مکان نام مقاله

چکیده:

صنعت بازی‌های رایانه‌ای امروزه به یکی از پرطرفدار‌ترین و سرگرم‌کننده‌ترین صنایع در جهان تبدیل شده است . یکی از اصلی ترین عوامل ایجاد هیجان و جذابیت در بازی‌های رایانه‌ای هوش مصنوعی می‌باشد و سعی سازندگان بازی‌‌های رایانه‌ای استفاده هرچه بیشتر از این عامل در محصولاتشان می‌باشد ، زیرا باعث بهبود روند بازی می‌شود. هوش مصنوعی بازی‌ها خود به چندین بخش مانند شبیه سازی جمعیت ، کنترل وسایل نقلیه و ... می‌باشد ، که از جمله پر استفاده‌ترین آنها می‌توان به طراحی هوش مصنوعی برای تعقیب و در نهایت دستگیری بازیکن اشاره کرد . در طراحی این سیستم ها که عموما چندعامله هستند ، یکی از مهم ترین چالش ها قرار دادن هر عامل در جای مناسب برای هرچه بهتر رسیدن به هدف است.

مدل پیشنهادی ما از یک سیستم رخداد محور و چندین سیستم پیش بینی کلی تشکیل شده است و با استفاده از این سیستم‌ها سعی دارد که بهترین چیدمان را برای عامل‌های در اختیارش پیدا کند . در ادامه توضیحات این سیستم را بررسی می‌کنیم.

کلمات کلیدی:

هوش مصنوعی ، سیستم های چند عامله ، هوش مصنوعی مرکزی ، هوش مصنوعی توزیع شده

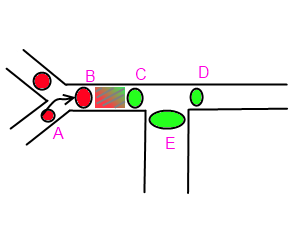
مقدمه:

در این مدل هوش مصنوعی ما یک عامل مرکزی داریم که نقش فرماندهی و ارتباط یک عامل با عامل های دیگر را ایفا می‌کند و یک هوش توزیع شده که هرکدام از عامل ها را در انتخاب تصمیم ها یاری می‌کند . همچنین تعدادی عامل در اختیار داریم که هر کدام از آنها در مکان های مختلفی از نقشه قرار گرفته اند. وظیفه‌ی کلی این سیستم دستگیری بازیکنی است که در مکانی از نقشه قرار دارد. بازیکن آزادانه می‌تواند در هر مسیری که بخواهد جا‌ به جا شود . عامل ها در شهر بر اساس نقاطی که برایشان از قبل منظور شده است به گشت زنی می‌پردازند و با دیدن بازیکن در صورتی که بازیکن تحت تعقیب باشد ، به تعقیب او می‌پردازند . مسیریابی عامل ها با استفاده از سیستم Navigation Mesh و الگوریتم A\* انجام می‌گیرد. هر عاملی که به تعقیب بازیکن بپردازد ابتدا به فرمانده (هوش مصنوعی مرکزی) موقعیت و وضعیت خود را گزارش می‌دهد سپس فرمانده بر مبنای اطلاعات رسیده ، عامل‌های مفید را از بقیه جدا می‌کند.(منظور از عامل مفید عاملی است که توانایی کمک به تعقیب بازیکن را داشته باشد ، برای مثال خیلی دور نباشد یا در ماموریت دیگری قرار نگرفته باشد) .

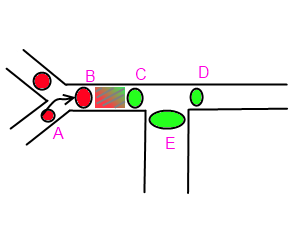
سپس با محاسبه‌ی فاصله‌ی واقعی هرکدام از عامل‌های مفید تا بازیکن که با استفاده از الگوریتم A\* بر روی خانه های ساخته شده با سیستم Navigation Mesh به دست می‌آید تصمیماتی را در خصوص اینکه هر کدام از عامل‌ها در کجا قرار بگیرند را اتخاذ می‌کند و به هر کدام از عامل ها تصمیم متناظر خودش را ارسال می‌کند و عامل آن را انجام می‌دهد. هدف اصلی فرمانده محاصره کردن بازیکن و بستن راه های فرار وی می‌باشد و این کار را با فرستادن عامل ها به انتهای مسیرهایی که احتمال حضور بازیکن در آنها بیشتر است انجام می‌دهد.

