
بسم الله الرحمن الرحيم
الحمد لله رب العالمين الرحمن الرحيم مالك يوم الدين
اياك نعبد واياك نستعين اهدنا الصراط المستقيم
صراط الذين انعمت عليهم غير المغضوب عليهم ولا الضالين

دانشگاه صنعتي اميركبير
(پلي تكنيک تهران)
دانشكده مهندسي كامپيوتر و فناوري اطلاعات

پايان نامه كارشناسي ارشد
گرايش هوش ماشين و رباتيك

طراحي الگوريتم‌هاي تكاملي مبتني بر اتوماتاي سلولي براي محيط‌هاي پويا

نگارش :
وحيد نوروزي

استاد راهنما :
دكتر محمدرضا ميبدي

تابستان ۱۳۹۰

به نام مادر
بوسه‌اي بايد زد
دست‌هايي را
كه مي‌شويند
غبار خستگي روزگار را
و سيراب مي‌كنند روح تشنه را

به نام پدر
بوسه‌اي بايد زد
دست‌هايي را
كه مي‌تابانند نيرو را
و محكم مي‌كنند
استواري پايه‌هاي زيستن را

تقديم به پدر و مادر عزيزم به پاس محبت‌هاي بي‌دريغشان

سپاسگزاري

مراتب تشكر و قدرداني‌ام را نسبت به استاد ارجمندم، جناب آقاي دكتر ميبدي و همچنين دكتر علي هاشمي براي تمامي زحمات و راهنمايي‌هاي بي‌دريغشان ابراز مي‌دارم. از عموي عزيزم دكتر مسعود نوروزي و بويژه همسر محترمشان كه مرا در تمامي اين سال‌هاي تحصيل مورد لطف و حمايت قرار دادند بسيار سپاسگزارم. همچنين از دوستان عزيزم بويژه آقايان روزبه نوروزي و علي شريفي كه در اين راه مرا ياري و همراهي نمودند تشكر مي‌كنم.

چکیده

بسياري از مسائل بهينه‌سازي در دنياي واقعي پويا مي‌باشند. در اين مسائل پهنه جستجو وابسته به زمان است و مكان بهينه‌هاي محلي و سراسري در طول زمان ممكن است تغيير كنند. نمونه‌هايي از اين گونه مسايل در دنياي واقعي عبارتند از مدل‌سازي در شرايط متغير بازار، مسائل زمانبندي با منابع پويا و مسيريابي پوياي وسايل نقليه. در اين پايان‌نامه چندين الگوريتم تكاملي مبتني بر اتوماتاي سلولي، اتوماتاي يادگير سلولي و اتوماتاي سلولي فازي براي محيط‌هاي پويا ارائه مي‌شود كه همگي بر اساس ايده تقسيم‌بندي فضاي جستجو طراحي شده‌اند.

در ابتدا يک الگوريتم مبتني بر مدل اتوماتاي سلولي با نام الگوريتم تكامل تفاضلي سلولي پيشنهاد مي‌شود كه در آن از الگوريتم تكامل تفاضلي جهت جستجو در هر قسمت و همسايگي آن به صورت جداگانه استفاده مي‌گردد. يک مدل تكاملي نيز براي ساخت بردار كانديد در الگوريتم تكامل تفاضلي پيشنهاد مي‌شود كه در الگوريتم پيشنهاد شده كارايي بالاتري نسبت به مدل‌هاي تكاملي رايج از خود نشان مي‌دهد و همچنين موجب كاهش تعداد پارامترهاي الگوريتم تكامل تفاضلي نيز مي‌شود. جهت بهبود دنبال كردن بهينه‌ها پس از هر تغيير در محيط نيز، يک روش جستجوي محلي بهبوديافته پيشنهاد مي‌شود كه كارايي الگوريتم را در محيط‌هايي با دوره تغيير بالا در حد بسيار قابل توجهي بهبود مي‌دهد. روش‌هاي پيشنهادي با چندين روش تكاملي ديگر كه بالاترين كارايي‌ها را تاكنون بر روي تابع محک قله‌هاي متحرک از خود نشان داده‌اند مقايسه شده است و در اكثر محيط‌هاي پوياي مورد آزمايش كارايي بالاتري نسبت به آن‌ها از خود نشان مي‌دهند.

سپس با انجام تغييراتي در الگوريتم پيشنهادي پايه، نسخه ديگري از آن به نام الگوريتم كوه‌نورد پيشنهاد مي‌شود. در اين الگوريتم از طريق تشخيص محل قله‌ها و دامنه‌هاي اطراف آن‌ها، كارايي جستجو و استفاده از منابع را بهبود مي‌دهيم. در آزمايشات، اين نسخه پيشنهادي از كارايي بالاتري نسبت به نسخه پايه برخوردار است.

براي افزايش تطبيق‌پذيري روش‌هاي پيشنهادي نسبت به محيط‌هاي پوياي مختلف و همچنين كاهش نياز به تنظيم مقادير از طريق سعي و خطا، دو نسخه ديگر از الگوريتم تكامل تفاضلي سلولي پيشنهاد مي‌شود كه در آن‌ها مقادير پارامترهاي الگوريتم تكامل تفاضلي به صورت تطبيقي با استفاده از يادگيري تقويتي و سيستم استنتاج فازي مبتني بر قوانين تنظيم مي‌شوند. اين دو نسخه پيشنهادي، يكي بر مبناي مدل اتوماتاي يادگير سلولي و ديگري بر مبناي اتوماتاي سلولي فازي طراحي شده‌اند. هر دو روش مورد آزمايش قرار مي‌گيرند و نشان داده مي‌شود كه آن‌ها از كارايي و تطبيق‌پذيري قابل قبولي نسبت به محيط‌هاي مختلف برخوردارند.

كلمات كليدی: محيط‌های پويا، الگوريتم‌های تكاملی، الگوريتم تكامل تفاضلی، اتوماتای سلولی، اتوماتای يادگير سلولی، اتوماتای سلولی فازی

فهرست مطالب

فهرست شکل‌ها

فهرست جدول‌ها

فصل اول

مقدمه

۱. مقدمه

در اكثر مسايل دنياي واقعي نوعي عدم قطعيت وجود دارد. اين عدم قطعيت‌ها كه در محيط‌هاي واقعي انواع مختلفي دارد را مي‌توان به چهار دسته‌ي محيط‌هاي نويزي، توابع شايستگي تقريبي، پايداري و محيط‌هاي پويا تقسيم‌بندي نمود. يكي از عمده‌ترين و چالش‌برانگيزترين آن‌ها، محيط‌هاي پويا مي‌باشد كه در آن‌ها تابع شايستگي به زمان وابسته مي‌باشد و در نتيجه پاسخ بهينه مسائل با مرور زمان تغيير مي‌كند. بنابراين در اين گونه محيط‌ها الگوريتم‌هاي مورد استفاده جهت بهينه سازي بايد توانايي دنبال كردن بهينه را به صورت پيوسته داشته باشد.

يكي از رايج‌ترين روش‌هاي بهينه‌سازي، الگوريتم‌هاي تكاملي مي‌باشند كه از آن‌ها معمولا براي حل مسايل دشواري كه راه‌حل قطعي سريع براي آن‌ها وجود ندارد استفاده مي‌شود. تا كنون نيز الگوريتم‌هاي تكاملي بسياري براي بهينه‌سازي مسايل مختلف پيشنهاد شده است كه اكثر آن‌ها براي بهينه‌سازي در محيط‌هاي ايستا طراحي و آزمايش شده‌اند. اما اين الگوريتم‌هاي تكاملي متداول، در محيط‌هاي پويا كارايي مناسبي از خود نشان نمي‌دهند.

در محيط‌هاي پويا هر از چند گاهي تغييري در محيط روي مي‌دهد و طبيعتاً مكان بهينه‌ها ممكن است كه جابه‌جا شود كه در نتيجه پس از هر تغيير نياز به جستجوي دوباره بهينه‌ها مي‌باشد. اما الگوريتم‌هاي تكاملي متداول كه در هنگام روي دادن تغيير در محيط دچار همگرايي شده‌اند به علت فقدان تنوع كافي در جمعيت، قادر به جستجوي دوباره محيط جديد به صورت مناسب نيستند. در نتيجه الگوريتم‌هاي تكامليِ‌اي كه براي محيط‌هاي پويا طراحي مي‌شوند، از مكانيزم‌ها و روش‌هايي استفاده مي‌كنند تا اين مشكل را تا حد امكان رفع نمايند.

موضوع مهم ديگري كه در طراحي الگوريتم‌هاي تكاملي براي محيط‌هاي پويا در نظر گرفته مي‌شود، استفاده از نتايج جستجوي پيشين براي بهبود و سرعت بخشيدن به جستجو مي‌باشد. معمولا منظور از محيط‌هاي پويا، محيط‌هاي هستند كه تغييرات محيط در آن‌ها پس از هر تغيير خيلي زياد نباشد. زيرا اگر اين تغيير بسيار كلي و زياد باشد، در آن صورت نيازي به طراحي يک الگوريتم جديد نمي‌باشد و مي‌توان از همان الگوريتم‌هاي متداول استفاده نمود. تنها لازم است كه تغيير شناسايي شود و الگوريتم از نو راه‌اندازي شود. در صورتي كه اگر تغيير در محيط جزيي باشد، مي‌توان از جستجوي‌هاي قبلي استفاده كرد، زيرا محيط جديد شباهت‌هاي زيادي با محيط قبلي دارد. به طور مثال مكان‌هايي كه قبلا جستجو شده‌اند را كمتر جستجو كنيم و يا جستجوهاي بيشتري در اطراف بهينه‌هاي قبلي انجام دهيم.

تا كنون جهت بهينه سازي در محيط پويا با استفاده از پردازش‌هاي تكاملي پژوهش‌هاي زيادي صورت گرفته است. ميزان اين پژوهش‌ها بويژه در ساليان اخير رو به افزايش است و يكي از موضوعات مورد علاقه در زمينه الگوريتم‌هاي تكاملي محسوب مي‌شود. از دسته روش‌هاي پيشنهاد شده مي‌توان به روش‌هاي شروع از نو [۱, ۲] ، تطبيقي سازي جهش [۳-۶]، حافظه آشكار يا تلويحي [۷-۱۶] ، بهبود روش انتخاب [۱۷, ۱۸] و روش‌هاي چند جمعيتي [۱۹-۳۸] نام برد.

از بين اين دسته روش‌ها، الگوريتم‌هاي چند جمعيتي به علت كارايي بالايي كه از خود نشان داده‌اند، از محبوبيت بالاتري نسبت به ساير روش‌ها برخوردارند. علت آن توانايي بالاي روش‌هاي چند جمعيتي در حفظ تنوع و جستجوي همزمان چندين بهينه مي‌باشد. اخيرا نيز اكثر كارهاي موفق پيشنهاد شده در زمينه بهينه‌سازي در محيط‌هاي پويا در اين دسته جاي مي‌گيرند.

۱-۱ اهداف و ساختار پايان‌نامه

الگوريتم‌هاي تكاملي كه تا كنون براي بهينه‌سازي در محيط‌هاي پويا طراحي و پيشنهاد شده‌اند، همچنان داراي مشكلاتي از جمله عدم توانايي كافي در حفظ تنوع كافي و همچنين عدم استفاده مناسب از تجربيات جستجوهاي پيشين مي باشند. در نتيجه نياز به الگوريتمي مي‌باشد كه پس از تغيير در محيط قادر باشد كه تنوع لازم جهت جستجوي دوباره را حتي در صورت همگرايي داشته باشد و همچنين بتواند از اطلاعات جستجوهاي قبلي به صورت مناسب‌تر استفاده نمايد.هدف از اين پروژه طراحي الگوريتم‌هاي تكاملي جهت جستجو در محيط‌هاي پويا مي‌باشد كه تا حد ممكن اين مشكلات را كاهش دهد. ايده كلي نيز در اين پروژه تقسيم‌بندي فضاي جستجو به قسمت‌هاي جداگانه و انجام جستجو به صورت جداگانه در اين قسمت‌ها مي‌باشد. تاكنون نسخه‌هايي از اين رويكرد جهت بهبود كارايي الگوريتم‌هاي تكاملي به صورت توزيع شده در محيط‌هاي ايستا پيشنهاد شده است [۳۹, ۴۰] ولي براي محيط‌هاي پويا تنها در الگوريتم Cellular PSO [۴۱, ۴۲] كه توسط هاشمي و ميبدي پيشنهاد شده است، از آن استفاده شده است. اين الگوريتم كه سلولي نيز مي‌باشد، كارايي نسبتا خوبي از خود نشان داده است ولي تا حد مطلوب فاصله دارد. در اين پروژه سعي شده با استفاده از اين ايده، يک الگوريتم تكاملي با كارايي بالاتر پيشنهاد شود.

به علت توزيع‌شدگي جستجو كه ناشي از تقسيم‌بندي فضا مي‌باشد، الگوريتم توانايي بالايي در حفظ تنوع دارد كه اين امر در محيط‌هاي پويا بسيار مهم مي‌باشد. علاوه بر اين‌ها، اين امكان فراهم مي‌شود كه الگوريتم جستجو، تمركز بيشتري بر روي جستجوي محلي و استخراج داشته باشد زيرا ديگر نياز زيادي به جستجوي عمومي در هر منطقه محلي وجود ندارد. همچنين از طريق حفظ و

نگهداري تاريخچه جستجو در هر قسمت و ردوبدل كردن اطلاعات بين اين قسمت‌ها مي‌توان تا حد زيادي با هدفمند تر كردن جستجو از اتلاف منابع و جستجوي‌هاي تكراري جلوگيري نمود كه اين امر پيدا كردن و دنبال كردن بهينه‌هاي در حال حركت بين قسمت‌هاي مختلف را تسريع مي‌بخشد. هدف ديگري كه در اين پروژه دنبال مي‌شود، انتخاب و طراحي يک روش جستجوي تكاملي مناسب جهت انجام جستجوي مورد نياز در هر قسمت مي‌باشد.

رويكردي كه در اين پروژه براي نيل به اين اهداف مورد توجه مي‌باشد استفاده از اتوماتاي سلولي مي‌باشد. بدين صورت كه يک اتوماتاي سلولي بر روي اين تقسيم‌بندي منطبق خواهيم كرد كه وظيفه كنترل روند جستجو در هر قسمت و روابط بين قسمت‌هاي مختلف را بر عهده دارد. براي اين منظور مي‌بايست قوانين محلي مناسبي براي اتوماتاي سلولي پيدا كرد كه از طريق آن بتوان اطلاعات مناسبي را در بين سلول‌ها درباره جستجوهاي انجام شده و تغييرات محيط ردوبدل كرد. همچنين مي‌بايست تدابيري جهت حفظ تنوع در هنگام تغيير محيط جهت يافتن بهينه‌هاي جابه‌جا شده انديشيده شود. الگوريتم‌هاي تكاملي مختلفي مبتني بر اتوماتاي سلولي و يا اتوماتاي يادگير سلولي براي محيط‌هاي ايستا پيشنهاد شده است [۴۳-۴۸] ولي تاكنون از اتوماتاي سلولي در محيط‌هاي پويا تنها در همان روش Cellular PSO استفاده شده است.

هدف ديگري كه در اين پروژه دنبال خواهد شد استفاده از يادگيري تقويتي و همچنين منطق فازي براي ايجاد توانايي يادگيري در الگوريتم‌هاي پيشنهادي مي‌باشد. از اين طريق قدرت تطبيق‌پذيري الگوريتم‌هاي پيشنهادي را نيز در مواجهه با شرايط متفاوتي كه بر محيط‌هاي پويا حاكم است افزايش مي‌دهيم و همچنين نياز به تنظيم جداگانه الگوريتم براي محيط‌هاي مختلف را نيز كاهش مي‌دهيم.

