```
colourable([WA,SA,NT,Q,NSW,V,T])                                    :-
rgb(WA),rgb(SA),rgb(NT),rgb(Q),rgb(NSW),rgb(V),rgb(T),


WA\=NT,NT\=Q,Q\=NSW,NSW\=V,SA\=WA,SA\=NT,SA\=Q,SA\=NSW,S
A\=V.
```

هر مسأله ی ارضای محدودیّت با دامنه ی محدود می تواند به صورت یک شرط صریح تکی با برخی واقعیّات اضافه بیان شود . هر اتصال [۱] ، یک محدودیت در متغیّرهای شامل شده می باشد . بنابراین ، مسأله ی ارضای محدودیت را به زبان پرولوگ بنویسید و اجازه بدهید که مفسر [۲] آن را حل کند . مسأله این است که روش اوّل – عمق پرولوگ در بیش تر موارد ، بهترین نمی باشد .

برنامه نویسی منطقی محدود

الگوریتم هایی از الگوریتم های مسأله ی ارضای محدودیّت به پرولوگ استاندارد اضافه شده اند که معمولاً دارای نام گستره یا پسوند یا دامنه ی CLP می باشند (به صورت فایل های CLP.* هستند . [۳]) . CLP(FD) در دامنه های محدود ؛ CLP(B) در مورد بولین ها و CLP(Q,R) در مورد عقلانیّات و واقعیّات می باشند . با استفاده از مکانیزم محلی [۴] پرولوگ برای توسعه ی زبان پیاده سازی شده اند .

مثال : مسأله ی رمزی ؛ SEND+MORE=MONEY

```
:- use_module(library('clp/bounds')).

cryptarithmetic(S,E,N,D,M,O,R,Y) :-
```

---

[۱] conjunct

[۲] interpreter

[۳] مترجم

[۴] native

```
Digits= [S,E,N,D,M,O,R,Y],
Carries = [C1,C2,C3,C4],
Digits in 0..9,
Carries in 0..1,
M #= C4,
O + 10 * C4 #= M + S + C3,
N + 10 * C3 #= O + E + C2,
E + 10 * C2 #= R + N + C1,
Y + 10 * C1 #= E + D,
M #>= 1,
S #>= 1,
all_different(Digits),
label(Digits).
```

۱

فصل هشتم

تئوری بازی ها [۱]

[۱] – تصویرهای بالا متعلق به گری کاسپاروف (Gary Kasparov) ، قهرمان شطرنج روسی است .

[۱] Game playing

مترجم: سهراب جلوه گر
ویرایش دوّم ، بهار ۱۳۸۸

رئوس مطالب

- بازی ها
- بازی کامل [۱]

– تصمیم گیری های مینیماکس [۲]

– هرس [۳] α-β

- محدودیت منابع و ارزیابی تقریبی [۴]
- بازی های تصادفی [۵]
- بازی های با اطّلاعات غیر کامل [۶]

---

[۱] Perfect play

[۲] Minimax decisions

[۳] pruning

[۴] Resource limits and approximate evaluation

[۵] Games of chance

[۶] Games of imperfect information

مترجم: سهراب جلوه گر
ویرایش دوّم ، بهار ۱۳۸۸

محیط های چندعاملی

الگوریتم های جستجویی که ما تا حالا بررسی کرده ایم ، دارای یک محیط تک عاملی بودند . و عامل در حال کار در یک جهان بی اثر [۱] می باشد به عبارت دیگر ، دیگر عامل ها بر روی آن اثری نداشتند ؛ امّا در مواردی که یک عامل باید دیگر عامل ها را به حساب بیاورد چه طور ؟ مثل محیط های رقابتی نظیر بازی های شطرنج ، چکرز ، go و ... ؛ یا حراجی ها [۲] ، فروشگاه های برخط و محیط های شراکتی [۳] مثل عامل های یک تیم فوتبال .

تئوری بازی ها – تئوری بازی ها ، شاخه ای از علم ریاضیّات یا علم اقتصاد می باشد که به اثرات متقابل میان چند عامل می پردازد و روش هایی را برای تشخیص رفتار بهینه ، تعیین می نماید .

بازی های با اطّلاعات کامل : عامل ها به همه ی دانش در مورد محیط ، دسترسی دارند . *ما این محیط را محیط کاملاً قابل مشاهده می نامیم .*

بازی های با اطّلاعات ناقص : عامل باید در مورد جهان یا حریفش ، استنتاج نماید . *ما این محیط را محیط قابل مشاهده به صورت جزیی می نامیم .* مثال ها ، بازی های شانسی ، بیش تر برهم کنش های دنیای واقعی و برخی از حراجی ها می باشند .

مفروضات تئوری بازی ها :

عقلانیّت کامل – عملکردها ، همیشه کار درست را انجام خواهند داد .

---

[۱] neutral

[۲] auctions

[۳] cooperative

مترجم: سهراب جلوه گر
ویرایش دوّم ، بهار ۱۳۸۸

محاسبه ی نامحدود – عامل ها همیشه قادر به تشخیص این هستند که چه کاری برای انجام دادن ، درست می باشد .

همان طور که دیده ایم ، این مفروضات در برخی از موارد ، مصداق ندارند (درست نمی باشند) . به هر حال ، ما هنوز از برخی از ایده های تئوری بازی ها برای کمک در مورد تصمیم گیری در مورد این که چه کاری برای انجام ، درست می باشد ، استفاده می نماییم . بنابراین ، یک عامل چگونه در یک محیط چند عاملی در مورد این که چه کار باید بکند تصمیم گیری نماید ؟

روش های بهینه

بیایید با ساده ترین نوع بازی ها شروع کنیم ؛ مثل ، بازی های دو نفره ، بازی های با اطّلاعات کامل و بازی های رقابتی به صورت کامل – که در آن ها یکی می برد و دیگری می بازد . ما می توانیم این بازی ها را به صورت یک مسأله ی جستجو تعریف نماییم . تئوری بازی ها یکی از قدیمی ترین مباحثی است که در هوش مصنوعی راجع به آن زیاد مطالعه شده است . دلیل این مطلب آن است که : اوّل ، افراد بازی ها را دوست دارند ! و خوب می توانند با بازی ها کار کنند . دوّم ، اغلب ، بازی ها به صورت یک نماینده ی هوش دیده می شوند . سوم ، بازی ها دارای یک توضیح واضح از محیط می باشند ، ولی در آن ها فضاهای حالت ، خیلی بزرگ و پیچیده اند ، بنابراین برخی اوقات اطّلاعات اتّفاقی و ناکار آمد می باشند و این باعث به وجود آمدن چالش در مسأله می شود . وقتی که هوش مصنوعی خوب کار می کند بازی ها واضح و صریح می باشند . بازی ها برای هوش مصنوعی مانند جایزه ای بزرگ برای مسابقه ی طراحی اتومبیل هستند .

مسایل جستجوی بازی ها – در بازی ها حرکت حریف [۱] ، غیر قابل پیش بینی می باشد در نتیجه راه حل ، یک روش مشخص کننده ی یک حرکت ، برای هر جواب ممکن حریف است .

---

[۱] opposite

مترجم: سهراب جلوه گر
ویرایش دوّم ، بهار ۱۳۸۸

محدودیت های زمانی موجود برای پیداکردن هدف ، نا مطلوب یا نـا خوشـایند هسـتند ، در نتیجـه ، بایـد تخمین زده شوند .

تاریخچـه : کـامپیوتر بـه مـوارد ممکـن در بـازی توجّـه مـی کنـد (Babbage,1846) | الگـوریتمی بـرای بـازی کامـل (Zermelo,1912;Von Neumann,1944) | وسـعت محـدود ، ارزیابی تقریبی (Zuse , 1945 ; Wiener , 1948 ; Shannon , 1950) | برنامه ی شطرنج اوّلیـه (Turing , 1951) | آموزش ماشینی [۱] برای بهبود بخشـیدن بـه درسـتی ارزیـابی (Samuel , 1952-57) | بازبینی یا هرس برای اجازه دادن به جستجوی عمیق تر (McCarthy , 1956)

انواع بازی ها

	قطعی	شانسی
اطّلاعات کامل	شطرنج[۲]،بازی چکرز[۳]،go	تخته نرد[۴]
اطّلاعات ناقص	تیک تا تو کور[۵]	پل[۶]،پوکر[۷]

[۱] machine learning
[۲] chess
[۳] checkers
[۴] backgammon
[۵] blind tictactoe
[۶] bridge
[۷] poker

مترجم: سهراب جلوه گر
ویرایش دوّم ، بهار ۱۳۸۸

شطرنج : بازی ای است که بر روی یک صفحه انجام می شود و برای دو بازیگر است که شانزده مهره را با توجّه به قوانینی معین حرکت می دهند . هدف مات کردن شاه طرف مقابل می باشد .

بازی چکرز : صفحه ی بازی چکرز برای دو نفر می باشد که هر نفر دارای دوازده مهره می باشد ؛ هدف پریدن و دستگیر نمودن مهره های حریف می باشد .