**سیستم ساخت نقشه:**

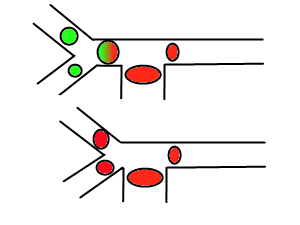
ما به منظور پیاده سازی ایده خود زمین و راه‌های آن را که معرّف نقشه بازی است، به یک گراف مدل کردیم. در این مدل راس‌ها به محل‌های ورود و خروج اطلاق میشود. به عنوان مثال در **شکل زیر**



نقاط رنگی نشان دهنده یک راس در گراف هستند ( چراکه محل ورود یا خروج یک مسیر جدید می‌باشند ) و خطوط درون شکل نشان دهنده یالها میباشند. در این گراف، یال AB نشان میدهد که یک مسیر از رأس A به B و بالعکس وجود دارد که نیاز به عبور از هیچ رأس دیگری ندارد. استفاده از این شیوه مدل سازی به ما کمک می‌کند که مستقل از مختصات جغرافیایی فرد در حال تعقیب ( بدون توجه به x و y مکانی ) و تنها به کمک آگاهی‌ از چند رأس آخری که از آن‌ها عبور کرده ، محل نسبی‌ فرد را تشخیص دهیم. همچنین به کمک روشی که در زیر معرفی‌ میشود ، ما قادر به تشخیص تغییر مسیرها خواهیم بود. بدین منظور ما گراف را به کمک ماتریس مجاورت (Adjacent Matrix) پیاده سازی کردیم. در نتیجه، اگر فرض کنیم  که می‌دانیم فرد در حال تعقیب به ترتیب از راس های A و B عبور کرده باشد، مشخص میشود که این فرد در حال حرکت بسمت راس C می‌باشد و در ناحیه‌‌ هاشور خورده مشخص شده در شکل میباشد. اینکار ( تشخیص مکان بعدی و در نتیجه آن، محل نسبی فرد) بکمک تفریق مجموعه رئوس مجاور B از مجموعه رئوس مجاور راس A بدست می آید، که مشخصا پردازش بسیاری کمی‌ را نیاز دارد). از آنجا که در مدل پیشنهادی ما، رئوس بعبارتی یک گذرگاه (trigger) می باشند، ما از نگرش اتفاق محور (Event Base) در انجام محاسبات فوق استفاده میکنیم. به این معنا که در هر بار عبور کردن فرد در حال تعقیب از یک گذرگاه، محاسبات فوق یکبار انجام میشود.   
از آنجا که ما مسیر پیشروی فرد در دست تعقیب را بکمک ۲ راس آخری که وی از آن ها عبور کرده است ، محاسبه ‌کنیم، این نکته مهم است که این ۲ راس باید متوالی باشند. از طرف دیگر از آنجا که یکی‌ از جنبه‌هایی که در این طرح پیشنهادی مبنا بوده است شبیه سازی بودن آن است، ما تنها در صورتی‌ رخداد عبور از یک راس را پردازش می‌کنیم که یکی‌ از عامل ها‌ی در حال تعقیب، شخص در حال فرار را در آن لحظه ببیند ، در غیر این صورت مانند دنیای واقعی هیچ رخدادی صادر نمی‌شود.  
به عنوان مثال اگر فرد در دست تعقیب، بعد از عبور از گذرگاه‌های A، B و C ، از گذرگاه E عبور کند ولی‌ عاملی به او نرسد و مدتی‌ بعد در حال عبور از گذرگاه D دیده شود، نشان میدهد که او توسط عاملی‌ در حال تعقیب نبوده است و ما اطلاعاتی‌ از وی نداشتیم. پس نمیتوانیم تنها بر اساس راس D نظر بدهیم. در اینجا ما بر اساس نقشه واقعی بازی ( و نه مدل گراف آن) مسیر بعدی را بدست می‌آوریم. اینکار میتواند بشکل های مختلف انجام گیرد که ما از روش شبیه سازی حرکت واقعی با سرعت بسیار بالا استفاده کردیم .  
همچنین ما در ادامه سعی‌ کردیم برای بهینه سازی گراف ، تغیراتی‌ در انتخاب رئوس داشته باشیم. بدین منظور ما سعی کردیم بر اساس فاصله دو راس متوالی و اهمیت ناحیه‌‌ بین  ۲ راس ( که نماینده یک یال در گراف، و یک مسیر در نقشه میباشد) رئوس را با یکدیگر ادغام کنیم. در **شکل زیر** (شکل2) دو نمونه از نحوه بهینه انتخاب رئوس **شکل زیر** آورده شده است.



