Pokroky matematiky, fyziky a astronomie

Ján Šturc

Počítače v šachu

Pokroky matematiky, fyziky a astronomie, Vol. 28 (1983), No. 2, 78--92

Persistent URL: http://dml.cz/dmlcz/138074

Počítače v šachu

Ján Šturc, Bratislava

Obsah

1. Úvod
2. Čo dáva do súvisu starú hru a jeden z posledných výdobytkov techniky
3. Prečo majú počítače hrať šach na veľmajstrovskej úrovni
4. Základné charakteristiky šachistu a ich modelovanie
   4.1. Pozičný cit
   4.2. Pamäť
   4.3. Prepočet variantov
5. Technické aspekty realizácie šachovej hry na počítači
6. Využívanie špecifických šachových poznatkov — základný rozdiel medzi hrou človeka a počítača
7. Ukážky hry počítača a vyhliadky do budúcnosti

1. Úvod

Ako stará je šachová hra sa presne nevie. Podľa názoru šachových historikov vznikla v 4. stor. n. l. v Indii. Vraj sa vyvinula z doskových hier starých Egypťanov, o čom svedčia nálezy datované až 12 000 rokov pred našim letopočtom.

Podľa starej indickej báje si vynálezca šachu od vladára, ktorému svojim vynálezom pomohol zahnať nudu, vyžiadal na prvé políčko šachovnice 1 zrnko ryže, na druhé dve zrnká a na každé nasledujúce dvakrát toľko čo na predchádzajúce. Úsmev vladára nad „skromnou" žiadosťou vynálezcu zamrzol, keď mu oznámili, že ju nemôže vyplniť ani keby vyprázdnil všetky sýpky v krajine. (Čiastočný súčet geometrického radu $2^{64} - 1 = 18\,446\,744\,073\,709\,551\,615$, čo predstavuje asi 4 úrody ryže z celého zemského povrchu, bol vraj už vtedy Indom známy [Souček 1981]).

Už touto bájou sa vytvorila nejaká súvislosť medzi šachom a matematikou. Je asi nemožné zistiť, koľko matematikov sa zaoberalo šachmi a koľkí z nich aj skutočne hrali. Je však isté, že v priebehu modernej historie šachu, t. j. od čias, čo sa organizujú oficiálne majstrovstvá sveta (1840), dvaja majstri sveta (Em. Lasker a M. Euwe) boli matematici. Podobne sa vie, že šachom sa zaoberali aj matematici do „počítačovej éry" (napr. Zermelo [1913]). Existovali pokusy zostrojiť mechanické zariadenie hrajúce šachy. Sporné zariadenie Jána Kempelena, ktoré sice hralo dobre, ale má sa za to, že v ňom bol ukrytý šachista malej postavy, neuvažujeme. Je však isté, že šachy chcel hrať Charles Babbage (1850) so svojim „analytical engine" a že Quevado Torres (1890) zostrojil mechanické zariadenie, ktoré naprosto spoľahlivo matovalo kráľa kráľom a vežou na rozdiel od dnešných „elektronických šachových kalkulačiek".

Myšlienka použiť počítač a naprogramovať algoritmus šachovej hry vznikla prakticky súčasne s počítačmi [Shannon 1950, Turing 1950]. Prvé programy hrajúce šach zostro-

jili Greenblatt a Bernstein roku 1958. V roku 1967 sa uskutočnil z iniciatívy profesora McCarthyho zo Stanfordskej univerzity prvý medzištátny zápas počítačov ZSSR – USA s výsledkom 3 : 1. Od roku 1974 sa každé tri roky súčasne s kongresom IFFIP konajú aj majstrovstvá sveta šachových programov. Doteraz zvíťazili nasledujúce programy: Kaisa roku 1974, CHESS 4.6 roku 1977, Belle roku 1980.

2. Čo dáva do súvisu starú hru a jeden z posledných výdobytkov modernej techniky

Ako vidíme, šachová hra sprevádza počítače od ich vzniku. Sme teda oprávnení položiť otázku, prečo tomu tak je. Prečo už v počiatkoch používania počítačov, keď počítač bol drahé zariadenie a zdalo by sa, že existuje množstvo významnejších úloh, sa vedci ako Shannon a Turing venujú takej hračke, ako je programovanie šachovej hry? Prečo dodnes výzkumné inštitúcie, ktoré majú cieľove organizované praktické zameranie, neľutujú prostriedky na vývoj šachového programu?

Je však prirodzené, že akonáhle sa vyjasnilo, že počítač okrem rýchleho počítania dokáže riešiť aj logické a kombinatorické úlohy, vznikly otázky: Môže stroj myslieť?; Existuje nejaká strojová „umelá inteligencia"?. Rozumná odpoveď na takéto otázky je nesmierne ťažká, pretože pojmy myslenie a inteligencia sú neformálne filozofické pojmy.

Pokusom riešiť túto otázku je Turingov test, ktorý predpokladá konverzáciu (vedenú buď ústne, alebo písomne). Ak účastník po 30 minutách nedokáže rozhodnúť, či jeho partnerom je človek alebo stroj a partnerom bol počítač, potom môžeme povedať, že stroj myslí. Aj takto formulovaná otázka je ťažká, pretože aj medzi dvoma ľuďmi môže prebiehať dosť nezmyselná konverzácia. Stačí napríklad, ak partneri vzájomne nerozumejú používanému jazyku (rôzna reč alebo odborníci vzdialených odborov) alebo vedú „duchaprázdnu" spoločenskú konverzáciu (viď Karel Čapek: *Válka s mloky*, protokol s „Andrias Scheuzerim"). Zdá sa, že je asi rozumné obmedziť sa na relatívne úzku konečnú oblasť ľudskej činnosti. Šachová hra vzhľadom k tomu, že nie je známy jej realizovateľný algoritmus na jednej strane a že existuje rozvinutá šachová kultúra na strane druhej, je vhodným kandidátom na takúto oblasť.

Principiálne sú možné tri výsledky Turingovho testu cez šachovú hru:

a) Počítač hrá strojovo dokonale, podstatne prevyšuje akéhokoľvek človeka.
b) Počítač hrá ako rovný s rovným s každým človekom, ale nie je podstatný rozdiel medzi ľudským a počítačovým šachom.
c) Nie je možné zostrojiť program, ktorý by hral veľmajstrovský šach.