بازی go : بازی ای بر روی صفحه برای دو بازیگر می باشد که شمارنده هایی را روی یک شبکه قرار می دهند ، هدف محاصره کردن و سپس دستگیر کردن شمارنده های حریف می باشد . [۱]

تیک – تاک – تو کور : بازی تیک – تاک – تو دارای دو بازیگر می باشد . برای هر دو بازیگر هدف این است که اوّلین کسی باشند که سه شی ء همانند را در یک ردیف ، ستون یا قطر قرار می دهند . صفحه ی این بازی دارای یک شبکه ی سه در سه می باشد . بنابراین دارای نه خانه می باشد . [۲]

تخته نرد : بازی ای بر روی صفحه است و دو بازیگر دارد . هر بازیگر دارای پانزده مهره می باشد که آن ها را در بیست و چهار خانه ی مثلثی شکل با توجّه به عدد ظاهر شده روی دو تاس حرکت می دهد . [۳]

پل : یک بازی کارتی حقّه ای برای چهار نفر می باشد که این چهار نفر به صورت دو گروه دو نفری با هم بازی می کنند و در هر طرف هر گروه مقابل هم می نشیند . بازی پل دارای دو مرحله می باشد : پیشنهاد و بازی . [۴]

پوکر : بازی ای کارتی می باشد ، محبوب ترین بازی از یک دسته از بازی ها به نام بازی های همچشمی می باشد که در آن بازیگران با کارت هایی کاملاً پنهان یا اندکی پنهان روی یک چیزی شرط

---

[۱] wordnet.princeton.edu/perl/webwn

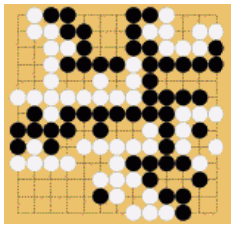
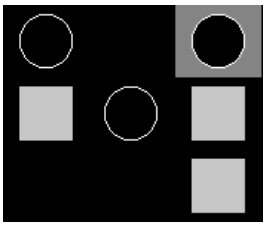
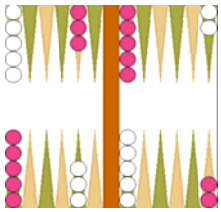
[۲] http://www.seeingwithsound.com/tictactoe.htm

[۳] wordnet.princeton.edu/perl/webwn

[۴] en.wikipedia.org/wiki/Bridge_game

مترجم: سهراب جلوه گر
ویرایش دوّم ، بهار ۱۳۸۸

بندی می کنند ، سپس چیزی که شرط بندی روی آن انجام شده است به بازیگر یا بازیگران باقی مانده ای که دارای بهترین ترکیب کارت ها هستند جایزه داده می شود .[۱] در زیر تصویر صفحه ی چند بازی ای که هم اکنون معرّفی نمودیم مشاهده می نمایید :

نام بازی	چکرز	go	تیک – تاک – تو کور	تخته نرد
تصویر صفحه				

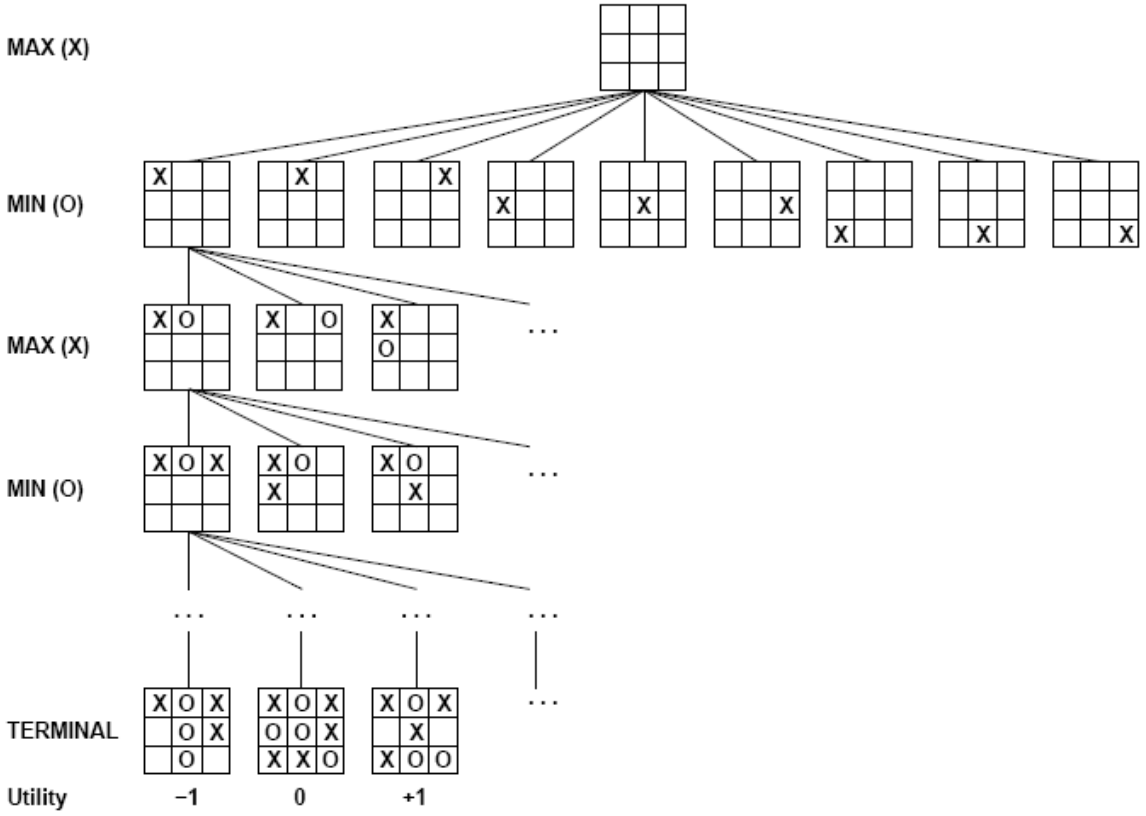
تئوری بازی به صورت جستجو – یک بازی دو نفره با اطّلاعات کامل را در نظر بگیرید ؛ آیا ما می توانیم این بازی را به صورت مسایل جستجو فرمول بندی نماییم ؟ ؛ وضعیّت صفحه ی بازی ، وضعیّت جستجوی مسأله می باشد ؛ حرکت های مجاز ، عملگرها هستند و وضعیّت های پایانی ، وضعیّت هایی هستند که درآن ها بازی را برده ایم یا بازنده شده ایم یا بازی بی نتیجه مانده است . ما می توانیم به بردن عدد ۱+ ، به باختن عدد ۱– و به بازی بی نتیجه مانده عدد ۰ را نسبت دهیم . ما می خواهیم یک روش (یک راه برای انتخاب حرکت ها) که بازی را می برد پیدا نماییم .

در این مورد ، یک حریف وجود دارد . که بداندیش است ؛ یعنی ، چیزهای خوب را برای خود می خواهد و چیز های بد را برای شما می خواهد . ما باید تصمیم گیری حریفمان را شبیه سازی نماییم . توجّه

[۱] en.wikipedia.org/wiki/Poker

کنید که ، یک بازیگر بیشینه وجود دارد که بیش ترین نفع را برای خودش می خواهد و یک بازیگر کمینه وجود دارد که کم ترین نفع را برای خودش می خواهد .

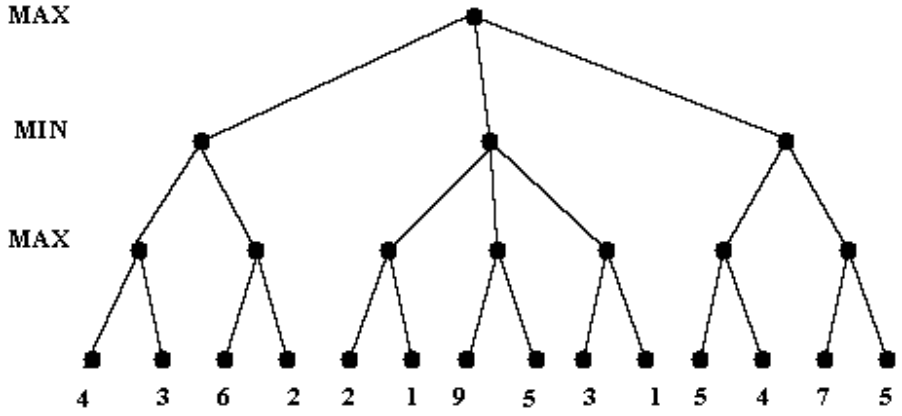
درخت بازی – در حالت کلّی ، ما می توانیم درخت بازی را برای هر بازی به صورت زیر رسم نماییم .



مینیماکس – [۱] MIN و MAX ، دو بازیگر هستند که MAX می خواهد بازی را ببـرد و MIN هم می خواهد بازی را ببرد و در واقع MAX و MIN ، رقیب هم هسـتند . و هـر دو بهتـرین بـازی ممکـن را انجام می دهند .