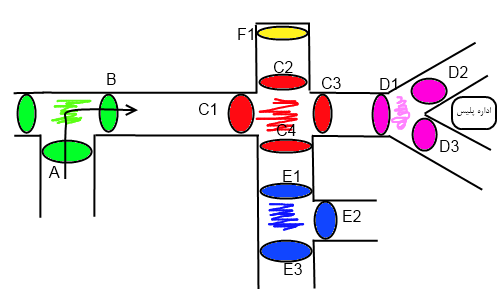
شکل 1



شکل2

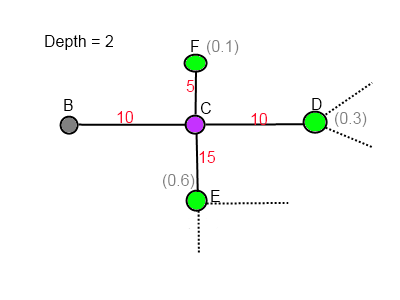
**درخت پیشبینی مسیر :**

همانطور که در بالا به درخت پیش بینی اشاره شد، در هر بار عبور از رئوس ، ما یک پردازش برای برسی‌ محل بعدی فرد در حال گریز انجام میدهیم. ما همچنین همزمان با بروز رسانی اطلاعات فوق، یک درخت پیشبینی‌ مسیر نیز تشکیل میدهیم. علت همزمانی این دو عملیات، عدم امکان تغییر مسیر در بین دو راس است. یعنی‌ پس از اینکه فرد در حال تعقیب از یک گذرگاه عبور کرد، تا رسیدن فرد به رأس بعدی ، تغییر مسیری نمیتواند صورت گیردکه این فرصت مناسبی را برای تشکیل درخت ما مهیا می‌کند. (لازم به ذکر است که گذر از رئوس که منجر به ورود به یک ناحیه‌‌ چند راهی‌ میشود بشکل دیگری بررسی‌ میشود که از بیان این حالت خاص صرفنظر میکنیم). هدف از تشکیل درخت فوق که بنوعی از گراف اصلی ما ولی‌ بشکل خاص استفاده می‌کند ، به دست آوردن احتمال حرکت فرد در حال گریز به محل‌های در پیش روی اوست. نتیجه ای که از این درخت برای ما اهمیت دارد، بدست آوردن مکان‌های مناسب برای قرار دادن عامل‌ها و بستن راه‌ها است.

برای نمونه و بر اساس **شکل** 3 ، رئوس C1 تا C4 از نظر مفهومی‌ نمایانگر یک منطق قابل پوشش توسط نیروهای تعقیب کننده می‌باشد، پس ما در درخت خود، برای چنین رئوسی، یک راس را به عنوان نماینده برای قرار گیری عامل استفاده میکنیم.   