در ادامه همين فصل به معرفي و تعريف مفاهيم اوليه مورد نياز در فصول آتي خواهيم پرداخت. در ابتدا به معرفي الگوريتم‌هاي تكاملي و تاريخچه آن‌ها مي‌پردازيم و روند كلي آن‌ها را به صورت مختصر شرح مي‌دهيم. با توجه به اينكه محيط‌هاي پويا نوعي از محيط‌هاي غيرقطعي محسوب مي‌شوند، در ابتدا انواع محيط‌هاي غيرقطعي را به همراه روش‌هاي كلي مواجهه با آن‌ها معرفي مي‌كنيم. سپس محيط‌هاي پويا را به همراه انواع روش‌هاي طبقه‌بندي آن‌ها معرفي خواهيم كرد. پس از آن معيارها و توابع محكي كه به طور معمول براي مقايسه روش‌هاي پيشنهاد شده در محيط‌هاي پويا به كار مي‌روند را معرفي مي‌كنيم. از بين اين توابع، تابع محک قله‌هاي متحرک به علت جامعيت و محبوبيت بيشتري كه نسبت به ساير توابع محک برخوردار است را به عنوان تابع محک براي ارزيابي و مقايسه كارايي الگوريتم‌هاي پيشنهادي در آزمايشات به كار خواهيم برد. در

فصل دوم نيز، پژوهش‌هاي انجام شده بر روي الگوريتم‌هاي تكاملي براي محيط‌هاي پويا را طبقه‌بندي كرده و توضيح مختصري در مورد روش‌هاي پيشنهاد شده تا به اكنون داده مي‌شود.

در ابتداي فصل سوم الگوريتم پايه را كه مبناي اوليه تمامي روش‌هاي پيشنهادي مي‌باشد را به طور كامل معرفي و شرح خواهيم داد. در اين روش پيشنهادي در ابتدا فضاي جستجو را تقسيم‌بندي نموده و سپس يک اتوماتاي سلولي هم‌اندازه با تقسيم‌بندي انجام شده بر آن منطبق خواهد شد. جستجو نيز به صورت محلي در هر منطقه و همسايگي آن به صورت همزمان و جداگانه انجام مي‌گيرد. هر سلول اين اتوماتا وظيفه كنترل روند جستجو در هر قسمت و همچنين تبادل اطلاعات بين قسمت‌هاي همسايه را بر عهده دارد. اين توزيع‌شدگي جستجو به حفظ تنوع الگوريتم و جستجوي مناسب محلي كمک فراواني مي‌كند. اين الگوريتم را كه بر اساس ايده تقسيم‌بندي فضاي جستجو و انطباق يک اتوماتاي سلولي طراحي شده است را الگوريتم تكامل تفاضلي سلولي (CellularDE) مي‌ناميم.

پس از آن براي بهبود روند بازيابي قله‌هاي گمشده پس از تغيير، يک روش جستجوي محلي مناسب پيشنهاد شده است كه كارايي الگوريتم را بويژه در دوره‌هاي تغيير بالا افزايش مي‌دهد. پس از شرح و توضيح كامل روند روش‌هاي پيشنهاد شده، آزمايشاتي جهت تنظيم مقادير پارامترهاي الگوريتم و همچنين بررسي تاثير اين پارامترها و ارتباط بين آن‌ها انجام شده است. آزمايش نهايي نيز كارايي روش‌هاي پيشنهادي را با چندين روش ديگر كه بر روي تابع محک قله‌هاي متحرک كارايي بالايي از خود نشان داده‌اند مورد ارزيابي قرار خواهيم داد.

در فصل چهارم با انجام تغييراتي در الگوريتم تكامل تفاضلي سلولي كه در فصل دوم پيشنهاد شده، نسخه ديگري از آن به نام الگوريتم تكامل تفاضلي سلولي كوه‌نورد[1] پيشنهاد مي‌شود. يكي از مشكلات الگوريتم تكامل تفاضلي سلولي اين بود كه در مناطق داراي قله، موجودات همگرا شده و ديگر نياز زيادي به جستجوي عمومي نمي‌باشد و بهتر است تا منابع را بر روي استخراج همان بهينه يافت شده متمركز كنيم. مشكل ديگر تجمع و جستجوي بي مورد در سلول‌هاي همسايه يک سلول داراي قله مي‌باشد و مي‌بايست با محدود كردن ظرفيت سلول‌هاي اطراف از اين اتفاق جلوگيري كرد. اين نسخه پيشنهادي با دستكاري متناسب ظرفيت سلول‌ها و تغيير به موقع روش جستجو در برخي سلول‌ها سعي در استفاده بهينه از منابع و بهبود جستجو دارد. در انتهاي فصل، در آزمايشاتي كارايي اين نسخه پيشنهادي با نسخه‌هاي قبلي و ساير الگوريتم‌ها مورد ارزيابي و مقايسه قرار داده شده است.

[1] Alpinist CellularDE

در فصل پنجم، سعي شده است با استفاده از دو روش يادگيري تقويتي و همچنين استنتاج فازي پارامترهاي الگوريتم تكامل تفاضلي سلولي را به صورت خودكار كنترل و تنظيم نماييم. اين امر براي محيط‌هاي پويا كه مدام در حال تغيير هستند و همچنين حالت‌ها و سناريوهاي مختلفي دارند مي‌تواند بسيار كمك‌كننده باشد. مزيت ديگر اين روش‌ها عدم نياز به تنظيم پارامترهاي الگوريتم تكاملي از طريق سعي و آزمايش براي هر محيط به صورت جداگانه، است. در اين فصل دو الگوريتم با نام‌هاي الگوريتم تكامل تفاضلي يادگير سلولي[1] و الگوريتم تكامل تفاضلي فازي سلولي[2] پيشنهاد شده است. الگوريتم اول بر مبناي مدل اتوماتاي يادگير سلولي طراحي شده است و با استفاده از يادگيري تقويتي سعي در يافتن مقدار مناسب براي پارامترها دارد و الگوريتم ديگر از سيستم استنتاج فازي جهت تنظيم مقدار پارامترها استفاده مي‌كند. در انتهاي فصل نيز آزمايشاتي جهت بررسي كارايي هر دو الگوريتم تطبيقي پيشنهاد شده انجام شده است.

در فصل ششم نيز با مرور كلي بر روي كارهاي پيشنهاد شده در پايان‌نامه و انجام نتيجه‌گيري كلي از آن‌ها، پيشنهاداتي نيز جهت كارهاي آينده پيشنهاد خواهد شد. سپس نيز مراجع به كار رفته در طول پايان‌نامه ذكر شده است. در فصل هشتم، واژه‌نامه لغات پاورقي به كار گرفته شده در اين پايان‌نامه به هر دو ترتيب حروف الفبايي فارسي و انگليسي درج شده است.

در فصل نهم، پيوست‌ها قرار دارد كه در ابتداي آن، مدل‌هاي اتوماتاي سلولي، يادگيري تقويتي و اتوماتاي يادگير سلولي را كه مبناي الگوريتم‌هاي پيشنهادي مي‌باشد را تعريف و شرح مي‌دهيم. در ادامه فصل نيز، الگوريتم تكامل تفاضلي كلاسيک را كه الگوريتم تكاملي اصلي مورد استفاده در مدل‌هاي پيشنهادي است را به همراه روند كلي آن شرح و توضيح خواهيم داد. در نهايت نيز كدهاي منبع شبيه‌سازي‌هاي انجام شده به همراه توضيحات، درج شده است.

۲-۱ جستجو و الگوريتم‌هاي تكاملي

نياز به جستجو در حل مسائل كاربردي امري غير قابل اجتناب و در عين حال دشوار است. به همين جهت تعداد زيادي از الگوريتم‌هاي جستجو با فلسفه‌ها و دامنه استفاده متفاوت به وجود آمده‌اند. اين الگوريتم‌هاي جستجو را مي توان به دو دسته كلي جستجوهاي كامل و جستجوهاي مكاشفه اي تقسيم كرد. فرق اساسي بين اين دو دسته در اين است كه در جستجوهاي كامل، تمام فضاي جستجو به طور كامل مورد جستجو و ارزيابي قرار مي گيرد تا جواب مورد نظر يافته شود، در

[1] Cellular Learning DE (CLDE)
[2] Fuzzy CellularDE (FCDE)

حاليكه در جستجوهاي مكاشفه اي تنها بخشي از فضا كه احتمال يافتن جواب در آن بيشتر است، مورد توجه قرار مي‌گيرد.

جستجوهاي مكاشفه‌اي به دو دسته الگوريتم‌هاي قطعي و غير قطعي تقسيم پذيرند. ويژگي اصلي الگوريتم‌هاي قطعي در اين است كه تحت شرايط يكسان، جواب‌هاي يكسان مي‌دهند. از جمله اين الگوريتم‌ها مي توان به تپه نوردي اشاره كرد. ايراد اساسي اين الگوريتم‌ها احتمال گير افتادن در كمينه‌هاي محلي مي‌باشد.

در مقابل، الگوريتم‌هاي مكاشفه‌اي غير قطعي با استفاده از احتمالات و جستجوهاي تصادفي، در شرايط يكسان جواب‌هاي متفاوتي بدست مي‌آورند. همينطور در صورت افتادن در كمينه‌هاي محلي، از آن‌ها مي‌گريزند. الگوريتم‌هاي مكاشفه‌اي غير قطعي را بر اساس تعداد جواب‌هايي كه در هر تكرار بررسي و ذخيره مي كنند به دو دسته تقسيم مي‌كنند. بعضي مانند تابكاري فلزات تنها يک جواب را در هر تكرار مورد بررسي قرار داده و ذخيره مي‌كنند. بعضي ديگر در هر تكرار دسته اي از جواب‌ها را ذخيره مي‌كنند، كه به اين الگوريتم‌هاي مبتني بر جمعيت مي‌گويند. دسته بزرگي از اين الگوريتم‌ها را الگوريتم‌هاي تكاملي تشكيل مي‌دهند.

الگوريتم‌هاي تكاملي دسته‌اي از الگوريتم‌هاي بهينه‌سازي تصادفي مي باشند كه در آن‌ها از قوانين تكاملي موجود در طبيعت به منظور بهينه‌سازي استفاده مي شود. اين الگوريتم‌ها معمولا براي حل مسائل بهينه سازي پارامتري كه ساير روش‌هاي فرمال از حل آن‌ها عاجزند مورد استفاده قرار مي گيرند.

اين الگوريتم‌ها براي اولين بار در سال ۱۹۶۰ توسط آقاي ريچنبرگ ارائه شد كه تحقيق وي در مورد استراتژي تكامل[1] بود. بعدها نظريه او توسط محققان زيادي مورد بررسي قرار گرفت. الگوريتم ژنتيک توسط جان هولند[2] در سال ۱۹۷۵ در دانشگاه ميشيگان ارائه شد. در سال ۱۹۹۲ جان كوزا[3] از الگوريتم ژنتيک براي حل و بهينه سازي مسائل مهندسي پيشرفته استفاده كرد و توانست براي اولين بار روند الگوريتم ژنتيک را به زبان كامپيوتر در آورد و براي آن يک زبان برنامه‌نويسي ابداع كند كه به اين روش برنامه‌نويسي، برنامه‌نويسي ژنتيک[4] گويند و نرم افزاري كه توسط وي ابداع گرديد به نرم

[1] Evolutionary Strategy
[2] John Holland
[3] John Koza
[4] Genetic Programming

افزار ليسپ[1] مشهور است كه هم اكنون نيز اين نرم افزار كاربرد زيادي در حل و بهينه سازي مسائل مهندسي پيدا كرده است.

الگوريتم‌هاي ژنتيكي كلاسيک به دليل سادگي و عدم نياز به معادلات ديفرانسيل پيچيده در حل مسائل پيچيده با فضاي جستجوي غير هموار مورد استفاده قرار مي گيرند. اين الگوريتم‌ها بر پايه انتخاب و بازتركيبي[2] جواب‌هاي مجاز مساله كار مي‌كنند. مجموعه جواب‌هاي توليد شده در هر دوره الگوريتم را يک نسل و هر كدام از جواب‌ها را يک ژنوم يا كروموزوم مي نامند. هر ژنوم تركيبي از متغيرهاي مساله است. در بعضي مسائل متغيرهاي مساله به هم وابسته و در بعضي ديگر از هم مستقل هستند. ولي همواره ارتباط و تبادل اطلاعات بين ژنوم‌ها از طريق انتخاب و تركيب مجدد ژنوم‌ها در يک نسل صورت مي پذيرد. اين جابجايي اطلاعات سبب مي شود تا جواب‌هاي جزئي[3] با يكديگر تركيب و احتمالا جواب‌هايي با كيفيت بالاتر بدست آيند.

محدوده كاري الگوريتم ژنتيک بسيار وسيع مي باشد و هر روز با پيشرفت روزافزون علوم و تكنولوژي استفاده از اين روش در بهينه سازي و حل مسائل بسيار گسترش يافته است الگوريتم ژنتيک را مي‌توان يک روش جستجوي كليناميد كه از قوانين تكامل بيولوژيک طبيعي تقليد مي‌كند. الگوريتم ژنتيک بر روي يک سري از جواب‌هاي مساله به اميد به دست آوردن جواب‌هاي بهتر قانون بقاي بهترين را اعمال مي كند. در هر نسل به كمک فرآيند انتخابي متناسب با ارزش جواب‌ها و توليد مثل جواب‌هاي انتخاب شده به كمک عملگرهايي كه از ژنتيک طبيعي تقليد شده‌اند ,تقريب‌هاي بهتري از جواب نهايي بدست مي‌آيد. اين فرايند باعث مي‌شود كه نسل‌هاي جديد با شرايط مساله سازگارتر باشد.

روند كلي الگوريتم‌هاي تكاملي بدين شكل مي‌باشد كه ابتدا يک جمعيت اوليه كه هر عضو آن مي‌توان جوابي براي مساله باشد به صورت تصادفي و با احتمال يكنواخت در بازه مجاز توابع مورد بررسي ساخته مي‌شوند و شايستگي هر موجود در اين مجموعه اوليه محاسبه مي‌گردد. سپس در هر نسل ابتدا با روش‌هاي مختلف انتخاب كه معرفي خواهند شد از بين جمعيت به تعداد فرزنداني كه نياز مي‌باشد افرادي را انتخاب مي‌كنيم كه چند برابر تعداد جمعيت مي‌باشد. معمولا در اين مرحله براي انتخاب از روش احتمال يكنواخت استفاده مي‌شود.

---

[1] LISP
[2] Recombination
[3] Partial Solution

پس از اين انتخاب با يک احتمال که احتمال بازترکيبي[1] ناميده مي‌شود بر روي افراد انتخاب شده که به صورت تصادفي نيز جفت شده‌اند عمل بازترکيبي انجام مي‌شود. پس از مرحله بازترکيبي بر روي تمامي‌فرزندان جهش صورت مي‌گيرد. البته اين جهش بر روي تمام ژن‌ها انجام نمي‌گيرد بلکه بر روي هر ژن با يک احتمال مشخص که احتمال جهش ناميده مي‌شود عمل جهش صورت مي‌گيرد. پس از اين مراحل با توجه به نوع مجموعه اي انتخاب ، تصميم گرفته مي‌شود که والدين نيز به فرزندان اضافه مي‌گردد يا خير. در نهايت با استفاده از يک روش مشخص شده براي انتخاب بازماندگان عمل انتخاب صورت مي‌گيرد و به اندازه جمعيت از بين فرزندان توليد شده انتخاب انجام مي‌شود. سپس افراد انتخاب شده جايگزين جمعيت قبلي شده و الگوريتم ادامه داده مي‌شود.

۱-۳ محيط‌هاي غيرقطعي

الگوريتم‌هاي تکاملي اغلب مي‌بايست در حضور دامنه وسيعي از ابهامات و عدم قطعيت‌ها در مسائل حقيقي بهينه سازي عمل کنند. به صورت کلي عدم قطعيت در پردازش تکاملي به چهار دسته تقسيم مي‌شود [۴۹]. اول، ممکن است تابع شايستگي تحت تاثير نويز باشد. دوم، متغيرهاي طراحي و يا پارامترهاي محيطي ممکن است بعد از بهينه سازي تغيير کنند و کيفيت پاسخ بهينه معين شده بايد در مقابل تغييرات محيط و انحراف از نقطه بهينه مقاوم[2] باشد. سوم، تابع شايستگي تخمين زده شده است، به اين معني که تابع شايستگي تحت تاثير خطاي تقريب قرار دارد. چهارم نيز اينکه پاسخ بهينه مسائل با مرور زمان تغيير مي‌کند و بنابراين الگوريتم بهينه سازي بايد توانايي دنبال کردن بهينه را به صورت پيوسته داشته باشد. در تمام موارد فوق محاسبات افزونه‌اي بايد صورت گيرد تا الگوريتم تکاملي بتواند همچنان به صورت شايسته عمل کند.

