محیط S=20

در t = 4 :

1. 01## : 0000 S=220 0001
2. 00#0 : 1100 S=208
3. 11## : 1000 S=196
4. ##00 : 0001 S=156

محیط S=20

و در t = 5 – درخواست به سیستم می آید و به آخرین طبقه بندی کننده ی فعّال نسبت داده می شود .

1) 01## : 0000 S=220
2) 00#0 : 1100 S=208
3) 11## : 1000 S=196
4) ##00 : 0001 S=206

محیط S=20

تولید قوانین جدید – دسته سطل ، یک روش انتخاب قوانین و انتساب اعتبار را ارایه می نماید . چگونه قوانین جدید را به دست بیاوریم ؟ الگوریتم ژنتیکی پایه ی ما از یک مدل جمعیّت که با هم اشتراک ندارند استفاده می نماید ؛ همه ی جمعیّت در زمان t ، در زمان t+1 جایگزین می شود . به خوبی برای بهینه سازی عمل می کند ، ولی برای یادگیری ، زیاد مناسب نمی باشد . در عوض ، ما از نخبه سالاری [1] برای حفظ

[1] elitism

برخی از قانون ها استفاده می نماییم ، از روش انتخاب رولت [۱] برای حفظ قانون ها استفاده نمایید ، البتّه این روش ، کندتر استنتاج می کند .

برنامه نویسی ژنتیکی – فراتر از قوانین – روش های تکاملی برای تولید برنامه ها به کار گرفته شده اند . ایده ی این روش این است که یک برنامه می تواند به صورت یک S- عبارتی بیان شود ، مثلاً :

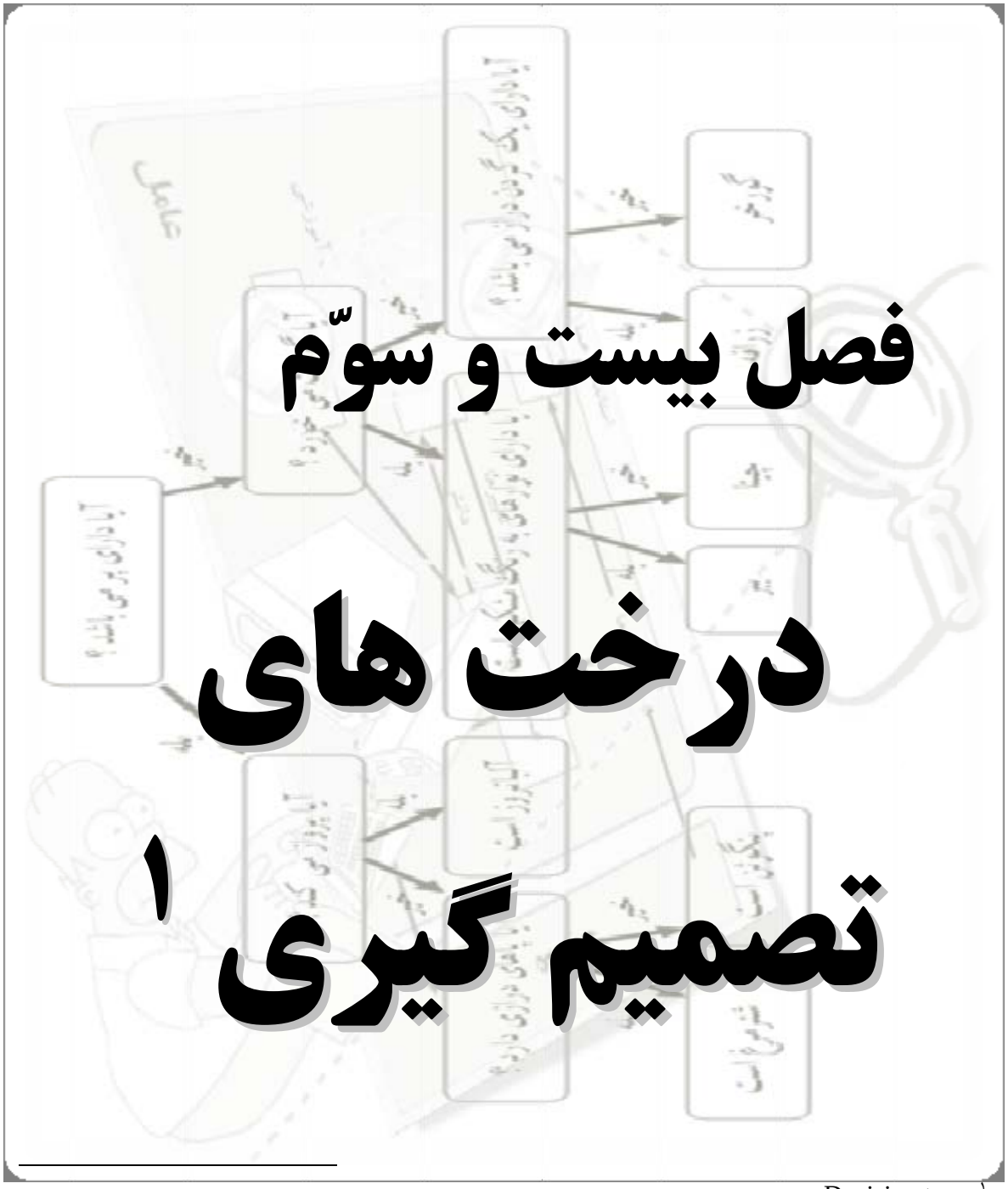
(+ 3 (* 4 x))

ما می توانیم این را به صورت یک درخت بکشیم . ما سپس تکامل را روی یک جمعیّت از برنامه ها انجام می دهیم . شایستگی ، کارایی یک برنامه برای یک عملکرد ارایه شده است . Crossover ، زیر درخت ها را با هم عوض می نماید . جهش ، عملگرها را عوض می نماید و برای تکامل استفاده می شوند ، مثال ها برنامه های فوتبال روبوت ، مدارهای کنترل آنالوگ ، فیلترها و مدارهای پیچیده می باشند . در این مورد چالش ها عبارتند از ، فضای پهناور (نامحدود) برنامه ها و کمبود ساختارهای سطح بالاتر برنامه .

[۱] roulette selection

مترجم: سهراب جلوه گر

ویرایش دوّم ، بهار ۱۳۸۸



فصل بیست و سوّم

درخت های تصمیم گیری [۱]

[۱] Decision trees

در گذشته ، ما فرض کردیم که دانش اوّلیّه توسّط خبره ها به ما داده شده است و بر چگونگی استفاده از این دانش تمرکز کردیم . حال می خواهیم در این مورد صحبت کنیم که چگونه دانش را از راه مشاهده به دست آوریم و بر قانون های گزاره ای تمرکز می کنیم . مثلاً اگر هوا آفتابی و گرم باشد ، آن گاه تنیس بازی کنید و می نویسیم :

$$\text{sunny} \wedge \text{warm} \rightarrow \text{PlayTennis}$$

یا اگر هوا خنک باشد و بارانی باشد یا باد شدید بوزد ، آن گاه تنیس بازی نکنید و می نویسیم :

$$\text{cool} \wedge (\text{rain} \vee \text{strongWind}) \rightarrow \neg\, \text{PlayTennis}$$