شکل 3

نحوه تشکیل این درخت را با یک مثال نشان میدهیم.  
اگر فرض کنیم فرد در دست تعقیب بترتیب از گذرگاه‌های A و B گذشته باشد ، برای عمق ۲، درختی بشکل روبرو خواهیم داشت :



شکل 4

در این درخت وزن یالها نمایانگر طول مسیر و وزن برگ‌ها نشان دهنده احتمال آن مکان است.( احتمال آنکه فرد درحال فرار به آنجا بگریزد )  
در محاسبه این احتمالات ما ۲ معیار اصلی‌ را مور توجه قرار دادیم :

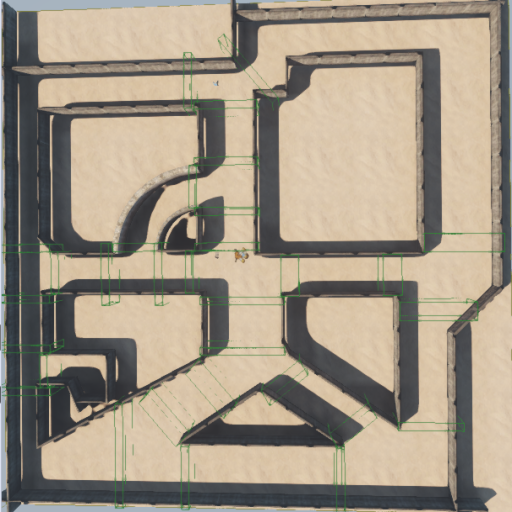
1**. میزان خطر گذرگاه ها :**

این معیار توسط طراح بازی ارزش دهی‌ میشود و به شرایط فیزیکی یک راس و عامل‌های از این دست بستگی دارد. به عنوان مثال، یک مکان که دارای آیتم‌هایی برای پنهان شدن بازیکن باشد، نسبت به یک مسیر که به بن‌بست ختم میشود، مناسب تر در نظر گرفته میشود و از ارزش بالاتری برخوردار خواهد بود.

2**. تاریخچه مسیر‌های انتخابی:**

در این قسمت، یک سیستم ارزش دهی جداگانه برای گذرگاه ها در نظر گرفته میشود که بر اساس مشاهدات عامل های تعقیب کننده در حین تعقیب و بر اساس انتخاب های فرد در دست تعقیب، ارزش دهی میشود. به عنوان مثال، اگر بازیکن چندبار در یک دوراهی از مسیر سمت راست برود و اینکار توسط عامل‌ها دیده شود، باعث میشود راس انتهای مسیر سمت راست از ارزش بیشتری نسبت به راس موجود در مسیر سمت چپ، برخوردار شود.

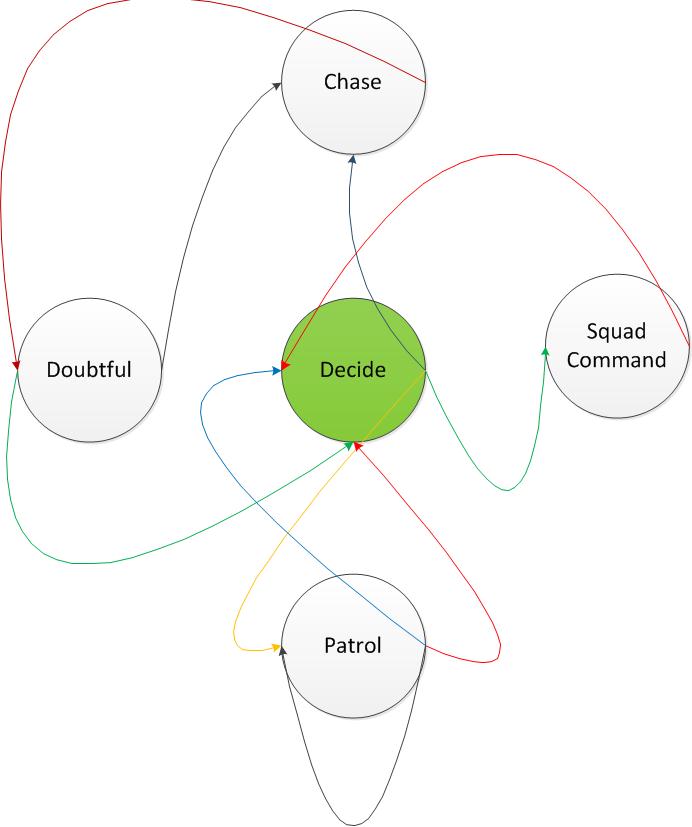
در زیر نمای نقشه ای که تیم ما آزمایشات خود را در آن انجام داده، در کنار گذرگاه های آن نمایش داده شده است :