۱-۳-۱ نويز

در اين گونه مسايل، ارزيابي شايستگي تحت تاثير نويز قرار دارد. نويز در ارزيابي شايستگي ممکن است از عوامل مختلفي چون خطاي اندازه گيري حسگرها و شبيه سازي‌هاي تصادفي ناشي مي‌شود. به صورت رياضي، يک تابع شايستگي نويزي مي‌تواند به صورت زير توصيف شود:

$$F(X) = \int_{-\infty}^{\infty} [f(X) + z]\, p(z)\, dz = f(X), z \sim N(0, \sigma^2) \qquad (۱-۱)$$

[1] Crossover
[2] Robust

هنگامي‌كه X برداري از پارامترها مي‌باشد كه مي‌توانند به وسيله الگوريتم تغيير كنند (اغلب با عنوان متغيرهاي طراحي شناخته مي‌شوند)، $f(X)$ يک تابع شايستگي مستقل از زمان است، z نويز اضافه شده مي‌باشد، كه اغلب يک نويز نرمال با ميانگين صفر و واريانس σ^2 تعيين مي‌شود. بايد توجه شود كه نويزهايي با توزيعي غير از گوسي نيز مانند نويز با توزيع كوشي[1] نيز مورد توجه قرار گرفته‌اند. تفاوت كيفي در كارايي يک الگوريتم استراتژي تكاملي در حضور نويز گوسي و كوشي مشاهده نشده است. ايده‌ال اين است كه الگوريتم تكاملي بايد روي تابع شايستگي مورد نظر يعني F(X) كار كند و به دليل حضور نويز گمراه نشود. هرچند، در طي بهينه سازي، تنها مقدار شايستگي قابل اندازه گيري مقدار تصادفي $f(X)+z$ است. بنابراين، در عمل، تابع مورد انتظار در رابطه (۱-۱) اغلب با ميانگين مجموع تعدادي نمونه تصادفي تخمين زده مي‌شود :

$$\hat{F}(X) = (1/N)\sum_{i=1}^{N}[f(X) + z_i] \qquad (۲-۱)$$

كه در آن N تعداد نمونه‌ها و $\hat{F}(X)$ تخميني از F(X) = f(X) است .

يک مثال خوب بهينه سازي تكاملي ساختار شبكه‌هاي عصبي به وسيله روش كد كردن غير مستقيم است. در اين موارد ارزيابي‌هاي شايستگي ژنوتيپ[2] كاملا نويزي است. مقادير متفاوت شايستگي براي يک ژنوتيپ به علت مقداردهي اوليه تصادفي وزن‌ها و چند قله‌اي بودن تابع وجود دارد. اداره كردن نويز در ارزشيابي‌هاي شايستگي اغلب از مهمترين مسائل در رباتيكز تكاملي محسوب مي‌شود. روش‌هايي كه در ادامه اين بخش توصيف مي‌شوند براي كاهش نويز در اين گونه توابع پيشنهاد شده‌اند.

۱-۳-۱-۱ ميانگين گيري آشكار[3]

يک روش معمول براي كاهش اثر نويز تخمين زدن شايستگي به وسيله ميانگين‌گيري روي تعدادي نمونه گرفته شده در زمان‌هاي متفاوت است، يا به صورت اختصار ميانگين گيري زماني[4]. افزايش تعداد نمونه‌ها معادل است با كاهش واريانس تابع تخمين زده شده. نمونه‌برداري از شايستگي يک فرد در N زمان انحراف معيار را با فاكتور $\sqrt{N}$ كاهش مي‌دهد. در يک روش ساده، تعداد نمونه‌ها براي هر عضو از پيش معين و ثابت است. از آنجايي كه هر ارزيابي شايستگي[5] ممكن است

---

[1] Cauchy
[2] Genotype
[3] Explicit Averaging
[4] Averaging over Time
[5] Fitness Evaluation

خيلي هزينه بر باشد، اين مناسب است كه تعداد ارزيابي‌هاي شايستگي را تا حد امكان بدون كم كردن كارايي كاهش دهيم. تعداد ارزيابي‌هاي شايستگي نيز مي‌تواند به صورت تطبيقي در هنگام اجرا تنظيم شود، و دو روش تطبيقي ارائه كردند: كاهش تعداد ارزيابي‌هاي شايستگي با تعداد مراحل الگوريتم و ديگري تعداد ارزيابي‌هاي شايستگي بيشتر براي افراد با واريانس تخمين زده ي بيشتر.

نظر به اينكه كاهش تعداد ارزيابي‌هاي شايستگي موفقيتي در جهت كاهش ميانگين هزينه محاسباتي در زمان مي‌باشد، يک روش جايگزين محاسبه شايستگي با ميانگين‌گيري مكاني[1] مورد محاسبه مي‌باشد. يک فرض ضمني براي جايگزيني ميانگين گيري در زمان به وسيله ميانگين‌گيري در مكان اين است كه نويز در همسايگان ويژگي‌هايي مشابه دارد و در واقع پهنه جستجو[2] مربوط به تابع شايستگي به صورت محلي هموار است.

۱-۳-۱-۲ ميانگين گيري ضمني [3]

از آنجاييكه نواحي اميد بخش‌تر از نظر شايستگي در فضاي جستجو در الگوريتم تكاملي سهم بيشتري در جمعيت دارند و به صورت مكرر توليد مثل در اين نواحي صورت مي‌گيرد، و اغلب تعدادي عضو بسيار مشابه در جمعيت وجود دارد، هنگامي‌كه جمعيت بزرگ باشد، احتمالا تاثير نويز در محاسبه شايستگي يک عضو به وسيله يک عضو مشابه آن جبران مي‌شود. در نتيجه يک روش ساده براي كاهش تاثير نويز در جمعيت استفاده از اندازه بزرگ جمعيت است. مي‌توان نشان داد كه هنگامي‌كه اندازه جمعيت بي نهايت باشد، انتخاب مناسب تحت تاثير نويز قرار نمي‌گيرد. الگوريتم‌هاي ژنتيک با يک جمعيت محدود مورد مطالعه قرار گرفته و نمايان كرده است كه افزايش اندازه جمعيت تاثير نويز گوسي را بر انتخاب به روش تابع بولتزمن[4] كاهش مي‌دهد.

يک سوال طبيعي اين است كه آيا ميانگين گيري آشكار به شكل ارزيابي شايستگي مكرر يا ميانگين گيري ضمني به شكل افزايش اندازه جمعيت موثرتر است، با ثابت نظر گرفتن تعداد دفعات محاسبه تابع شايستگي در هر مرحله از الگوريتم. نتايج متناقضي در تحقيق‌هاي متفاوت در اين باره گزارش شده است. براي مثال از مطالعه روي الگوريتم ژنتيک اين نتيجه گرفته شده كه افزايش اندازه جمعيت از افزايش اندازه ارزيابي شايستگي بهتر عمل مي‌كند.

---

[1] Averaging over Space
[2] Landscape
[3] Implicit Averaging
[4] Boltzmann

۱-۳-۱-۳ اصلاح روش انتخاب

تعدادي از پژوهشگران اصلاح روش انتخاب را به عنوان راهي براي مقابله با نويز پيشنهاد مي‌كنند. يک مثال آن، تعيين يک آستانه در طي انتخاب در يک الگوريتم تكاملي است. بدين صورت كه يک فرزند در انتخاب پذيرفته خواهد شد اگر و تنها اگر شايستگي آن حداقل به اندازه يک آستانه معين از پيش تعيين شده از والدينش بهتر باشد.

۱-۳-۲ پايداري و مقاومت[1]

متغيرهاي طراحي تحت تاثير اختلال قرار دارند يا بعد از اين كه پاسخ بهينه تعيين مي‌شود تغيير مي‌كنند. بنابراين يک نياز معمول اين است كه هنگامي كه متغيرهاي طراحي كمي‌تغيير مي‌كنند به عنوان مثال بر اثر خطاي مجاز توليد صنعتي پاسخ‌ها همچنان قابل قبول باشند. به چنين پاسخ‌هايي مقاوم گويند. براي جستجوي اين گونه پاسخ‌ها الگوريتم تكاملي بايد روي تابع شايستگي مورد انتظار مبني بر توزيع احتمالاتي $P(\delta)$ از احتمال اختلال δ، كه معمولا به صورت مستقل مقدار دهي مي‌شوند و داراي توزيع نرمال است:

$$F(X) = \int_{-\infty}^{\infty} f(X+\sigma)\ p(\sigma)\ d\sigma \qquad (۳-۱)$$

كه در آن $F(X)$ به طور كلي تابع شايستگي موثر ناميده مي‌شود. از آنجايي كه يک فرم بسته تحليلي از تابع شايستگي موثر معمولا قابل دسترسي و محاسبه نيست، معمولا با مجموع مونت كارلو[2] بدين شكل تقريب زده مي‌شود:

$$\hat{F}(X) = (1/N) \sum_{i=1}^{N} [f(X+\sigma_i) \qquad (٤-۱)$$

توجه كنيد كه شباهت بسياري بين اين رابطه و رابطه تقريبي كه براي حضور نويز بيان شد وجود دارد و به راستي دو مورد بسيار با يكديگر مرتبط هستند. هر چند تفاوت‌هاي مفهومي بسياري نيز با يكديگر دارند. در مورد نويز، به صورت كلي نويز بر مقدار شايستگي اثر مي‌گذارد و هنگامي‌كه براي جستجو براي پاسخ مقاوم صورت مي‌گيرد، ابهام روي متغيرهاي طراحي است. در نتيجه حتي اگر δ يک توزيع نرمال با ميانگين صفر باشد مقدار موثر شايستگي $F(X)$ وابسته به فرم $f(X)$ در نقطه X است و برابر با $f(X)$ است تنها اگر $f(X)$ خطي باشد. نتيجه اين كه يک پاسخ بهينه مقاوم

[1] Robustness
[2] Monte Carlo

لزوما یک پاسخ بهینه براي f(X) نیست، اما همیشه یک تعادل[1] مابین کیفیت و خاصیت مقاومت پاسخ وجود دارد. اگر نویز غیر قابل اجتناب باشد شایستگي یک فرد به طور دقیق قابل محاسبه نیست. از سوي دیگر در جستجو براي پاسخ مقاوم شایستگي هر فرد به صورت دقیق مشخص است و ابهام به صورت عمومي تنها بعد از اینکه بهینه سازي به پایان مي‌رسد بروز مي‌کند. جستجو براي پاسخ بهینه و مقاوم از مفاهیم عمده در دامنه وسیعي از کاربردهاي جهان حقیقي است، همچون بهینه سازي طراحي و زمانبندي تولید کارگاهي[2] است .

مقاومت یک پاسخ بهینه معمولا مي‌تواند از دو منظر مورد بحث قرار گیرد :

۱) پاسخ بهینه نسبت به تغییرات کوچک متغیرهاي طراحي حساس نباشد.
۲) پاسخ بهینه نسبت به تغییرات کوچک پارامترهاي محیطي حساس نباشد.

۳-۳-۱ تقریب شایستگي

هنگامي‌که محاسبه تابع شایستگي بسیار هزینه بر باشد، یا یک تابع شایستگي تحلیلي قابل محاسبه قابل دسترسي نباشد، تابع شایستگي اغلب براساس داده‌هاي تولید شده در آزمایشات و شبیه سازي‌ها تقریب زده مي‌شود. تابع شایستگي تقریب زده شده معمولا با عنوان ابرمدل[3] شناخته مي‌شود. چنانچه در [۵] پیشنهاد شده است ابر مدل معمولا به همراه تابع شایستگي اصلي مورد استفاده قرار مي‌گیرد.

اگر از تابع شایستگي اصلي استفاده شود $F(X) = f(X)$ (۵-۱)

اگر از ابر مدل استفاده شود $F(X) = f(X) + E(X)$ (٦-۱)

که در آن E(X) خطاي تقریب ابر مدل است. تفاوت اصلي میان تابع شایستگي نویزي و تابع شایستگي تخمین زده شده این است که خطاي با میانگین صفر در تابع شایستگي تخمین زده شده زماني که یکبار ابر مدل ساخته شود قطعي و مشخص است. بنا براین خطا با ارزیابي مجدد شایستگي با استفاده از تخمین تابع شایستگي کاهش پیدا نمي‌کند. درعوض خطا باید با استفاده از یک تابع شایستگي صحیح به جاي تقریب محاسبه شود. یافتن تعادل صحیح بین تابع تقریب کم هزینه ولي همراه با خطا و محاسبه دقیق اما پر هزینه تابع دقیق شایستگي یک چالش محسوب مي‌شود.

---

[1] Trade off
[2] Job Shop Scheduling
[3] Meta Model

۱-۳-۴ تابع شايستگي متغير با زمان (محيطهاي پويا[1])

در اين گونه مسايل تابع شايستگي در هر نقطه از زمان مشخص و قطعي است اما وابسته به زمان است. بدين صورت كه :

$$F(X) = f_t(X) \qquad (۱-۷)$$

در نتيجه، پاسخ بهينه نيز با زمان تغيير خواهد كرد. بنا براين الگوريتم تكاملي بايد اين توانايي را داشته باشد كه به صورت پيوسته تغييرات پاسخ بهينه را به نحوي دنبال كند كه نيازي به شروع دوباره روند بهينه‌سازي نباشد. چالش پيش رو استفاده مجدد از اطلاعات محيطهاي قبل از تغيير براي تسريع بهينه سازي بعد از بروز تغيير در محيط است. اكثر توجه ما نيز به اين گونه محيطها معطوف خواهد بود.

۱-٤ محيطهاي پويا

ممكن است كه تمامي مسايل تغيير پذير با زمان به عنوان مسايل پويا در نظر گرفته شوند، هر چند از ديدگاه الگوريتم‌هاي تكاملي تمامي‌اين مسايل به يک اندازه جذابيت ندارند. در اين قسمت ما توجه خود را به مسايلي معطوف مي‌كنيم كه پهنه جستجوي تابع شايستگي قبل و بعد از تغيير داراي شباهت‌هاي قابل استخراج زيادي باشند. اگر مساله تقريبا به صورت كامل تغيير كند بدون هيچ گونه ارجاع و شباهتي به گذشته خود ، بايد به عنوان يک سري مسايل مستقل از هم مورد توجه قرار گيرد و به طور كامل از ابتدا حل شود.

با اين حال حتي اگر ما ديدمان را نسبت به مسايل پويا همان گونه كه ذكر شد نيز محدود كنيم تمامي مسايل يكسان نيستند و تفاوت در پويا بودن محيطهاي متفاوت احتمالا به روش‌هاي بهينه سازي متفاوتي احتياج خواهند داشت.

۱-٤-۱ معيارهاي طبقه بندي محيطهاي پويا

در اين قسمت معيارهايي را كه در طبقه‌بندي محيطهاي پويا به ما كمک مي‌كنند مطرح مي‌كنيم :

- فركانس تغييرات

[1] Dynamic Environments

محيط اغلب چگونه تغيير مي‌كند (از تغييرات بسيار كم و گسسته تا تغييرات پيوسته)؟ و يا مهمتر اينكه الگوريتم چقدر زمان براي به وجود آوردن يک پاسخ مطابق با محيط دارد؟ از آنجايي كه بررسي اين مطلب خيلي به سخت افزار و جزييات پياده سازي بستگي دارد و از آنجايي كه تعداد دفعات محاسبه تابع شايستگي معيار تعيين زمان در الگوريتم‌هاي تكاملي است پس ميانگين تعداد محاسبه شايستگي بين دو تغيير معيار مناسبي محسوب مي‌شود.

- شدت تغييرات

شدت تغييرات سيستم در چه حدي است؟ فقط به يک تغيير جزئي است يا كاملا به وضعيت جديد را موجب مي‌شود، اين به صورت عمده با فاصله ژنوتيپي پاسخ بهينه گذشته با پاسخ بهينه جديد تعيين مي‌شود اما ممكن است از جنبه‌هاي ديگري نيز مورد توجه قرار گيرد مثلا آيا مي‌توان از پاسخ بهينه قبلي با يک تپه نوردي ساده به پاسخ بهينه جديد رسيد و يا احتمال اينكه پاسخ بهينه جديد با يک جهش روي پاسخ بهينه قبلي حاصل مي‌شود يا خير، بينش بيشتر مي‌تواند با توجه كردن به ميانگين تغييرات روي هر نقطه از فضاي جستجو حاصل شود و ممكن است با توجه بيشتر به نقاط با شايستگي بيشتر صورت گيرد.