Zakiaľ možnosti b) stroj myslí a c) stroj nemyslí dávajú „jasnú" odpoveď na Turingov test, možnosť a) ho necháva nerozhodnutým. Hovorí len, že sme našli realizovateľný algoritmus šachovej hry a to je prínos pre šach samotný. Jedným z cieľov tohto článku je ukázať, že je malá nádej na výsledok a).

Druhý dôvod snáď najjasnejšie vystihol Vorobjev [1961], keď hry, kde majú obaja súperi úplnú informáciu o postavení, vylúčil z predmetu teórie hier s odôvodnením, že majú charakter všeobecne matematického (kombinatorického alebo analytického) problému. Riešiť takéto problémy na počítači je obvyklé, a keď ide navyše o dobre formulovaný problém, ktorý dlhý čas odoláva pokusom o riešenie, má sa za to, že jeho riešenie

môže byť podstatným prínosom z hľadiska rozvoja matematických metód a metód používania počítačov.

3. Prečo majú počítače hrať šach na veľmajstrovskej úrovni

Z doteraz uvedených dôvodov vyplýva, že programovanie šachu je zaujímavá matematická úloha, ktorej riešenie môže prispieť aj k riešeniu istého filozofického problému. Ťažko by sme tým ale zdôvodnili, prečo treba, aby počítač hral dobre šachy. Súčasné programy už hrajú lepšie ako priemerný alebo aj nadaný človek, ktorý nemá špecifické šachové vedomosti, a teda alternatíva b) je najpravdepodobnejší výsledok Turingovho testu. Načo treba plytvať prostriedkami na to aby počítač porazil majstra sveta? (Pre prvý program, ktorému sa to podarí, je určená Fredkinova cena $ 100 000.)

Otázka je oveľa vážnejšia, než sa zdá na prvý pohľad. Podobné kombinatorické úlohy sa riešia na počítači často. Napríklad rozmiestňovanie obvodov na doskách, nastreľovanie ložísk nafty ap. Pritom sa má za to, že počítač vybavený dobrým programom, rieši tieto úlohy lepšie než „skúsený inžinier". Treba však priznať, že skúsený elektroinžinier rozmiestňuje obvody či navrhuje tlačené spoje približne raz za päť rokov a príležitostí pre geológa streliť si do naftového ložiska je ešte menej.

Využiť preto existenciu šachovej kultúry a postaviť rovnakými metódami šachový program a porovnať ho s kvalifikovaným šachistom je preto aj nepriamym testom kvality týchto programov. Ak navyše uvážime, že šachové myslenie zahrnuje aj niektoré črty strategického plánovania (či vojenského alebo priemyselného), je jasné, že na prvý pohľad hračka, programovanie šachu, vždy nájde svojich mecenášov.

4. Základné charakteristiky šachistu a ich modelovanie

Súčasná matematická a šachová kultúra majú ešte jednu spoločnú črtu a to je množstvo a spôsob publikovania. Tým sa šach podobá ktorejkoľvek vednej disciplíne. Ak by sme chceli zrovnávať matematika, šachistu a šachový program podľa troch najdôležitejších vlastností, vyzerala by príslušná tabuľka asi takto:

matematik	šachista	šachový program
intuícia	pozičný cit	ohodnocovacia funkcia
znalosť metód a techník	znalosť teórie	pamäť
schopnosť formálne dokazovať	prepočet variantov	prehľadávanie stromu

Je tu však podstatný rozdiel medzi matematikou a šachom. Šach poskytuje len konečný počet možností, a teda každá z uvedených vlastností, keby bola „absolútna", stačí. Znamená to, že stačí presne rozoznať, ktorá pozícia je lepšia a ktorá horšia práve tak, ako zapamätanie si všetkých možností lebo schopnosť dopočítať každú partiu do konca.

Poznámka: Šachová teória sa skladá prevažne z typických pozícií (zahájení, koncoviek a strednej hry) a spôsobov ich rozohrávania. Je tak skôr praxou než teóriou v matematickom slova zmysle. Je to prevažne množina príkladov, z ktorej len zriedka sú vyabstrahované všeobecné poznatky.

Vzhľadom na uvedenú poznámku môžme v počítači znalosť teórie nahradzovať zapamätanými šablónami, teda pamäťou. Budeme sa teraz podrobne zaoberať jednotlivými vlastnosťami.

4.1. Pozičný cit

Pre klasického matematika je šachový problém jednoducho vyriešený napríklad takto: Nech $f: \mathbf{P} \to \mathbf{R}$ je funkcia z množiny pozícií (šachových postavení) do množiny racionálnych čísiel definovaná takto:

$$f(p) = \begin{cases} \frac{1}{k} & \text{ak pozícia je vyhraná pre bieleho a pri najlepšej možnej hre oboch strán} \\ & \text{biely môže vynútiť mat na } k \text{ ťahov} \\ 0 & \text{ak pozícia je remízová} \\ -\frac{1}{k} & \text{ak pozícia je vyhraná pre čierneho a pri nejlepšej možnej hre oboch strán} \\ & \text{čierny môže vynútiť mat na } k \text{ ťahov} \end{cases}$$

Algoritmus hry je teraz už veľmi jednoduchý. Pre danú pozíciu p si generuje všetky pozície $p_1, \ldots p_d$, ktoré z nej môžu vzniknúť jedným polťahom (ťahom bieleho, resp. čierneho). Z týchto generovaných pozícií biely vyberá pozíciu p_i takú, že $f(p_i) = \max\{f(p_j) : 1 \leqq j \leqq d\}$ a čierny analogické minimum. Dôkaz, že tento algoritmus končí a je korektný, je priamočiary.