شکل 5

هوش مصنوعی فردی:

سیستم هوش مصنوعی فردی با استفاده از ساختار Finite State Machine یا FSM ساخته شده است. در این سیستم هر عامل دارای چندین حالت می‌باشد که با توجه به شرایط و فرمان هایی که از سیستم هوش مرکزی می‌گیرد ، وارد یکی از این حالت ها می‌شود. در شکل رو‌به‌رو ما حالت های مختلف در نظر گرفته شده را می‌بینیم .

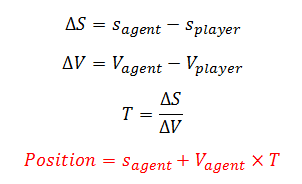


شکل 6

حالت تعقیب:

این حالت هنگامی رخ می‌دهد که عامل کاربر را مشاهده کرده و از طرف فرمانده دستوری نداشته باشد. در این حالت بسته به موقعیت بازیکن و عامل اگر عامل از بازیکن عقب تر باشد و هر دو در یک جهت حرکت کنند (عامل در حال تعقیب از پشت سر بازیکن ) عامل به مکانی که بازیکن را می‌بیند می‌رود ودر هر لحظه مکان بازیکن را به روز رسانی کرده و به فرمانده اطلاع می‌دهد.

در صورتیکه عامل از پشت سر بازیکن در تعقیب او نباشد (به طور مثال از رو به رو یا از طرفین) مکان احتمالی برخورد با بازیکن توسط فرمول زیر به دست آمده و عامل را به Position می‌فرستیم تا در یک زمان به بازیکن برسد و او را دستگیر کند.



در صورتیکه در حین مراحل تعقیب بازیکن از دید عامل ناپدید شد ، عامل وارد حالت مشکوک شده و در صورت مشاهده بازیکن به حالت تعقیب باز می‌گردد.

حالت مشکوک:

این حالت زمانی رخ می‌دهد که عامل در هنگام تعقیب دیگر موفق به دیدن بازیکن نشود. در این حالت ابتدا به آخرین جایی که بازیکن را دیده رفته و سپس در صورت مشاهده کردن بازیکن دوباره وارد حالت تعقیب می‌شود ، در صورت عدم مشاهده بازیکن ، مسیر روبه‌رو (در صورت وجود) یا یکی از مسیرهای ممکن را تا مدتی ادامه می‌دهد و در صورت عدم مشاهده بازیکن به حالت تصمیم گیری باز می‌گردد.

حالت گشت زدن:

در این حالت وظیفه‌ی عامل رفتن به نقاط خاصی در نقشه و انجام کار خاصی در آن مکان به صورت تکرار شونده است. برای اجرای این حالت ما باید از قبل برای عامل ها مکان هایی را در نقشه مشخص کرده باشیم ، برای مثال می‌توان گشت زنی در یک محله را با این سیستم بدین صورت شبیه سازی کرد که عامل از اولین نقطه‌ی داده شده شروع کرده و به صورت ترتیبی به تمام نقاط مربوطه رفته و کار متناظر آن نقطه را انجام می‌دهد. (برای مثال انداختن نور چراغ قوه در مغازه ها و بررسی امنیت آنها)

در صورت مشاهده مورد خاصی یا دریافت فرمانی از فرمانده وارد حالت تصمیم گیری می‌شود در غیر این صورت به گشت زدن خود می‌پردازد.

حالت دستور فرمانده:

این حالت مربوط به انجام دستورات فرمانده است و در هنگامی که فرمانی از فرمانده می‌رسد ، عامل وارد این حالت می‌شود. این حالت بر طبق قابلیت‌هایی که برای سیستم در نظر گرفته‌ایم ، دارای پیاده‌سازی های متفاوتی است ، که عمده ترین تفاوت آن‌‌ها فرمان های پشتیبانی شده توسط عامل است. در مدل ما به دلیل آنکه هدف نهایی محاصره‌ و دستگیری بازیکن است ، تنها فرمان ارسالی فرمانده به هر عامل دستور حرکت به نقطه‌ای خاص است.