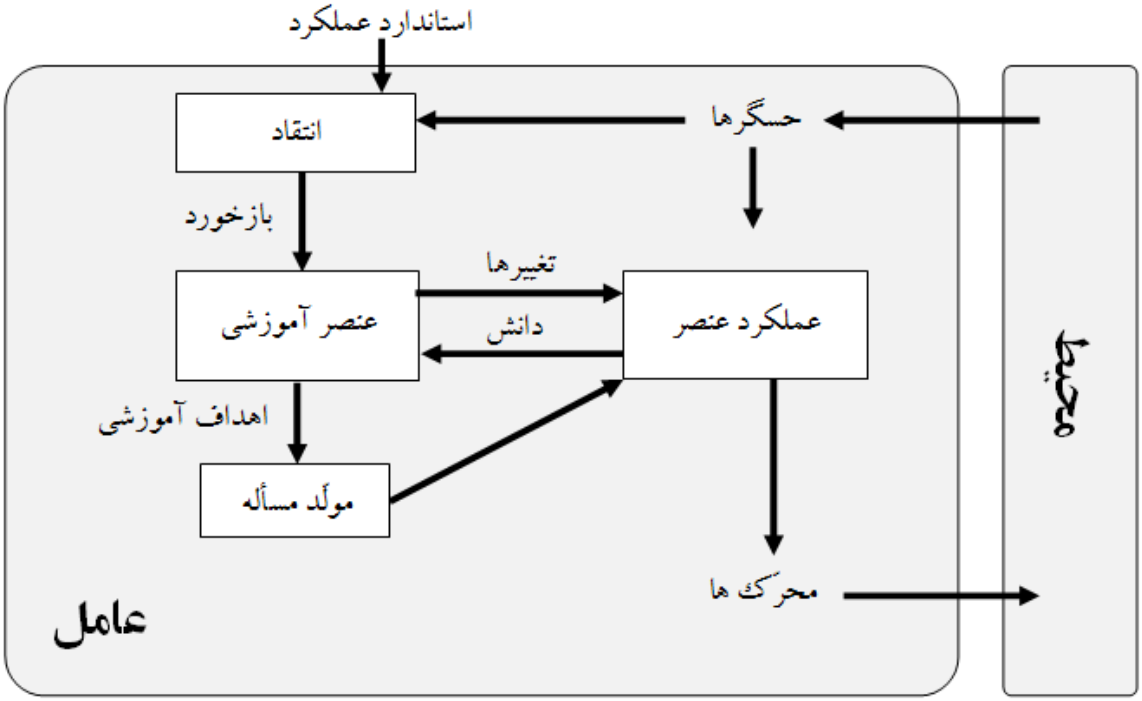
یادگیری

برای یک عامل ، یادگیری به چه معنی می باشد ؟ عامل ، دانش جدید را دریافت می نماید ، دانش جدید را به کار می گیرد ، رفتارش را تغییر می دهد و در یک کار معیّن ، معیار کارایی خود را بهبود می بخشد .

عامل های آموزشی

مترجم: سهراب جلوه گر
ویرایش دوّم ، بهار ۱۳۸۸

به یاد بیاورید که ما قبلاً در مورد عامل های آموزشی صحبت کردیم :



یک عامل آموزشی دارای یک *عنصر کارایی* [۱] و یک *عنصر آموزشی* [۲] می باشد . عنصر کارایی ، چیزی است که یک عامل برای تصمیم گیری در مورد این که چه کاری انجام دهد از آن استفاده می کند و این چیزی است که ما تا کنون مطالعه کرده ایم . عنصر آموزشی ، چیزی است که به عامل برای بازنگری عنصر کارایی اجازه می دهد ، این ممکن است به معنی اضافه نمودن یا تغییر قانون ها یا واقعیّات ، بازنگری

---

[۱] performance element

[۲] learning element

مترجم: سهراب جلوه گر
ویرایش دوّم ، بهار ۱۳۸۸

یک مکاشفه و تغییر یک تابع جانشین [۱] باشد . یک عامل ، برای بازنگری کردن رفتارش ، به اطّلاعاتی که به عامل بگوید چگونه به خوبی کارش را انجام می دهد نیازمند می باشد ، این اطّلاعات ، بازخورد [۲] نام دارد .

انواع بازخورد

در اصل سه نوع عملکرد آموزشی وجود دارد که هر کدام بازخورد متفاوتی را دارد :

یادگیری نظارت شده [۳] (کنترل شده) که در این مورد ، یک منبع خارجی (که اغلب آموزش دهنده [۴] نام دارد) ، عامل را با *نمونه های برچسب زده شده* [۵] ارایه می نماید . عامل ، موارد / عملکردهای معینی را در طول طبقه بندی اشان می بیند .

یادگیری بدون نظارت [۶] (کنترل نشده) که در این مورد ، آموزش دهنده ای برای ارائه ی نمونه ها وجود ندارد و عامل ، معمولاً برای پیداکردن الگوهایی در داده ها تلاش می نماید .

یادگیری تقویتی [۷] که این ، یک نوع مخصوص از آموزش است که در آن عامل فقط یک پاداش را برای انجام یک عمل دریافت می نماید . ممکن است نداند یک پاداش بهینه چگونه می باشد و "بهترین" عملکرد را برای انجام نمی داند .

یادگیری نظارت شده (کنترل شده)

---

[۱] Successor function
[۲] feedback
[۳] Supervised learning
[۴] teacher
[۵] labled examples
[۶] unsupervised learning
[۷] reinforcement learning

مترجم: سهراب جلوه گر
ویرایش دوّم ، بهار ۱۳۸۸

یکی از عمومی ترین شکل های آموزش می باشد . عامل با مجموعه ای از داده های برچسب زده شده ارایه می شود که باید از این داده ها برای تشخیص قانون های کلّی تر استفاده نماید . نمونه ها ، لیست بیماران و ویژگی ها ، چه عامل هایی مرتبط با سرطان می باشند ؟ چه عواملی ، فردی را دارای خطر می داند ؟ بهترین سؤالات برای طبقه بندی حیوانات چیست ؟ صورت چه کسی در این تصویر می باشد ؟ این پردازش یادگیری قانون های کلی از واقعیّات مشخص ، *استنتاج* [۱] نام دارد .

تعریف یک مسأله ی یادگیری – ما می توانیم مسأله ی یادگیری را با تخمین ، به صورت یک تابع f که به ما می گوید چگونه یک مجموعه از ورودی ها را طبقه بندی نماییم تفسیر کنیم . یک مثال در این مورد یک مجموعه از ورودی های x و f(x) متناظر می باشد :

<<Mammal,Eats-Meat,Black-Stripes,Tawny>,Tiger>

ما می توانیم یک عملکرد آموزشی را به این صورت ، تعریف نماییم : یک مجموعه ی ارایه شده از نمونه های f ، یک تابع H که f را برای مثال ما تخمین می زند پیدا نمایید ، H ، *فرض* [۲] نام دارد .

استنتاج – ما ترجیح می دهیم H را تعمیم بدهیم ، این به این معنی است که H به درستی نمونه های دیده نشده را طبقه بندی می کند . در صورتی که فرض به درستی بتواند همه ی نمونه های آموزشی را طبقه بندی نماید ، ما آن را *فرض سازگار* [۳] می نامیم . هدف ، پیداکردن یک فرض سازگار که در نمونه های دیده نشده هم به خوبی کار کند . ما می توانیم یادگیری را به صورت جستجو در میان یک فضا از فرض ها تصوّر نماییم .

---

۱ induction

۲ hypothesis

۳ consistent hypothesis

مترجم: سهراب جلوه گر
ویرایش دوّم ، بهار ۱۳۸۸

بایاس استنتاجی – توجّه نمایید که استنتاج ، بی عیب نمی باشد . در انتخاب یک فرض ، ما یک گمان آموزش داده شده را به وجود می آوریم . *روشی که ما با استفاده از آن ، این گمان را به وجود می آوریم بایاس نام دارد* . نمونه ها ، اصل اکام [۱] ، معیّن ترین فرض ، کلّی ترین فرض و تابع خطی می باشند .

