♕d4 ♘xf3+ 11. gxf3 ♗xf3 12. ♗xc4 ♕d6 13. ♘b5 ♕f6 14. ♕c5 ♕b6 15. ♕a3 e6



16. ♘c7+ ♕xc7 17. ♗b5+ ♕c6 18. ♗xc6+ bxc6 19. ♗c5 ♗xc5 20. ♕xf3 ♗b4+ 21. ♔e2 cxd5 22. ♕g4 ♗e7 23. ♖hc1 ♔f8 24. ♖c7 ♗d6 25. ♖b7 ♘f6 26. ♕a4 a5 27. ♖c1 h6 28. ♖c6 ♘e8 29. b4 ♗xh2 30. bxa5 ♔g8 31. ♕b4 ♗d6 32. ♖xd6 ♘xd6 33. ♖b8+ ♖xb8 34. ♕xb8+ ♔h7 35. ♕xd6 ♖c8 36. a4 ♖c4 37. ♕d7 1–0

Partida 26 : **Anatoly Karpov — Deep Thougth**
Harvard (exhibición) 1990

**1. e4 c6 2. d4 d5 3. ♘d2 g6 4. c3 ♗g7 5. e5 f6 6. f4 ♘h6 7. ♘gf3 O–O 8. ♗e2 fxe5 9. fxe5 c5 10. ♘b3 cxd4 11. cxd4 ♘c6 12. O–O ♕b6 13. ♔h1 a5 14. a4 ♗f5 15. ♗g5 ♗e4 16. ♘c5 ♕xb2 17. ♘xe4 dxe4 18. ♖b1 ♕a3 19. ♗c1 ♕c3 20. ♗d2 ♕a3 21. ♗c1 ♕c3 22. ♖b3 ♕a1 23. ♗c4+ ♔h8 24. ♗xh6 ♕xd1 25. ♗xg7+ ♔xg7 26. ♖xd1 exf3 27. gxf3 ♖a7 28. ♗d5 ♖d8 29. ♖b5 ♖a6 30. ♗e4 ♖a7 31. ♗d5 ♖a6 32. ♖c5 ♖d7 33. ♔g2 ♖b6 34. ♗xc6 bxc6 35. ♔f2 ♖d5 36. ♖xd5 cxd5 37. ♖c1 ♖b4 38. ♔e3 ♖xa4 39. ♖c5 e6 40. ♖c7+ ♔g8 41. ♖e7 ♖a3+ 42. ♔f4 ♖d3 43. ♖xe6 ♖xd4+ 44. ♔g5 ♔f7 45. ♖a6 a4 46. f4 h6+ 47. ♔g4 ♖c4 48. h4 ♖d4 49. ♖f6+ ♔g7 50. ♖a6 ♔f7 51. h5 gxh5+ 52. ♔f5 ♔g7 53. ♖a7+ ♔f8 54. e6 ♖e4 55. ♖d7 ♖c4 56. ♖xd5 h4 57. ♖d3 ♔e7 58. ♖d7+ ♔f8 59. ♖h7 h5 60. ♔e5 h3 61. f5 ♔g8 62. ♖xh5 a3 63. ♖xh3 a2 64. ♖a3 ♖c5+ 65. ♔f6 1–0**

Partida 27 : **Deep Blue — Gary Kasparov**
Filadelfia (m/1), 10 de Febrero de 1996

**1. e4 c5 2. c3 d5 3. exd5 ♕xd5 4. d4 ♘f6 5. ♘f3 ♗g4 6. ♗e2 e6 7. h3 ♗h5 8. O–O ♘c6 9. ♗e3 cxd4 10. cxd4 ♗b4 11. a3 ♗a5 12. ♘c3 ♕d6 13. ♘b5 ♕e7 14. ♘e5 ♗xe2 15. ♕xe2 O–O 16. ♖ac1 ♖ac8 17. ♗g5 ♗b6 18. ♗xf6 gxf6 19. ♘c4 ♖fd8 20. ♘xb6 axb6 21. ♖fd1 f5 22. ♕e3 ♕f6 23. d5 ♖xd5 24. ♖xd5 exd5 25. b3 ♔h8 26. ♕xb6 ♖g8 27. ♕c5 d4 28. ♘d6 f4 29. ♘xb7 ♘e5 30. ♕d5 f3 31. g3 ♘d3 32. ♖c7 ♖e8 33. ♘d6 ♖e1+ 34. ♔h2 ♘xf2 35. ♘xf7+ ♔g7 36. ♘g5+ ♔h6 37. ♖xh7+ 1–0**