برای اجرای این دستور فرمانده باید نقطه‌ای را که برای محاصره‌ی بازیکن بهتر است ، به عامل معرفی کند . سپس عامل به سمت آن نقطه حرکت می‌کند و در صورتیکه بازیکن را در طول پیمودن مسیر مشاهده نکند خود را به آن نقطه می‌رساند. سپس از فرمانده یا دستور سکون یا حرکت به نقطه ای جدید دریافت می‌کند که به دستور جدید عمل می‌کند.

حالت تصمیم گیری:

این حالت که اولین و اصلی ترین حالت عامل و به نوعی مغز عامل است ، بر اساس اولویت‌های تعیین شده در داخل آن تصمیم به رفتن به حالت دیگری می‌گیرد.  
اولویت‌های قرار گرفته در مدل ما برای محاصره بازیکن به صورت زیر است:

1. در صورت داشتن دستوری از فرمانده رفتن به حالت دستور فرمانده
2. در صورت دیدن بازیکن و تحت تعقیب بودن آن رفتن به حالت تعقیب
3. در صورت داشتن مسیر برای گشت زدن رفتن به حالت گشت زنی

همچنین اکثر حالت ها پس از انجام کارهای تعیین شده به حالت تصمیم گیری وارد می‌شوند و به این صورت حلقه‌ی اصلی تفکر عامل تشکیل می‌شود.

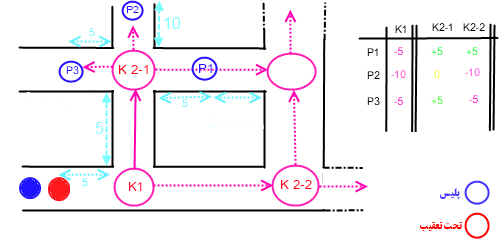
**هوش مصنوعی مرکزی:**

ما بمنظور کنترل و هماهنگی‌ بین نیروهای موجود از هوش جمعی‌ استفاده میکنیم. این به اصطلاح واحد مرکزی وظیفه دارد که اطلاعات و مشاهدات تمام نیروها را گرفته و پس انجام پردازش ها، دستورت لازم را به عامل ها ارسال می‌کند. واحد مرکزی در دو مقطع زمانی‌ به تصمیم گیری می‌پردازد:

1. هنگامی که بازیکن در حال تعقیب از یک گذرگاه عبور کند و در عین حال توسط یک عامل قابل مشاهده باشد

2. دریافت اطلاعات از عامل‌ها  
  
**حالت اول:**

حالت ۱ همانطور که در بخش‌های قبل معرفی‌ شد ، با ایجاد یک درخت پیشبینی‌ مسیر آغاز میگردد. عمق این درخت بر اساس معیار‌های گوناگونی از قبیل  تعداد عامل‌ها ، موقعیت عامل‌ها و ... تعیین میگردد. در مرحله اول ما درختی با عمق یک ایجاد میکنیم. بر اساس برگ این درخت، امکان بسته شدن مسیر توسط یکی‌ از نیروها برسی‌ میگردد. اگر بتوان عاملی بدین منظور پیدا کرد مشکل حل شده هست، در غیر این صورت عمق درخت یک واحد افزایش می‌یابد. ما بر اساس عمق درخت و وزن یالها و مکان فعلی‌ عامل‌ها **جدولی‌ مانند زیر** به وجود می‌آوریم .