Moderná informatika (computer science) si však už automaticky kladie otázku, ako ťažké je funkciu f vypočítať, t. j. pre danú mieru zložitosti pozície $|p|$ definovať funkciu $\varphi_f : \mathbf{N} \to \mathbf{N}$ ($\mathbf{N}$ je množina prirodzených čísiel) takú, že $\varphi_f(|p|)$ určuje počet krokov, resp. nejakých operácií potrebných na výpočet funkcie f. Aby sme to však vedeli urobiť, potrebujeme niektoré zovšeobecnenia:

a) Pretože väčšinou sa nám nedarí vypočítať funkcie φ_f presne, odhadujem ich asymptoticky, a teda potrebujeme zásobu pozícií takých, že $|p|$ môže ľubovoľne rásť. K tomu stačí uvažovať hru na šachovnici $n \times n$ takú, že v nej vystupujú králi, dámy, veže, strelci a pešiaci, kde ich pohyb je prirodzene zovšeobecnený na $n \times n$ šachovnicu. Biely a čierny majú práve jedného kráľa a cieľom hry je dať mat kráľovi. Od jazdcov jednoducho abstrahujeme. Znamená to, že závery sa budú vzťahovať na pozície bez jazdcov.

b) Nebudeme uvažovať funkciu $f(p)$, ale zjednodušenú funkciu $g : \mathbf{P} \to$ **boolean**, takú, že $g(p) =$ **if** $f(p) > 0$ **then** true **else** false, čo je vlastne rozhodovací problém, či pozícia je pre bieleho vyhraná.

Skôr než pristúpime k vlastnému hodnoteniu zložitosti φ_g funkcie g, stručne uvedieme triedy zložitosti ťažkých algoritmických problémov, ako sa používajú v informatike (viď napr. [Garey a Johnson 1979]).

$$P \subseteq NP \subseteq PSPACE \subseteq EXPTIME,$$

kde

P sú problémy, ktoré sa dajú vypočítať polynomiálnym počtom krokov (t. j. $\varphi(|p|)$ sa dá asymptoticky ohraničiť polynómom premennej $|p|$) na deterministickom zariadení.

NP sú problémy, ktoré sa dajú vypočítať polynomiálnym počtom krokov na nedeterministickom zariadení.

PSPACE sú problémy, ktoré sa dajú vypočítať pri použití polynomiálneho množstva pamäti (nepočítame kroky, ale množstvo použitej pamäti).

EXPTIME sú problémy, ktoré sa dajú vypočítať v exponenciálnom počte krokov (t. j. $\varphi(|p|)$ sa dá asymptoticky ohraničiť exponenciálnou funkciou).

Problém či inklúzie sú vlastné, patrí dlhý čas k otvoreným problémom. (Špeciálne známy je P=NP problém.) Vie sa len, že P ≠ EXPTIME. Ďalej sa vie, že každá z uvedených tried obsahuje tzv. úplné problémy. Problém sa nazýva S-úplným, ak patrí do triedy S a každý problém z triedy S sa dá polynomiálne redukovať na tento problém.

Fraenkel a Lichtenstein [1980] dokázali tvrdenie: Rozhodovací problém pre $n \times n$ šach (výpočet funkcie $g(|p|)$) je EXPTIME úplný problém.

Znamená to, že každý algoritmus, počítajúci funkciu $g(p)$, teda aj funkciu $f(p)$, potrebuje exponenciálne veľa krokov v $|p|$, a teda je prakticky nerealizovateľný. Navyše, pretože prehľadávanie stromu sa dá robiť v polynomiálnom priestore, ak PSPACE ≠ ≠ EXPTIME (na čo je podozrenie), potom realizácia vyhrávajúcej stratégie niekedy vyžaduje exponenciálne veľa ťahov.

Dôkaz uvedeného tvrdenia je značne zdĺhavý a spočíva v simulácii umelej boooleovskej hry G3, o ktorej Stockmeyer a Chandra [1979] dokázali, že je EXPTIME úplná. Pozície, ktoré pri tom vznikajú, sa málo podobajú skutočnej partii. Svojim blokovaným charakterom skôr pripomínajú mnohoťahové úlohy.

Hoci nie je nikdy celkom zaručené, že výsledky zovšeobecnenia na $n \times n$ šach musia platiť v plnej miere aj pre 8 × 8 šach, existencia úloh „rekordérov" mat 255 ťahom (viď [Formánek 353]), ako aj skutočnosť, že vo viacerých jednoduchých koncovkách prijala FIDE výnimku z pädesiatťahového pravidla, svedčí o tom, že matematická abstrakcia sa od šachovej praxe príliš neodchyluje.

Nie je teda nádej na jednoduché rozhodnutie, či pozícia je vyhraná a na nejaké presné ohodnocovanie pozícií. Musíme sa uspokojiť nejakými jednoducho počítateľnými aproximatívnymi ohodnocovacími funkciami. Takéto funkcie sa v počítačovom i ľudskom šachu používali dávno. Všeobecne známa je funkcia pre výpočet materiálovej bilancie na šachovnici. Základnou jednotkou je pešiak (1) a jednotlivým figurám sú priradené váhy: jazdec a strelec 3, veža 5 a dáma 9. Tieto hodnotové vzťahy boli prijaté aj pre počítačový šach a upravené na hodnoty : pešiak 100, jazdec 300, strelec 325, veža 500, dáma 900.

V praxi programovania šachu bolo vymyslené množstvo heuristických ohodnocovacích funkcií. V poslednej dobe sa má za to, že najlepšie fungujú ohodnocovacie funkcie tvaru:

$$\sum_{1 \leq i \leq N} v(F_i)\, t(F_i),$$

kde F_i pre $1 \leqq i \leqq N$ sú rôzne faktory hodnotenia šachovej pozície napríklad materiál, priestor, rozsah pôsobnosti figúr, súhrn figúr, prevaha na krídle, slabý kráľ a pod.; $v(F_i)$ je číselné ohodnotenie daného faktoru a $t(F_i)$ je funkcia nadobudajúca hodnoty 0 alebo 1 podľa toho, či daný faktor je pre hodnotenie danej pozície významný. Šachová prax ukazuje, že väčšinou je v pozícii len niekoľko málo faktorov (obvykle len jeden), pre ktoré $t(F_i) = 1$. Obľúbené funkcie tohto typu sú tzv. prahové funkcie, kde sa počíta F_i ako rozdiel hodnôt faktora F_i pre bieleho a čierneho a vlastná prahová funkcia t nadobudne hodnotu 1, až keď tento rozdiel prekročí určitú vopred stanovenú hodnotu.

Nie je možné v krátkom článku uviesť prehľad všetkých možných pozičných faktorov šachovej hry, naznačíme len jeden možný spôsob, ako niektoré z nich reprezentovať geometricky.

Sledujúc autorov uvádzaného prístupu [Atkin a Witten 1975] zavádzame označenia:

Nech $W = \{W_i : 0 \leqq i \leqq 15\}$ je množina bielych kameňov,

$B = \{B_i : 0 \leqq i \leqq 15\}$ je množina čiernych kameňov a

$S = \{S_i : 0 \leqq i \leqq 63\}$ je množina polí šachovnice.

