« به نام او »

آموزش شبیه سازی دو بعدی فوتبال

نویسنده : محمد علی میرزایی - علی یعقوبی آزاد

مرجع روبوکاپ ایران

www.iranrcss.com

جلسه دوازدهم

۱- آموزش بیس UVA_Trilearn Base

۲- آموزش هوش مصنوعی، جست و جوی آگاهانه و اکتشاف، **الگوریتم های ژنتیک**

در طی جلسات گذشته ما به طور مفصل به توضیح deMeer۵ از توابع کلاس Player پرداختیم . در این جلسه قصد داریم به توضیح تابع زیر که یکی از کلیدی ترین توابع بیس می باشد بپردازیم . با ما همراه باشید :

```
۱ - SoccerCommand BasicPlayer::moveToPos( VecPosition posTo, AngDeg angWhenToTurn,
                        double dDistBack, bool bMoveBack, int iCycles )
۲ - {
۳ - // previously we only turned relative to position in next cycle, now take
۴ - // angle relative to position when you're totally rolled out...
۵ - //  VecPosition posPred   = WM->predictAgentPos( ۱, ۰ );
۶ - VecPosition posAgent  = WM->getAgentGlobalPosition();
۷ - VecPosition posPred    = WM->predictFinalAgentPos();
۸ - AngDeg      angBody   = WM->getAgentGlobalBodyAngle();
۹ - AngDeg      angTo     = ( posTo - posPred ).getDirection();
۱۰ -            angTo     = VecPosition::normalizeAngle( angTo - angBody );
۱۱ - AngDeg    angBackTo = VecPosition::normalizeAngle( angTo + ۱۸۰ );
۱۲ - double     dDist        = posAgent.getDistanceTo( posTo );
۱۳ - Log.log( ۵۰۹, "moveToPos (%f,%f): body %f to %f diff %f now %f when %f",
```

```
                posTo.getX(), posTo.getY(), angBody,
            ( posTo - posPred ).getDirection(), angTo,
            ( posTo -  WM->predictAgentPos( ١, ٠ )).getDirection(),
            angWhenToTurn );

١٤ - if( bMoveBack )
١٥ - {
١٦ - if( fabs( angBackTo ) < angWhenToTurn )
١٧ - return dashToPoint( posTo, iCycles );
١٨ - else
١٩ - return turnBackToPoint( posTo );
٢٠ - }
٢١ - else if(  fabs( angTo     ) < angWhenToTurn ||
             (fabs( angBackTo ) < angWhenToTurn && dDist < dDistBack ) )
٢٢ - return dashToPoint( posTo, iCycles );
٢٣ - else
٢٤ - return directTowards( posTo, angWhenToTurn );
٢٥ - //return turnBodyToPoint( posTo );
```

۲۶ - }

از خط ابتدایی تابع شروع میکنیم :

۱ - SoccerCommand BasicPlayer::moveToPos(VecPosition posTo, AngDeg angWhenToTurn,
double dDistBack, bool bMoveBack, int iCycles)

این تابع یکی از توابع کلاس BasicPlayer است که به عنوان خروجی یک SoccerCommand بر می گرداند . آرگومان های این توابع عبارتند از :

آ) VecPosition posTo : نقطه ای که میخواهید بازیکن به آنجا برود .

ب) AngDeg angWhenToTurn : زاویه ی گردن بازیکن هنگام حرکت .

ج) double dDistBack

د) bool bMoveBack : برای حرکت به سوی عقب به کار میرود (اکیدا توصیه نمی شود شما به سوی عقب حرکت کنید.)

ه) int iCycles

که البته شما میتوانید سه آرگومان انتهایی را به تابع نفرستید .

توابعی را که تا قبل از خط ۱۷ آمده است ، در جلسات گذشته به طور کامل توضیح داده ایم ؛ اما در خط شماره ۱۷ تابعی به کار رفته که تا بدین روز بررسی نشده است :

۱۷ - return dashToPoint(posTo, iCycles);

این تابع یک حرکت ساده به سمت نقطه posTo میکند . شما با این تابع به بازیکن دستور میدهید که یک حرکت مستقیم به سمت نقطه ی گفته شده ، داشته باشد .

```
۱۹ - return turnBackToPoint( posTo );
```

این تابع به بازیکن دستور برگشتن (چرخاندن گردن و بدن) به سمت نقطه ی داده شده ، می دهد .