مشاهده ی داده ها – عامل ها ، ممکن است دارای ابزار مختلف مشاهده کننده ی نمونه های یک فرض باشند . یک الگوریتم یادگیری *دسته ای* [۲] ، یک مجموعه ی بزرگ داده ها که همه با هم ارایه می شوند و یک فرضیّه ی منفرد را انتخاب می کنند ، می باشد . یک الگوریتم یادگیری *افزایشی* ، نمونه های همه در یک زمان را دریافت می کند و دایماً فرض آن را بازنگری می نماید ؛ دسته معمولاً با دقّت تر می باشد ، اما افزایشی ممکن است با محیط عامل ، مناسب تر باشد . یک عامل یادگیری فعّال می تواند نمونه ها را انتخاب نماید . یک عامل یادگیری *غیرفعّال* [۳] ، دارای نمونه های ارایه شده با آن به وسیله ی یک منبع خارجی هستند می باشد ؛ یادگیری فعال توانمندتر می باشد ، اما ممکن است با محدودیّت های دامنه ، مناسب نباشد .

درخت های تصمیم گیری یادگیری – درخت های تصمیم گیری ، ساختمان داده هایی هستند که یک عامل را با ابزارهای طبقه بندی کننده ی نمونه ها ارایه می نمایند . در هر گره در درخت ، یک ویژگی ، تست می شود .

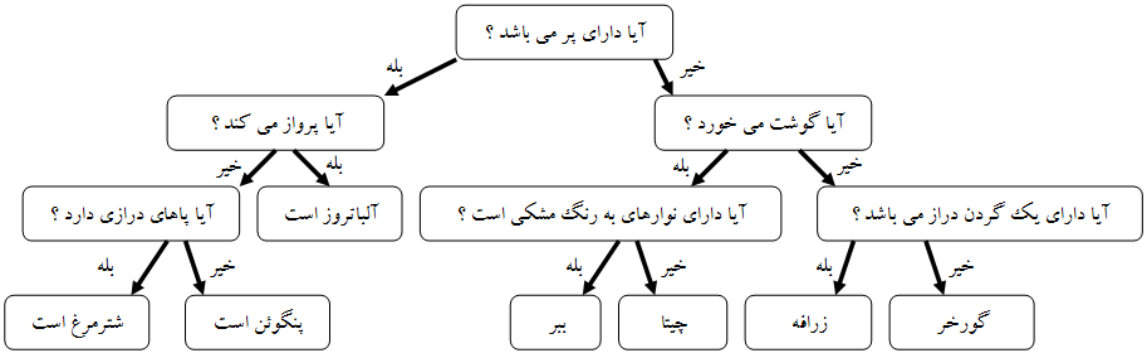
---

[۱] Occam's razor ، برای دو توصیف ارایه شده ، آسان ترین آن ها را انتخاب نمایید – توضیحی که به مفروضات کم تری نیاز دارد (لغت نامه ی وب برنامه ی babylon)

[۲] batch

[۳] passive

مترجم: سهراب جلوه گر
ویرایش دوّم ، بهار ۱۳۸۸



اطّلاعات

در مورد دنیای جاروبرقی ، اتاق ها ممکن است تمیز یا کثیف باشند که برای بیان این مورد به یک بیت نیاز داریم . اگر یک اتاق دارای چهار وضعیّت باشد یا هشت وضعیّت چه طور ؟ اگر یک اتاق فقط دارای یک وضعیّت باشد چه طور ؟ به چند بیت برای بیان نیاز داریم ؟

تئوری اطّلاعات

به صورت رسمی تر ، بیایید بگوییم که n پاسخ ممکن $v_1, v_2, ..., v_n$ برای یک سؤال وجود دارد و هر جواب دارای احتمال رویداد $P(v_n)$ می باشد . *محتوای اطّلاعات* [1] جواب برای سؤال به این صورت می باشد :

$$I = \sum_{i=1}^{n} - P(v_i) \log_2 P(v_i)$$

برای یک سکّه ، این برابر است با :

$$-\frac{1}{2}\log_2\frac{1}{2} + -\frac{1}{2}\log_2\frac{1}{2} = 1$$

[1] information content

مترجم: سهراب جلوه گر
ویرایش دوّم ، بهار ۱۳۸۸

سؤالاتی با یک احتمال زیاد جواب ، دارای محتوای اطّلاعاتی کمی خواهند بود . (در صورتی که سکّه در یک بار ۹۹/۱۰۰ شیر بیاید ، I=0.08 خواهد بود .) ، محتوای اطّلاعاتی ، برخی اوقات بی نظمی (آنتروپی [۱]) نامیده می شود و اغلب در فشرده سازی و الگوریتم های انتقال داده ها استفاده می شود .

استفاده از تئوری اطّلاعات

برای درخت های تصمیم گیری ، ما می خواهیم بدانیم ارزشمندی هر مورد ممکن چگونه می باشد ، یا چه میزان از اطّلاعات را نتیجه می دهد . ما می خواهیم احتمال جواب های ممکن از یک مجموعه ی مورد آموزش را تخمین بزنیم . معمولاً ، یک مورد تکی برای به طور کامل مجزا نمودن نمونه های مثبت و منفی کافی نمی باشد . در عوض ، ما نیاز داریم در مورد این فکر کنیم که چه مقدار بهتری بعد از سؤال وجود خواهد داشت و این ، *سود اطّلاعات* [۲] نام دارد .

سود اطّلاعات

در صورتی که یک مجموعه ی مورد آزمایش دارای p نمونه ی مثبت و n نمونه ی منفی باشد ، بی نظمی یا آنتروپی به صورت زیر خواهد بود :

$$I(\frac{p}{p+n},\frac{n}{p+n}) = -\frac{p}{p+n}\log_2\frac{p}{p+n} - \frac{n}{p+n}\log_2\frac{n}{p+n}$$

ما می خواهیم خصوصیّتی که برای مجزّا کردن نمونه های مثبت ومنفی ، نزدیک ترین می باشد را پیدا نماییم . ما با محاسبه ی باقی مانده شروع می کنیم – این اطّلاعاتی است که هنوز در داده ها بعد از این که ما ویژگی A را تست نمودیم وجود دارد . می گوییم ویژگی A می تواند مقادیر ممکن v را به کار بگیرد . این تست ،

---

[۱] entropy

[۲] information gain

مترجم: سهراب جلوه گر
ویرایش دوّم ، بهار ۱۳۸۸

v زیر مجموعه ی جدید از داده ها را به نام $E_1, E_2, ..., E_v$ خواهد ساخت . باقی مانده ، مجموع اطّلاعات موجود در هر کدام از این زیرمجموعه ها می باشد .

$$\text{Re } mainder \quad (A) = \sum_{i=1}^{v} \frac{p_i + n_i}{p + n} * I\left(\frac{p_i}{p_i + n_i}, \frac{n_i}{p_i + n_i}\right)$$

سود اطّلاعات سپس می تواند به صورت تفاوت میان اطّلاعات اصلی (قبل از آزمایش) و اطّلاعات جدید (بعد از آزمایش) بیان شود .

$$Gain(A) = I\left(\frac{p}{p+n}, \frac{n}{p+n}\right) - \text{Re } mainder\ (A)$$

مکاشفه : ویژگی با بیش ترین سود اطّلاعات را انتخاب نمایید .