بازی کامل قطعی [۲] ***، بازی های با اطّلاعات کامل می باشند .***

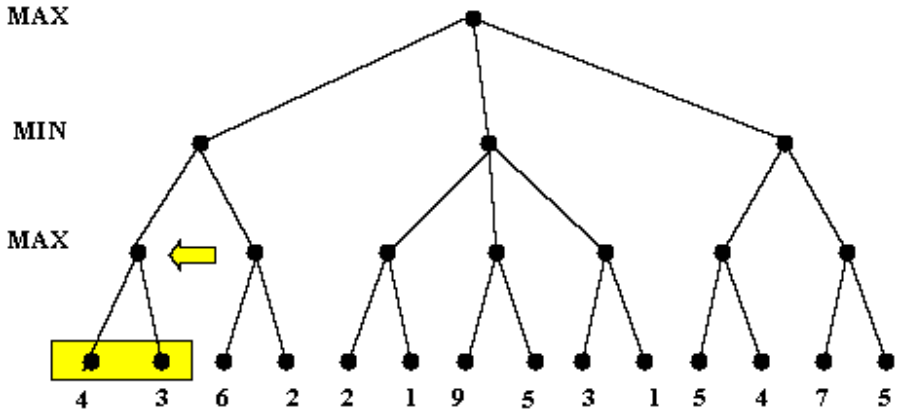
مثال زیر گویای این روش است :

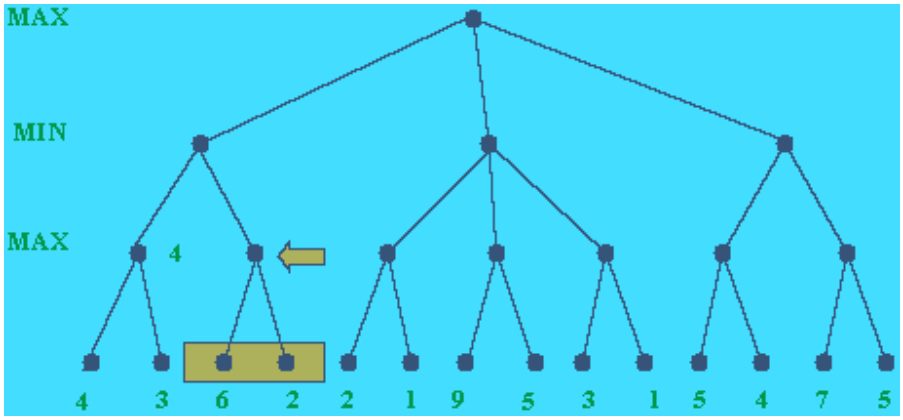


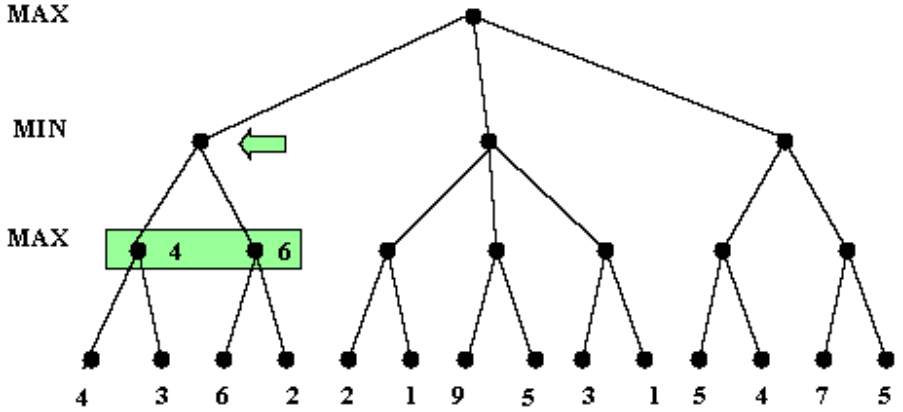
داریم :

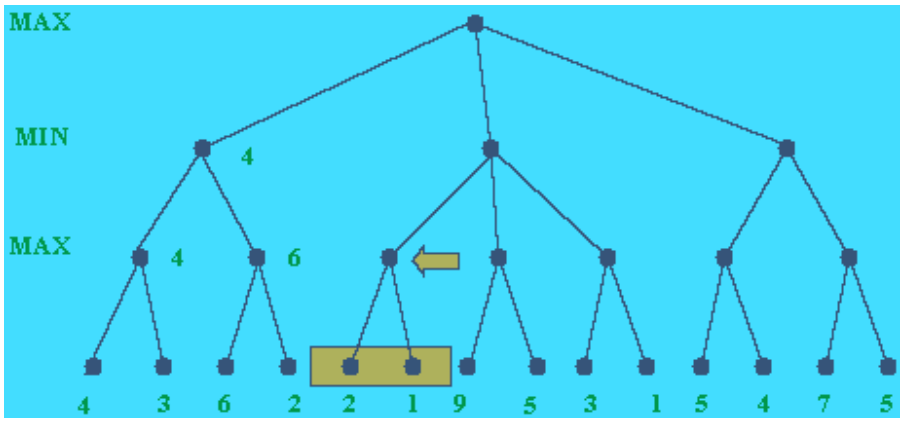
[۱] www.informatics.susx.ac.uk/books/computers-and-thought/gloss/node1.html

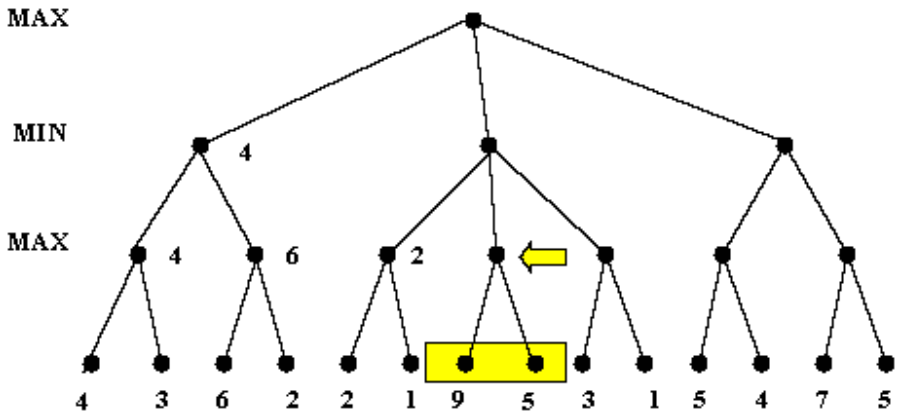
۱ Deterministic در کامپیوتر ، نتیجه ی فرایندی که به موقعیّت های اولیّه ورودی ها بستگی دارد