و در انتها :

```
۲۴ - return directTowards( posTo, angWhenToTurn );
```

در زیر تابع را می آوریم . در جلسه بعد به توضیح این تابع خواهیم پرداخت :

```
۱ - SoccerCommand BasicPlayer::directTowards( VecPosition posTurnTo,
    AngDeg angWhenToTurn, VecPosition *pos, VecPosition *vel, AngDeg *angBody  )
۲ - {
۳ -   //  return turnBodyToPoint( posTurnTo );
۴ -   // copy values or initialize when not set
۵ -   VecPosition posAgent= (pos    ==NULL)?WM->getAgentGlobalPosition ():*pos;
۶ -   VecPosition velAgent= (vel    ==NULL)?WM->getAgentGlobalVelocity ():*vel;
۷ -   AngDeg  angBodyAgent= (angBody==NULL)?WM->getAgentGlobalBodyAngle():*angBody;
۸ -   // first predict what happens when the agents rolls out.
```

```
٩ -  VecPosition posPred    = WM->predictFinalAgentPos();

١٠ - AngDeg      angTo      = ( posTurnTo - posPred ).getDirection();

١١ - AngDeg      ang        = VecPosition::normalizeAngle( angTo - angBodyAgent );

١٢ - AngDeg      angNeck = ٠;

١٣ - int iTurn = ٠;
١٤ - while( fabs( ang ) > angWhenToTurn && iTurn < ٥ )

١٥ - {

١٦ -   iTurn++;

١٧ -   WM->predictStateAfterTurn(

١٨ -         WM->getAngleForTurn( ang, velAgent.getMagnitude() ),
        &posAgent,
        &velAgent,
        &angBodyAgent,
        &angNeck );

١٩ -   ang = VecPosition::normalizeAngle( angTo - angBodyAgent );

٢٠ - }

٢١ - Log.log( ٥٠٩, "direct towards: %d turns", iTurn );

٢٢ - posAgent = (pos    ==NULL)?WM->getAgentGlobalPosition ():*pos;

٢٣ - velAgent = (vel    ==NULL)?WM->getAgentGlobalVelocity ():*vel;
```

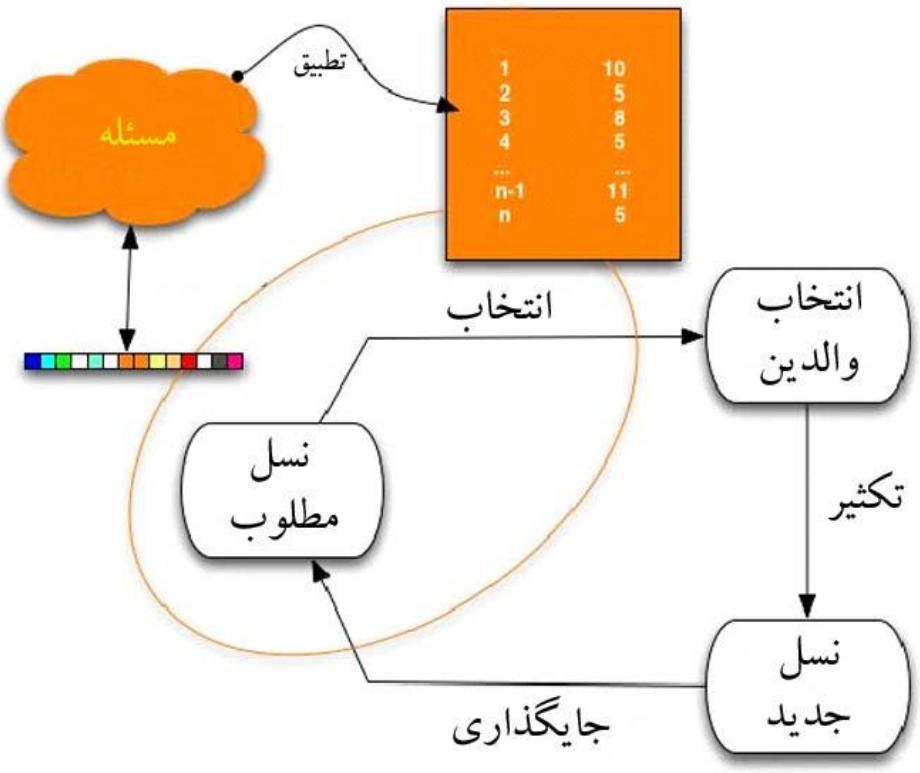
```
۲۴ -  angBodyAgent = (angBody==NULL)?WM->getAgentGlobalBodyAngle():*angBody;

۲۵ -  switch( iTurn )

۲۶ -  {

۲۷ -    case ۰:   cerr << "direct towards: ۰ turns" ;

۲۸ -              return SoccerCommand( CMD_ILLEGAL );

۲۹ -    case ۱:

۳۰ -    case ۲:   return turnBodyToPoint( posTurnTo, ۲ );

۳۱ -   default:  return dashToPoint(
            (pos==NULL)?WM->getAgentGlobalPosition ():*pos   );  // stop

۳۲ -  }

۳۳ - }
```