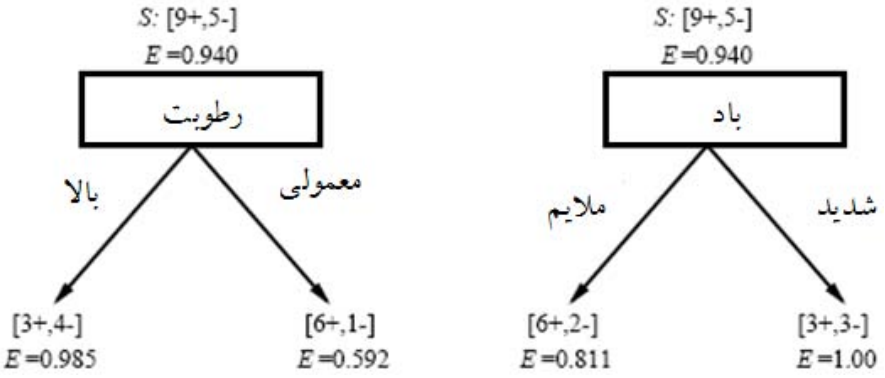
سؤال : این کدام نوع از جستجو می باشد ؟

مثال :

روز	پیش بینی	دما	رطوبت	باد	تنیس بازی
D1	آفتابی	گرم	بالا	ملایم	نه
D2	آفتابی	گرم	بالا	شدید	نه
D3	ابری	گرم	بالا	ملایم	بله
D4	بارانی	متوسط	بالا	ملایم	بله
D5	بارانی	خنک	معمولی	ملایم	بله
D6	بارانی	خنک	معمولی	شدید	نه
D7	ابری	خنک	معمولی	شدید	بله
D8	آفتابی	متوسط	بالا	ملایم	نه

بله	ملایم	معمولی	خنک	آفتابی	D9
بله	ملایم	معمولی	متوسط	بارانی	D10
بله	شدید	معمولی	متوسط	آفتابی	D11
بله	شدید	بالا	متوسط	ابری	D12
بله	ملایم	معمولی	گرم	ابری	D13
نه	شدید	بالا	متوسط	بارانی	D14

کدام ویژگی ، بهترین طبقه بندی می باشد ؟



Grain(S , رطوبت) = 0.940 - (7/14).985-(7/14).592 = .151

Gain(S , باد) = .940-(8/14).811-(6/14)1.0=.048

اغتشاش یا پارازیت [1]

[1] Noise

در صورتی که دو نمونه ی دارای ویژگی های یکسان ولی مقدارهای مختلف وجود داشته باشد ، یک درخت تصمیم گیری قادر به طبقه بندی آن ها به صورت مجزّا نخواهد بود و در این صورت می گوییم که این داده ها دارای اغتشاش می باشند .

ورودی های پیوسته ی مقداردهی شده – درخت های تصمیم گیری همچنین می توانند برای کار با ورودی های صحیح و مقداردهی شده به صورت پیوسته توسعه داده شوند . در این مورد برای پیداکردن مقداری که بالاترین سود اطّلاعات را نتیجه می دهد از جستجوی تپّه نوردی استفاده نمایید .

مترجم: سهراب جلوه گر

ویرایش دوّم ، بهار ۱۳۸۸

فصل بیست و چهارم

یادگیری Q ای [۱]

[۱] Q - learning

ما به محیط های تصادفی جذب شده ایم . ما می توانیم این مسأله را به صورت یک پروسه ی تصمیم گیری مارکوفی فرمول بندی نماییم ؛ مسأله دارای یک حالت اولیّه ی s_0، یک مجموعه ی گسسته از وضعیّت ها و عملکردها ، یک مدل انتقالی $T(s,a,s')$ که احتمال رسیدن وضعیّت s' از s در موقع انجام عمل a می باشد و یک تابع پاداش ($R(s)$) می باشد .

روش ها [۱]

یک روش ، راه حلّی برای یک پردازه ی تصمیم گیری مارکوفی می باشد و عملکرد بهینه را برای انجام در هر وضعیّت ، مشخّص می نماید . عملکرد بهینه ، عملکردی است که سودمندی مورد انتظار کاهش داده شده از این وضعیّت را بیشینه می نماید . بنابراین ، سودمندی یک وضعیّت ، پاداشی برای آن وضعیّت ، به اضافه ی پاداش کم شده ی مورد انتظار برای دنبال کردن روش بهینه از این وضعیّت می باشد . داریم :

$$U(s) = R(s) + \gamma \max_a \sum_{s'} T(s,a,s')U(s')$$

[۱] policies

که عبارت فوق معادله ی بلمن [۱] نام دارد .

تکرار مقدار [۲]

به صورت مستقیم نمی توان معادله ی بلمن را حل نمود . ما می توانیم از تکرار مقدار برای حل یک سیستم معادلات بلمن استفاده نماییم : به سودمندی های غیرپایانی ، مقدارهای تصادفی نسبت دهید . برای هر وضعیّت ، عملکرد بهینه ی ارایه کننده ی این مقادیر را محاسبه نمایید . از این روش برای به روز رسانی سودمندی تخمین زده شده ی وضعیّت استفاده نمایید و این کار را تا زمان نزدیک شدن به وضعیّت مطلوب (همگرایی) ، ادامه دهید .

تکرار روش [۳]

در روش تکرار مقدار ؛ حتّی در زمانی که روش تغییر نمی کند ، همگرایی ، کند می باشد و ما واقعاً نمی توانیم نسبت به سودمندی یک وضعیّت ، بی تفاوت باشیم . تکرار روش ، روش ها را به صورت مستقیم به روز رسانی می نماید .

یادگیری یک روش

تا حالا ، ما فرض کردیم که دارای دانش زیادی هستیم . *در حالت به خصوص ، ما فرض کرده ایم که مدلی از جهان شناخته شده است ؛ این ، مدل انتقال وضعیّت می باشد* . اگر ما یک مدل نداشته باشیم چطور ؟ همه ی چیزی که ما می دانیم این است که مجموعه ای از وضعیّت ها و یک مجموعه از عملکردها وجود دارند . ما هنوز می خواهیم یک روش بهینه را یاد بگیریم .

---

[۱] Bellman equation

[۲] Value Iteration

[۳] Policy Iteration

یادگیری Q ای

آموزش یک روش به صورت مستقیم ، مشکل می باشد . مسأله این است که داده های ما به یک شکل نمی باشند : < وضعیّت ، عملکرد> . در عوض ، به شکل $s_1, s_2, s_3, ..., R$ می باشد . از آنجایی که ما تابع انتقال را نمی دانیم ، مشکل است که سودمندی یک وضعیّت را یاد بگیریم . در عوض ، ما یک تابع Q(s,a) را یاد خواهیم گرفت که " سودمندی " انجام عملکرد a را در وضعیّت s ، تخمین می زند . به طور صریح تر ، Q(s,a) مقدار a را در وضعیّت s را ارایه می کند و بعد از این به صورت بهینه عمل می کند .

$$Q(s,a) = R(s,a) + \gamma \max_a \sum_{s'} T(s,a,s')U(s')$$

روش بهینه این است که عملکرد با بالاترین مقدار Q را در هر وضعیّت ، به دست آوریم . اگر عامل بتواند Q(s,a) را یاد بگیرد ، می تواند عملکردهای بهینه را حتّی بدون دانستن تابع پاداش یا تابع انتقال داشته باشد .