Potom pozícia je funkcia $p : W \cup B \to S$ (čiastočná funkcia z $W \cup B$ do S). Pre každú pozíciu p definujeme relácie $\Gamma_W \subset W \times S$ a $\Gamma_B \subset B \times S$ takto: $\langle W_i, S_j \rangle \in \Gamma_W$ (resp. $\langle B_i, S_j \rangle \in \Gamma_B$) práve vtedy, ak kameň W_i (resp. B_i) napadá pole S_j.

Relácia Γ_W definuje simpliciálny komplex $K_W(S)$ tak, že k-simplex $\sigma_k = \{S_{i_0}, S_{i_1}, \ldots, S_{i_k}\} \in K_W(S)$ práve vtedy, ak existuje aspoň jeden kameň W_σ taký, že pre každé j, $0 \leqq j \leqq k$, $\langle W_\sigma, S_{i_j} \rangle \in \Gamma_W$. Obdobne relácie $\tilde{\Gamma}_W$, Γ_B, a $\tilde{\Gamma}_B$ definujú simpliciálne komplexy $K_S(W)$, $K_B(S)$ a $K_S(B)$. Rozmernosť simplexu σ_k je $\dim \sigma_k = k$ a rozmernosť celého simpliciálneho komplexu K je $\dim K = \max \{k : \sigma_k \in K\}$.

Dva simplexy q-súvisia, ak majú q spoločných bodov (prvkov). Súvislosť v $K_W(S)$, resp. $K_B(S)$ určuje súhrn kameňov rovnakej farby. Súvislosť v $K_S(W)$, resp. $K_S(B)$ určuje spoločné postavenie kameňov rovnakej farby.

Napríklad mat čiernemu kráľovi BK možno vyjadriť formulou:

$$((\sigma_{BK} - B) \cup p(BK)) \text{ je podkomplexom } K_W(S).$$

Zavedením $K_{W \cup B}(S)$ a $K_S(W \cup B)$ možno analyzovať aj vzájomné pôsobenie bielych a čiernych kameňov.

4.2. Pamäť

Z predchádzajúceho je vidieť, že ohodnocovanie pozícií je ťažké a nie je nádej úspešne hrať šach len na základe hodnotenia pozície. Je tu však jednoduchý nápad: nehodnotiť pozície vôbec, ale jednoducho si ich aj s ohodnotením zapamätať, alebo skúsiť urobiť učiaci sa program, ktorý by si pamätal všetky partie, čo hral a jednoducho vždy, keď prehrá, zmenil posledný ťah, ktorý už viedol do nutne prehraného postavenia. Ukážeme, že neexistujú prostriedky, ktorými bylo možné tento prístup realizovať.

Základom dôkazu je odhad počtu možných postavení. Zhruba by sa dalo povedať,

že máme 32 figúr a každá z nich môže stáť na 64 poliach tak, že na ich zakódovanie treba 32×6 ($\log_2 64$) bitov. Tu však rátame mnoho zo základnej pozície neodvoditeľných postavení a na druhej strane nerátame postavenia, kde nejaká figúra chýba. Vylepšiť tento odhad možno tým, že budeme všetkých pešiakov bodovať tak, že využijeme skutočnosť, že títo môžu stáť len na 48 poliach. Zapamätáme si ich bitový obrazec a potom postupnosť 16 bitov, ktorá udáva, či je pešiak biely alebo čierny. Zvyšných 16 figúr si pamätáme normálne. Takto na zapamätanie pozície potrebujeme len 160 bitov.

Samozrejme potrebujeme ešte ďalšie bity na rochády, branie mimochodom a pod. Nie je asi dôvod pridávať ďalší bit na údaj, kto je na ťahu, pretože skúmame pozície z hľadiska hráča, ktorý je na ťahu (považujeme ho za bieleho). Pozície, kde je čierny na ťahu jednoducho „prefarbíme". Určiť presnú hodnotu počtu postavení alebo čo najlepší horný odhad je dosť ťažké. Spokojíme sa s odhadom $2^{160} \doteq 10^{48}$ postavení. Aj keby to bolo len 10^{40}, naše úvahy sa príliš nezmenia. Na uchovanie 10^{48} postavení potrebujeme približne 10^{50} bitov (sme zámerne skromní).

Pre tých, čo majú slabšiu predstavivosť o veľkých číslach uvádzame, že protón váži približne $1{,}66 \cdot 10^{-24}$ g. To znamená, že keby sme chceli uchovávať 1 bit informácie v jednej jadrovej častici (systém protón, neutrón) a to je určite hranica fyzikálnych možností, potrebovali by sme $1{,}66 \cdot 10^{20}$ ton hmoty a to je približne hmota celej zeme. Navyše rok má len asi $3{,}15 \cdot 10^{13}$ μsec. Pokus naplniť takúto pamäť, aj keby sme na ňu zohnali dosť hmoty a dokázali ju plniť rýchlosťou jednu pozíciu za 1 μsec, by nemal nádej na úspech ani za čas celej existencie vesmíru.

Na druhej strane, ak sa pozrieme na množstvo informácií partiového materiálu, ktorým disponuje silný šachista, tak za posledných 10 rokov je zhrnutý v 32 *Šahovskich informatoroch*, čo je kniha, ktorá obsahuje zhruba 1000 partií (preháňame) o 100 pozíciách. Predstavuje to teda $3{,}2 \cdot 10^6$ pozícií. Keby sme na kódovanie jednej pozície použili 200 bitov, je to len $6{,}4 \cdot 10^8$ bitov a všetko sa pohodlne umiestni na jeden disk o 100 M bytoch. Navyše šachista si obvykle vyberá svoj repertoár otvorení, čím eliminuje veľkú časť tohto materiálu.