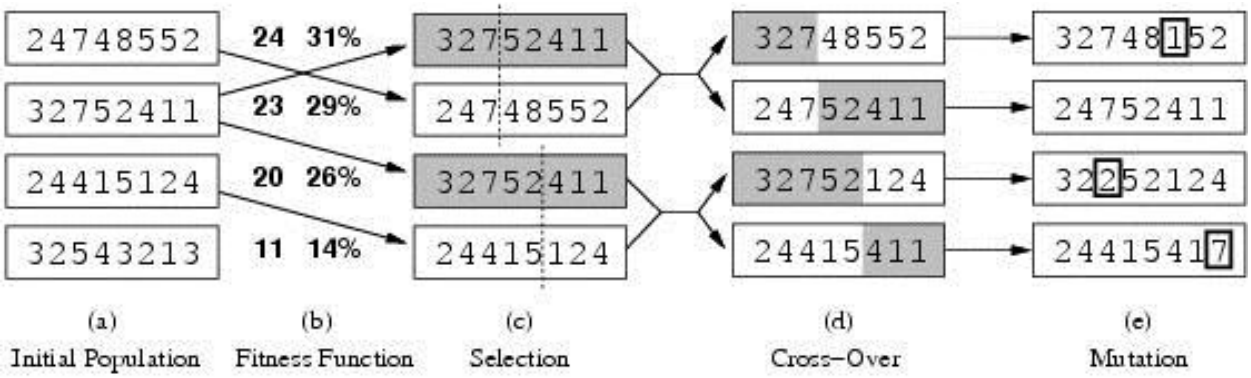
این آموزش ها صرفا جنبه ی آموزش دارد اما نباید خیال کنید با خواندن این آموزش ها میتوانید تیم بنویسید. این آموزش ها تنها برای یادگیری راحت تر شما می باشد ولی شما باید تلاش کنید و تیم را تکمیل کنید . شما هیچگاه نباید وابسته به بیس باشید بلکه باید سعی کنید بعد از مدتی کار، خودتان از پایه یک تیم بنویسید . نباید اینگونه باشد که شما بعد از اینکه با بیس UVA کار کردید نتوانید با بیس دیگری کار کنید . شما نباید بیس را یاد بگیرید بلکه باید چگونگی نوشتن بیس را یاد بگیرید . لطفا این نکته را همیشه در نظر بگیرید .

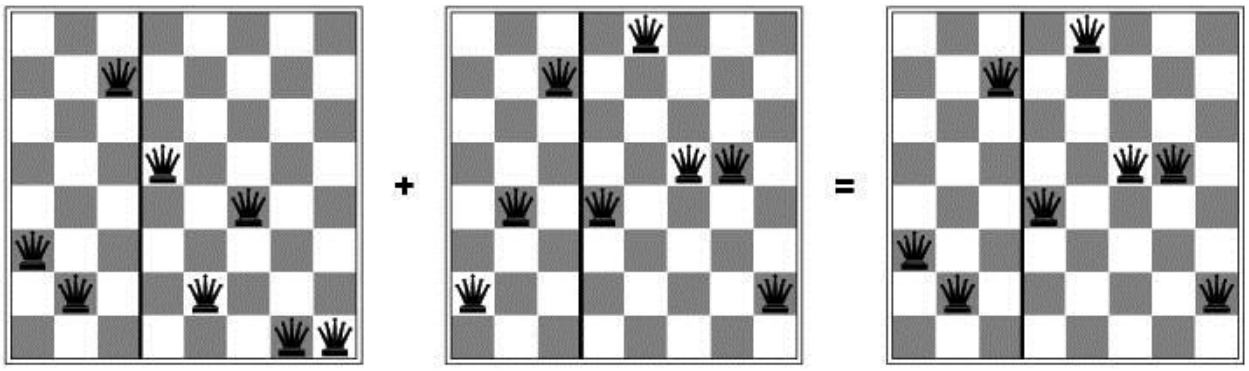
جست و جوی آگاهانه و اکتشاف

الگوریتم های ژنتیک

شکلی از جست و جوی پرتو غیر قطعی که حالتهای جانشین از طریق ترکیب دو حالت والد تولید میشود.







جست و جوی محلی در فضاهای پیوسته

گسسته در مقابل محیط های پیوسته

- در فضاهای پیوسته، تابع جانشین در اغلب موارد، حالتهای نامتناهی را بر میگرداند.

حل مسئله:

- گسسته کردن همسایه هر حالت

- استفاده از شیب منظره

$$\mathbf{x} \leftarrow \mathbf{x} + \alpha \nabla f \quad where\ \nabla f = \left\{ \frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots \right\}$$

- روش نیوتن رافسون

عاملهای جست و جوی Online و محیطهای ناشناخته

تا به حال همه الگوریتمها برون خطی بودند.