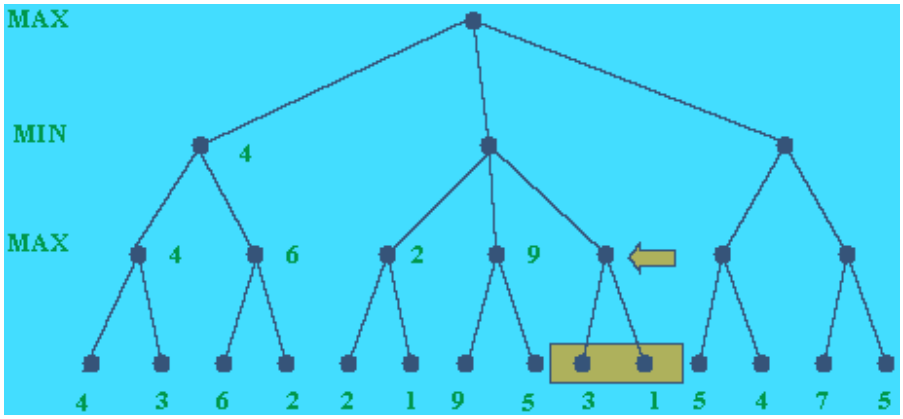

MAX
MIN
MAX
4 3 6 2 2 1 9 5 3 1 5 4 7 5

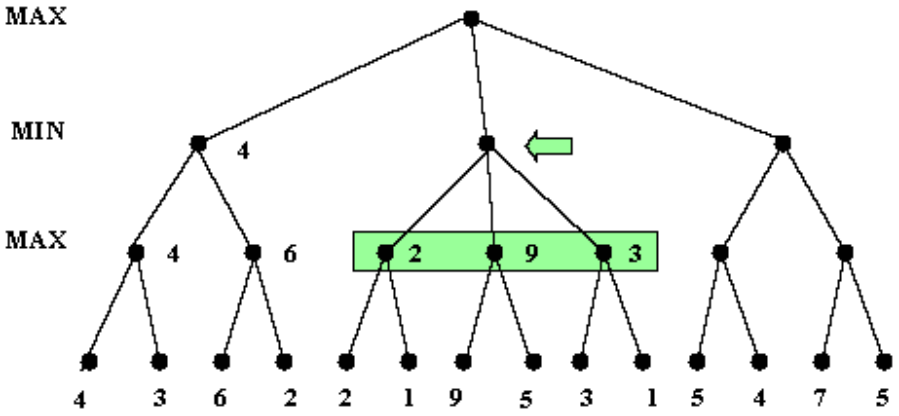

MAX
MIN
MAX
4
4 3 6 2 2 1 9 5 3 1 5 4 7 5


MAX
MIN
MAX
4 6
4 3 6 2 2 1 9 5 3 1 5 4 7 5

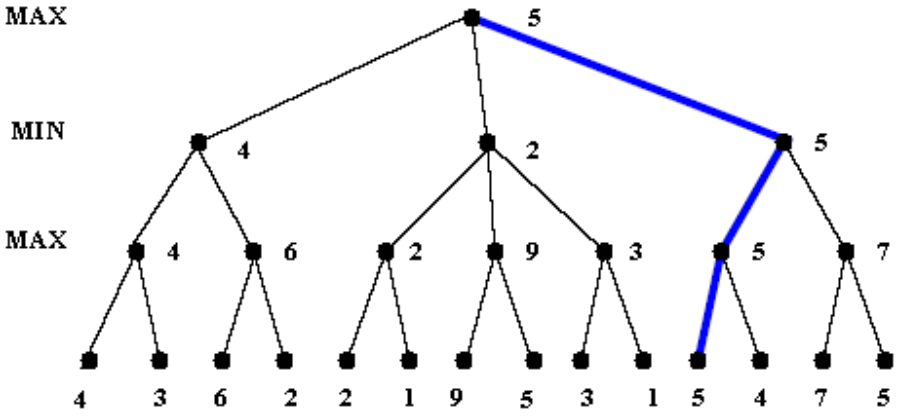

MAX
MIN
MAX
4
4
6
4 3 6 2 2 1 9 5 3 1 5 4 7 5


MAX
MIN
MAX
4
4
6
2
4 3 6 2 2 1 9 5 3 1 5 4 7 5


MAX
MIN
MAX
4
4
6
2
9
4 3 6 2 2 1 9 5 3 1 5 4 7 5

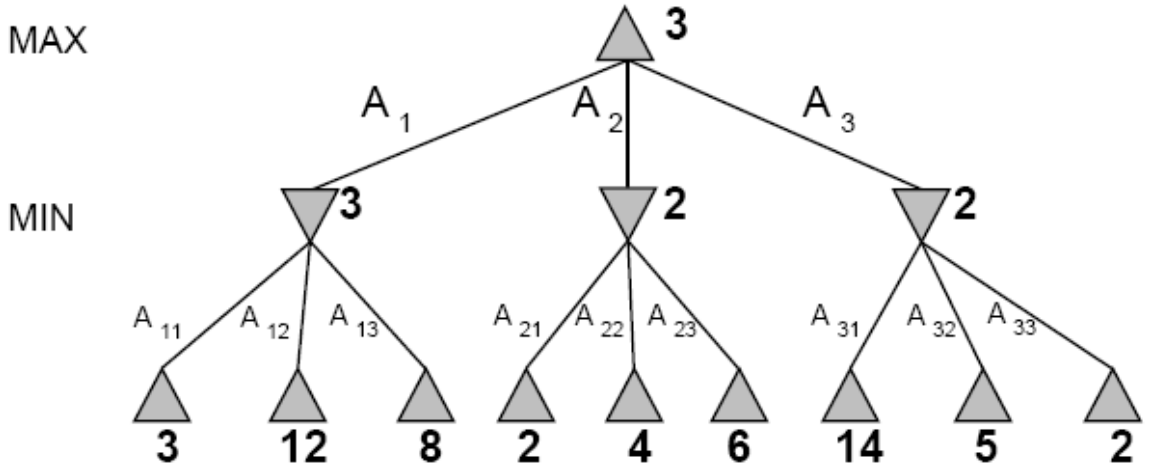


و به همین ترتیب موارد دیگر را هم به دست می آوریم تا در نهایت به شکل زیر برسیم :



پس جواب این مثال عدد ۵ است .

به عنوان مثال دیگر ، برای بازی دو نفره داریم :



که در نتیجه حاصل برابر با ۳ خواهد بود .

الگوریتم مینیماکس

تابع Minimax-Decision(state) یک عملکرد را برمی گرداند

ورودی : state ، که حالت جاری بازی می باشد

a ای را کـه در Actions(state) بیشـینه کننـده ی Min-Value(Result(a ,state)) مـی باشـد را برگردان

تابع Max-Value(state) یک مقدار مفید را یرمی گرداند

در صورتی که تابع Terminal-Test(state) دارای مقدار بازگشتی درست می باشـد Utility(state) را برگردان

$v \leftarrow -\infty$

برای a,s در تابع Successor(state) ، $v \leftarrow \text{Max}(v, \text{Min-Value}(s))$

مقدار v را برگردان

تابع Min-Value(state) یک مقدار مجاز را برمی گرداند

در صورتی که Terminal-Test(state) درست است ، Utility(state) را برگردان

$v \leftarrow \infty$

برای a,s درون Successors(state) ، $v \leftarrow Min(v,Max\text{-}Value(s))$

v را برگردان

خصوصیّات مینیماکس

کامل بودن ؟؟ فقط در صورتی که درخت محدود باشد ، کامل است (شطرنج قوانین معینی برای این کار دارد) .

بهینه بودن ؟؟ بله

پیچیدگی زمانی ؟؟ $O(b^m)$

پیچیدگی فضا ؟؟ O(bm) (مثل جستجوی اوّل عمق)

برای شطرنج ، $b \approx 35$ ، $m \approx 100$ (برای بازی های مستدل) ← راه حل دقیق کاملاً اجرا نشدنی است .

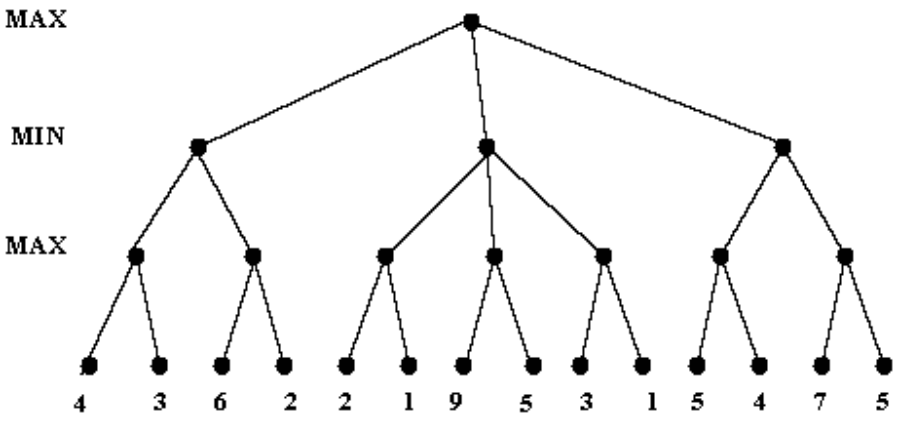
امّا آیا ما احتیاج داریم که هر مسیر را پیدا نماییم ؟

هرس α - β

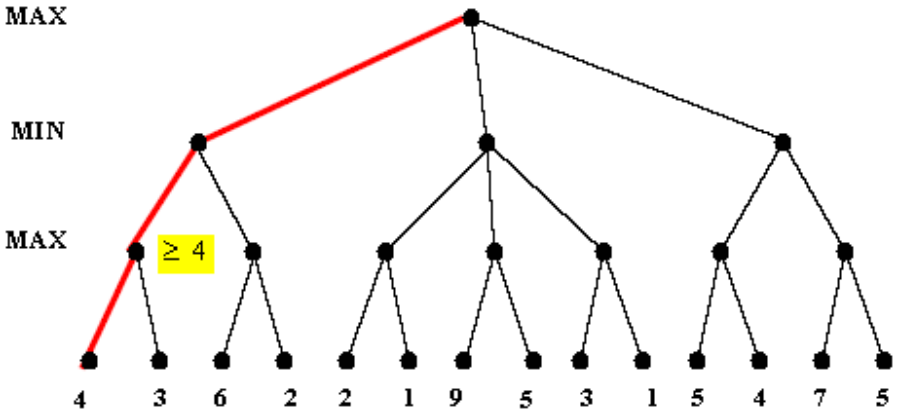
مترجم: سهراب جلوه گر
ویرایش دوّم ، بهار ۱۳۸۸

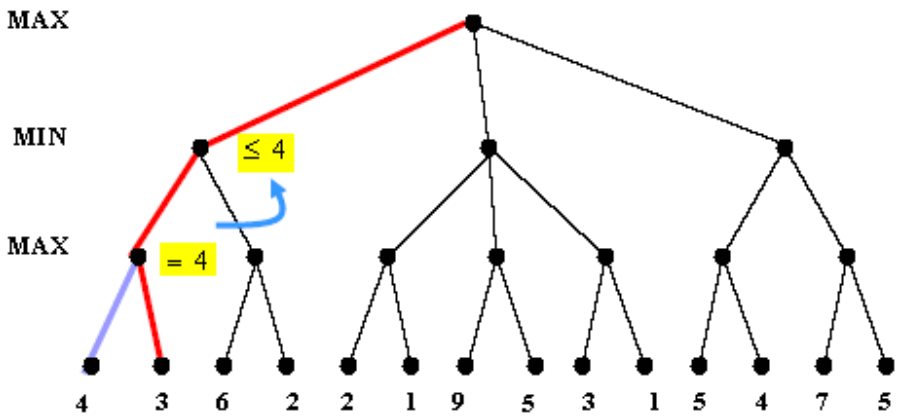
روشی استاندارد برای بازی های قطعی و با اطّلاعات کامل می باشد ؛ در این روش ، شاخه ای که هیچ وقت مورد استفاده قرار نمی گیرد را از درخت جستجو حذف می کنیم . توجّه کنید که این شاخه ها هیچ وقت توسط یک بازیکن عاقل مورد استفاده قرار نمی گیرند ، در ضمن ، توسط بازیکن حریف هم مورد استفاده قرار نمی گیرند .