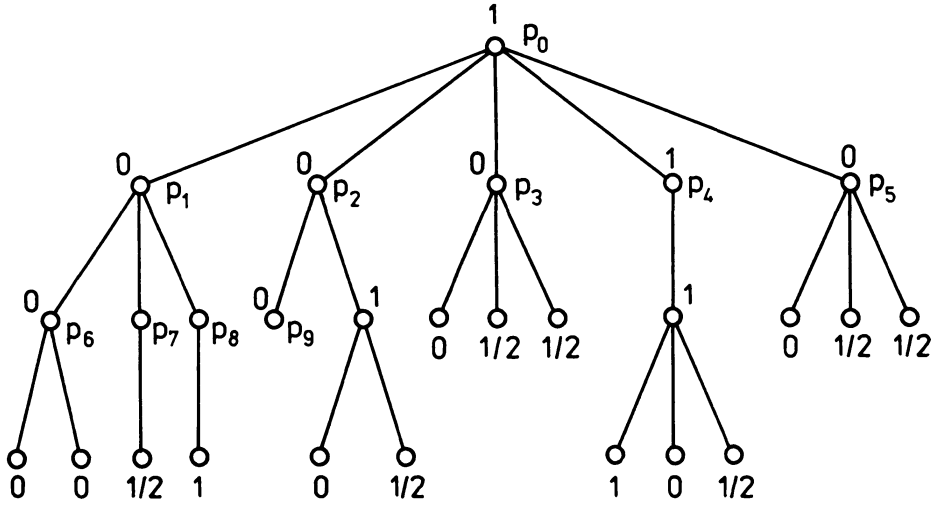
Vytvorenie bázy dát partiového materiálu je teda reálne a bolo by veľkou pomocou nie len pre počítačový šach, ale aj pre praktických hráčov. V súvislosti s tým chceme upozorniť ešte na to, že tlač *Šahovského informatora* a jemu ekvivalentných materiálov je značne štandardizovaná a nebolo by veľkým technickým problémom naprogramovať pre ňu optický čítač, čím by sa vyhlo prakticky nezvládnuteľnej práci s dierovaním 32 kníh.

4.3. Prepočet variantov

Prepočet variantov spočíva vo vytváraní stromu možností hry. V koreni stromu je pozícia, v ktorej sa práve nachádzame, z nej sa rozbiehajú vetve všetkých možných ťahov a vznikajú nové pozície zodpovedajúce stavu hry o polťah neskôr. Tie sa ďalej rozvinujú, až pokiaľ nedosiahneme pozície, ktoré sú podľa pravidiel hry konečné (terminálne), predstavujú listy stromu a sú teda ohodnotené výsledkom hry. Z hodnôt listov sa spätne ohodnocujú aj vnútorné uzly stromu procedúrou, ktorá vždy vyberá najlepšiu možnosť

pre hráča, ktorý bol práve na ťahu. Vetvy (hrany) stromu sú ohodnotené ťahmi, ktorými sa dosahuje prechod medzi pozíciami.

Príklad stromu:



V uvedenom strome kvôli jednoduchosti vynechávame označenie hrán. Vidíme, že je len nejakým koncovým úsekom hry. Takouto procedúrou môžme riešiť každú hru s úplnou informáciou. Teoreticky to vždy vedie k cieľu. Nevýhodou je, že strom sa strašne rýchlo rozrastá s počtom ťahov, ktoré musíme prepočítať do hĺbky. Ak predpokladáme uniformnú hru so stupňom uzla d, t. j. hru, kde v každej neterminálnej pozícii je d možných pokračovaní, potom veľkosť stromu pri prepočítaní n polťahov (ťah bieleho, resp. čierneho) je d^n. Šach nie je uniformná hra, $1 \leqq d \leqq 130$ a priemerné vetvenie d^* sa odhaduje na $d^* = 35$.

Poblém prepočtu variantov je tedy práve tak nezvládnuteľný ako predošlé dva problémy. Aby sme sa z toho nejak vymotali, musíme si vziať za vzor myslenie človeka – šachistu. Šachista sa obvykle nesnaží dopočítať hru až do konca, ale len po nejakú pozíciu, ktorú už vie ohodnotiť a navyše počíta za seba len tie možnosti (ťahy), ktoré sú v súlade s jeho strategickým zámerom.

Skôr než si uvedieme nejaké metódy ohraničovania prepočtu, budeme sa chvíľku zaoberať prehľadávaním stromu. Základom je tzv. „minimaxová" procedúra, ktorá predpokladá hru dvoch hráčov Mina a Maxa, z ktorých jeden sa snaží maximalizovať výhru a druhý minimalizovať prehru. Aby program nemusel striedať dva rôzne kroky, budeme v každom kroku pre Mina radšej obracať znamienko ohodnotenia pozície a aj Min bude maximalizovať. Algoritmus prehľadávania potom môžme jednoducho napísať takto:

```
function F (p: position): integer;
var m, i, t, d: integer; function f: integer;
begin
if terminal (p) then F := f(p) else
  begin generate (p, d); ⟨determine the succesor positions p_1, ..., p_d⟩
      m := −∞;
      for i := 1 to d do
        begin t := −F(p_i);
            if t > m then m := t
        end;
        F := m
  end
end;
```

Táto rekurzívna funkcia síce správne ohodnotí strom, ale robí množstvo zbytočnej práce. V našom príklade, ak ohodnotíme $F(p_6) = 0$, potom už nemusíme hodnotiť p_7 a p_8. A ak už raz ohodnotíme $F(p_4) = 1$, je zbytočné ďalšie pokračovanie, pretože viac ako vyhrať sa už nedá. Zahrnutím tejto myšlienky do programu vzniklo tzv. alfa-beta prehľadávanie (alpha-beta pruning).

```
function AB (p :↑ position; a, b: integer): integer;
var m, t: integer; q :↑ position;
function f: integer;
begin generate (p);
    q := first (p); ⟨p refers to the list of successor positions⟩
    if q = nil then AB := f(p) else
    begin m := a;
        while m < b ∧ q ≠ nil do
            begin t := −AB (q, −b, −m);
                if t > m then m := t;
                q := next (q)
            end;
        AB := m
    end
end;
```

Ohodnotenie celého stromu hry sa vykoná vyvolaním *AB* (koreň, $-\infty$, ∞).

Pri praktickom programovaní sa samozrejme používa táto procedúra s odstránenou rekurziou a optimalizovaná.

Knuth a Moore [1975] ukázali, že pre uniformný strom výšky h každá ohodnocujúca procedúra musí ohodnotiť aspoň $d^{\lceil h/2 \rceil *} + d^{\lfloor h/2 \rfloor} - 1$ terminálnych pozícií ($\lceil h/2 \rceil *$ je najmenšie celé číslo väčšie alebo rovné $h/2$, podobne $\lfloor h/2 \rfloor$ je najväčšie celé číslo menšie alebo rovné $h/2$) a *AB* procedúra dosahuje tohto minima, ak sa najlepší ťah generuje ako prvý. Je teda v istom zmysle optimálna.