- پيش‌بيني پذيري تغييرات

آيا طرح و الگويي در تغييرات وجود دارد يا كاملا تصادفي هستند؟ آيا اين امكان وجود دارد كه جهت ، شدت يا زمان تغييرات بعدي از روي تغييراتي كه تا كنون رخ داده پيش بيني شود؟

- طول دوره/ دقت دوره

آيا پاسخ بهينه به مكانهاي قبلي و يا حداقل به نزديكي آنها باز مي‌گردد؟ و چگونه باز مي‌گردد؟ يک معيار مناسب تعداد حالت‌هاي محيطي ما بين وقوع پي در پي يک حالت با خودش مي‌باشد. اگر حالت جديد دقيقا همان حالت نباشد و انحراف كوچكي داشته باشد آنگاه فاصله بين حالت جديد و حالت قبلي مشابه با آن مهم مي‌باشد. طول و دقت دوره معين مي‌كند كه آيا نگهداري حالت‌هاي قبلي در حافظه مي‌تواند مفيد باشد يا خير.

از آنجايي كه محاسبه برخي از ويژگي‌هاي فوق دشوار به نظر مي‌رسد ممكن است كه مقايسه مسايل پوياي متفاوت دست يافتني نباشد اما حداقل پارامترهاي فوق مي‌توانند براي تعيين يک الگوريتم تكاملي ويژه براي يک مساله و بررسي كيفي آن الگوريتم مناسب باشند.

علاوه بر ويژگي‌هاي محيطي ممكن است براي طراحي يک الگوريتم تكاملي توجه به جنبه‌هاي زير نيز مفيد باشد :

- رويت پذير بودن تغيير

رخدادهاي مربوط به يک تغيير براي سيستم به صورت واضح مشخص هستند يا بايد شناسايي شوند؟

- لزوم تغيير بازنمايي

آيا بازنمايي تحت تغيير قرار مي‌گيرد به دليل اينکه مساله از نظر بعد دچار تغيير شده است؟

- نوع تغيير

تغيير در چه عناصري از مساله مي‌باشد؟ تابع بهينه سازي ، نمونه مساله و يا در محدوديت‌ها؟

به نوعي ديگر نيز مي‌توان تغييرات در محيط‌هاي پويا را به دسته‌هاي زير طبقه بندي کرد :

- ثابت : تغييرات يکسان در هر پريود
- ايستا : بدون تغيير
- دوره‌اي : بازگشت به حالت‌هاي قبلي به صورت متناوب
- همگن : تمام پهنه‌ها جستجو به صورت متصل حرکت مي‌کنند
- تعويضي : پاسخ بهينه از يک قله در پهنه جستجو به قله ديگري جهش مي‌کند

تشخيص تغيير در محيط

پيش از توانايي پاسخ به تغييرات محيط ابتدا بايد تشخيص داده شود که تغيير در محيط واقعا رخ داده است. براي بسياري از کاربردها، تغييرات مي‌توانند به صورت آشکار براي سيستم شناخته شده باشند، براي مثال هنگامي‌که يک روند جديد به صف روندهاي که بايد براي اجرا زمان بندي شوند اضافه مي‌شود. اما در برخي ديگر از کاربردها مانند هنگامي‌که کيفيت مواد اوليه تغيير مي‌کند، چنين تغييراتي بايد توسط سيستم شناسايي شوند.

يکي از شاخص‌هاي عمومي براي تشخيص تغيير در محيط، کاهش کارايي جمعيت و يا ميانگين زماني بهترين کارايي الگوريتم ژنتيک، مي‌باشد. اگر چه فرض بر آن است که يک تغيير در محيط به طور کم و بيش موجب کاهش کارايي راه‌حل‌هاي پيشين مي‌شود که البته اين امر الزامي نيز نمي‌باشد. به طور مثال مي‌توان در هر نسل شايستگي تعدادي از موجودات را دوباره محاسبه نماييم و در صورتي که حداقل يکي از مقادير شايستگي تغيير کرده باشند تغيير در محيط شناسايي مي‌شود.

راه ديگر استفاده از يک ماژول اعتبار سنجي مي‌باشد که کارايي الگوريتم تکاملي را ارزيابي مي‌کند و زماني که از حدود خاصي تخطي شد الگوريتم را دوباره از نو راه‌اندازي مي‌کند زيرا محيط آن قدر تغيير کرده است که کارايي الگوريتم تکاملي به هيچ وجه راضي کننده نيست. البته هيچ اشاره اي تا کنون به روش و چگونگي ساخت چنين ماژولي نشده است.

در بعضي از ديگر رويکردها به طور صريح يک مدل از محيط ساخته مي‌شود و به طور مرتب محيط واقعي با اين مدل مقايسه مي‌شود. اگر پاسخي که مدل پيش بيني مي‌کند با پاسخي که از محيط واقعي دريافت مي‌شود بسيار با يکديگر متفاوت باشند، به اين نتيجه خواهيم رسيد که محيط تغيير نموده است. در آن صورت مدل بروز مي‌شود و الگوريتم از نو راه‌اندازي مي‌گردد.

۱-۵ مسائل محک[1] جهت ارزيابي

تا به حال، دهها نوع متفاوت از مسائل بهينه سازي پويا براي آزمايش الگوريتم‌هاي تکاملي مورد استفاده قرار گرفته‌اند. از توابع ساده رياضي تا انواع مسائل زمانبندي براي کاربردهايي در مسائل مصنوعي شبيه به واقعيت. هر چند، بيشتر اين مسائل آزمايشي بسيار خاص هستند و مورد علاقه عمومي براي اکثريت خوانندگان و پژوهشگران نيستند. بنابراين توجه خود را به يک زير مجموعه از مسائل که نهايتا ممکن است از يک مجموعه از مسائل آزمايشي متداول باشند محدود مي‌کنيم.

جنبه‌هاي که در ادامه آمده براي مناسب بودن يک مجموعه از مسائل آزمايشي مورد توجه قرار مي‌گيرند:

- بايد تغيير دادن تعداد زيادي از متغيرهاي محيطي ممکن باشد.
- اين مسائل بايد براي پياده سازي و توصيف ساده باشند.
- بايد براي تحليل ساده باشند.
- بايد از نظر محاسباتي کارآمد باشند.
- بايد بتوان مثال‌هايي از مسائل حقيقي معادلشان را برشمرد.

در اينجا چند مساله انتخاب شده که کم و بيش جنبه‌هاي ذکر شده را دارا مي‌باشند را شرح مي‌دهيم. برخي از اين مسايل مانند تطبيق بيت‌هاي پويا مخصوص محيط‌هاي گسسته مي‌باشند و برخي نيز مانند کوله‌پشتي متغير با زمان و زمانبندي کارها در هر دو حالت پيوسته و گسسته قابل

[1] Benchmark

تعريف مي‌باشند. برخي توابع نيز مانند مساله قله‌هاي نوساني، قله‌هاي متحرک، سهمي‌متحرک در اصل به صورت پيوسته مي‌باشند که مي‌توان آن‌ها را گسسته کرد که البته موجب کاهش دقت مي‌شود.

۱-۵-۱ تطبيق بيت‌هاي پويا[1]

در مساله ايستاي بيشترين يک[2]، الگوريتم‌هاي تکاملي بايد تعداد يک‌هاي رشته کروموزوم‌ها[3] را حداکثرسازي کند، اين مساله به خوبي در مقالات مورد آزمايش قرار گرفته است. تطبيق بيت‌ها يک نوع ديگر از همين مساله است به صورتي که شايستگي افراد تعداد بيت‌هاي مطابق با يک الگو يا قالب معين مي‌باشد. اگر قالب با زمان تغيير کند، مي‌گوييم مساله پويا است. اين مساله براي تعريف نسبتاً آسان است، براي آناليز آسان است و به ويژه براي بازنمايي دودويي مناسب است. فرکانس تغييرات مي‌تواند به روش سرراستي تنظيم شود. شدت تغييرات نيز مي‌تواند با تصميم گيري در مورد اينکه چه تعداد از بيت‌هاي قالب در هر تغيير مي‌توانند عوض شوند تنظيم شود. همچنين مساله به صورت دلخواه توسط افزايش رشته کروموزوم مقياس‌پذير است. با تکرار آشکار يک دنباله مشخص از الگوها، جنبه‌هايي مانند قابل پيش بيني بودن و طول دوره مي‌توانند به خوبي ترکيب شوند. محدوديت‌هاي اين مساله تک قله‌اي بودن مساله (يک تپه نوردي ساده احتمالا کاراتر از يک الگوريتم تکاملي عمل مي‌کند) و محدوديت ديگر اين است که اين مساله محدود به بازنمايي دودويي است. تاکنون در مقالات مختلف از ۱۰ بيت تا ۵۰ بيت مورد استفاده قرار گرفته است.

۱-۵-۲ کوله‌پشتي متغير با زمان

يکي از معمول‌ترين توابع محک که تا به حال مورد استفاده قرار گرفته‌اند مساله کوله‌پشتي با يک حد وزن مجاز متغير با زمان مي‌باشد. اين مساله ويژگي خاصي دارد که اگر حد وزن کاهش داده شود. اغلب پاسخ‌هاي قبلي غير معتبر مي‌شوند چنانچه پاسخ‌هاي قديمي را از اساس ريشه کن مي‌کند و الگوريتم تکاملي را وادار به جستجو براي يافتن پاسخ‌هاي جديد مي‌کند.

بعلاوه در بيشتر مقالات حد وزن مجاز فقط ميان دو مقدار تغيير مي‌کند، يعني محيط بين دو وضعيت نوسان مي‌کند. البته در حالت حافظه بسيار کارآمد است. در بيشتر موارد مساله بر اساس نمونه اوليه اصلي منتشر شده مي‌باشد . اين مساله ۱۷ شي دارد و حد وزن در هر ۱۵ نسل تغيير

---

[1] Dynamic Bit Matching
[2] ONEMAX
[3] Chromosome

مي‌كند(۲۲۵۰ بار محاسبه شايستگي) و بين دو مقدار نوسان مي‌كند. در برخي ديگر از مقالات، حد وزن بين سه مقدار تغيير مي‌كند ، تغيير يا به صورت تصادفي است و، يا به صورت گام به گام در فواصل معين كاهش مي‌يابد. مقاله‌هاي نيز از تمامي۱۰۰ شي استفاده مي‌كنند.

۱-۵-۳ سهمي متحرک[1]

مساله سهمي متحرک يک مساله مشابه با تطبيق بيت‌هاي پويا اما با کد کردن اعداد حقيقي مي‌باشد. همانند مساله تطبيق بيت‌هاي پويا اين مساله نيز خوب ارزيابي و آزمايش شده است، براي آناليز آسان است،در تعداد ابعاد به صورت دلخواه مقياس‌پذير است. يک تغيير با شيفت دادن سهمي قابل انجام است. شدت تغييرات مي‌تواند با ميزان شيفت سهمي‌كنترل شود. با استفاده از يک مسير مناسب در طول زمان مي‌توان طول دوره و دقت دوره يا تناوب تغييرات را تعيين کرد.

اين مساله مي‌تواند سه حالت مختلف داشته باشد: خطي، دوراني و تصادفي. روابط بين اين تغييرات در معادلات (۱-۸) تا (۱-۱۰) ذکر شده است. در اين روابط، پارامتر s براي تنظيم شدت تغيير، n بيان كننده تعداد ابعاد مساله، و $N(0,1)$ يک متغير تصادفي با توزيع نرمال مي‌باشد. توجه شود كه t به عنوان انديس براي حالت محيط استفاده شده است، هنگامي‌كه محيط تغيير مي‌كند (هر Δe محاسبه)، t يک واحد افزايش مي‌يابد.

$$f(x,t)=\sum_{i=1}^{n}(x_i+\delta_i(t))^2 \qquad (۱-۸)$$

Linear translation: (۱-۹)

$$\delta_i(0)=0 \;\forall i \in \{1,...,n\}$$
$$\delta_i(t)=\delta_i(t-1)+s$$

Random dynamics: (۱-۱۰)

$$\delta_i(0)=0 \;\forall i \in \{1,...,n\}$$
$$\delta_i(t)=\delta_i(t-1)+s.N_i(0,1)$$

[1] Moving Parabola

Circular dynamics:

$$\delta_i(0) = \begin{cases} 0 : i \ \ odd \\ 1 : i \ \ even \end{cases} \qquad (۱۱-۱)$$

$$\delta_i(t) = \delta_i(t-1) + s$$

و γ برابر است با تعداد حالت‌هاي متفاوتي كه محيط از آنها قبل از اينكه حلقه تكرار حالات مجددا شروع شود عبور مي‌كند (γ طول دوره تناوب را معين مي‌كند، با يک دقت دوره ۱۰۰% زيرا دوره دقيقا تكرار مي‌شود).

۱-۵-۴ قله‌هاي نوسان‌گر[1]

در برخي از الگوريتم‌هاي تكاملي پيشنهاد شده كه از حافظه كمک گرفته شده است، از اين محيط نوساني استفاده شده است. در اين محيط K پهنه جستجو كه معمولا دو تا در نظر گرفته مي‌شود كه هر كدام شامل m قله انتخاب شده به صورت تصادفي مي‌باشند، به همان شكلي كه در مساله قله‌هاي متحرک شرح داده شد. هر كدام از اين پهنه‌هاي جستجو با توجه به يک تابع كسينوسي در حال نوسان مي‌باشند:

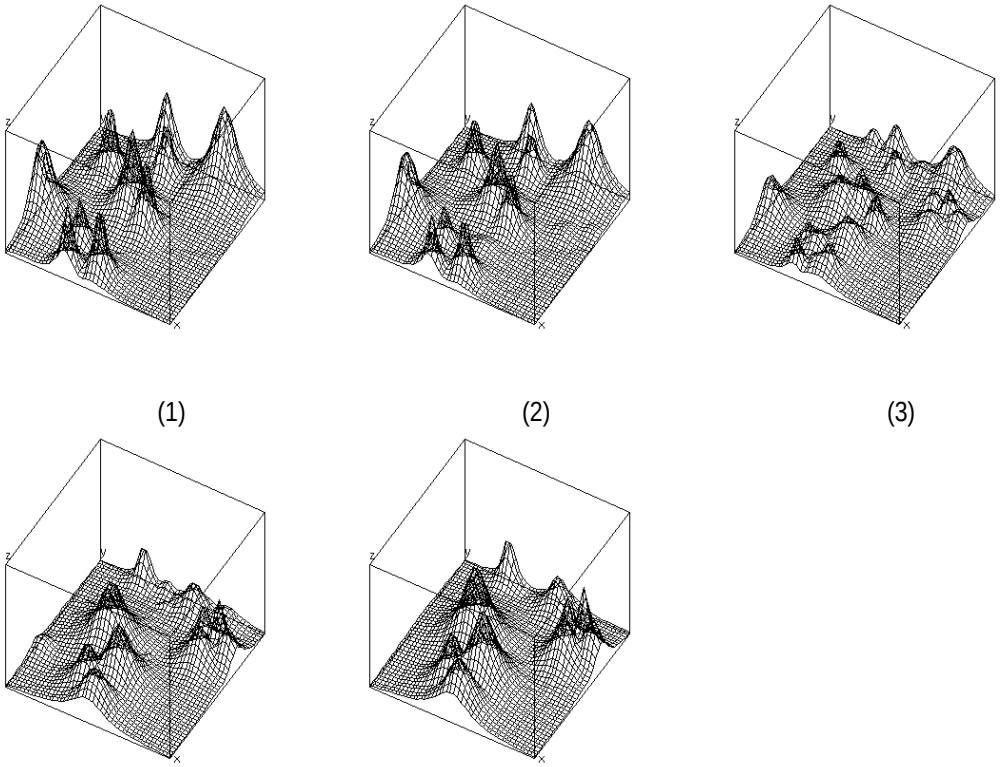
$$f_i(t) = \omega(t) f_i(0) \qquad (۱۲-۱)$$

$$\omega(t) f_i(0) = 0.5 \cos\left(\frac{2t\pi}{steps} + 2\pi\frac{i-1}{K}\right) + 0.5 \qquad (۱۳-۱)$$

$$i = 1 \ldots K$$

كه در آن $steps$ تعداد مراحل مياني در هر دوره را تعيين مي‌كند. در نتيجه مساله بين اين l پهنه جستجو نوسان مي‌كند. در شكل ۱-۱ چند مرحله مياني از يک دوره از نوسان قله‌ها با دو پهنه و هر كدام با ده قله نشان داده شده است.