Partida 28 : **Gary Kasparov — Deep Blue**
Filadelfia (m/2), 11 de Febrero de 1996

**1. ♘f3 d5 2. d4 e6 3. g3 c5 4. ♗g2 ♘c6 5. O–O ♘f6 6. c4 dxc4 7. ♘e5 ♗d7 8. ♘a3 cxd4 9. ♘axc4 ♗c5 10. ♕b3 O–O 11. ♕xb7 ♘xe5 12. ♘xe5 ♖b8 13. ♕f3 ♗d6 14. ♘c6 ♗xc6 15. ♕xc6 e5 16. ♖b1 ♖b6 17. ♕a4 ♕b8 18. ♗g5 ♗e7 19. b4 ♗xb4 20. ♗xf6 gxf6 21. ♕d7 ♕c8 22. ♕xa7 ♖b8 23. ♕a4 ♗c3 24. ♖xb8 ♕xb8 25. ♗e4 ♕c7 26. ♕a6 ♔g7 27. ♕d3 ♖b8 28. ♗xh7 ♖b2 29. ♗e4 ♖xa2 30. h4 ♕c8 31. ♕f3 ♖a1 32. ♖xa1 ♗xa1 33. ♕h5 ♕h8 34. ♕g4+ ♔f8 35. ♕c8+ ♔g7 36. ♕g4+ ♔f8 37. ♗d5 ♔e7 38. ♗c6 ♔f8 39. ♗d5 ♔e7 40. ♕f3 ♗c3 41. ♗c4 ♕c8 42. ♕d5 ♕e6 43. ♕b5 ♕d7 44. ♕c5+ ♕d6 45. ♕a7+ ♕d7 46. ♕a8 ♕c7 47. ♕a3+ ♕d6 48. ♕a2 f5 49. ♗xf7 e4 50. ♗h5 ♕f6 51. ♕a3+ ♔d7 52. ♕a7+ ♔d8 53. ♕b8+ ♔d7 54. ♗e8+ ♔e7 55. ♗b5 ♗d2 56. ♕c7+ ♔f8 57. ♗c4 ♗c3 58. ♔g2 ♗e1 59. ♔f1 ♗c3 60. f4 exf3 61. exf3 ♗d2 62. f4 ♔e8 63. ♕c8+ ♔e7 64. ♕c5+ ♔d8 65. ♗d3 ♗e3 66. ♕xf5 ♕c6 67. ♕f8+ ♔c7 68. ♕e7+ ♔c8 69. ♗f5+ ♔b8 70. ♕d8+ ♔b7 71. ♕d7+ ♕xd7 72. ♗xd7 ♔c7 73. ♗b5 ♔d6 1–0**

Partida 29 : **Deep Blue — Gary Kasparov**
Filadelfia (m/3), 13 de Febrero de 1996