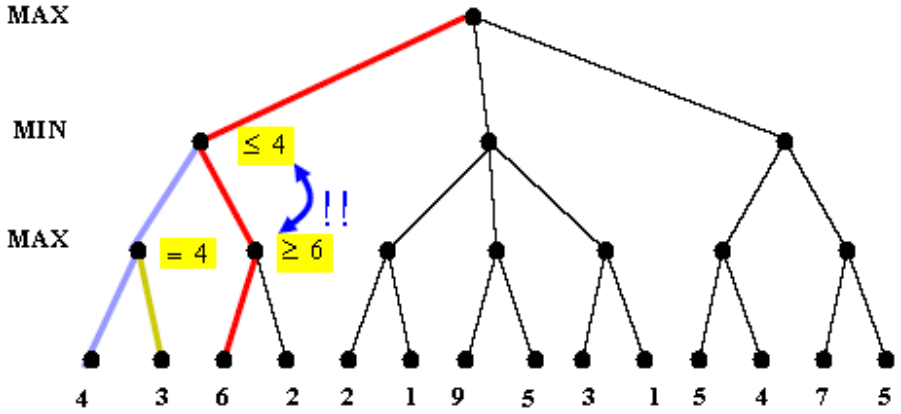
مثال هرس α - β : به درخت زیر توجّه کنید .

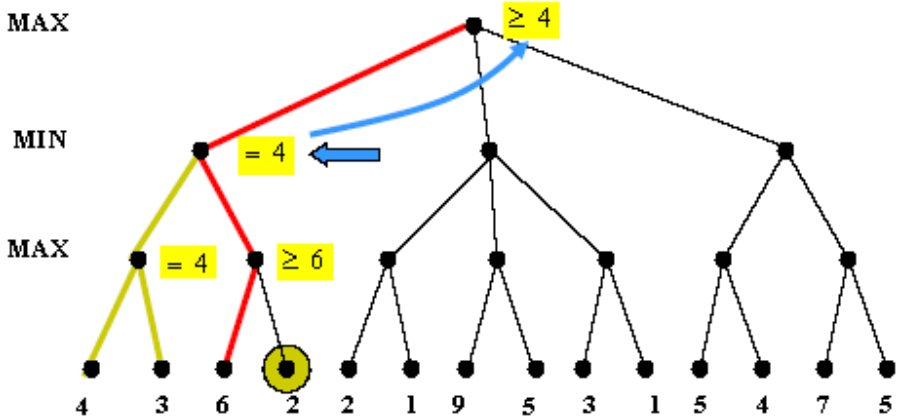


داریم (توجّه کنید که گره های به شکل ⊙ هیچوقت مورد استفاده قرار نمی گیرند) :

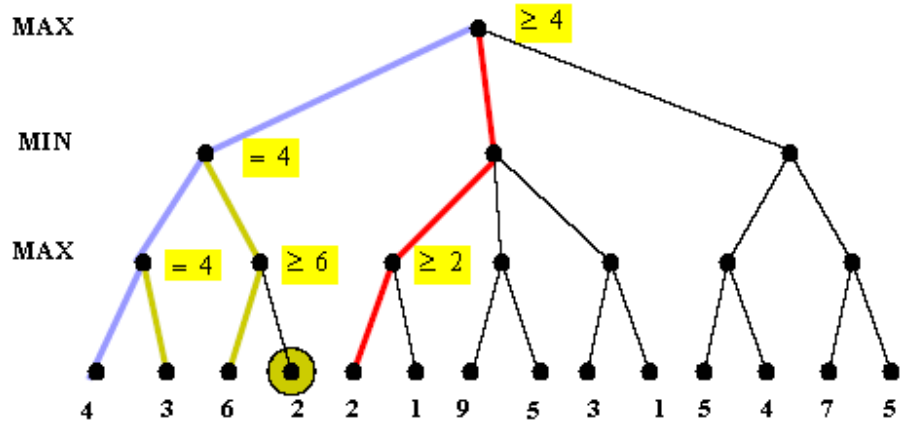


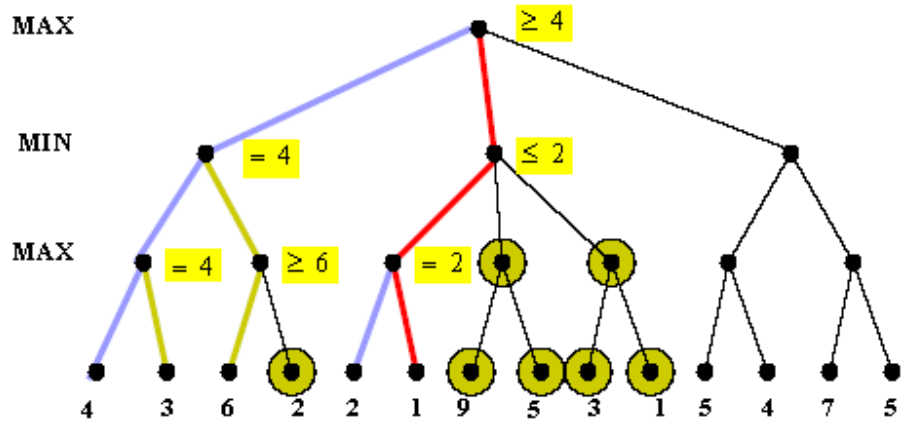

MAX
MIN
MAX
≤ 4
= 4
4 3 6 2 2 1 9 5 3 1 5 4 7 5

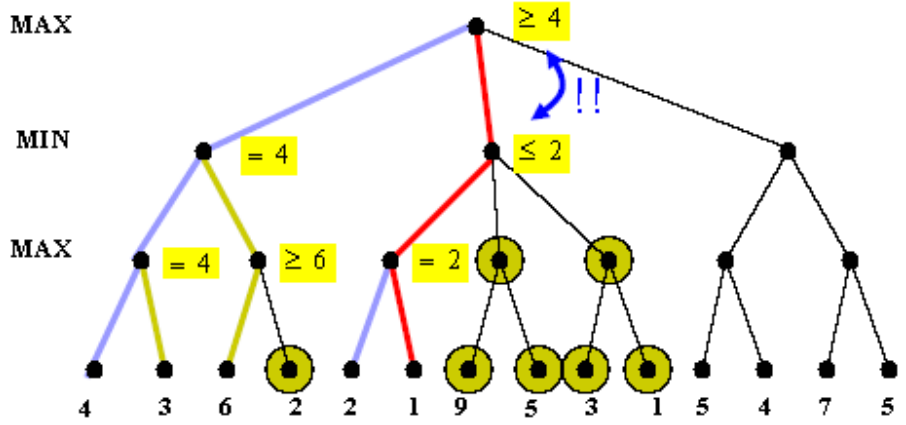

MAX
MIN
MAX
≤ 4
!!
= 4
≥ 6
4 3 6 2 2 1 9 5 3 1 5 4 7 5


MAX
MIN
MAX
≥ 4
= 4
= 4
≥ 6
4 3 6 2 2 1 9 5 3 1 5 4 7 5

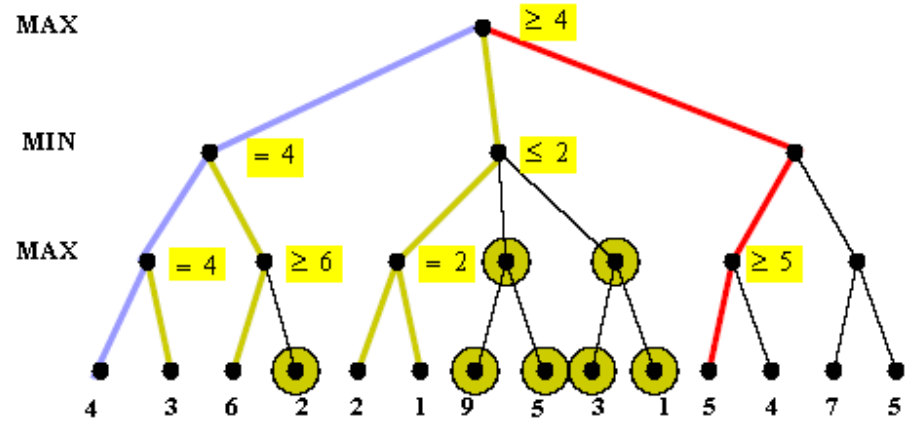
مترجم: سهراب جلوه گر
ویرایش دوّم ، بهار ۱۳۸۸