[1] Oscillating Peaks



شكل ١-١) چند مرحله از يک دوره از يک تابع قله‌هاي نوساني، دو پهنه جستجو با هر کدام ده قله [٤٩]

١-٥-٥ تابع قله‌هاي متحرک[1]

با هدف پرکردن شکاف ميان مسائل واقعي دشوار و مسائل غير واقعي بسيار ساده، يک مساله با يک پهنه جستجوي چند بعدي شامل چندين قله در [٤٩, ٥٠] پيشنهاد شده است که ارتفاع، پهنا و مکان هر قله هنگامي که تغيير کوچکي در محيط رخ مي‌دهد به آهستگي تغيير مي‌کند.

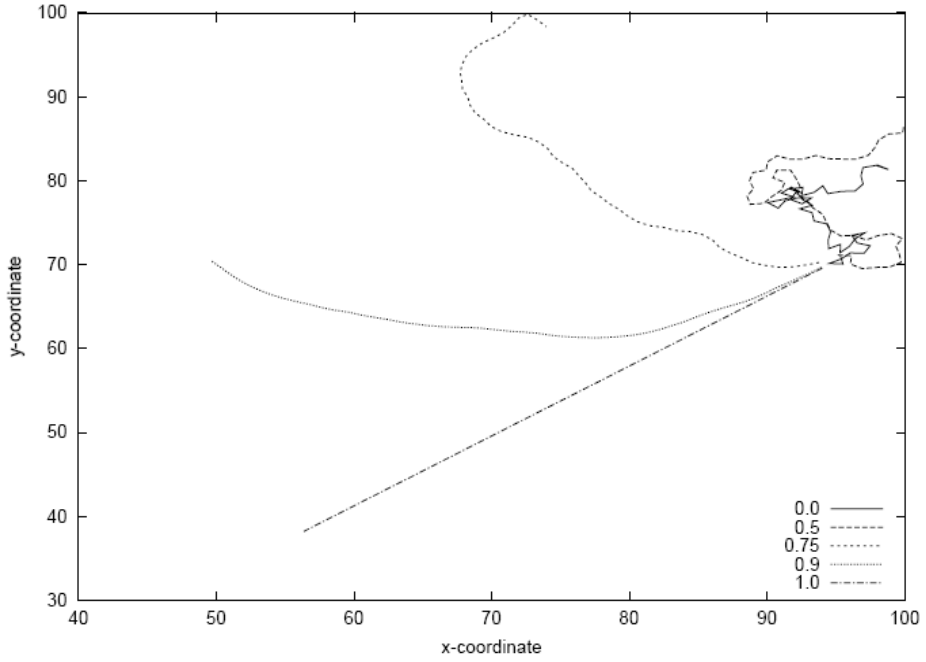
تابع قله‌هاي متحرک شامل m قله و يک دامنه حقيقي n بعدي است. پهنه جستجوي تابع شايستگي به صورت بيشينه بر روي تمام توابع قله معين مي‌شود که به صورت زير فرموله‌بندي مي‌شود :

$$F(\vec{x},t) = \max(B(\vec{x}), \max_{i=1\ldots m} P(\vec{x}, h_i(t), w_i(t), \vec{p}_i(t))) \qquad (١٤\text{-}١)$$

[1] Moving Peaks Function (MPB)

که در آن $B(\vec{x})$ یک پهنه جستجوي پايه تغييرناپذير با زمان مي‌باشد و P تابعي مي‌باشد که شکل يک قله را تعيين مي‌کند به شکلي که هر يک از m قله پارامترهاي متغير با زمان ارتفاع (h)، پهنا(w)، و مکان $(\vec{p})$ مختص به خود را دارا مي‌باشند.

در هر Δe بار محاسبه شايستگي، ارتفاع، پهنا و مکان هر قله تغيير مي‌کند. ارتفاع و پهناي هر قله به وسيله اضافه کردن يک متغير تصادفي گوسي تغيير مي‌کند. مکان هر قله به وسيله بردار v با طول ثابت s حرکت مي‌کند. بنابراين پارامتر s اجازه کنترل شدت تغيير را مي‌دهد. Δe مي‌تواند فرکانس تغيير را معين کند. يک پارامتر جديد، λ، تعيين مي‌کند که تا چه اندازه تغيير مکاني يک قله وابسته به حرکت قبلي آن است. اگر λ=0.0، حرکت کاملا تصادفي است، براي λ=1.0، قله هميشه در يک جهت حرکت خواهد کرد تا هنگامي‌که با مرزهاي فضاي مساله برخورد کند که در اين حالت بازگشتي مشابه توپ بيليارد خواهد داشت. تاثير λ روي مسير يک قله در شکل ۱-۲ نمايش داده شده است.



شکل ۱-۲) مسير حرکت يک قله تنها طي ۱۰۰ مرحله به ازاي مقادير مختلف λ [۴۹]

به طور کلي پارامتر s اجازه کنترل شدت تغيير را مي‌دهد، Δe فرکانس تغيير را تعيين مي‌کند و λ نيز اجازه کنترل مسير تغييرات را مي‌دهد.

به صورت صريح‌تر، يک تغيير در يک قله به صورت زير توصيف مي‌شود :

$\sigma \in N(0,1)$

$$h_i(t) = h_i(t-1) + height\_severity.\sigma \quad (۱۵-۱)$$

$$w_i(t) = w_i(t-1) + width\_severity.\sigma \quad (۱۶-۱)$$

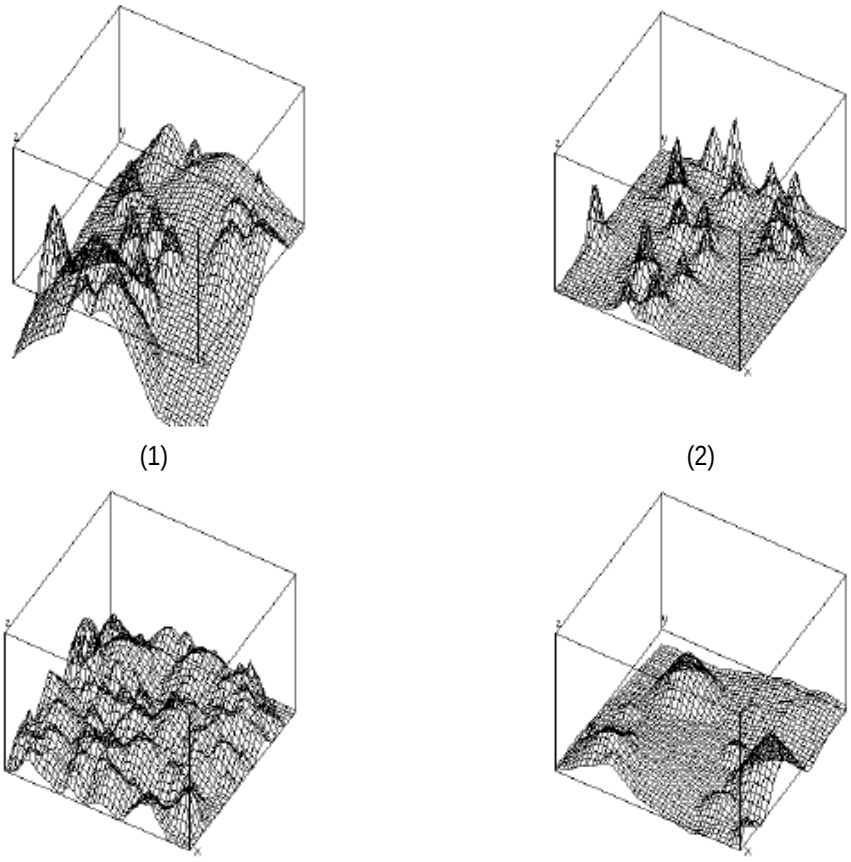
$$\vec{p}_i(t) = \vec{p}_i(t-1) + \vec{v}_i(t) \quad (۱۷-۱)$$

بردار شيفت $\vec{v}_i$ يک ترکيب خطي از بردار تصادفي $\vec{r}$ و بردار شيفت قبلي $\vec{v}_i(t-1)$ است و به بازه [s ۰] نرمال شده است.

$$\vec{v}_i(t) = \frac{s}{\left|\vec{r} + \vec{v}_i(t-1)\right|}((1-\lambda)\vec{r} + \lambda\vec{v}_i(t-1)) \quad (۱۸-۱)$$

بردار تصادفي $\vec{r}$ با توليد اعداد تصادفي براي هر بعد و نرمال کردن طول آن به s ساخته مي‌شود. مختصات X ، ارتفاع H و پهناي هر قله W به صورت تصادفي در بازه محدوده قابل قبول مقدار دهي اوليه مي‌شوند. يک توليد کننده عدد تصادفي ساخته شده در الگوريتم اجازه تکرار دقيقا يک حالت مساله را فراهم مي‌کند. با دادن دانه‌هاي مختلف به اين توليد کننده عدد تصادفي ، مي‌توان بيشمار مساله آزمايشي با ويژگي‌هاي بنيادي يکسان ساخت. پيچيدگي تابع را مي‌توان به سادگي با افزايش تعداد ابعاد و يا تعداد قله‌ها تغيير داد. بعلاوه، مساله يک تعداد از توابع قله P متفاوت مهيا مي‌کند و اجازه تغيير اندازه گام s به صورت متغير با زمان مي‌دهد.

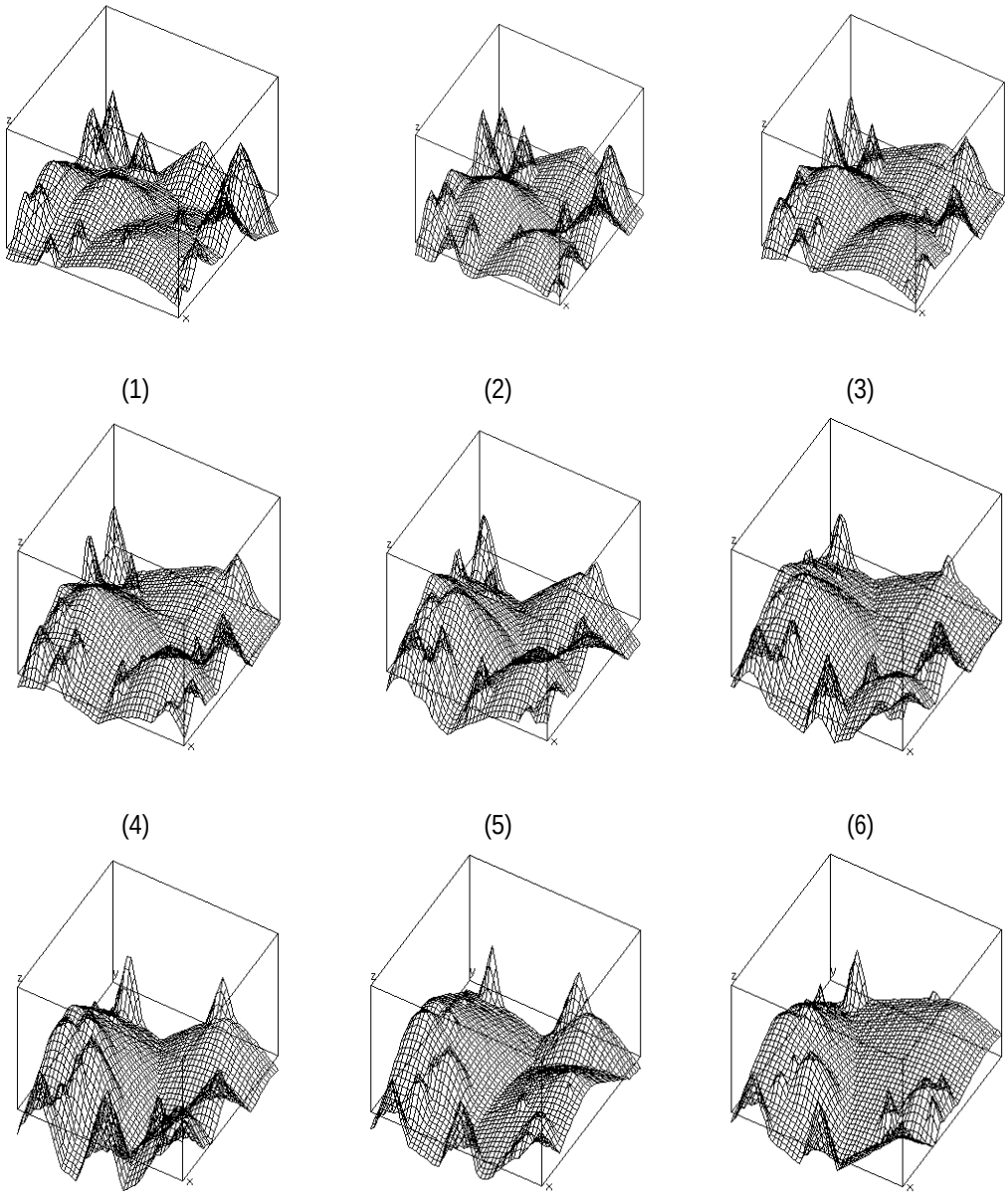
براي ايجاد امکان بازسازي و تکرار يک مساله ، تابع آزمون براي تمام گزينه‌هاي تصادفي از يک توليدکننده اعداد تصادفي جداگانه استفاده مي‌کند. تعدادي مثال براي پهنه‌هاي جستجوي دو بعدي که به وسيله توليدکننده قله‌هاي متحرک ايجاد شده‌اند را مي‌توان در شکل ۱-۳ مشاهده نمود. همچنين يک دنباله از تغييرات براي يک پهنه جستجو در شکل ۱-۴ نمايش داده شده است.



(1) (2)

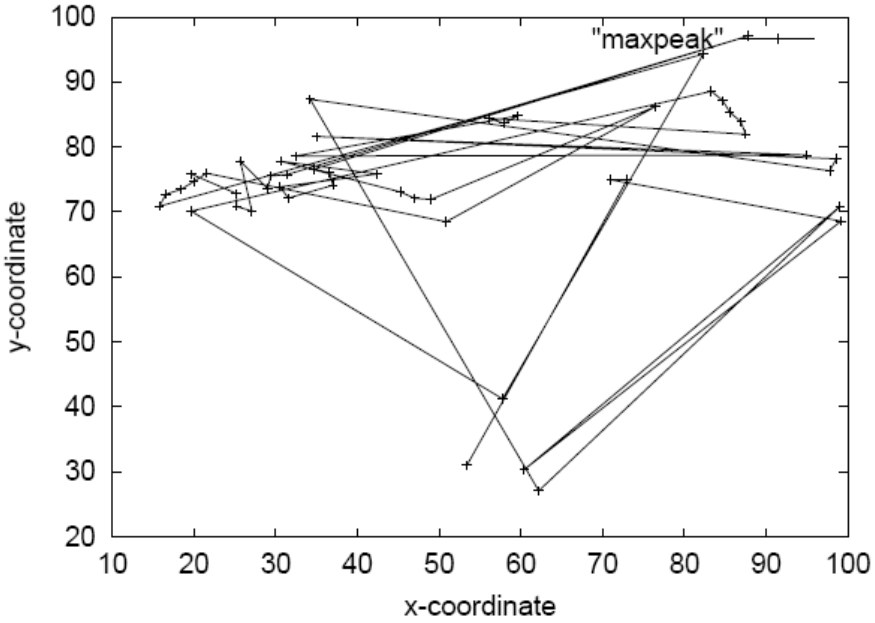
(3) (4)

شکل ۱-۳) مثال‌هاي از پهنه‌هاي جستجوي دو بعدي که توسط محک قله‌هاي متحرک توليد شده‌اند [۴۹]



شكل ١-٤) مثالي از ٨ تغيير پشت سر هم از تغييرات توليد شده توسط محک قله‌هاي متحرک [٤٩]

شكل ١-٥ مثالي از چگونگي حركت پاسخ بهينه در ٥٠ تغيير و در دو بعد را نمايش مي‌دهد. گام‌هاي كوچک از حركت‌هاي قله بهينه ناشي مي‌شود و پرش‌هاي بلند هنگامي رخ مي‌دهند كه در اثر تغيير در ارتفاع قله‌ها، قله ديگري قله بهينه مي‌شود.