**1. e4 c5 2. c3 d5 3. exd5 ♕xd5 4. d4 ♘f6 5. ♘f3 ♗g4 6. ♗e2 e6 7. O–O ♘c6 8. ♗e3 cxd4 9. cxd4 ♗b4 10. a3 ♗a5 11. ♘c3 ♕d6 12. ♘e5 ♗xe2 13. ♕xe2 ♗xc3 14. bxc3 ♘xe5 15. ♗f4 ♘f3+ 16. ♕xf3 ♕d5 17. ♕d3 ♖c8 18. ♖fc1 ♕c4 19. ♕xc4 ♖xc4 20. ♖cb1 b6 21. ♗b8 ♖a4 22. ♖b4 ♖a5 23. ♖c4 O–O 24. ♗d6 ♖a8 25. ♖c6 b5 26. ♔f1 ♖a4 27. ♖b1 a6 28. ♔e2 h5 29. ♔d3 ♖d8 30. ♗e7 ♖d7 31. ♗xf6 gxf6 32. ♖b3 ♔g7 33. ♔e3 e5 34. g3 exd4+ 35. cxd4 ♖e7+ 36. ♔f3 ♖d7 37. ♖d3 ♖axd4 38. ♖xd4 ♖xd4 39. ♖xa6 1/2–1/2**

Partida 30 : **Gary Kasparov — Deep Blue**
Filadelfia (m/4), 14 de Febrero de 1996

**1. ♘f3 d5 2. d4 c6 3. c4 e6 4. ♘bd2 ♘f6 5. e3 ♘bd7 6. ♗d3 ♗d6 7. e4 dxe4 8. ♘xe4 ♘xe4 9. ♗xe4 O–O 10. O–O h6 11. ♗c2 e5 12. ♖e1 exd4 13. ♕xd4 ♗c5 14. ♕c3 a5 15. a3 ♘f6 16. ♗e3 ♗xe3 17. ♖xe3 ♗g4 18. ♘e5 ♖e8 19. ♖ae1 ♗e6 20. f4 ♕c8 21. h3 b5 22. f5 ♗xc4 23. ♘xc4 bxc4 24. ♖xe8+ ♘xe8 25. ♖e4 ♘f6 26. ♖xc4 ♘d5 27. ♕e5 ♕d7 28. ♖g4 f6 29. ♕d4 ♔h7 30. ♖e4 ♖d8 31. ♔h1 ♕c7 32. ♕f2 ♕b8 33. ♗a4 c5 34. ♗c6 c4 35. ♖xc4 ♘b4 36. ♗f3 ♘d3 37. ♕h4 ♕xb2 38. ♕g3 ♕xa3 39. ♖c7 ♕f8 40. ♖a7 ♘e5 41. ♖xa5 ♕f7 42. ♖xe5 fxe5 43. ♕xe5 ♖e8 44. ♕f4 ♕f6 45. ♗h5 ♖f8 46. ♗g6+ ♔h8 47. ♕c7 ♕d4 48. ♔h2 ♖a8 49. ♗h5 ♕f6 50. ♗g6 ♖g8 1/2–1/2**

Partida 31 : **Deep Blue — Gary Kasparov**
Filadelfia (m/5), 15 de Febrero de 1996

**1. e4 e5 2. ♘f3 ♘f6 3. ♘c3 ♘c6 4. d4 exd4 5. ♘xd4 ♗b4 6. ♘xc6 bxc6 7. ♗d3 d5 8. exd5 cxd5 9. O–O O–O 10. ♗g5 c6 11. ♕f3 ♗e7 12. ♖ae1 ♖e8 13. ♘e2 h6 14. ♗f4 ♗d6 15. ♘d4 ♗g4 16. ♕g3 ♗xf4 17. ♕xf4 ♕b6 18. c4 ♗d7 19. cxd5 cxd5 20. ♖xe8+ ♖xe8 21. ♕d2 ♘e4 22. ♗xe4 dxe4 23. b3 ♖d8 24. ♕c3 f5 25. ♖d1 ♗e6 26. ♕e3 ♗f7 27. ♕c3 f4 28. ♖d2 ♕f6 29. g3 ♖d5 30. a3 ♔h7 31. ♔g2 ♕e5 32. f3 e3 33. ♖d3 e2 34. gxf4 e1=♕ 35. fxe5 ♕xc3 36. ♖xc3 ♖xd4 37. b4 ♗c4 38. ♔f2 g5 39. ♖e3 ♖d2+ 40. ♔e1 ♖d3 41. ♔f2 ♔g6 42. ♖xd3 ♗xd3 43. ♔e3 ♗c2 44. ♔d4 ♔f5 45. ♔d5 h5 0–1**