یادگیری تابع Q

برای یادگیری Q ، ما باید بتوانیم مقدار یک عملکرد را در یک وضعیّت ، حتّی اگر پاداش ها در طول زمان پخش شده اند تخمین بزنیم . ما می توانیم این کار را به صورت تکراری انجام دهیم . توجّه کنید که $U(s) = \max_a Q(s,a)$. سپس ما می توانیم معادله را برای Q به صورت زیر بازنویسی نماییم :

$$Q(s,a) = R(s,a) + \gamma \max_{a'} Q(s',a')$$

بیایید تخمین Q(s,a) را به صورت $\hat{Q}(s,a)$ ، مشخّص نماییم . ما یک جدول لیست کننده ی هر جفت وضعیّت – عملکرد و مقدار تخمین زده شده ی Q را نگهداری می نماییم . عامل ، وضعیّت s را درک می کند ، عملکرد a را انتخاب می کند و سپس درک می کند که پاداش برابر با r = R(s,a) می باشد ، که

آن را دریافت می کند و وضعیّت جدید 's می باشد . سپس جدول Q را با توجه به فرمول زیر به روز می کند :

$$\hat{Q}(s,a) = r + \gamma \max_{a'} \hat{Q}(s',a')$$

عامل ، تخمینی از $\hat{Q}$ را برای 's برای تخمین $\hat{Q}$ برای s ، استفاده می نماید . توجّه کنید که عامل نیاز به هیچ دانشی در مورد R یا تابع انتقال برای اجرای این کار ندارد . آموزش Q ای برای نزدیک شدن (همگرایی) ضمانت شده است به شرطی که پاداش ها محدود شده باشند ، عامل جفت وضعیّت – عملکرد های این شکلی را به روشی که معمولاً نامحدود می باشد انتخا ب می کند . این به این معنی است که یک عامل باید دارای احتمالی غیر صفر از انتخاب هر a در هر s به صورت رشته ای از جفت های وضعیّت – عملکرد با نگرش نامحدود باشد .

اکتشاف

بنابراین چگونه آن را به وجود بیاوریم ؟ یادگیری Q ای دارای یک تفاوت مشخّص از دیگر الگوریتم هایی که تا کنون دیده ایم می باشد و آن این است که *عامل می تواند عملکردها و دیدی که آن ها به دست می آورند را انتخاب نماید و این یادگیری پویا [1] نام دارد* . عامل ، مایل به بیشینه کردن عملکرد می باشد ؛ این به این معنی است که کاری را که در حال حاضر به نظر می رسد بهترین باشد را انجام می دهد ، امّا اگر عامل هیچ گاه عملکردهای با " دید بد " نداشته باشد ، نمی تواند از خطاها بیرون بیاید . $\hat{Q}$ خیلی بادقّت نمی باشد ، بنابراین ، عملکردهای غیر بهینه را امتحان خواهیم کرد . از این پس ، در صورتی که $\hat{Q}$ بهتر شود ، ما عملکردهای بهینه را انتخاب خواهیم کرد .

---

[1] active learning

اکتشاف بولتزمان [۱]

یک راه برای انجام این کار استفاده از اکتشاف بولتزمان است . ما یک عمل با احتمال زیر را انجام می دهیم :

$$P(a \mid s) = \frac{k^{\hat{Q}(s,a)}}{\sum_j k^{\hat{Q}(s,a_j)}}$$

که k پارامتر دما می باشد . این فرمول ، شبیه به فرمولی است که ما در روش شبیه سازی گرم و سرد کردن استفاده کردیم .

یادگیری مستحکم [۲]

یادگیری Q ای نمونه ای از آن چه که ما یادگیری مستحکم می نامیم می باشد . در یادگیری مستحکم ، عامل ها نمونه هایی از این که چگونه عمل نمایند را نمی بینند . در عوض ، آن ها عملکردها را انتخاب می نمایند و پاداش ها یا مجازات ها را دریافت می نمایند . یک عامل می تواند یاد بگیرد ، حتّی در زمانی که عمل غیر بهینه ای را انجام می دهد . تعمیم ها به پاداش تأخیر داده شده و پردازه های غیرقطعی تصمیم گیری مارکوفی رسیدگی می کنند .

خلاصه

یادگیری Q ای برخی اوقات به صورت یادگیری مستقل از مدل [۱] شناخته می شود . در این روش ، عامل فقط نیاز به انتخاب عملکردها و دریافت پاداش ها دارد . مشکلات این روش ، چگونه تعمیم بدهد ؟ ،

---

[۱] Boltzmann exploration

[۲] reinforcement learning

مقیاس پذیری [۲] و سرعت نزدیک شدن (همگرایی) می باشد . یادگیری Q ای برای یک الگوریتم مفید تجربی به سطح مطلوبی رسیده است .

---

[۱] model-free learning

[۲] scalability

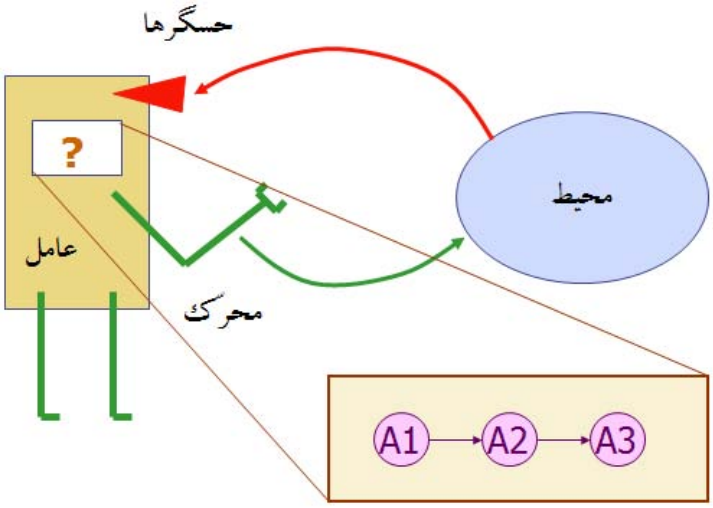
مترجم: سهراب جلوه گر
ویرایش دوّم ، بهار ۱۳۸۸

فصل بیست و پنجم

برنامه ریزی [۱]