شكل ١-٥) مثالي از تغيير مكان بهينه در طي ٥٠ بار تغيير با پارامترهاي ٥ قله، $s = 0.2$ و $\lambda = 0.5$ [٤٩]

٦-١ روش‌هاي محاسبه كارايي

به دليل اينكه در مسائل بهينه سازي پويا يک پاسخ بهينه ثابت وجود ندارد، هدف يافتن اكسترمم‌هاي مساله به صورت دقيق نيست بلكه رديابي مسير پيشروي نقاط بهينه در فضاي مساله مطلوب است. براي اثبات برتري يک روش نسبت به روش‌هاي ديگر، در مقالات حاضر يک روش عمومي نمايش نمودارهاي همگرايي مربوط به بهترين يا ميانگين شايستگي مربوط به روش‌هاي مختلف و مقايسه ديداري آنهاست. براي يک مقايسه دقيق، محاسبه عددي قطعا ترجيح داده مي‌شود.

كارايي آنلاين[1] و آفلاين[2] ابتدا براي مسايل ايستا پيشنهاد شد اما بسياري از نويسندگان از آن‌ها براي مسايل پويا نيز استفاده نمودند. اگر e_t، t امين ارزيابي و T تعداد ارزيابي‌هاي در نظر گرفته شده باشد در آن صورت بسته به نوع مساله‌اي كه در دست داريم، يكي از روش‌هاي زير ممكن است انتخاب خوبي براي مقايسه باشد :

- كارايي آنلاين x كه به صورت محاسبه ميانگين محاسبه شايستگي در تمام دفعات اجرا است و به صورت :

[1] Online
[2] Offline

$$x = \frac{1}{T}\sum_{t=1}^{T} e_t \qquad (۱۹-۱)$$

بيان مي‌شود . فرض بر اين است كه هر ارزيابي نياز به آزمايش دنياي واقعي دارد.

- كارايي آفلاين x* كه به صورت محاسبه ميانگين بهترين مقدار شايستگي محاسبه شده در هر گام[1] از الگوريتم بيان مي‌شود :

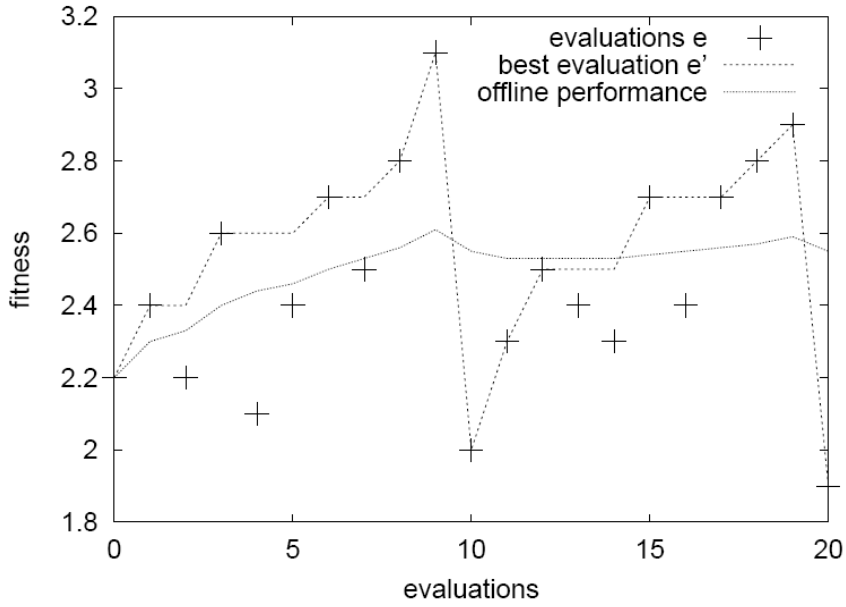
$$x^* = \frac{1}{T}\sum_{t=1}^{T} e_t^* \qquad (۲۰-۱)$$

كه در آن $e_t^* = \max\{e_1, e_1, ..., e_t\}$ مي‌باشد. براي مسائل پويا اين محاسبه پيچيده است زيرا بايد مطمئن باشيم كه پاسخي كه پيدا كرده‌ايم بهترين پاسخ براي محيط جاري است. از آنجايي كه اين مساله تنها با محاسبه دوباره تضمين مي‌شود زيرا شايستگي‌هاي محاسبه شده در گام‌هاي قبل از بروز تغيير در محيط ديگر معتبر نيستند و بايد دوباره محاسبه شوند، پس اين معيار يعني كارايي آفلاين بايد از زمان بروز آخرين تغيير قبل از مرحله‌اي كه قرار داريم محاسبه شود كه محاسبات شايستگي در اين بازه معتبر مي‌باشند.

$$x' = \frac{1}{T}\sum_{t=1}^{T} e_t' \qquad (۲۱-۱)$$

كه در آن $e_t' = \max\{e_\tau, e_{\tau+1}, ..., e_t\}$ و τ شروع كننده بازه زماني است كه محاسبات معتبر هستند و در واقع بيان كننده زماني است كه آخرين تغيير در محيط قبل از لحظه t رخ داده است. البته احتياج است كه تغييرات محيطي و لحظه اتفاق افتادن آن‌ها و يا حداقل فركانس تغييرات محيط شناخته شده باشند. در شكل ۱-۶ مفهوم كارايي آفلاين نمايش داده شده است.

[1] Iteration



شكل ١-٦) نمايش محاسبات كارايي آفلاين. فرض شده است كه محيط پس از ارزيابي‌هاي نهم و نوزدهم تغيير كرده‌اند [٤٩]

اگر پاسخ بهينه واقعي در هر لحظه از زمان شناخته شده باشد، ميانگين خطا مي‌تواند به عنوان معيار محاسبه كارايي استفاده شود. اگر بزرگ كردن خطاهاي بزرگ مد نظر باشد، ميانگين مربع خطا ممكن است استفاده شود. همچنين مي‌توان به جاي خطاي مطلق از خطاي نسبي نيز استفاده كرد (بهترين پاسخ جاري تقسيم بر پاسخ بهينه واقعي) .

براي حالتي كه تنها يک پاسخ بعد از هر تغيير واقعا پياده سازي مي‌شود، ميانگين خطاي بهترين فرد گرفته شده براي هر وضعيت محيط ممكن است مناسب باشد. اين معيار كارايي ممكن است هنگامي‌كه پاسخ پياده سازي شده توانايي تطبيق به صورت پيوسته با محيط پويا را نداشته باشد مناسب باشد، اما هنگامي‌كه بعد از هر تغيير، يک پاسخ در يک فاصله زماني معين خواسته شده است. بايد توجه داشت كه ميانگين گيري روي تعداد دفعات محاسبه بيشتر و بيشتر و نزديكتر شدن به اجراي پيوسته الگوريتم باعث مي‌شود كه منحني‌ها هموارتر و هموارتر باشند.

١-٧ نتيجه‌گيري

در اين فصل در ابتدا اهداف و ساختار كلي پايان‌نامه شرح داده شد. سپس به معرفي و شرح مقدمات و مفاهيم اوليه مورد استفاده در پايان‌نامه پرداختيم. در ابتدا معرفي كوتاهي بر الگوريتم‌هاي تكاملي انجام شد. سپس محيط‌هاي پويا، معيارهاي طبقه‌بندي آن‌ها و همچنين انواع معيارهاي ارزيابي كارايي در اين محيط‌ها را مورد بررسي قرار داديم.

فصل دوم

کارهاي گذشته

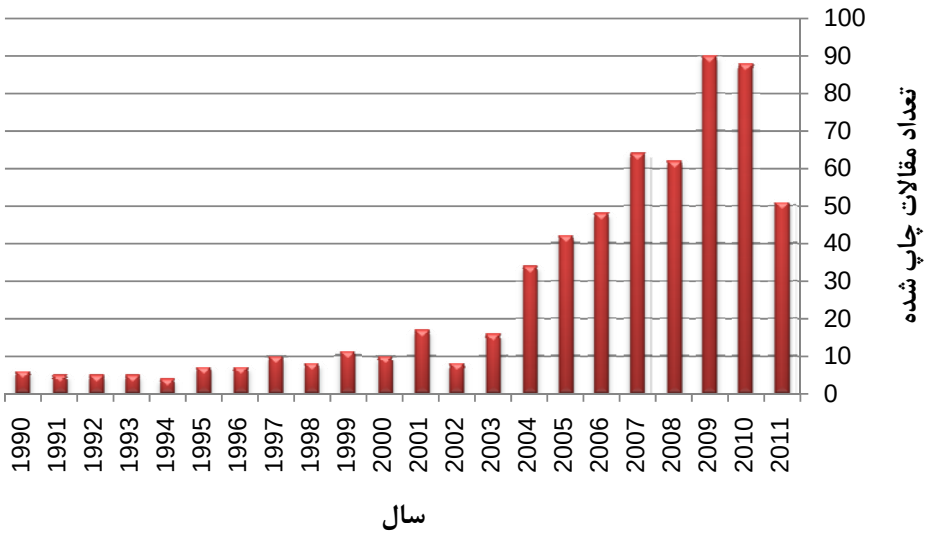
۲. کارهاي گذشته

زماني که تغييري در يک مساله بهينه سازي روي مي‌دهد به طور مثال زماني که هدف بهينه سازي، نمونه مساله، و يا برخي از محدوديت‌هاي مساله تغيير مي‌کند، در آن صورت بهينه آن مساله نيز به طبع آن تغيير مي‌کند. در اين حالت نياز است تا پاسخ قبلي با شرايط جديد تطبيق داده شود. راه‌کارهاي بسيار متفاوتي تا کنون براي مواجهه با اين مشکل پيشنهاد گرديده که در اينجا به بررسي آن‌ها مي‌پردازيم.

يک راهکار براي مواجهه با اين پويايي اينست که هر تغييري را به عنوان يک مساله جديد بهينه سازي که مي‌بايست از ابتدا دوباره حل گردد در نظر گرفت. اين راهکار ساده در عمل غير ممکن است زيرا حل کردن يک مساله از ابتدا و بدون استفاده از اطلاعات بدست آمده قبلي بسيار زمانبر مي‌باشد زيرا ممکن است يک تغيير به صورت مستقيم و دقيق قابل شناسايي نباشد و حتي شايد مساله جديد با راه حل مساله قديمي‌تفاوت شاياني نداشته باشد.

در اين صورت مناسب به نظر مي‌رسد که الگوريتم تکاملي‌اي داشته باشيم که قادر باشد به صورت پيوسته راه حل را با محيط در حال تغيير با استفاده از اطلاعات بدست آمده در گذشته، متناسب سازد. با توجه به اينکه الگوريتم‌هاي تکاملي با تکامل طبيعي مشترکاتي دارند و در طبيعت تطبيق پذيري يک فرايند پيوسته و ادامه دار مي‌باشد، به نظر براي اين گونه محيط‌ها مناسب به نظر مي‌رسند.

قديمي‌ترين کاربرد الگوريتم‌هاي تکاملي براي محيط‌هاي پويا به سال ۱۹۶۶ باز مي‌گردد. البته اين موضوع تا ميانه دهه ۸۰ مورد استقبال چنداني قرار نگرفت. پس از آن تمرکز بسياري از تحقيقات به آن جلب شد و مقالات نسبتا زيادي نيز به چاپ رسيد. همان طور که در شکل ۲-۱ نيز مشاهده مي‌شود اين محبوبيت از ميانه دهه ۹۰ مرتبا در حال افزايش بوده است و اکنون از زمينه‌هاي مورد علاقه در بهينه‌سازي‌هاي تکاملي محسوب مي‌شود. آمارهاي ارايه شده در اين نمودار از طريق جستجو در مقالات موجود بر روي اينترنت، بدست آمده که حتي تمامي مقالات ارايه شده در اين زمينه را پوشش نمي‌دهد.



شكل ۲-۱) تعداد مقالات چاپ شده بر روي الگوريتم‌هاي تكاملي در محيط‌هاي پويا در هر سال، بين سال‌هاي ۱۹۹۰ تا ۲۰۱۰

مشكل اصلي الگوريتم تكاملي استاندارد به نظر مي‌رسد اين مي‌باشد كه به ناگهان به سمت بهينه همگرا مي‌شود و در نتيجه تنوع لازم جهت اكتشاف فضاي جستجو را از دست مي‌دهد. در نتيجه زماني كه جمعيت الگوريتم تكاملي همگرا مي‌شود قابليت خود را براي تطبيق با تغييرات محيطي زماني كه تغييري روي مي‌دهد از دست مي‌دهد. در سال‌هاي اخير نويسندگان مختلفي از راه‌هاي متفاوتي براي حل اين مشكلات تلاش نموده‌اند كه در ادامه برخي از جديدترين ايده‌هاي مطرح شده مرور خواهند شد. در مراجع [۳۴, ۴۹, ۵۱, ۵۲] نيز بررسي‌هاي كاملي بر روي روش‌هاي تكاملي ارايه شده براي بهينه‌سازي در محيط‌هاي پويا انجام شده است.

۲-۱-۱ شروع از نو[1]

يک راه ساده براي حفظ تنوع جمعيت و يا عكس‌العمل نشان دادن به تغيير راه‌اندازي دوباره الگوريتم بدون در نظر گرفتن اطلاعات مساله قبل از تغيير مي‌باشد و البته اين راه حل بسيار ساده‌لوحانه ولي در عين حال سر راست مي‌باشد. اگر مقدار تغيير تقريبا ناچيز باشد، اين شبيه به حالتي است كه پاسخ بهينه جديد از بسيار جهات وابستگي به بهينه قبلي دارد. بنابراين ممكن است به سادگي بتوان از اطلاعات جمعيت قبلي براي مقدار دهي اوليه به جمعيت جديد استفاده كرد به عنوان مثال با انتقال افراد به جمعيت بعدي.

[1] Restart/Reinitialization

ميان اكتشاف[1] يا همان جستجوي عمومي و استخراج[2] يا همان جستجوي محلي تعادلي وجود دارد. حفظ مقدار زيادي از جمعيت و شروع از نزديكي يک بهينه محلي ممكن است ما را به همگرايي زودرس دچار كند ولي از سوي ديگر از دست دادن مقدار زيادي از اطلاعات ممكن است به صورت جدي زمان زيادي براي پيدا كردن بهينه جديد بگيرد.

اين تعادل مورد بررسي قرار گرفته شده و در حدود ۱۰% تا ۵% جمعيت قبلي را در جمعيت جديد قرار مي‌دهند و بهبود قابل ملاحظه‌اي را گزارش كرده‌اند، هرچند كه هنگامي‌كه ميزان جمعيتي بسيار بيشتري براي شرايط بعد از تغيير حفظ شود و يا تغييرات به صورت گسترده‌اي مانند حذف يک كار صورت بگيرد، اين روش جواب نمي‌دهد. تحقيقات بيشتر روي كيفيت شايستگي جمعيتي كه به مرحله بعد پاس داده مي‌شوند صورت گرفته و گزارش شده به عنوان مثال در [۴۹] محقق گزارش كرده است كه شباهت بسياري بين تاثير تعداد جمعيت و شايستگي جمعيت در كاركرد الگوريتم وجود دارد. بدين صورت كه ارسال افرادي از جمعيت با شايستگي پايين‌تر به اكتشاف و ارسال افراد با شايستگي بيشتر به استخراج كمک خواهد كرد. بنابراين بهتر اين است كه الگوريتم تكاملي توانايي تطبيقي داشته باشد.