Partida 32 : **Gary Kasparov — Deep Blue**
Filadelfia (m/6), 16 de Febrero de 1996

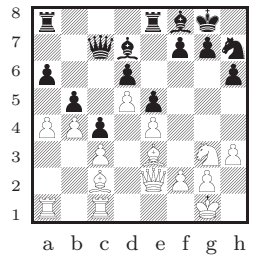
**1. ♘f3 d5 2. d4 c6 3. c4 e6 4. ♘bd2 ♘f6 5. e3 c5 6. b3 ♘c6 7. ♗b2 cxd4 8. exd4 ♗e7 9. ♖c1 O–O 10. ♗d3 ♗d7 11. O–O ♘h5 12. ♖e1 ♘f4 13. ♗b1 ♗d6 14. g3 ♘g6 15. ♘e5 ♖c8 16. ♘xd7 ♕xd7 17. ♘f3 ♗b4 18. ♖e3 ♖fd8 19. h4 ♘ge7 20. a3 ♗a5 21. b4 ♗c7 22. c5 ♖e8 23. ♕d3 g6 24. ♖e2 ♘f5 25. ♗c3 h5 26. b5 ♘ce7 27. ♗d2 ♔g7 28. a4 ♖a8 29. a5 a6 30. b6 ♗b8 31. ♗c2 ♘c6 32. ♗a4 ♖e7 33. ♗c3 ♘e5 34. dxe5 ♕xa4 35. ♘d4 ♘xd4 36. ♕xd4 ♕d7 37. ♗d2 ♖e8 38. ♗g5 ♖c8 39. ♗f6+ ♔h7 40. c6 bxc6 41. ♕c5 ♔h6 42. ♖b2 ♕b7 43. ♖b4 1–0**

Partida 33 : **Kasparov — DeepBlue**
New York (m/1), 3 de Mayo de 1997

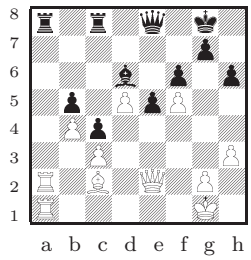
**1. ♘f3 d5 2. g3 ♗g4 3. b3 ♘d7 4. ♗b2 e6 5. ♗g2 ♘gf6 6. O–O c6 7. d3 ♗d6 8. ♘bd2 O–O 9. h3 ♗h5 10. e3 h6 11. ♕e1 ♕a5 12. a3 ♗c7 13. ♘h4 g5 14. ♘hf3 e5 15. e4 ♖fe8 16. ♘h2 ♕b6 17. ♕c1 a5 18. ♖e1 ♗d6 19. ♘df1 dxe4 20. dxe4 ♗c5 21. ♘e3 ♖ad8 22. ♘hf1 g4 23. hxg4 ♘xg4 24. f3 ♘xe3 25. ♘xe3 ♗e7 26. ♔h1 ♗g5 27. ♖e2 a4 28. b4 f5 29. exf5 e4 30. f4 ♗xe2 31. fxg5 ♘e5 32. g6 ♗f3 33. ♗c3 ♕b5 34. ♕f1 ♕xf1+ 35. ♖xf1 h5 36. ♔g1 ♔f8 37. ♗h3 b5 38. ♔f2 ♔g7 39. g4 ♔h6 40. ♖g1 hxg4 41. ♗xg4 ♗xg4 42. ♘xg4+ ♘xg4+ 43. ♖xg4 ♖d5 44. f6 ♖d1 45. g7 1–0**