[۱] planning

عامل برنامه ریزی



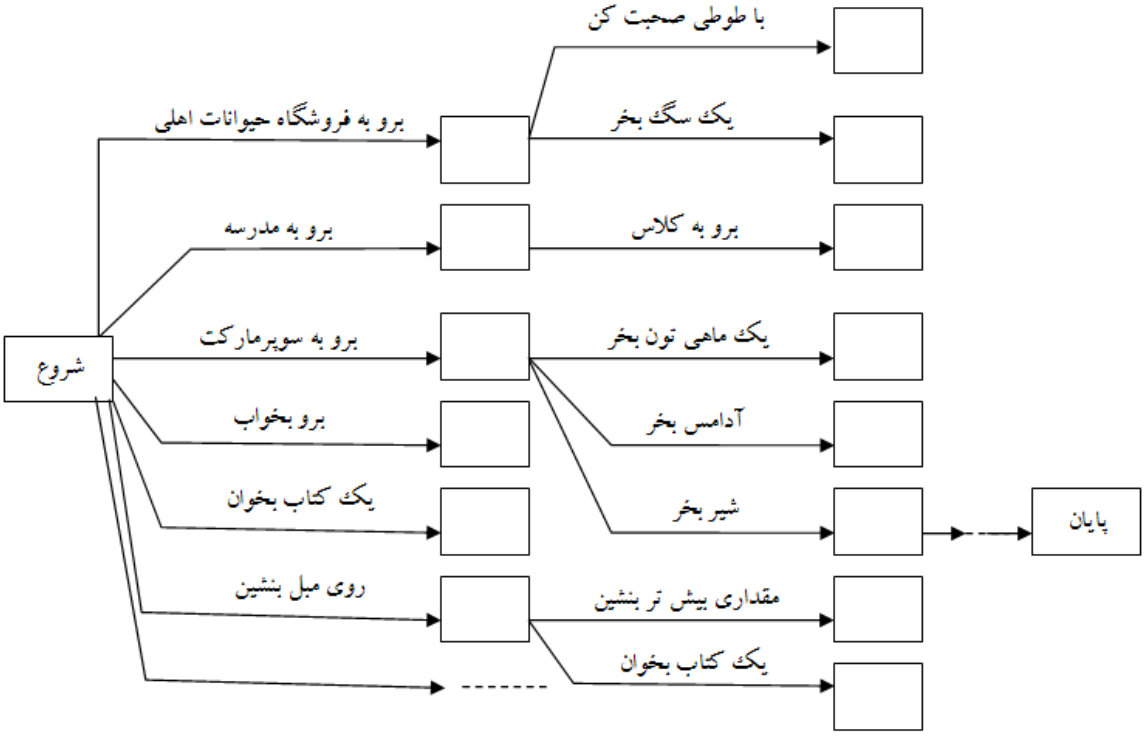
برنامه ریزی

مترجم: سهراب جلوه گر
ویرایش دوّم ، بهار ۱۳۸۸

برنامه ریزی ، تعبیه کردن یک رشته از عملکردها برای دسترسی به یک هدف می باشد . برنامه ریزی کلاسیک ، کاملاً قابل مشاهده ، قطعی ، محدود ، ثابت [۱] و گسسته است . زبان مشخصی برای ارایـه ی مسـایل برنامه ریزی وجود دارد .

جستجو و برنامه ریزی

الگوریتم های جستجوی استاندارد به نظر می رسد که به سختی شکست بخورند :



[۱] static

مترجم: سهراب جلوه گر
ویرایش دوّم ، بهار ۱۳۸۸

مشکلاتی که در روش جستجو وجود دارد عبارتند از : تعداد زیادی عملکردهای نامربوط وجود دارد ، پیدا کردن مکاشفه های خوب مشکل می باشد و از تجزیه ی مسأله نمی توان سودی برد . به صورت خودکار مکاشفه ها از ارایه مشتق می شوند ؛ مانند مسأله ی ارضای محدودیّت .

مقایسه ی برنامه ریزی با منطق

	جستجو	برنامه ریزی
وضعیّت ها	ساختمان داده های جاوا	عبارت های منطقی
عملکردها	کد جاوا	پیش شرط ها [۱] / نتیجه ها
هدف	کد جاوا	عبارت منطقی
برنامه	رشته ای از S_0	محدودیّت های روی عملکردها

زبانی برای مسایل برنامه ریزی

حل کننده ی مسأله ی مؤسّسه ی تحقیق استنفورد [۲] (اِستریپس) : جهان توسّط شرط های منطقی توصیف می شود ، وضعیّت به صورت اجتماع حرف های مثبت می باشد و می تواند به صورت

---

[۱] preconditions

[۲] STanford Research Institute Problem Solver (STRIPS)

گزاره ای ؛ به عنوان مثال ، $Happy \wedge Hungry$ برای نمایش وضعیّت عامل و یا منطق مرتبه ی اوّل و عبارت های مستقل از تابع باشد ؛ به عنوان مثال ،

$$At(Plane_1, Verona) \wedge At(Plane_2, Malpensa)$$

فرض جهان بسته ؛ هر شرطی که نام برده نشده غلط می باشد .

هدف ، وضعیّتی است که به صورت جزیی مشخّص شده است . اگر محدودیّت ها همه لفظ هایی از هدف باشند یک وضعیّت یک هدف را ارضا می نماید . به عنوان مثال ،

$$At(Plane_1, Verona) \wedge At(Plane_2, Malpensa)$$

هدف $At(Plane_2, Malpensa)$ را ارضا می نماید .

عملکردهای استریپس

عملکردها توسّط موارد زیر مشخّص می شوند :

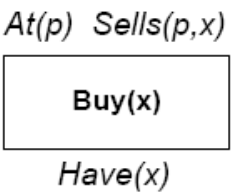
پیش شرط ها [1] **:** زمانی را مشخّص می کنند که عملکرد می تواند به کار گرفته شود .

اثرات : وضعیّت توسّط عملکردها تغییر می یابد . در این مورد ، لیست اضافه ، گزاره هایی هستند که صحیح از آب در می آیند و لیست حذف ، گزاره هایی که غلط از آب در می آیند .

عملکردها شامل متغیّرها می باشند ، یک طرح عملکرد منفرد عملکردهای مختلف را نمایش می دهد (نمونه ای از متغیّرها) . توصیف عملکردها به صورت منظّم می باشد و زبان ، محدود شده است .

طرح عملکرد :

[1] preconditions



Buy(x)

پیش شرط ، $At(p) \wedge Sells(p, x)$ می باشد و اثر ، Have(x) است و باید به این نکته توجه کنیم که ، هیچ اطّلاعاتی در مورد چگونگی اجرای عملکرد وجود ندارد ! . چون که زبان محدود شده است پس الگوریتم کارا می باشد . عملکرد ، نام و لیست پارامتر را مشخّص می نماید . پیش شرط ، اجتماع لفظ های مثبت و اثر ، اجتماع لفظ ها (مثبت یا منفی) است . یک مجموعه ی کامل از عملگرهای استریپس می تواند به یک مجموعه از اصول جانشین وضعیّت [۱] ترجمه شود .

معانی [۲]

با داشتن یک وضعیّت (اجتماع لفظ ها) ، اگر انتساب یک متغیّر ، در ارتباط با لفظ های شامل شده ی وضعیّت باشد ، پیش شرط ارضا می شود ؛ به عنوان مثال ، وضعیّت $At(HW) \wedge Sell(HW, Drill)$ ، پیش شرط $At(p) \wedge Sell(p, x)$ را با انتساب p/HW و x/Drill راضی می کند .

عملکردهای با پیش شرط های ارضا شده می توانند در این موارد به کار گرفته شوند : حذف عناصر از لیست حذف ، اضافه نمودن عناصر به لیست اضافه ، وضعیّت جدید :

$$At(HW) \wedge Sell(HW, Drill) \wedge Have(Drill)$$

مثال : خرید

---

[۱] successor – state axioms

[۲] semantics

عملکردها عبارتند از :

Buy(x)

پیش شرط : At(store) و Sells(store,x)

اثر : Have(x)

Go(x,y)

پیش شرط : At(x)

اثر : At(y) و $\neg At(x)$

شروع :

$$At(Home) \wedge Sells(SM, Milk) \wedge Sells(SM, Banana) \wedge Sells(HWS, Drill)$$

هدف :

$$Have(Milk) \wedge Have(Banana) \wedge Have(Drill)$$

مثال : انتقال محموله ی هوایی

عملکردها :

Load(c,p,a)

پیش شرط : $At(c,a) \wedge At(p,a) \wedge C\arg o(c) \wedge Plane(p) \wedge Airport(a)$

اثر : $\neg At(c,a) \wedge In(c,p)$

Unload(c,p,a)

پیش شرط : $In(c, p) \wedge At(p, a) \wedge C\arg o(c) \wedge Plane(p) \wedge Airport(a)$

اثر : $At(c, a) \wedge \neg In(c, p)$

Fly(p,from,to)

پیش شرط : $At(p, from) \wedge Plane(p) \wedge Airport(from) \wedge Airport(to)$

اثر : $\neg At(p, from) \wedge At(p, to)$

عملکردها عبارتند از Load ، Unload و نتیجه ی Fly . گزاره ها ، In(0,0) و At(0,0) می باشند . انـواع گزاره ها هم Cargo(0) ، Plane(0) و Airport(0) هستند .