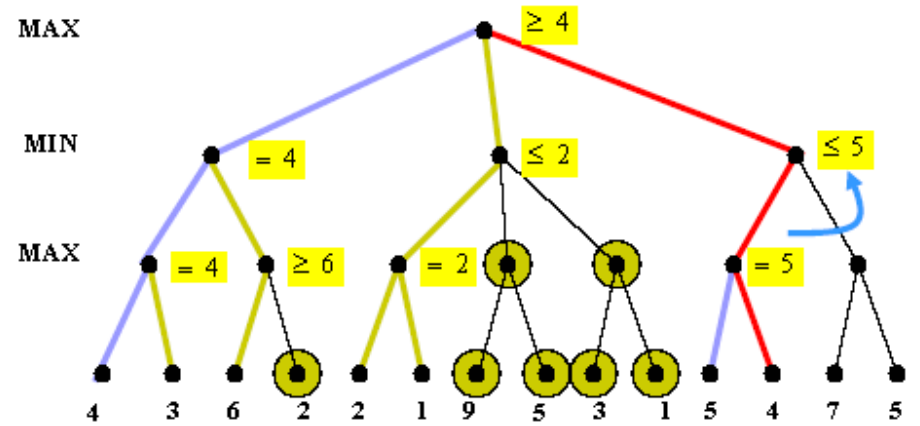


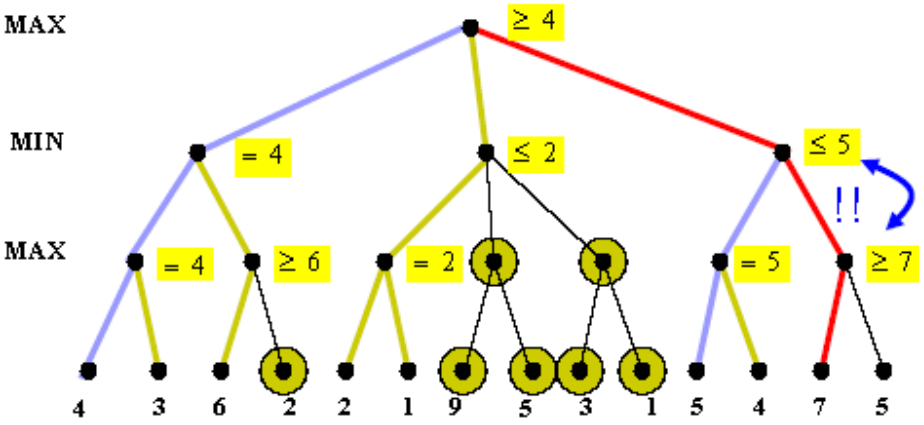




مترجم: سهراب جلوه گر
ویرایش دوّم ، بهار ۱۳۸۸







α مقدار بهترین برای MAX در طول مسیر به state می باشد .

β مقدار بهترین برای MIN در طول مسیر به state می باشد .

در صورتی که Terminal-Test(state) درست می باشد ، Utility(state) را برگردان

$V \leftarrow -\infty$

برای a,s در تابع Successors(state) کارهای زیر را انجام بده

$v \leftarrow Max(v,Min\text{-}Value(s,\alpha,\beta))$

در صورتی که $v \geq \beta$ باشد ، v را برگردان

$\alpha \leftarrow Max(\alpha,v)$

انتهای حلقه

v را برگردان

تابع Min-Value(state, α,β) یک مقدار مجاز را بر می گرداند

شبیه مقدار Max-Value ، امّا با قوانین رزرو شده ی α و β

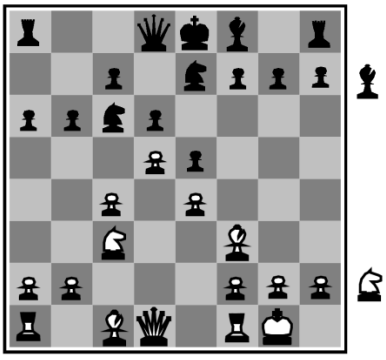
خصوصیّات α-β – بازبینی یا هرس بر روی نتیجه ی نهایی اثری ندارد . حرکت خوب و منظم اثر هرس را بهبود می بخشد . با " منظّم کننده ی کامل " ، پیچیدگی زمانی برابر با $O(b^{m/2})$ است ، درنتیجه عمق جستجو را مضاعف می کند . بنابراین ، به سادگی می تواند به عمق هشت برسد و بازی شطرنج خوبی را ارایه نماید . متأسّفانه 35^{50} هنوز غیر ممکن است !

مترجم: سهراب جلوه گر
ویرایش دوّم ، بهار ۱۳۸۸

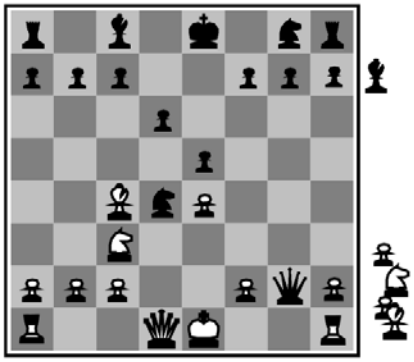
محدودیّت منابع – فرض کنید ما صد ثانیه زمان داریم و می توانیم 10^4 گره در ثانیه را ملاقات کنیم درنتیجه در صد ثانیه 10^6 گره را می توانیم ملاقات نماییم .

توابع ارزیابی – یک تابع ارزیابی ، تابعی است که خوبی یک وضعیّت را اندازه گیری می نماید (به عنوان مثال ، شانس برنده شدن در آن وضعیّت را اندازه می گیرد) و این تابع می تواند توسط طراح ارایه شود یا با آزمایش به دست آید . اگر ترکیبات صفحه بتواند به صورت مستقل ارزیابی شود آن گاه یک انتخاب خوب یک تابع توزیع شده ی خطی می باشد :

$w_1 f_1(s) + w_2 f_2(s) + ... + w_n f_n(s)$ ، که s وضعیّت صفحه ی بازی می باشد و f_i تعداد یک نوع مهره بر روی صفحه ی بازی می باشد (مثلا فیل) و w_i ارزش مهره ها (مثلا یک برای سرباز و سه برای فیل) . توجّه کنید که برنامه های پیشرفته از ترکیبات غیرخطّی هم استفاده می نمایند ، در ضمن ، ترکیبات و توزیع ها جزیی از قوانین شطرنج نمی باشند .



با حرکت سیاه ، وضعیّت سفید بهتر می شود



با حرکت سفید ، سیاه برنده می شود

مترجم: سهراب جلوه گر
ویرایش دوّم ، بهار ۱۳۸۸

بازی های قطعی در عمل [۱]

– بازی چکرز : چینوک [۲] به چهل سال قهرمانی قهرمان جهان ماریون تینزلی [۳] در سال ۱۹۹۴ خاتمه داد. از پایگاه داده ی مشخص کننده ی پایان بازی شطرنج برای تعریف تمام بازی و برای وضعیّت های شامل هشت یا کم تر مهره در صفحه استفاده شد ، یک مجموعه از ۲۴۷و۴۰۱و۷۴۸و۴۴۳ حالت . Deep Blue قهرمان شطرنج جهان ، گری کاسپاروف [۴] را در شش بازی در سال ۱۹۹۷ شکست داد. Deep Blue ، دویست میلیون حالت را در هر ثانیه جستجو می کرد و از یک ارزیابی خیلی ماهرانه استفاده می کرد و از متدهای فاش نشده برای توسعه دادن تعدادی از خطوط جستجو تا ۴۰ تا استفاده می نمود.

اتلو [۵] : قهرمانانی که خیلی خوب هستند نمی پذیرند که با کامپیوتر رقابت نمایند .

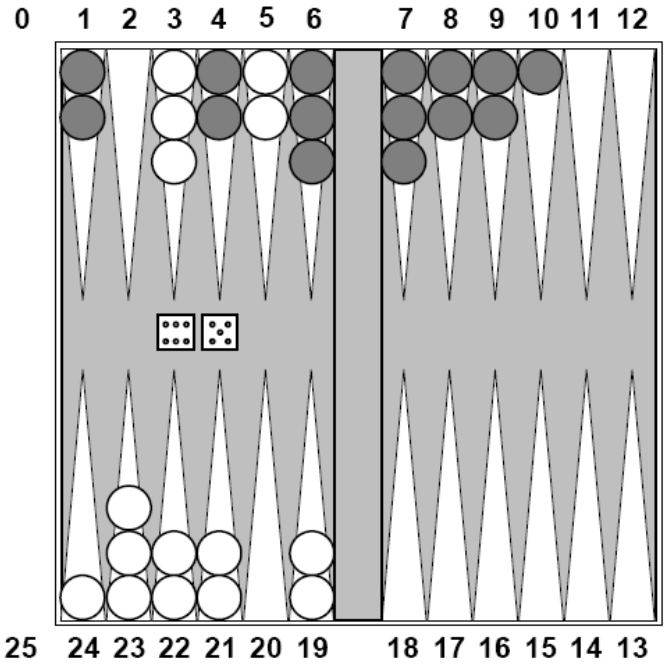
Go : قهرمانان نمی پذیرند که با کامپیوتر رقابت کنند . در این بازی ، $b>300$ ، بنابراین بیش تر برنامه ها از الگوی براساس دانش برای پیشنهاد حرکت های ممکن استفاده می نمایند .