Nevýhodou alfa-beta prehľadávania je, že vyžaduje pomerne presné určenie, kedy možno pozíciu považovať za terminálnu a presné ohodnotenie terminálnych pozícií. Prvé šachové algoritmy sa definovali ako terminálne pozície v určitej hĺbke. Takáto definícia terminálnych pozícií má za následok snahu programu odsunúť nevyhnutné stráty až za túto hĺbku na jednej strane a preceňuje prechodné výhody získané na hranici tejto hĺbky na strane druhej. V oboch prípadoch sa jedná o odsunutie nepríjemností za „horizont"; hovoríme preto o posuve horizontu. Posuv horizontu je nežiadúci jav, ktorému sa nemôže vyhnúť žiaden program, ktorý prehľadáva strom do konštantnej hĺbky.

Berliner [1979] prišiel so zaujímavou ídeou, že v hre nejde vlastne o hodnotenie pozície, ale o nájdenie najlepšieho ťahu. K tomuto účelu používa dve ohodnocovacie funkcie – optimistické ohodnotenie a pesimistické ohodnotenie. Prehľadávanie pokračuje až dovtedy, pokiaľ pesimistické ohodnotenie nejakého ťahu (t. j. pozície vzniklej po tomto ťahu) nie je väčšie než optimistické ohodnotenie ľubovoľného iného ťahu. Ak budeme predpokladať, že o jednotlivých uzloch stromu vedieme nasledujúci dátový typ

```
node: record pos: position;
             opt : pes : integer;
             pred :↑ node;
             expanded : boolean
      end;
```

kde pos je vlastná pozícia na šachovnici, opt a pes sú jej optimistické, resp. pesimistické ohodnotenia, expanded je údaj o tom, či z danej pozície sa už uvažoval nasledujúci ťah a pred je smerník na predošlú pozíciu, potom ídeu Berlinerom navrhnutého prehľadávania „najlepší – najprv" (best-first search) popisuje nasledujúci algoritmus. Výsledok kvôli zjednodušeniu v tejto schéme nie je ťah, ale pozícia, ktorá týmto ťahom vzniká.

```
procedure BF (p : node; maxopt, maxpes : integer) RESULT (pl : node);
var p, q, pl, best, altern : node;
    maxopt, maxpes : integer;
function optimistic, pesimistic (p : node) : integer;
begin if ¬ expanded (p) then
```

```
begin generate (p); q := first (p);
      while q ≠ nil do begin q . pred := p;
                             q . opt := optimistic (q);
                             q . pes := pesimistic (q);
                             next (q)
                       end
end;
if p . pred = nil then begin first (q); maxopt := −∞; maxpes := −∞;
                             best := q;
                       while q ≠ nil do
                       begin if q . opt > maxopt then
                         begin maxopt : = q . opt; altern := best;
                               best := q
                         end;
                         if q . pes > maxpes then maxpes := q . pes;
                         next (q)
                       end;
                       if best . pes ≧ altern . opt then pl := best
                         else begin decide strategy;
                                    if provebest then p := best
                                                 else p := altern
                              end
                  end;
   while (maxopt < p . pes) ∨ (maxpes > p . opt) do
    begin p . pes := maxopt; p . opt := maxpes;
          if p . pred ≠ nil then p := p . pred
    end;
  BF (p, maxopt, maxpes, pl)
end;
```

Obe uvedené procedúry sa líšia od spôsobu prepočtu používaného človekom hlavne tým, že počítač využíva hrubú silu (brute force) a počíta všetky možnosti.

5. Technické aspekty realizácie šachovej hry na počítači

Šach na rozdiel od iných hier spôsobuje programátorom ťažkosti aj rozmanitosťou svojich ťahov. Zakiaľ čo človeku geometrické predstavy a rozmanitosť figúr uľahčujú zapamätanie si pozície, pri programovaní táto rozmanitosť spôsobuje narastanie programu a podstatne komplikuje program generovania ťahov.

Boli vymyslené tri pomerne rafinované spôsoby, ako sa s problémom generovania ťahov vysporiadať. Prvý z nich používa tabuľky, ktoré sa ako šablony prikladajú na každé pole. V podstate je v nich pre každé pole a pre každú figúru zaznačené, kam môže ísť z tohto poľa. Druhý simuluje ťahy na 12 × 12 šachovnici a krajné rady a stĺpce slúžia ako zarážky. Ďalší spôsob využíva priamo bitové obrazce šachovnice. Sú to v podstate 64bitové počítačové slová: W slovo, ktoré má obsadené bity, kde stoja kamene hráča na ťahu, slovo A pre všetky kamene a slovo P pre pešiakov hráča na ťahu. Tieto slová sa využívajú pri generovaní ťahov pomocou strojových logických operácií. Napríklad postupy pešiakov sa vypočítajú naraz priamo strojovými operáciami:

SHR 8, P ⟨Posuv o osem doprava⟩

AND $\neg A$ ⟨Prienik negáciou A⟩

Výsledkom sú všetky polia, na ktoré môžu pešiaci ísť. Pre branie používane posuvy o 7, resp. 9 a špeciálne konštanty, ktoré zabraňujú cyklickému prechodu krajných pešiakov. Ťahy líniovými figúrami sú trochu komplikovanejšie, pretože treba sledovať ich pohyb.

Pre atypické ťahy ako branie mimochodom, rochády a premeny pešiakov sa používajú špeciálne programy a techniky. Aj najlepší program generovania ťahov však potrebuje rádove desiatky strojových inštrukcií pre jeden ťah.

Programy úspešné v majstrovstve sveta počítačov 1980 používali špecializovaný hardware, ktorý je realizovaný pomocou VLSI technologie a vlastne simuluje ťahanie na šachovnici. Takéto zariadenie umožňuje generovať ťah na jednu strojovú operáciu. Taktiež zápis pozície do tohto zariadenia je jeden takt práce stroja. Dokonca boli navrhnuté a realizované špecializované počítačové systémy pre hranie šachu. Napríklad sysém CHEOPS rozpracovaný na MIT [Moussouris, Holloway, Greenblatt, 1980] pozostáva z 16-bitovej aritmetickej jednotky, špeciálneho zariadenia pre generovanie ťahov, špeciálnej šachovej zásobníkovej pamäti (1 pozícia = 1 slovo) a dvoch počítačov PDP.