شروع :

$$At(C_1, SFO) \wedge At(C_2, JFK) \wedge At(P_1, SFO) \wedge At(P_2, JFK)$$
$$\wedge C\arg o(C_1) \wedge C\arg o(C_2) \wedge Plane(P_1)Plane(P_2)$$
$$\wedge Airport(JFK) \wedge Airport(SFO)$$

هدف : $At(C_1, JFK) \wedge At(C_2, SFO)$.

یک راه حل به صورت زیر می باشد :

$$Load(C_1, P_1, SFO), Fly(P_1, SFO, JFK), Unload(C_1, P_1, JFK),$$
$$Load(C_2, P_2, JFK), Fly(P_2, JFK, SFO), Unload(C_2, P_2, SFO)$$

برنامه ریزی استریپس

مسأله ی برنامه ریزی استریپس : یک رشته از عملکردهایی که به هدف منجر می شـوند را پیـدا نمایید . وضعیّت ها و هدف ها توسّط یک اجتماع از لفظ ها تعریف می شوند .

مترجم: سهراب جلوه گر
ویرایش دوّم ، بهار ۱۳۸۸

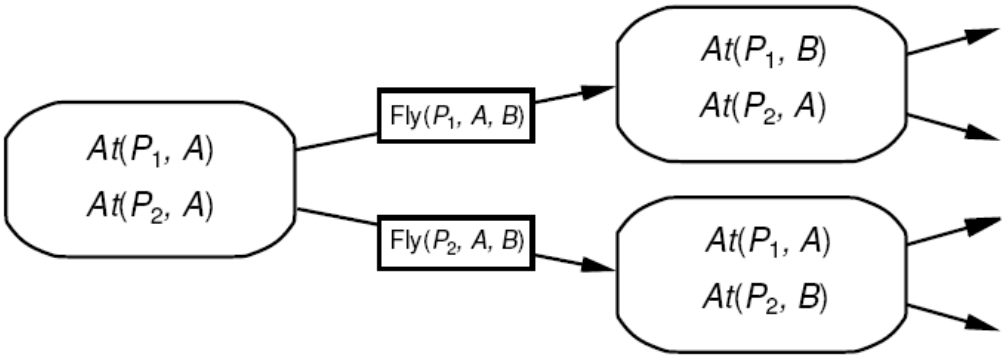
فضای حالت جستجو : جستجوی مستقیم از وضعیّت اولیّه تلاش می کند تا به هدف برسد . جستجوی معکوس از هدف تلاش و طرح ریزی می کند که به وضعیّت ابتدایی برسد .

فضای برنامه ی جستجو : برنامه ریزی با مرتبه ی جزیی [۱] ، فضای جزیی ساخته شده توسّط طرح ها را جستجو می کند .

فضای حالت جستجو

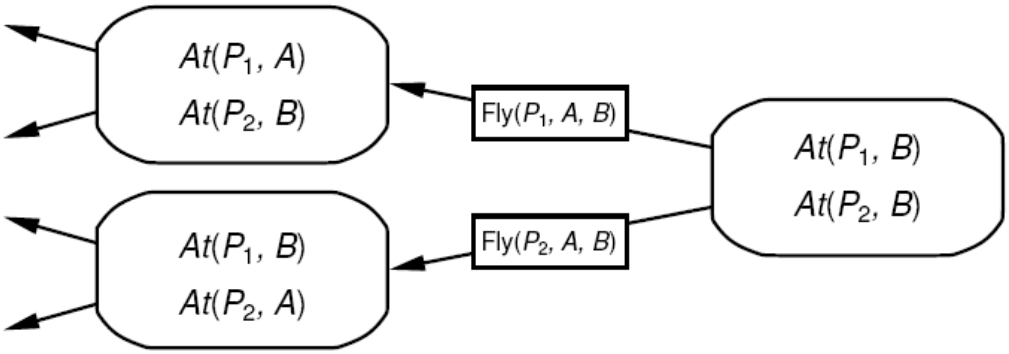
مسأله ی برنامه ریزی ، مسأله ی جستجو را تعریف می کند . وضعیّت اوّلیّه ، وضعیّت شروع می باشد ، آزمون هدف ، بررسی می کند که آیا هدف برآورده (ارضا) می شود یا نه ، عملکردها ، عملگرها را تعریف می کنند ، هزینه مرحله ، معمولاً ۱ می باشد . در زیر ، شکل a ، جستجوی مستقیم و شکل b ، جستجوی معکوس می باشد :

(a)



[۱] Partial-Order Planning(POP)

(b)



جستجوی مستقیم – حلقه ی جستجوی اصلی : یک عملکرد را انتخا ب نمایید و پیش شرط را با وضعیّت یکی نمایید . در صورتی که پیش شرط برآورده شد ، عملکرد تولید کننده ی یک وضعیّت جدید را به کار ببرید . برّرسی کنید که آیا وضعیّت جدید ، هدف را برآورده می نماید یا نه . در صورتی که فضای حالت ، محدود باشد ، برای الگوریتم های جستجوی کامل ، کامل می باشد . این روش به دلیل عملکردهای نامربوط ، ناکارآمد می باشد و به مکاشفه های خوب ، نیازمند می باشد .

جستجوی معکوس – ایده ی این روش این است که عملکردهای مربوط را فقط با شروع از هدف انتخاب نمایید . عملکرد ، در زمانی که به یکی از متّحدانش دست می یابد (لیست اضافه) ، مربوط می باشد . در زمانی که هر لفظ مربوط را بی اثر [1] نمی کند (لیست حذف) ، سازگار می باشد .

یک عملکرد مربوط و سازگار را انتخاب نمایید و وضعیّت جدید را تولید نمایید : لیست اضافه را پاک نمایید و پیش شرط ها را اضافه نمایید .

این روش با یک وضعیّت برآورده شده توسّط وضعیّت ابتدایی خاتمه می یابد و روی وضعیّت های مربوط و سازگار منشعب می شود .

فضای حالت و فضای برنامه

[1] undo

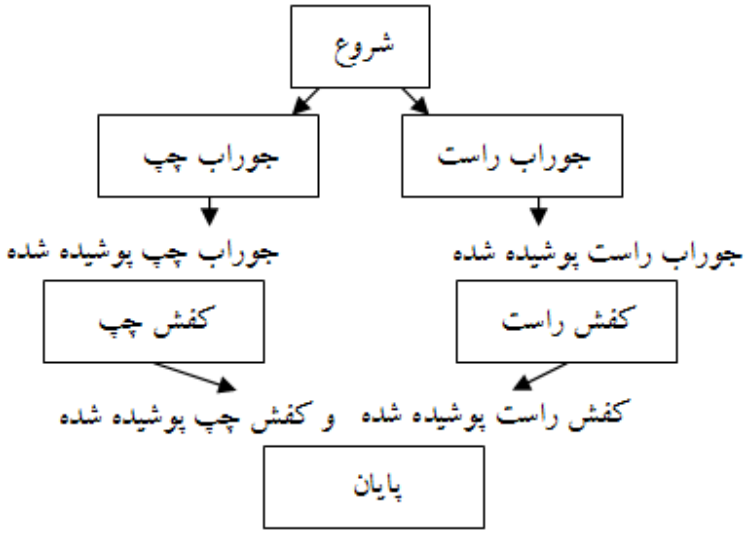
جستجوی مستقیم و معکوس ، رشته های خطی عملکردها را اکتشاف می نماید که این کار لازم نمی باشد ؛ به راه حل انتقال هوایی محموله توجّه نمایید :

$$Load(C_1, P_1, SFO), Fly(P_1, SFO, JFK), Unload(C_1, P_1, JFK),$$
$$Load(C_2, P_2, JFK), Fly(P_2, JFK, SFO), Unload(C_2, P_2, SFO)$$

تنها ترتیب لازم ، میان Load ، Fly و Unload می باشد که توسط اثرات سببی به هم متّصل شده اند ؛ به عنوان مثال ، $Load(C_1, P_1, SFO)$ به $In(C_1, P_1)$ استفاده شده توسط پیش شرط Unload دست می یابد . نیازی نیست که $Load(C_2, P_2, JFK)$ را بعد از $Unload(C_1, P_1, JFK)$ قرار دهیم .