Partida 34 : **DeepBlue — Kasparov**
New York (m/2), 4 de Mayo de 1997

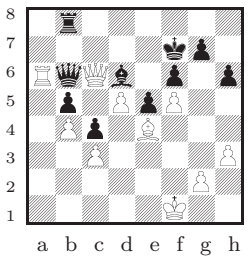
**1. e4 e5 2. ♘f3 ♘c6 3. ♗b5 a6 4. ♗a4 ♘f6 5. O–O ♗e7 6. ♖e1 b5 7. ♗b3 d6 8. c3 O–O 9. h3 h6 10. d4 ♖e8 11. ♘bd2 ♗f8 12. ♘f1 ♗d7 13. ♘g3 ♘a5 14. ♗c2 c5 15. b3 ♘c6 16. d5 ♘e7 17. ♗e3 ♘g6 18. ♕d2 ♘h7 19. a4 ♘h4 20. ♘xh4 ♕xh4 21. ♕e2 ♕d8 22. b4 ♕c7 23. ♖ec1 c4**



**24. ♖a3 ♖ec8 25. ♖ca1 ♕d8 26. f4 ♘f6 27. fxe5 dxe5 28. ♕f1 ♘e8 29. ♕f2 ♘d6 30. ♗b6 ♕e8 31. ♖3a2 ♗e7 32. ♗c5 ♗f8 33. ♘f5 ♗xf5 34. exf5 f6 35. ♗xd6 ♗xd6 36. axb5 axb5**



**37. ♗e4 ♖xa2 38. ♕xa2 ♕d7 39. ♕a7 ♖c7 40. ♕b6 ♖b7 41. ♖a8+ ♔f7 42. ♕a6 ♕c7 43. ♕c6 ♕b6+ 44. ♔f1 ♖b8 45. ♖a6**



**1–0**

Partida 35 : **Kasparov — DeepBlue**
New York (m/3), 6 de Mayo de 1997

**1. d3 e5 2. ♘f3 ♘c6 3. c4 ♘f6 4. a3 d6 5. ♘c3 ♗e7 6. g3 O–O 7. ♗g2 ♗e6 8. O–O ♕d7 9. ♘g5 ♗f5 10. e4 ♗g4 11. f3 ♗h5 12. ♘h3 ♘d4 13. ♘f2 h6 14. ♗e3 c5 15. b4 b6 16. ♖b1 ♔h8 17. ♖b2 a6 18. bxc5 bxc5 19. ♗h3 ♕c7 20. ♗g4 ♗g6 21. f4 exf4 22. gxf4 ♕a5 23. ♗d2 ♕xa3 24. ♖a2 ♕b3 25. f5 ♕xd1 26. ♗xd1 ♗h7 27. ♘h3 ♖fb8 28. ♘f4 ♗d8 29. ♘fd5 ♘c6 30. ♗f4 ♘e5 31. ♗a4 ♘xd5 32. ♘xd5 a5 33. ♗b5 ♖a7 34. ♔g2 g5 35. ♗xe5+ dxe5 36. f6 ♗g6 37. h4 gxh4 38. ♔h3 ♔g8 39. ♔xh4 ♔h7 40. ♔g4 ♗c7 41. ♘xc7 ♖xc7 42. ♖xa5 ♖d8 43. ♖f3 ♔h8 44. ♔h4 ♔g8 45. ♖a3 ♔h8 46. ♖a6 ♔h7 47. ♖a3 ♔h8 48. ♖a6 1/2–1/2**