6. Využívanie špecifických šachových poznatkov — základný rozdiel medzi hrou človeka a počítača

Je niekoľko rozdielov v práci šachistu a šachového programu. Šachista na rozdiel od programu za seba počíta len ťahy, ktoré sa zhodujú s jeho strategickými zámermi. Ak je v obrane prinútený počítať viac, snaží sa v prvej rade eliminovať vyslovene zlé ťahy. Väčšinou sa ťah, ktorý v danej pozícii vôbec neprichádza v úvahu, dá rozpoznať veľmi rýchlo (pri hĺbke prehľadávania 0—4 poľťahy). Avšak rozhodnutie o tom, ktorý z „kandidátov" na pokračovanie v partii je lepší, sa často skrýva až veľmi hlboko v stro-

me hry (12–20 polťahov prípadne ešte hlbšie). Nepovažujeme to však za podstatný rozdiel, pretože veríme, že úprava uvedených prehľadávacích algoritmov tak, aby pracovali takým istým spôsobom ako šachisti, je asi jednoduchá.

Najpodstatnejší rozdiel je však asi v tom, že človek sa pri výbere ťahu riadi nielen prepočtom, ale aj všeobecnými šachovými vedomosťami a poučkami. Aj pri tvorbe šachových programov je snaha využívať špecifické šachové poznatky a pomocou nich generovať len ťahy, ktoré v danom postavení majú nejaký význam. Takýto program pozostáva z nejakého systému reprezentácie šachových poznatkov, všeobecného mechanizmu riešenia problémov (general problem solver) a systému rozpoznávania obrazcov (pattern recognition system). Na postavenie najprv aplikuje systém rozpoznávania, ktorý rozpozná typ postavenia. Podľa typu postavenia sa rozhodne, ktoré poznatky treba aplikovať a potom mechanizmus riešenia problému vygeneruje plán, prípadne ťahy, ktoré prichádzajú do úvahy na splnenie tohto plánu. Bohužiaľ, programy pracujúce na základe takejto a podobných schém (Botvinik 1980, Wilkins 1980) neboli zatiaľ dovedené do takého stavu, aby mohli hrať celé partie. Ich silu môžme odhadovať len na základe malého množstva experimentov s riešením niektorých postavení.

Máme za to, že hlavná ťažkosť tohto prístupu spočíva v nedostatočnej pripravenosti samotnej šachovej teórie. Len malá časť šachových poznatkov má totiž charakter tvrdení a poučiek. Vo veľkej väčšine prípadov ide len o súhrn príkladov, z ktorých vlastný poznatok treba vyabstrahovať. Vzhľadom k tomu, že v mnohých prípadoch ide o poznatky, ktoré nemožno považovať za definitívne pravdivé, musel by asi dobrý program postavený na tejto báze vedieť čítať šachovú literatúru a aktualizovať svoju bázu poznatkov. Ak sa raz podarí takýto program vytvoriť, bude to definitívna odpoveď na otázku, či stroj môže myslieť.

7. Ukážky hry počítača a vyhliadky do budúcnosti

Pokúsime sa v tomto odseku charakterizovať doterajší vývoj šachových programov, súčasný stav a urobiť nejakú extrapoláciu do budúcnosti. Prvé šachové programy hrali na úrovni začiatočníkov a vzhľadom k tomu, že aj rýchlosť počítačov bola menšia, potrebovali na ťah približne 8 minút. Od sedemdesiatych rokov hrajú počítače podľa turnajových pravidiel, t. j. môžu spotrebovať v priemere 3,5 min na ťah.

Začneme partiou Moskva – Stanford zo zápasu USA – ZSSR 1967.

1.e4 e5 2.Jf3 Jc6 3.Je3 Sc5 4.Jxe5 Jxe5 5.d4 Sd6 6.dxe5 Sxe5 7.f4 Sxc3 8.bxc3 Jf6 9.e5 Je4 10.Dd3 Jc5 11.Dd5 Je6 12.f5 Jg5 13.h4 f6 14.hxg5 fxg5 15.Vxh7 Vf8 16.Vxg7 c6 17.Dd6 Vxf5 18.Vg8+ Vf8 19.Dd8x.

Ako vidno z partie, stanfordský program nepočítal ani 3 ťahy dopredu a nestačil klásť seriózny odpor, takže slabiny sovietskeho programu úplne zanikli.

Z prvého majstrovstva sveta počítačov v Stockholme 5.–8. augusta 1974 uvádzame partiu víťazného programu.

Kaisa – Chaos

1.e4 c5 2.Jf3 Jc6 3.c3 d5 4.exd5 Dxd5 5.d4 Sg4 6.Se2 e6 7.0–0 Jf6 8.Se3 cxd4 9.Sxd4 e5 10.h3 exd4 11.hxg4 Sd6 12.cxd4 Jxg4 13.Jc3 Dh5 14.g3 Kd7 15.Jh4 f5 16.d5 Jce5 17.Dc2 Vhf8 18.Sd3 Jxd3 19.Dxd3 Vae8 20.Jb5 f4 21.Jxd6 Kxd6 22.Da3 Kc7 23.Dxa7 Df7 24.Vfc1+ Kd6 25.Dc5+ Ke5 26.d6+ Ke6 27.Ve1+ Je3 28.gxf4 Vd8 29.Vxe3 + Kf6 30.f5 Dd7 31.Ve7 Da4 32. De5+ Kg5 33.Jf3+ Kg4 34.Vxg7+ Kh5 35.Dh2+ Dh4 36.Dxh4 x

Ďalšia partia je partia víťazného programu z majstrovstva sveta počítačov v Toronte 1977.

Belle – Chess 4.6.