بازی های غیرقطعی در حالت کلّی

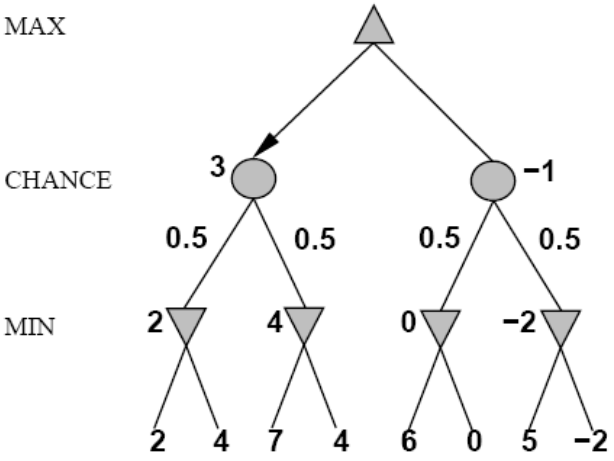
در بازی های غیر قطعی مثل تخته نرد که تصویری از صفحه ی این بازی را در شکل زیر مشاهده می کنید ، شانس به وسیله ی تاس [۶] و برهم زدن کارت [۱] معرّفی می شود.

---

[۱] deterministic games in practice
[۲] Chinook
[۳] Marion Tinsley
[۴] Gary Kasparov
[۵] Othello
[۶] dice



مثال ساده شده با سکه ی دورویه [۲] :



---

[۱] card-shuffling

[۲] coin-flipping

الگوریتمی برای بازی های غیرقطعی

از آمار و احتمالات داریم ؛ مقدار مورد انتظار ار یک متغیّر تصادفی x میانگینی از مقدارهای ممکن x می باشد و توسط احتمالات رخ داده ی $P(x)$ توزیع شده است :

$$E(x) = \sum_{x \in X} xP(x)$$

الگوریتم Expectminimax : شبیه Minimax می باشد ، تفاوت آن با Minimax در این است که از مقدارهای مورد انتظار در گره های تصادفی هم استفاده می نماید . EXpectminimax بازی کامل را ارایه می نماید ؛ مینیماکس هم بازی کامل را ارایه می کند در ضمن ما باید از گره های تصادفی هم استفاده نماییم :

...

در صورتی که حالت (state) یک گره ماکزیمم است ، بالاترین ExpectiMinimax-Value از Successors(state) را برگردان

در صورتی که حالت (state) یک گره مینیمم است ، کم ترین ExpectiMinimax-Value از Successors(state) را برگردان

در صورتی که حالت (state) یک گره شانسی است ، میانگین ExpectiMinimax-Value از Successors(state) را برگردان

...

توضیحی در مورد Deep Blue

مترجم: سهراب جلوه گر
ویرایش دوّم ، بهار ۱۳۸۸

Deep Blue ، یک مثال خوب از ترکیب توان محاسباتی مدرن و تکنیک ها (روش ها) ی هوش مصنوعی می باشد . در سال ۱۹۹۷ میلادی ، Deep Blue گری گاسپاروف (بهترین شطرنج باز بشری) را مغلوب نمود . و این کار ، یک هدف هوش مصنوعی از دهه ی ۵۰ میلادی بود . Deep Blue یک ابررایانه ی 32-Node RS/6000 می باشد . برنامه ی Deep Blue به زبان سی نوشته شده بود و از جستجوی آلفا – بتا استفاده می نمود و دارای سخت افزار مخصوص محاسبه کننده ی توابع ارزیابی بود . تعداد ارزیابی ها ، دویست میلیون حالت در ثانیه بود که ۱۰۰ – ۲۰۰ بیلیون وضعیّت را در سه دقیقه ، ارزیابی می کند . در ضمن میانگین فاکتور انشعاب در شطرنج ۳۵ می باشد و فضای حالت در آن حدود 10^{70} می باشد . مقدار زیادی از دانش با دامنه ی مخصوص افراد برای به وجود آوردن دقت در توابع ارزیابی به کار گرفته شد و از پایگاه داده های استادان بزرگ شطرنج در بازی های قبلی و همچنین از تعداد زیادی کتاب های پیشرفته استفاده شده بود .

مترجم: سهراب جلوه گر

ویرایش دوّم ، بهار ۱۳۸۸

فصل نهم

عامل های منطقی [۱]

[۱] Logical Agents

مترجم: سهراب جلوه گر
ویرایش دوّم ، بهار ۱۳۸۸

ریوس مطالب

- عامل ها براساس دانش [۱]
- دنیای Wumpus
- منطق در حالت کلّی – مدل ها و دارایی ها [۲]
- منطق گزاره ای [۳](بولین)
- تساوی [٤]، صحت [٥]، توانایی راضی کردن [٦]
- قوانین استنتاج [۷] و اثبات قضیه [۸]
  - زنجیره ی مستقیم [۹]
  - زنجیره ی معکوس [۱۰]
  - تحلیل [۱]

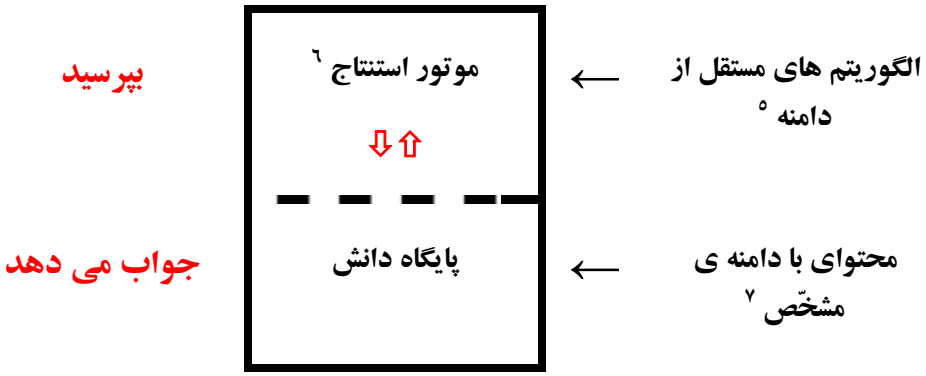
---

[۱] Knowledge-based agents
[۲] entailment
[۳] Propositional
[٤] Equivalence
[٥] validity
[٦] satisfiability
[۷] Inference rules
[۸] theorem proving
[۹] forward chaining
[۱۰] backward chaining

مترجم: سهراب جلوه گر
ویرایش دوّم ، بهار ۱۳۸۸

پایگاه های دانش

یک پایگاه دانش [۲] ، تشکیل شده از جملاتی که واقعیّاتی را در مورد جهان [خود] ارایه می کنند . به بیان دیگر ، پایگاه دانش ، مجموعه ای از عبارت ها است که به وسیله ی یک زبان رسمی [۳] بیان شده اند . حال این سؤال مطرح می شود که تفاوت میان پایگاه دانش و پایگاه داده [۴] چیست ؟ : به طور کلّی ، در طرز بیان و کاربرد آن هاست و در عمل ، یک پایگاه دانش ممکن است از یک پایگاه داده استفاده کند . جملات ، در مورد چیزهایی که در جهان وجود دارد مثل Wumpus ها ، طلا ، چاله ها ، فضاها و ارتباط میان عامل ها توضیح می دهند .



---

[۱] resolution

[۲] Knowledge Base

[۳] formal

[۴] Database

[۵] domain-independent algorithms

[۶] inference engine

[۷] domain-specific content

قانون استنتاج : وقتی کسی از پایگاه دانش می پرسد ، جواب باید از آن چه که قبلا به پایگاه دانش گفته ایم بیاید .

شیوه ی [۱] اعلانی یا اظهاری برای ساختن یک عامل (یا یک سیستم دیگر) :

به پایگاه دانش **بگویید** ( ) [۲] که می خواهید چه چیزی را بدانید . سپس از خودش **سؤال می کند** ( ) [۳] که چه کاری باید انجام دهد ؛ جواب هایی که می دهد باید از KB یا پایگاه دانش گرفته شوند . عامل ها می توانند در **سطح دانش** [٤] دیده شوند . توجه نمایید که **چه چیزی را آن ها می دانند** ، بدون این که بدانند چگونه به کار گرفته می شوند ، یا در **مرحله ی کاربرد** [٥] به ساختمان های داده ای [٦] در پایگاه دانش و الگوریتم هایی که آن ها را به کار می برند توجه می کنند .

یک عامل ساده ی براساس دانش

الگوریتم :

تابع KB-Agent(percept) یک عمل را برمی گرداند

متغیّر KB که یک متغیّر static است ، یک پایگاه دانش می باشد

t ، یک شمارنده با مقدار اوّلیه ی صفر و نشان دهنده ی زمان می باشد

---

[۱] approach
[۲] Tell ( )
[۳] Ask ( )
[٤] knowledge level
[٥] implementation level
[٦] data structures

Tell(KB,Make-Percept-Sentence(percept,t))

action ← Ask(KB,Make-Action-Query(t))

Tell(KB,Make-Action-Sentence(action,t))

t ← t+1

عمل (action) را برگردان

عامل باید قادر باشد : حالت ها ، عملکرد ها را ارایه نماید ؛ دریافت های ادراکی [۱] جدید را یکی کند [۲] ؛ ارایه های درونی جهان را به روز نماید ؛ خصوصیات پنهان جهان خود را استنباط [۳] نماید و عملکردهای مناسب را استنباط نماید .