در [۹] از يک پايگاه دانش براي نگهداري موجودات موفقيت استفاده گرديده است. سيستم فرض مي‌كند كه وضعيت‌هاي محيطي قابل اندازه گيري مي‌باشند. در بازه‌هاي زماني معيني، بهترين موجود در آن زمان به همراه داده‌هايي كه وضعيت محيط را در آن زمان مشخص مي‌كند در داخل پايگاه دانش ذخيره و انديس گذاري مي‌شود. هر زمان كه الگوريتم با محيط جديدي روبرو مي‌شود الگوريتم از نو راه‌اندازي مي‌گردد. براي شروع از نو، نيمي‌از جمعيت را از افراد ذخيره شده در پايگاه دانش كه در محيطي مانند محيط جديد موفق بوده‌اند انتخاب مي‌گردند. شباهت محيط‌ها با استفاده از آناليز همسايه‌هاي مجاور[3] محاسبه مي‌شود. آزمايشات نشان داده است كه اين پايگاه دانش به الگوريتم تكاملي اجازه مي‌دهد كه بر اساس دانش بدست آمده از گذشته عمل كند.

در مسايلي كه تغيير در مساله موجب تغيير در بازنمايي افراد مي‌شود، به طور مثال زماني كه در مساله زمانبندي كارها يک كار جديد اضافه مي‌گردد و مي‌بايست به ژنوم افراد يک ژن جديد افزود، انتقال ساده افراد از پايگاه دانش به جمعيت جديد بدين صورت امكان پذير نمي‌باشد. در عوض در موجودات قديمي‌مي‌بايست تغييرات لازم صورت گيرد و با مساله جديد تطبيق پيدا كنند.

---

[1] Exploration
[2] Exploitation
[3] Nearest Neighbor Analysis

با اين حال حتي در اين شرايط زمان زيادي صرفه جويي خواهد شد زيرا حتي زماني كه مساله تغيير كند قسمتي از راه حل‌هاي گذشته مي‌توانند در راه حل جديد دوباره مورد استفاده قرار گيرند.

مي‌توان زماني كه مساله تغيير مي‌كند، تمامي‌افراد جمعيت با شرايط جديد تطبيق داده شوند و به عنوان جمعيت اوليه جديد مورد استفاده قرار گيرند. مساله آزمايش مساله زمانبندي كارها مي‌باشد و زماني كه يک كار جديد در زمان‌هاي تصادفي اضافه مي‌گردد از راه حل‌هايي كه تا كنون براي كارهاي ناتمام بدست آمده‌اند براي توليد جمعيت اوليه استفاده نموده‌اند. اين ايده اخيرا آزمايش شده است بدين صورت كه كارهاي جديد به صورت تصادفي در داخل جمعيت نسل قبلي اضافه مي‌گردند. با وجودي كه كيفيت جواب بدست آمده نسبت به حالتي كه جمعيت اوليه به صورت تصادفي انتخاب شوند تغيير محسوسي نداشته است ولي سرعت به شدت افزايش يافته است.

اگر يک مساله زمانبندي وظيفه‌ها بر روي يک پردازنده كه هدف حداقل سازي مدت زمان انتظار مي‌باشد را در نظر بگيريم. براي مساله از يک بازنمايي چينشي با طول ثابت استفاده شده است يعني از صف وظيفه‌ها تنها n وظيفه اول براي بهينه سازي در نظر گرفته مي‌شوند. زماني كه اجراي يک وظيفه به پايان برسد اولين وظيفه از بهترين كروموزوم انتخاب شده و با يک وظيفه جديد از داخل صف جابجا مي‌گردد.

در مسايل زمانبندي كارهاي مغازه زمان آماده بودن يک كار و يا زمان پردازش ممكن است تغيير كند كه در اين صورت نيازي به تغيير بازنمايي نمي‌باشد. در هر صورت پيشنهاد مي‌شود كه پس از يک تغيير الگوريتم ژنتيک مي‌بايست تنها بر روي مجموعه كاهش يافته‌اي از كارها كه به طور مستقيم تحت تاثير تغيير مساله قرار مي‌گيرند از نو راه‌اندازي شود. تمامي‌كارهايي كه تحت تاثير قرار نمي‌گيرند را ثابت در نظر مي‌گيريم و در دسترس بودن ماشين‌ها با توجه به آن‌ها به صورت تطبيقي تنظيم مي‌گردند. بايد در نظر داشت كه اين روش تضمين نمي‌كند كه جواب بهينه در داخل فضاي جستجوي جديد وجود داشته باشد زيرا ممكن است يک تغيير در مساله نياز به دوباره زمانبندي كردن تمامي‌كارها داشته باشد، حتي كارهايي كه به صورت مستقيم تحت تاثير تغيير مساله نبوده‌اند. اما اين مساله كاهش يافته بسيار سريعتر حل مي‌شود و همچنين بايد در نظر داشت كه بين سرعت و كيفيت تعادلي وجود دارد.

يک نوع عملگر شروع از نو كه تا حدي عجيب نيز به نظر مي‌آيد در [۲] پيشنهاد شده است. در آن از يک جمعيت به نسبت كوچک با اندازه ۵ به همراه انتخاب شايسته‌ترين فرد و بدون وجود جهش استفاده مي‌شود. در عوض پس از آنكه همگرايي در جمعيت تشخيص داده شد (به طور مثال

با اندازه گيري ميانگين فاصله همينگ[1] تمامي‌افراد جمعيت با يكديگر)، بهترين عضو حفظ مي‌شود و مابقي افراد به صورت تصادفي توليد مي‌شوند و الگوريتم تكاملي از نو شروع به كار مي‌كند.

اين رويكرد شايد همان رويكرد شروع از نو به همراه نگهداري بهترين فرد در نظر گرفته شود. با اين وجود شروع از نو با تشخيص همگرايي جمعيت آغاز مي‌گردد و نه تغيير در محيط. در آن مقاله، از الگوريتم پيشنهاد شده به صورت موفقيت آميز جهت راهنمايي يک وسيله پرنده در داخل يک چاله هوايي[2] استفاده شده است. اين الگوريتم در مقايسه با ساير الگوريتم تكاملي بر روي مسايل پويا با مهاجرت تصادفي تا حدودي بدتر عمل مي‌كند ولي داراي پيچيدگي محاسبات پايين‌تري مي‌باشد.

شايد بتوان گفت كه يک شروع از نو ساده به طور قطع بهترين گزينه نمي‌باشد زيرا هيچ گونه اطلاعاتي از هر مرحله به مرحله بعدي انتقال پيدا نمي‌كند. شروع از نوهاي بهتري كه تلاش مي‌كنند تا قسمت‌هاي از اطلاعات را حفظ كنند در بعضي مسايل نويد بخش مي‌باشند. اگر تغيير در مساله موجب تغيير در بازنمايي گردد، شروع از نو شايد تنها گزينه پيش رو باشد.

۲-۱-۲ تطبيقي سازي جهش

به صورت آشكارا تطبيق جهش بعد از بروز تغيير در محيط يک استراتژي معمول است . علت آنست كه افزايش نرخ جهش بعد از تغيير به جمعيت همگرا شده كمک مي‌كند كه در فضا پخش شوند و دوباره جستجو را تا حدودي از نو آغاز كنند. براي مثال مي‌توان از روشي به نام ابر جهش[3] به عنوان قسمتي از الگوريتم ژنتيک استفاده كرد : هر گاه ميانگين زماني بهترين كارايي الگوريتم ژنتيک كاهش پيدا مي‌كند نرخ جهش نيز به شدت افزايش مي‌يابد. در واقع ايده اين روش از روشي كه باكتريها براي زنده ماندن در محيط‌هاي پويا استفاده مي‌كنند الهام گرفته شده است. در يک مطالعه اخير [۵۳]، ابر جهش مورد بررسي بيشتري قرار گرفته و مشاهده شد كه فركانس تغييرات بر روي ميزان افزايش لازم نرخ جهش تاثير دارد. هر چه فركانس تغييرات بيشتر باشد نياز به ابر جهش قوي‌تري مي‌باشد.

در [۵] روشي از مهاجرت تصادفي[4] را معرفي گرديده كه در هر نسل تقريبا جمعيت موجود با اعضاي جديدي كه به صورت تصادفي توليد شده‌اند جايگزين مي‌شود. از يک نظر اين روش مشابه

---

[1] Hamming Distance
[2] Wind Shear
[3] Hyper Mutation
[4] Random Immigrants

اعمال یک جهش بسیار قوي بر روي بخشي از جمعیت به نظر مي‌رسد. همچنین بازترکیبي بین یک عضو تصادفي تولید شده و یک عضو قدیمي‌از جمعیت نیز مشابه یک جهش با شدت زیاد مي‌باشد. علت محدود کردن جهش به تنها قسمتي از جمعیت این است که تنوع را بدون ایجاد گسستگي در روند در حال پیشرفت جستجو، حفظ کرد.در حالتي که پهنه جستجو دوران دارد این روش کمک خاصي نمي‌کند.

در [۴] مقایسه‌اي بین سه روش مختلف جهش در زماني که در محیط‌هاي پویا مورد استفاده قرار مي‌گیرند انجام گرفته است :

- جهش ثابت: مانند الگوریتم ژنتیک استاندارد
- ابر جهش محرکي[1]: هنگامي‌که کاهش در کارایي صورت مي‌گیرد شدت جهش افزایش مي‌یابد در واقع هنگامي‌که الگوریتم موفقیت کمتري به دست مي‌آورد افزایش جهش انجام مي‌شود
- مهاجرت تصادفي: با بررسي چندین مساله‌پویاي مختلف مشاهده گردیده است که ابر جهش براي محیط‌هایي که به آرامي‌تغییر مي‌کنند بهتر عمل مي‌کند. در محیط‌هایي که تغییر شدیدتر مي‌باشد مهاجرت تصادفي بهتر عمل مي‌کند.

در [۶] یک عملگر جهش به نام جستجوي محلي متغیر[2] پیشنهاد مي‌شود که در آن محدوده جهش هنگامي‌که تغییر در محیط شناسایي شد به آهستگي افزایش مي‌یابد. در ابتدا تنها جهش‌هاي کوچک اعمال مي‌شود . اگر شایستگي جمعیت بعد از گذشت یک زمان معین بهبود نیابد در آن صورت دامنه جستجوي محلي با زیاد و زیاد کردن شدت جهش گسترش مي‌یابد. توجه کنید که این روش به شدت به تغییرات مقادیر شایستگي وابسته است و به همین خاطر به صورت فراگیر قابل کاربردي نمي‌باشد. هنگامي‌که روش جستجوي محلي متغیر با روش ابر جهش محرکي مقایسه مي‌شود مشاهده مي‌شود که روش ابر جهش محرکي براي جابجایي‌هاي بزرگ و روش جستجوي محلي متغیر در جابجایي‌هاي کوچک در پهنه جستجو بهتر عمل مي‌کند.

از این عملگر جهت بهینه سازي تنظیمات دریچه هواي پیوسته یک سیستم متشکل از چندین کوره با لوله مشترک و بر روي مساله بهینه سازي پرس چغندر قند[3] استفاده شده است. در مقایسه با ابرجهش و شروع از نو این روش در محیط‌هاي با تغییر کوچک بهترین مي‌باشد. در صورتي که براي

---

[1] Triggered Hypermutation
[2] Variable Local Search (VLS)
[3] Suger-beet-presses

تغييرات بسيار شديد شروع از نو بهترين روش مي‌باشد. ابرجهش در تغييرات كوچكتر بهتر از شروع از نو عمل مي‌كند اما به خوبي عملگر جستجوي محلي متغير نمي‌باشد.

تاثير فركانس تغييرات بر روي روش‌هاي جستجوي محلي متغير، شروع از نو و ابرجهش مورد بررسي قرار گرفته است. نتايج نشان از برتري روش جستجوي محلي متغير نسبت به دو روش ديگر مي‌باشد بويژه در محيط‌هاي با فركانس تغيير بالا كه در آن‌ها ميزان تغيير به نسبت كوچك است. علاوه بر آن يک رويكرد يادگيري جهت تنظيم بازه جهش پيشنهاد شده است كه تلاش دارد تا براساس كاهش ميزان شايستگي به وجود آمده بر اثر تغيير محيط محدوده صحيح جستجو را بياموزد. با اين رويكرد بازدهي روش جستجوي محلي متغير افزايش نيز مي‌يابد.

روش‌هاي جهش تطبيقي ديگر كه مخصوص محيط‌هاي پويا طراحي نشده‌اند نيز در الگوريتم‌هاي تكاملي و برنامه‌نويسي تكاملي استفاده مي‌شوند. يک بررسي كوتاه بر روي اين رويكرد ها انجام شده است. معمولا جهش به صورت پويا و در پاسخ به مشخصات محلي شايستگي پهنه مورد جستجو تنظيم مي‌گردد. در حالت جهش خود تطبيقي پارامترهاي روش مورد نظر كه معمولا واريانس جهش گوسي مي‌باشد در داخل كروموزوم‌ها كدگذاري مي‌شوند و همراه به ساير پارامترهاي مساله تكامل مي‌يابند. جهت جهش دادن اين پارامترها نسخه عمده وجود دارد :

- جهش خود تطبيقي افزاينده[1]: $\sigma_{new} = \sigma_{old} + \tau N(0,1)$
- جهش خود تطبيقي لگاريتمي[2]: $\sigma_{new} = \sigma_{old} e^{\tau N(0,1)}$

كه اين روش‌ها بر روي مسايل ساده پويا آزمايش شده‌اند و نتايج زير بدست آمده :

روش خود تطبيقي با استفاده از روش بروز رساني افزاينده، با يک فرم ديگر از روش‌هاي تطبيقي كه در آن واريانس جهش در هنگام رسيدن به اكسترمم كاهش مي‌باشد مقايسه شده است. مشاهده گرديده است كه در بسياري از مسايل پويا روش خود تطبيقي قادر است تا در همان اوايل اجراي اول به سرعت به سمت اكسترمم‌ها حركت كند ولي اين بازدهي با مرور زمان به شدت كاهش مي‌يابد در حالي كه روش تطبيقي در تمامي‌طول اجرا يک كارايي تقريبا ثابت از خود نشان مي‌دهد. به طور كلي روش خود تطبيقي مقداري بهتر عمل مي‌كند و كمتر قابل پيش بيني مي‌باشد.

استراتژي تكاملي با جهش خود تطبيقي لگاريتمي و برنامه‌نويسي تكاملي با هر دو روش خود تطبيقي لگاريتمي و افزايشي بر روي همان مسايل مقايسه شده‌اند. مشاهده مي‌شود كه استراتژي

[1] Multiplicative Self Adaptation
[2] Log-normal Self Adaptation

تكاملي با جهش خود تطبيقي لگاريتميتا حدودي بهتر از برنامه‌نويسي تكاملي با جهش خود تطبيقي مي‌باشد اما هر دو بسيار بهتر از برنامه‌نويسي تكاملي با جهش خود تطبيقي افزايشي مي‌باشند. به اين نتيجه رسيده‌اند كه قانون خود تطبيقي لگاريتميبراي محيط‌هاي پويا مناسب‌تر از جهش خودتطبيقي افزايشي مي‌باشد. با اين وجود در اين آزمايشات علاوه بر تغيير قانون بروز رساني ترتيب به روز رساني نيز تغيير داده شده است. هنگامي‌كه از جهش خودتطبيقي لگاريتمياستفاده مي‌شود ابتدا پارامترهاي جهش، جهش داده مي‌شوند و سپس از آن‌ها جهت جهش دادن فرزندان استفاده مي‌شود در صورتي كه در هنگام استفاده از جهش افزايشي ابتدا فرزندان جهش داده مي‌شوند و پارامترهاي جديد جهش در نسل آينده مورد استفاده قرار مي‌گيرند. حتي ممكن است بيان شود كه ترتيب اين بروز رساني‌ها در واقع تاثير بيشتري نسبت به خود قانون بروزرساني دارد.

همچنين استراتژي‌هاي مختلف خود تطبيقي با يكديگر مقايسه شده‌اند. در آزمايشات گزارش شده بر روي پهنه‌هاي جستجوي در حال چرخش انتخاب يک مقدار براي تمامي‌ابعاد بهتر از انتخاب مقادير مختلف براي هر بعد عمل مي‌كند. احتمالا تعداد زياد پارامترها باعث كندي بيش از حد الگوريتم در محيط‌هاي پويا مي‌شود. با توجه به اين مشاهدات يک قانون جهش عمومي‌بويژه در محيط‌هاي پويا با نرخ تغييرات زياد بهتر عمل مي‌كند.