1.e4 Jc6 2.Jf3 e6 3.d4 d5 4.Jc3 Sb5 5.e5 Jge7 6.a3 Sxc3 7.bxc3 Ja5 8.Sb5 Sd7 9.Sd3 Vc8 10.Jg5 h6 11.Jf3 c5 12.dxc5 Vxc5 13.Se3 Vxc3 14.Sxa7 Jc4 15.0–0 Vxa3 16.Vxa3 Jxa3 17.Sc5 Da5 18.Sd6 Jc4 19.Da1 Jc6 20.Dxa5 J6xa5 21.Va1 Sc8 22.c3 Jc6 23.Va4 Jxd6 24.exd6 Kd7 25.Vg4 g5 26.Sc2 Kxd6 27.Va4 b5 28.Va1 b4 29.cxb4 Jxb4 30.Sb1 Sd7 31.Kh1 f5 32.Jd4 Vc8 33.Je2 Sb5 34.Jg1 Vc1 35.Va5 Vxb1 36.f3 Sf1 37.h4 Vb2 38.hxg5 Sxg2+ 39.Kh2 hxg5 40.Va4 Sxf3+ 41.Kg3 Sh5 42.Kh3 f4 43.Va8 Sg6 44.Kg4 Vg2+ 45.Kh3 Vxg1 46.Kh2 Vg3 47.Vd8 + Ke5 48.Vg8 Se4 49.Vg7 Sf3 50.Vh7 Jd3 51.Vh3 Vg2 + 52.Kh1 Jf2x.

Na majstrovstve sveta počítačov v Linci 1980 sa o prvenstvo rozdelili programy Belle a Chaos, ktoré sa vzájomne nestretli (turnaj sa hrá švajčiarskym systémom). Uvádzame ich dodatočnú partiu mimo súťaž (podľa pravidiel turnaja sa majstrom sveta stal program Belle).

Belle – Chaos.

1.e4 Jf6 2.e5 Jd5 3.d4 d6 4.Jf3 dxe5 5.Jxe5 g6 6.g3 Sf5 7.c4 Jb4 + 8.Da4 J4c6 9.d5 Sc2 10.Db5 Dd6 11.Jxc6 Jxc6 12.Jc3 Sg7 13.Dxb7 0–0 14.Dxc6 Db4 15.Kd2 Se4 16.Vg1 Vfb8 17.Sh3 Sh6+ 18.f4 Da5 19.Ve1 f5 20.De6 + Kf8 21.b3 Sg7 22.Sb2 Sd4 23.g4 Vb6 24.Dd7 Vd6 25.Da4 Db6 26.Sa3 Sxc3 + 27.Kxc3 Vdd8 28.Vad1 Df2 29.gxf5 Dc2+ 30.Kd4 gxf5 31.Dc6 Df2+ 32.Ke5 Kg8 33.Vg1 + Kh8 34.Sxe7 Db2+ 35.Vd4 Dg2 36.Df6+ Kg8 37.Sxg2 Vxd5 + 38.Ke6 h6 39.Dxh6 Ve5+ 40.fxe5 Vf8 41.Sf3x

Ako vidíme, aj najlepším súčasným programom k majstrovskému šachu ešte niečo chýba. Na druhej strane však existuje dnes už veľa šachových kalkulačiek, ktoré sice

hrajú na úrovni šachistu 3.—2. triedy (s výnimkou koncoviek, ktoré hrajú slabo). Uplatňujú sa ako výborní pomocníci skladateľov a riešiteľov dvoj- a trojťahových úloh. Špecializovaný veľký program použili Arlazarov, Donskoj a Bittman [1980] pri rozpracúvaní systematickej teórie koncoviek veža a pešiak proti veži. (Pomocou tohto programu našli pozíciu, kde k vynúteniu postupu pešiaka treba 60 ťahov!) V praktickom šachu však programy dosahujú najviac ak úroveň kandidáta majstra.

Napriek tomu sú dosť dobré vyhliadky na to, aby sa počítače ešte podstatne zlepšili. Je tu, ako sa domnievame, istá rezerva v samotných algoritmoch prehľadávania stromu hry a istá rezerva vo vypracovaní repertoáru otvorení systematickým spôsobom. Cesty k zlepšeniu programov formalizáciou šachových poznatkov sú asi najperspektívnejšie, ale vzhľadom k tomu, že si ťažko vieme predstaviť využitie výsledkov iné ako pre šach samotný, sú aj najmenej lákavé. Záverom možno povedať, že v roku 1978 vyhral anglický majster Levy sázku, ktorú uzavrel pred desiatimi rokmi. Sázka znela tak, že ho najbližších desať rokov žiadny počítač neporazí v zápase na 10 partií. Domnievame sa, že uzavrieť podobnú sázku dnes by bolo značne riskantné.

Literatúra

ARLAZAROV, V. L.; DONSKOJ, M. V.; BITTMAN, A. C.: *Complete treating of the Rook and Pawn vers. Rook chess endgames.* Meždunarodnoje soveščanie po iskustvennomu intelektu. Leningrad 1980.

ATKIN, R. H.; WITTEN, I. H.: *A multidimensional approach to positional chess.* Int. journal Man-machine studies 7 (1975), str. 727—750.

BERLINER, H.: *The B* tree search algorithm: A best-first proof procedure.* Artificial Intelligence *12* (1979), str. 23—40.

BOTVINNIK, M. M. a kol.: *Myšlenije čelovjeka i kompjutera.* Predvariteľnaja publikacia VNIIE, Moskva 1980.

FORMÁNEK, B.: *353 šachových problémov.* Bratislava 1962.

FRAENKEL, A. S.; LICHTENSTEIN D.: *Computing a perfect strategy for n × n chess requires time exponential in n.* Nepublikovaný materiál.

GAREY, M. R.; JOHNSON, D. S.: *Computers and Intractability: A guide to the theory of NP-completeness* San Francisco 1979.

MOUSSOURIS, J.; HOLLOWAY, J.; GREENBLATT, R.: CHEOPS: *A chess-oriented processing system.* Meždunarodnoje soveščanie po iskustvennom intelektu. Leningrad 1980.

SHANNON, C. E.: *Programming a computer for playing chess.* Phil. Mag. (London) 7 ser. *41* (1950), str. 256—275.

SOUČEK, L.: *Tušenie tieňa.* Bratislava 1981.

STOCKMEYER, L. J.; CHANDRA, A. K.: *Provably difficult combinatorial games.* SIAM Journal on Computing *8* (1979), str. 151—174.

TURING, A. M.: *Computing machinary and intelligence.* Mind, *59* (1950), str. 433—460.

VOROBJEV, N. N.: *Matričnyje igry.* Moskva 1961.

WILKINS, D. E.: *Using patterns and plans in chess.* Artificial Intelligence *14* (1980), str. 165—203.

ZERMELO, E.: *Über eine Anwendung der Mengenlehre auf die Theorie des Schachspiels.* Proc. of Fifth International Congress of Mathematicians. Cambridge 1913, str. 501—504.