توضیح : عامل پایگاه دانش ، شبیه عامل های با وضعیّت درونی می باشد . عامل ها می توانند در سطح های مختلف ، بیان شوند :

۱- **سطح دانش :** چه می دانند ، بدون توجه به پیاده سازی واقعی .
۲- **سطح پیاده سازی :** ساختمان های داده در پایگاه دانش و الگوریتم هایی که آن ها به کار می گیرند ؛ به عنوان مثال ، منطق گزاره ای و نتیجه [۴] .

انواع دانش

---

[۱] percepts
[۲] Incorporate
[۳] deduce
[۴] resolution

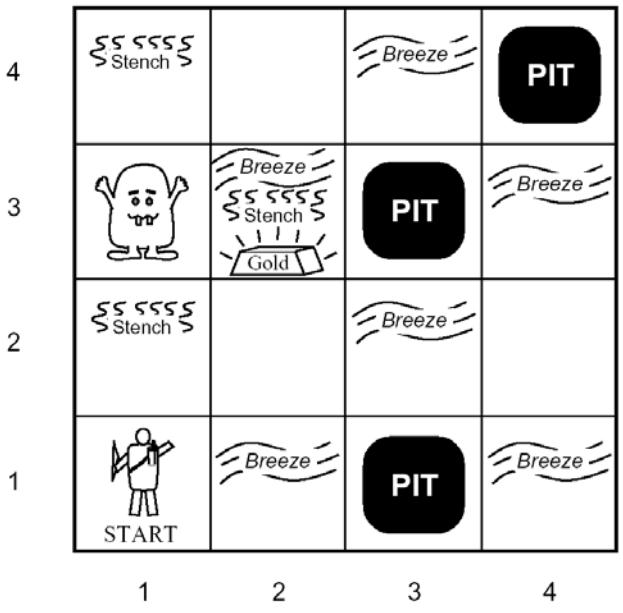
مترجم: سهراب جلوه گر
ویرایش دوّم ، بهار ۱۳۸۸

۱- **رویه ای :** به عنوان مثال ، تابع ها که مثل دانش می توانند فقط بـه یـک روش و آن هـم در زمان اجرا مورد استفاده قرار گیرند .

۲- **بیانی :** محدودیّت ها و قانون ها که می توانند برای انجام انواع زیادی از اسـتنباط هـا مـورد استفاده قرار گیرند .

دنیای Wumpus

Wumpus ، نام یک بازی کامپیوتری است ؛ در ایـن بـازی ، عامـل یـک غـار را کـه از اتـاق هـای متصل شده توسط راهرو تشکیل شده کشف می کند . *کمین گـاه ، جـایی اسـت کـه در آن Wumpus قـرار دارد و Wumpus ، جانوری است که هر عاملی را که به اتاقش وارد شـود مـی خـورد . همچنـین ، برخـی از اتاق ها دارای چاله های بدون کف هستند که هر عاملی را که در اتاق سرگردان است را بـه تلـه مـی اندازنـد . گاهی از وقت ها کپه ای از طلا هم در اتاق هسـت .* ***هدف ما در بازی این است که طلا هـا را جمـع کنیم و بدون اینکه خورده شویم خارج شویم .***



مترجم: سهراب جلوه گر
ویرایش دوّم ، بهار ۱۳۸۸

دنیای Wumpus معمولی

عامل ، معمولاً از خانه ی [۱و۱] شروع می کند . کار عامل این است که طلا را پیدا کند ، بـه خانـه ی [۱و۱] برگردد و از غار خارج شود .

خصوصیات دنیای wumpus

قابل مشاهده بودن ؟؟ نه – فقط دارای ادراک محلی است .

قطعیّت ؟؟ بله – نتایج دقیقا مشخص است .

دوره ای نه – دارای ترتیب در سطح عملکردها است .

ایستایی ؟؟ بله – wumpus و چاله ها نمی توانند حرکت نمایند .

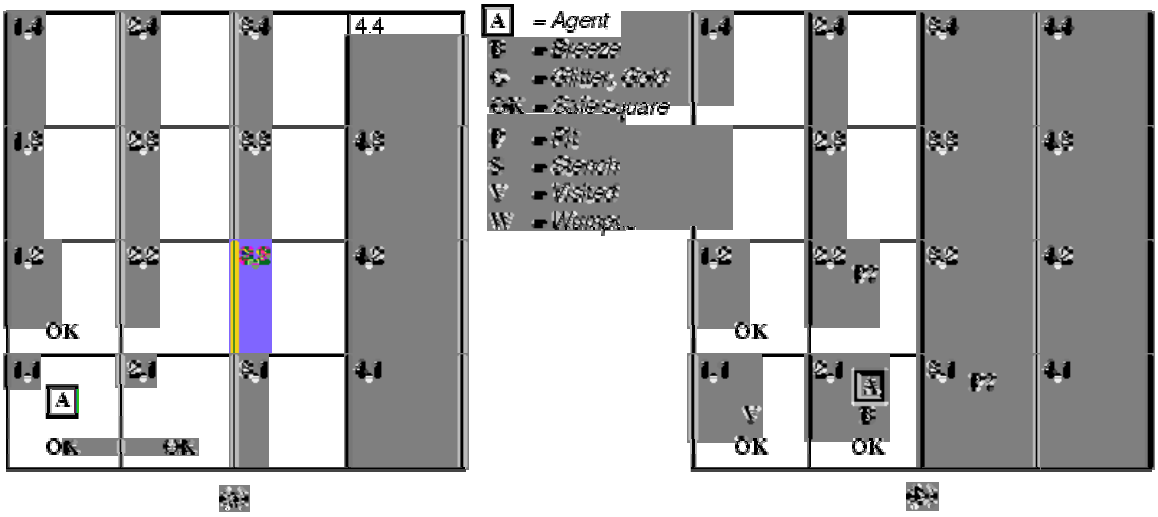
گسسته؟؟ بله .

تک عاملی ؟؟ بله wumpus در اصل یک طرح طبیعی می باشد .

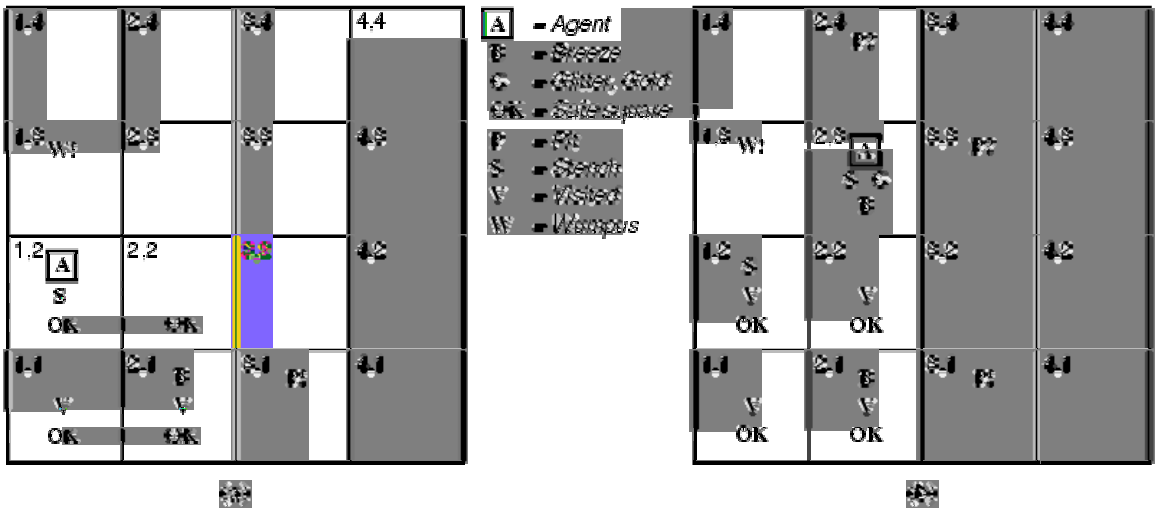
حرکت عامل در دنیای Wumpus

---

[۱] Left turn
[۲] Right turn
[۳] Forward
[۴] Grab
[۵] Release
[۶] Shoot



در خانه ی [۱و۱] ، پایگاه دانش دارای قانون های محیط است . اوّلین دریافت یا ادراک [هیچ و هیچ و هیچ و هیچ]است ، به یک خانه ی امن [1] مثل [۱و۲] می رویم . خانه ی [۱و۲] چون خوش بو است نتیجه می گیریم که چاله در [۲و۲] یا [۱و۳] است ؛ در نتیجه به خانه ی [۱و۱] برمی گردیم و خانه ی امن بعدی را امتحان می نماییم .



[1] safe square