طرح با مرتبه ی جزیی [1] : گراف عملکردها شامل شروع و پایان می باشد (گراف ≡ مرتبه ی جزیی ≺)

مثال : طرح با مرتبه ی جزیی :



[1] partial-order plan

طرح های با مرتبه ی کلّی :

طرح های منظّم شده به صورت جزیی - مجموعه ای از مراحل با **مرحله ی شروع** [۱] دارای توصیف وضعیّت اوّلیّه به صورت اثرش می باشد ، **مرحله ی پایانی** [۲] دارای توصیف هدف به صورت پیش شرطش می باشد ، **اتصالات سببی** [۳] از نتیجه های یک مرحله به پیش شرط دیگری به وجود می آیند و **مرتب سازی زمانی** [٤] میان جفت های مراحل می باشد .

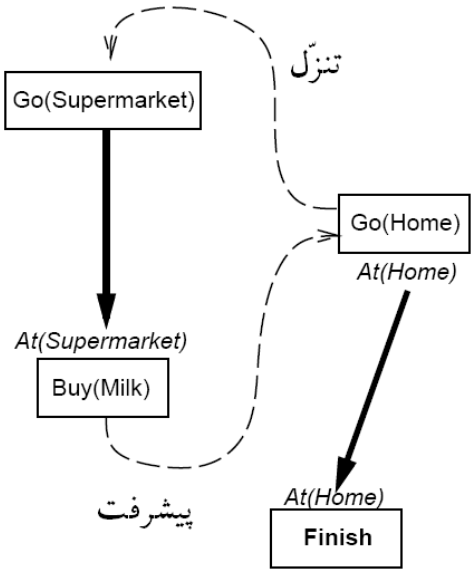
شرط باز [٥] ، پیش شرط یک گام که هنوز به صورت سببی متّصل نشده است می باشد . **طرح** در صورتی کامل است که هر پیش شرط در دسترس باشد ، **پیش شرط** در صورتی قابل دسترس می باشد که اثر یک

---

[۱] start step
[۲] finish step
[۳] causal links
[٤] temporal ordering
[٥] open condition

مرحله قبل تر باشد و هیچ مرحله ی میانی ، آن را بی اثر ننماید (تناقض نداشته باشند) . طرح ها نمی توانند دارای حلقه باشند .

ناسازگاری ها و پیشرفت [۱] / **تنزل** [۲] – یک عملکرد ناسازگار یک مرحله ی میانی بالقوّه است که شرط قابل دسترس توسط اتّصال سببی را خراب می نماید . به عنوان مثال ، Go(Home) ، At(Supermarket) را حذف می نماید :



تنزّل : قبل از Go(Supermarket) قرار دهید

پیشرفت : بعد از Buy(Milk) قرار دهید

---

[۱] promotion

[۲] demotion

فهرست برخی از منابع و مآخذ

بیش تر مطالب استفاده شده برای ترجمه ی این کتاب ، مطالب به زبان انگلیسی موجود در پایگاه های اینترنتی دانشگاه های کشور ایالات متحده ی آمریکا می باشد . در زیر ، فهرست برخی از پایگاه های اینترنتی ای که در ترجمه ی این کتاب از آن ها استفاده شده را مشاهده می نمایید :

۱- مطالب موجود در پایگاه اینترنتی دانشگاه سانفرانسیسکو ایالات متحده ی آمریکا ، به نشانی اینترنتی www.cs.usfca.edu

۲- مطالب موجود در پایگاه اینترنتی دانشکده ی برکلی دانشگاه کالیفرنیا ایالات متحده ی آمریکا ، با آدرس اینترنتی http://www.cs.berkeley.edu (مطالب استوارت راسل و پیتر نورویگ)

۳- مطالب موجود در پایگاه اینترنتی دانشکده ی علوم کامپیوتر دانشگاه دولتی تگزاس ایالات متحده ی آمریکا ، به آدرس اینترنتی www.cs.txstate.edu

۴- مطالب موجود در پایگاه اینترنتی http://www-formal.stanford.edu (مطالب جان مک کارتی ، استاد دانشگاه استنفورد)

۵- مطالب گردون اس. نواک جی. آر [۱] استاد دانشگاه تگزاس ایالات متحده ی آمریکا به نشانی اینترنتی www.cs.utexas.edu

۶- مطالب Milos Hauskrecht استاد دانشگاه شهر پیتسبورگ کشور ایالات متحده ی آمریکا به نشانی اینترنتی www.cs.pitt.edu

۷- مطالب موجود در پایگاه اینترنتی دانشگاه مریلند کشور ایالات متحده ی آمریکا با آدرس اینترنتی http://www.cs.umd.edu

۸- مطالب موجود در پایگاه اینترنتی مدرسه ی علوم کامپیوتر و مهندسی ، به نشانی اینترنتی www.cse.unsw.edu.au

۹- مطالب موجود در پایگاه اینترنتی جامعه ی هوش محاسباتی ، به نشانی اینترنتی

http://ebrains.la.asu.edu/~jennie/tutorial/index.htm

۱۰- مطالب موجود در پایگاه اینترنتی مهندسان سیاهپوست ، به آدرس اینترنتی www.nsbe.org

۱۱- مطالب موجود در پایگاه اینترنتی دانشگاه آزاد بوزن ایتالیا ، به نشانی اینترنتی http://www.inf.unibz.it

۱۲- مطالب موجود در پایگاه اینترنتی دانشگاه کوبلنز آلمان ، به نشانی اینترنتی http://www.uni-koblenz.de

---

[۱] - Gordon S. Novak Jr.

۱۳- مطالب موجود در پایگاه اینترنتی دانشکده ی علوم کامپیوتر دانشگاه کورک ایرلند ، با آدرس اینترنتی http://www.cs.ucc.ie

۱۴- مطالب موجود در پایگاه اینترنتی دانشکده ی علوم کامپیوتر و اطّلاعات نروژ ، به آدرس اینترنتی www.idi.ntnu.no

۱۵- مطالب موجود در پایگاه اینترنتی دانشگاه ملی تایوان ، به آدرس اینترنتی www.csie.ntu.edu.tw

۱۶- مطالب موجود در پایگاه اینترنتی دانشگاه لیورپول [۱] انگلستان به نشانی اینترنتی http://www.csc.liv.ac.uk

۱۷- مطالب موجود در پایگاه اینترنتی دانشگاه خصوصی سیبنسای [۲] کشور ترکیه

۱۸- مطالب موجود در پایگاه اینترنتی دانشگاه شمال غربی [۳] کشور ایالات متّحده ی آمریکا به نشانی اینترنتی http://www.eecs.northwestern.edu

۱۹- دانشکده ی علوم کامپیوتر دانشگاه بولوگنا کشور ایتالیا به نشانی اینترنتی http://www.cs.unibo.it

---

۱- Liverpool

۲- Sabancı University

۳- NORTHWESTERN UNIVERSITY