- برون خطی(Offline):راه حل قبل از اجرا مشخص است.
- درون خطی(Online):با یک در میان کردن محاسبات و فعالیت عمل میکند.

جستجوی درون خطی در محیطهای پویا و نیمه پویا مفید است.

- آنچه را که باید واقعا اتفاق بیفتد، در نظر گرفته نمیشود.

جست و جوی درون خطی ایده ضروری برای **مسئله اکتشاف** است.

- فعالیتها و حالتها برای عامل مشخص نیستند.
- مثال:قرار گرفتن روبات در محیطی جدید، نوزاد تازه بدنیا آمده

مسئله های جست و جوی Online

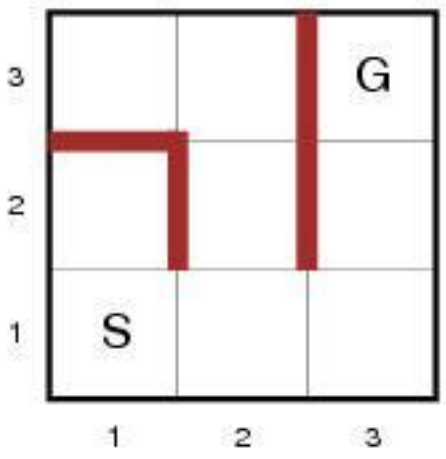
اطلاعات عامل

- Actions(s): لیستی از فعالیتهای مجاز در حالت s
- تابع هزینه مرحله ای c(s,a,s'): استفاده وقتی که بداند s' نتیجه است.
- Goal-Test(s): آزمون هدف

عامل حالت ملاقات شده قبلی را تشخیص میدهد.

فعالیتها قطعی اند.

دسترسی به تابع اکتشافی قابل قبول h(s)



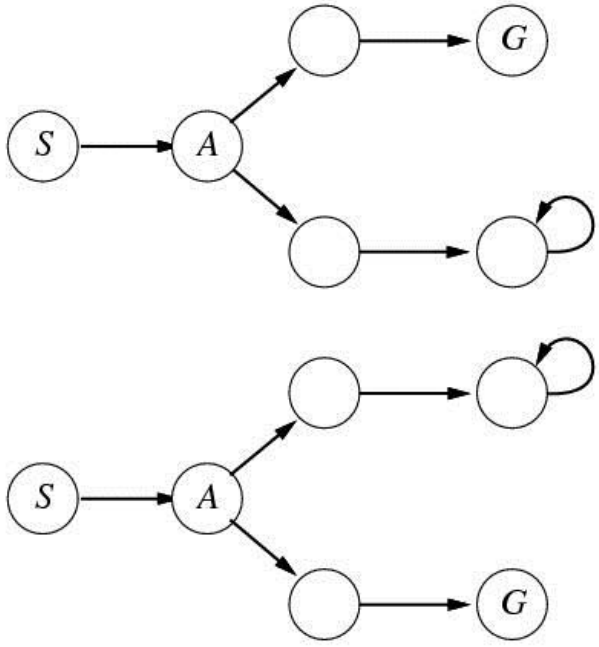
هدف: رسیدن به G با کمترین هزینه

هزینه: مجموع هزینه های مراحل مسیری است که عامل طی میکند.

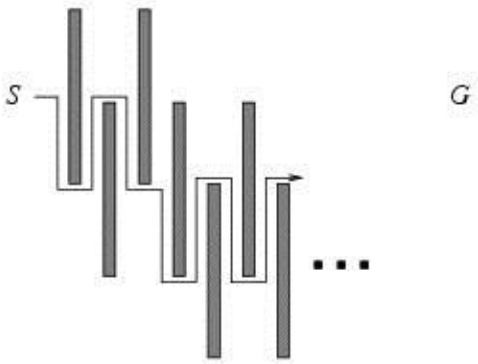
نسبت رقابتی: مقایسه هزینه با هزینه مسیری که اگر عامل فضای حالت را از قبل میشناخت، طی میکرد.

- در بعض موارد، بهترین نسبت رقابتی نامتناهی است.
- ممکن است جستجو به یک حالت بن بست برسد که نتوان از طریق آن به هدف رسید.

دو فضای حالت که عامل جست و جوی Online را به بن بست میرسانند. هر عاملی حداقل در یکی از این دو فضا شکست می خورد.



یک محیط دو بعدی که موجب میشود عامل جست و جویOnline مسیر دلخواه ناکارآمدی را برای رسیدن به هدف حل کند.



در جلسه بعد به مبحث مسائل **ارضای محدودیت** خواهیم پرداخت.

پایان