  
  
در توضیح این جدول ، فرض کنید تمامی‌ افراد از جمله فرد در حال فرار، بتوانند با سرعت ثابت ۱ واحد بر ثانیه بدوند. در این صورت مدت زمانی‌ که برای پیمودن M واحد لازم است، برابر M ثانیه خواهد بود. با این فرض ، خانه های جدول بالا، اختلاف زمان رسیدن فرد در حال فرار به یکی از مکان های حاصل از برگ درخت پیشیبینی، نسبت به هر یک از عامل‌های P1 تا P3 را نشان میدهد. یعنی :

Δt = (Kx-) – (Kx-)

= yزمان رسیدن فرد درحال فرار به خانه

= y به خانهPزمان رسیدن عامل

همانطور که از شکل و جدول مشخص است، ما در مرحله اول سعی‌ می‌کنیم که در لایه‌ی اول به موفقیت برسیم. بدین منظور باید بدنبال عاملی باشیم که بتواند قبل از رسیدن بازیکن در حال گریز، خود را به مکان K1 برساند. ولی‌ از آنجا که برای این لایه تمام اختلاف زمان‌ها منفی‌ است، مشخص است که امکان چنین چیزی وجود ندارد. پس باید به برسی‌ لایه بعدی بپردازیم.

موفقیت در هر لایه منوط به بسته شدن تمام برگ های آن لایه می‌باشد که در مورد لایه دو ، بسته شدن مکانهای و مدنظر است. همینطور که در جدول مشخص است، اینکار با قرار دادن نیروی P1 در مکان و یکی‌ از دو نیروی P2 یا P3 در مکان قابل انجام است.

اگر در شرایطی در این لایه نیز به قادر به بستن راه‌ها نمیشدیم، میبایستی لایه بعدی مورد برسی‌ واقع میشد.

روند افزایش لایه ها زمانی‌ خاتمه میابد که یا در لایه فعلی‌ به موفقیت برسیم و بتوانیم تمامی‌ مکان‌های آن لایه را (برگ های آن لایه را) پوشش دهیم ویا مجموعه اختلاف زمان‌ها لایه فعلی‌ از لایه قبلی‌ کمتر شود.  
هم چنین بمنظور کاهش محاسبات و نتیجه بهتر، گسترش درخت بصورت ناحیه ای (Partial) صورت میپذیرد. به این معنا که اگر در یک لایه قسمتی‌ از مکان‌ها قابل پوشش و قسمتی‌ غیر قابل پوشش باشند ما تنها برسی‌ لایه بعد را از سمت برگ های غیر قابل پوشش ادامه میدهیم و گسترش بصورت ناحیه ای صورت میپذیرد که باعث کاهش چشمگیری در هزینه پردازش ها میگردد.

در رابطه با نحوه چیدمان عامل‌ها در مکان‌های، چند حالت ممکن است اتفاق بیافتد، که هر یک را جداگانه شرح میدهیم:

1. **تعداد عامل‌ها بیشتر یا برابر تعداد برگ ها باشد:**

این حالت به معنی‌ وجود عامل‌های اضافی می‌باشد. در این حالت ما از عامل‌ها با چند رویکرد متفاوت و بسته به مکان و فاصله ی آنها از فرد در حال گریز و فاصله با سایر برگ ها استفاده می‌کنیم.

**الف)** بستن مسیر از پشت (اینکار بمنظور پشتیبانی‌ عامل‌های اصلی بکار میرود)  
**ب)** افزایش عامل‌ها در مکان‌های محتمل تر ( با توجه به وزن برگ‌ها )

1. **تعداد** عامل‌ها **کمتر از تعداد برگ ها باشد:**

در این حالت که معمول تر از حالت قبلی است، سعی‌ بر آن است که یک عامل‌ به دو یا چند ناحیه‌‌ اختصاص داده شود. در این حالت یک عامل‌ بر اساس میزان احتمال برگ هایی که به وی اختصاص داده شده و فاصله وی از هر یک از آنها، در حد فاصل برگ ها قرار میگیرد.مثالی از انجام گیری این کار بدین شکل است که فرض کنید یک عامل داریم و دو نقطه باید بسته شود ، ما مدت زمان رسیدن بازیکن به هر کدام از این دو نقطه را محاسبه و سپس بر مبنای کمترین زمان ، عامل را در مسیر بین دو نقطه و در فاصله ای از نقطه‌ی نزدیک تر به بازیکن قرار می‌دهیم که اگر از طرف فرمانده متوجه شد که بازیکن وارد کدام مسیر شده است ، وقت برای رسیدن به آن نقطه را داشته باشد و خود را به آنجا برساند.