Partida 36 : **DeepBlue — Kasparov**
New York (m/4), 7 de Mayo de 1997

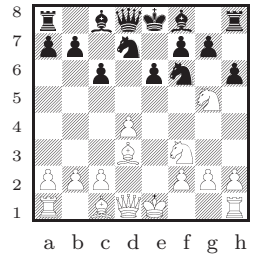
**1. e4 c6 2. d4 d6 3. ♘f3 ♘f6 4. ♘c3 ♗g4 5. h3 ♗h5 6. ♗d3 e6 7. ♕e2 d5 8. ♗g5 ♗e7 9. e5 ♘fd7 10. ♗xe7 ♕xe7 11. g4 ♗g6 12. ♗xg6 hxg6 13. h4 ♘a6 14. O–O–O O–O–O 15. ♖dg1 ♘c7 16. ♔b1 f6 17. exf6 ♕xf6 18. ♖g3 ♖de8 19. ♖e1 ♖hf8 20. ♘d1 e5 21. dxe5 ♕f4 22. a3 ♘e6 23. ♘c3 ♘dc5 24. b4 ♘d7 25. ♕d3 ♕f7 26. b5 ♘dc5 27. ♕e3 ♕f4 28. bxc6 bxc6 29. ♖d1 ♔c7 30. ♔a1 ♕xe3 31. fxe3 ♖f7 32. ♖h3 ♖ef8 33. ♘d4 ♖f2 34. ♖b1 ♖g2 35. ♘ce2 ♖xg4 36. ♘xe6+ ♘xe6 37. ♘d4 ♘xd4 38. exd4 ♖xd4 39. ♖g1 ♖c4 40. ♖xg6 ♖xc2 41. ♖xg7+ ♔b6 42. ♖b3+ ♔c5 43. ♖xa7 ♖f1+ 44. ♖b1 ♖ff2 45. ♖b4 ♖c1+ 46. ♖b1 ♖cc2 47. ♖b4 ♖c1+ 48. ♖b1 ♖xb1+ 49. ♔xb1 ♖e2 50. ♖e7 ♖h2 51. ♖h7 ♔c4 52. ♖c7 c5 53. e6 ♖xh4 54. e7 ♖e4 55. a4 ♔b3 56. ♔c1 1/2–1/2**

Partida 37 : **Kasparov — DeepBlue**
New York (m/5), 10 de Mayo de 1997

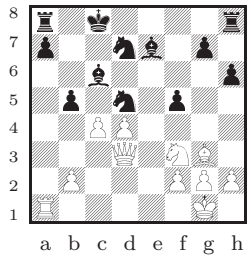
**1. ♘f3 d5 2. g3 ♗g4 3. ♗g2 ♘d7 4. h3 ♗xf3 5. ♗xf3 c6 6. d3 e6 7. e4 ♘e5 8. ♗g2 dxe4 9. ♗xe4 ♘f6 10. ♗g2 ♗b4+ 11. ♘d2 h5 12. ♕e2 ♕c7 13. c3 ♗e7 14. d4 ♘g6 15. h4 e5 16. ♘f3 exd4 17. ♘xd4 O–O–O 18. ♗g5 ♘g4 19. O–O–O ♖he8 20. ♕c2 ♔b8 21. ♔b1 ♗xg5 22. hxg5 ♘6e5 23. ♖he1 c5 24. ♘f3 ♖xd1+ 25. ♖xd1 ♘c4 26. ♕a4 ♖d8 27. ♖e1 ♘b6 28. ♕c2 ♕d6 29. c4 ♕g6 30. ♕xg6 fxg6 31. b3 ♘xf2 32. ♖e6 ♔c7 33. ♖xg6 ♖d7 34. ♘h4 ♘c8 35. ♗d5 ♘d6 36. ♖e6 ♘b5 37. cxb5 ♖xd5 38. ♖g6 ♖d7 39. ♘f5 ♘e4 40. ♘xg7 ♖d1+ 41. ♔c2 ♖d2+ 42. ♔c1 ♖xa2 43. ♘xh5 ♘d2 44. ♘f4 ♘xb3+ 45. ♔b1 ♖d2 46. ♖e6 c4 47. ♖e3 ♔b6 48. g6 ♔xb5 49. g7 ♔b4 1/2–1/2**

Partida 38 : **Kasparov — DeepBlue**
New York (m/6), 11 de Mayo de 1997

**1. e4 c6 2. d4 d5 3. ♘c3 dxe4 4. ♘xe4 ♘d7 5. ♘g5 ♘gf6 6. ♗d3 e6 7. ♘1f3 h6**



**8. ♘xe6 ♕e7 9. O–O fxe6 10. ♗g6+ ♔d8 11. ♗f4 b5 12. a4 ♗b7 13. ♖e1 ♘d5 14. ♗g3 ♔c8 15. axb5 cxb5 16. ♕d3 ♗c6 17. ♗f5 exf5 18. ♖xe7 ♗xe7 19. c4**



**1–0**