يک استراتژي تكاملي با استفاده از جهش خود تطبيقي را با يک الگوريتم ژنتيک زاينده[1] بدون استفاده از جهش خود تطبيقي مقايسه شده‌است و مشاهده گرديده كه استراتژي تكاملي خود به خود ميزان سيگماي جهش را به يک مقدار بهينه كه در اندازه‌هاي سرعت حركت نقطه بهينه مي‌باشد تنظيم مي‌كند و در نتيجه نتايج بسيار بهتري به دست مي‌آيد.

مفيد بودن جهش خود تطبيقي در هنگامي‌كه ميزان احتمال بازتركيبي نيز در كروموزوم كد گذاري شده‌است نيز تاييد شده‌است.

روش‌هاي فوق گاهي در ابتداي كار الگوريتم مي‌توانند به خوبي تغييرات را دنبال كنند ولي با مرور زمان كارايي خود را از دست مي‌دهند . روش دوم براي محيط‌هاي پويا مناسب‌تر مي‌باشد .

[1] Breeder Genetic Algorithm

۲-۱-۳ حافظه آشکار[1] یا تلویحي[2]

در روش‌هاي مورد بحث در اين قسمت ، الگوريتم تكاملي به وسيله حافظه‌اي كه بهترين پاسخ‌ها را ذخيره نموده‌است گسترش يافته است تا در صورت نياز دوباره مورد استفاده قرار گيرند. اين حافظه ممكن است به صورت تلويحي با افزونه كردن بازنمايي و يا به صورت آشكار با معرفي حافظه‌اي براي ذخيره سازي و فرموله‌بندي يک سري راهكار به منظور ذخيره سازي و بازيابي پاسخ‌ها از آن حافظه صورت مي‌گيرد.

استراتژي‌هاي همراه با يک حافظه ممكن است به صورت ويژه براي محيط‌هاي با تغييرات دوره‌اي سودمند باشند ، هنگامي‌كه مجموعه كوچكي از وضعيت‌هاي محيط تكرار مي‌شود. علاوه بر آن اين بازنمايي افزونه شده همگرايي را به تاخير مي‌اندازد و تنوع را افزايش مي‌دهد.

به نظر مي‌رسد مهمترين روش از نوع افزونگي بازنمايي ، جفت در نظر گرفتن كروموزوم‌ها مي‌باشد. در [۱۰] گزارشهايي از آزمايشات روي بازنمايي جفت كروموزومي و ژن‌هاي غالب و مغلوب ارائه شده است. از آنجايي كه از قبل واضح نيست كه كدام مقدار آلل[3] مي‌تواند در يک مكان ژن خاص غالب شود، گلدبرگ و اسميت پيشنهاد كردند كه از ژن‌هاي سه مقداري[4] كه مي‌توانند سه مقدار "۰" ، " يک مغلوب[5]" يا يک غالب[6]" را داشته باشند استفاده شود. در آزمايشات كه بر روي مساله كوله‌پشتي متغير با زمان انجام دادند، كيفيت تطبيق پذيري بيشتري از الگوريتم ژنتيک ساده گزارش كردند.

آن‌ها ادعا مي‌كردند كه روش ژن‌هاي سه مقداري دچار بايا‌س است و نتايجي كه مبني بر كارايي بهتر آن در مقايسه با الگوريتم ژنتيک ساده در آن گزارش آمده بود به دليل كندتر بودن روند همگرايي در روش ژن‌هاي سه مقداري مي‌باشد كه براي محيط‌هاي با تغييرات تكرار شونده زياد مناسب به نظر مي‌رسد. در ساير محيط‌هاي پويا حتي ممكن است الگوريتم ژنتيک ساده بهتر عمل كند. در [۵۴] طرح جفت كروموزومي‌جديد با چهار آلل براي هر ژن با مقادير ۰ يا ۱ براي هر دو حالت با دو وضعيت غالب يا مغلوب ارائه دادند. اين روش توانايي تطبيق سريع با تغييرات را دارا است، آنها سازوكاري را براي تعيين آلل غالب و اينكه كدام جفت از آلل‌ها بايد از نظر غالب و

---

[1] Explicit
[2] Implicit
[3] Allele
[4] Triallelic
[5] Recessive
[6] Dominant

مغلوب بودن معکوس شوند ارائه مي‌کنند. اين عمل در زماني که شايستگي جمعيت بيش از ۲۰% کاهش مي‌يابد انجام مي‌شود.

روش‌هاي ديگري نيز مبتني بر چندرشته‌اي[1] با اضافه شدن يک بردار براي تعيين وضعيت غلبه هر آلل در هر رشته نيز پيشنهاد شده است .

هر چند افزونگي بازنمايي به الگوريتم تکاملي اجازه مي‌دهد که در هنگام اجرا به صورت تلويحي اطلاعات مفيدي را ذخيره کند ولي اين موضوع که آيا الگوريتم به صورت کارآمد از اين اطلاعات استفاده مي‌کند يا خير به طور کامل روشن نيست. در روش‌هاي ارائه شده در قسمت‌هاي بعدي اين بخش از حافظه آشکار به عنوان جايگزين استفاده مي‌شود که در آن اطلاعات به صورت مشخص ذخيره مي‌شوند و در مراحل بعدي در صورت لزوم دوباره به جمعيت معرفي يا تزريق مي‌شوند. در يکي ديگر هر عضو به وسيله يک حافظه اضافي از نياکانش گسترش يافته مي‌يابد. اين اعضاي ذخيره شده بعد از تشخيص تغيير در محيط دوباره مورد ارزيابي قرار مي‌گيرند و اگر بهتر از اعضاي موجود در جمعيت باشند جايگزين آنها مي‌شوند.

۴-۱-۲ بهبود روند انتخاب

بسياري از محققين براي حفظ تنوع در طول اجراي الگوريتم، بهبود روند انتخاب را توصيه مي‌کنند. روش معمول براي استفاده در بهينه سازي در محيط‌هاي ثابت استفاده از اشتراک‌گذاري است. در واقع اشتراک گذاري براي جستجوي موازي و همزمان چند پاسخ بهينه در مسائل چند قله‌اي طراحي شده‌اند. ايده اصلي اين است که افراد موجود در يک ناحيه از محيط شايستگي خود را با يکديگر به اشتراک بگذارند.

در [۱۰] يک روش اشتراک بر اساس بيت‌هاي برچسب دار آزمايش شده است. در اين روش بيت‌هاي برچسب دار به ژنوتيپ هر عضو اضافه مي‌شود و تنها اعضاي با برچسب‌هاي يکسان مي‌توانند با يکديگر عمل بازترکيبي انجام دهند. اين عمل مانند آن مي‌باشد که زيرجمعيت‌هايي با اندازه‌هاي متفاوت در نظر گرفته شود. همسايگي‌هايي که براي اشتراک گذاشتن شايستگي در نظر گرفته مي‌شوند اعضاي با برچسب‌هاي يکسان در نظر گرفته مي‌شوند.

مي‌توان شايستگي را با در نظر گرفتن طول عمر هر عضو بهبود دهيم. يعني :

[1] Multiploidy

$$f_{\mathrm{mod}} = g(f_{old}, age) \qquad (۱-۲)$$

نشان داده شده است که این روش تنوع بیشتري را نسبت به الگوریتم ژنتیک ساده بدون در نظر گرفتن عمر حفظ مي‌کند و این ممکن است که باعث تطبیق بیشتري با محیط‌هاي پویا شود.

ایده کلیدي در الگوریتم ژنتیک ترمودینامیکي[1] که ، کنترل کردن تنوع به صورت آشکار توسط کنترل معیاري که انرژي آزاد (F) نامیده مي‌شود است. براي حداقل سازي مساله این معیار به صورت زیر محاسبه مي‌شود :

$$F = \langle E \rangle - TH \qquad (۲-۲)$$

که در آن E میانگین شایستگي جمعیت و TH معیاري براي میزان تنوع در جمعیت مي‌باشد. ضریب گرما T منعکس کننده اهمیت تنوع مي‌باشد و از پارامترهاي الگوریتم مي‌باشد. در هر مرحله از الگوریتم بهترین فرد جمعیت به عنوان برگزیده[2] حفظ مي‌شود، سپس n عضو جمعیت با یکدیگر براي تولید n فرزند جفت مي‌شوند سپس جهش روي والدین و فرزندان اعمال مي‌شود. سپس از میان این $2n+1$ عضو عمل انتخاب بازماندگان براي مرحله بعد صورت مي‌گیرد. براي این انتخاب در هر مرحله انرژي F این جمعیت جدید در حال شکل گیري محاسبه مي‌شود و تعیین مي‌کند که آیا عضو i (i=1,2,….,$2n+1$) مي‌تواند به جمعیت جدید اضافه شود یا خیر. عضوي که F را کمینه مي‌کند به جمعیت جدید اضافه مي‌شود و همین روند تا زماني که جمعیت جدید شامل n عضو شود ادامه مي‌یابد.

براي محاسبه انرژي آزاد F در فرمول قبل، $\langle E \rangle$ میانگین میزان شایستگي جمعیت جدید مي‌باشد و H آنتروپي است که به صورت زیر محاسبه مي‌شود :

$$H = -\sum_{K=1}^{M} \sum_{j \in \{0,1\}} p_j^k \log p_j^k \qquad (۳-۲)$$

که در آن P_j^k نسبت وجود ژن j ام در مکان kام در جمعیت جدید مي‌باشد.

[1] Thermodynamical Genetic Algorithm (TDGA)
[2] Elite

اين روش نسبت به الگوريتم ژنتيک استاندارد در مسايلي که نرخ تغييرات محيط کوچک مي‌باشد مانند مساله کوله‌پشتي کاراتر مي‌باشد. البته محاسبات سرباري را که براي انتخاب به وجود مي‌آيد نيز بايد در نظر داشت.

۲-۱-۵ الگوريتم تکاملي با نسخه‌هاي موازي

نسخه‌هاي پراکنده[1] الگوريتم‌هاي تکاملي نسبت به يک مدل جمعيت عمومي داراي برتري مي‌باشند. علت آن به طور واضح اثبات نگرديده ولي شايد به علت همگرايي کندتر مدل پراکنده مي‌باشد. در هر صورت چون مدل پراکنده تنها مي‌تواند همگرايي را به تاخير بياندازد و از آن جلوگيري نمي‌کند، کاربرد آن براي محيط‌هاي با فرکانس تغييرات کم شايد موثر نباشد.

۲-۱-۶ مدل سازي سيستم

در اين روش از الگوريتم تکاملي جهت يافتن مدلي براي محيط مشاهده شده استفاده مي‌شود و سپس از يک روش بهينه سازي آناليزي براي يافتن پاسخ بهينه مدل داده شده استفاده مي‌کند. در اين روش پيشنهاد مي‌شود که ساخت مدل توسط الگوريتم تکاملي ي تواند حتي به صورت آنلاين نيز صورت گيرد و با تغييرات محيط خود را سازگار کند. اگر چه تمامي‌مدل‌ها قابل پياده‌سازي نمي‌باشند. کنترل کننده از يک مدل تنها تا زماني که خطاي سيستم از يک حد آستانه مشخص کمتر باشد استفاده مي‌کند. اين خطا از طريق تفاضل رفتار پيش بيني شده توسط مدل سيستم و رفتار واقعي آن بدست مي‌آيد. تنها در آن صورت يک مدل جديد از الگوريتم تکاملي تقاضا مي‌شود.

۲-۱-۷ الگوريتم ژنتيک تصادفي

در الگوريتم ژنتيک تصادفي[2] که در [۵۵] پيشنهاد شده است از ايده اتوماتاي يادگير تصادفي براي الگوريتم‌هاي ژنتيک استفاده شده است. هر کدام از اعضاي جمعيت يک عمل ممکن در نظر گرفته مي‌شود. عمل بعدي با احتمال به نسبت شايستگي هر فرد انتخاب مي‌گردد. پس از آن که يک عمل به محيط تصادفي اعمال گرديد، يک پاداش و يا تنبيه به سيستم باز گردانده مي‌شود. بر اساس اين پاسخ ميزان شايستگي تمامي‌افراد دوباره محاسبه مي‌گردد.اين عمل مشابه يک الگوي يادگيري

[1] Diffusion
[2] Stochastic Genetic Algorithm (StGA)

خطي در اتوماتاهاي يادگير تصادفي انجام مي‌گيرد كه در آن اگر يک عمل با موفقيت همراه بود پاداش داده مي‌شود و ساير افراد تنبيه مي‌شوند و در صورتي كه عمل با شكست روبرو گردد برعكس.

اين روش در محيط‌هاي تصادفي در حال تغيير با ميزان نرخ تغييرات مختلف آزمايش گرديده و نشان داده شده است كه نسبت به نسخه‌هاي ديگر اتوماتاي يادگير تصادفي با سرعت بيشتري به سمت پاسخ‌هاي بهتري همگرا مي‌شود و داراي قابليت‌هاي سازگاري بيشتري نيز مي‌باشد. همچنين از نظر تئوري اثبات گرديده كه در نهايت پاسخ بهينه را خواهد يافت.

۸-۱-۲ تكامل فيله‌اي

در [۵۶] رويكرد تكاملي‌اي جهت خوشه‌بندي داده‌هاي در حال حركت با زمان پيشنهاد شده است. در اين رويكرد عملگر بازتركيبي تنها به افراد با والدين مشترک اعمال مي‌شود. علاوه بر آن فرزندان بيشتري با استفاده از جهش‌هاي بزرگتر بر روي افراد با شايستگي كمتر توليد مي‌گردد و فرزندان كمتري نيز با استفاده از جهش‌هاي كوچک‌تر بر روي افراد با شايستگي بيشتر توليد مي‌شود. نويسنده ادعا مي‌كند كه اين رويكرد براي محيط‌هاي در حال تغيير مناسب مي‌باشد.

۹-۱-۲ سيستم ايمني مصنوعي

يک الگوريتم زيستي ديگر براي بهينه سازي الگوريتم سيستم ايمني مصنوعي مي‌باشد. در [۵۶] يک سيستم ايمني مصنوعي براي زمانبندي در محيط‌هاي پويا پيشنهاد گرديده است. در اصل سيستم ايمني مصنوعي يک كتابخانه از سري نيمه كاره خوب از كارها مي‌باشد. اين روش در دو مرحله عمل مي‌كند : در مرحله اول از يک الگوريتم ژنتيک براي تكامل سيستم ايمني مصنوعي استفاده مي‌شود بدين معني كه يک سري از برنامه ريزي‌هاي خوب را بيايد كه داراي الگوهاي مشترک در برنامه ريزي مي‌باشند. زماني كه سيستم ايمني آماده گرديد، اين زمانبندي‌هاي ناقص به طور مناسب با يكديگر تركيب مي‌شوند تا زمانبندي نهايي مناسب براي مساله مورد نظر را بيابد. مزيت اين رويكرد اينست كه مي‌توان عمليات‌هاي پيش پردازش زيادي قبل از اينكه محيط شناسايي گردد انجام داد و پس از آن ساخت زمانبندي نهايي را مي‌توان بر پايه يک منبع غني از بلوک‌هاي ذخيره شده در سيستم ايمني انجام داد. رويكردهاي ديگري نيز كه در آن‌ها نيازي به مثال‌هاي خوب زمانبندي نمي‌باشد نيز پيشنهاد شده‌اند كه كارايي مناسبي ندارند.

۲-۱-۱۰چیدمان پویاي شبکه

انجام تنظیمات و اصلاحاتي در توپولوژي به اشتراک گذاشتن اطلاعات بین موجودات مي‌تواند با قصد کاهش موقتي میل حرکت در مسیر بهترین مکان بافت شده توسط گروه، به وجود آید، که این امر سبب حفظ تنوع جمعیت مي‌شود. در الگوریتم بهینه‌ساز گروه ذرات شبکه‌اي[1] [۵۷] از یک همسایگي با ساختاري به صورت شبکه چهارخانه و در الگوریتم بهینه‌ساز گروه ذرات سلسله مراتبي[2] [۵۸] از یک ساختار سلسله مراتبي استفاده شده است و در هر دو نیز افزایش کارایي نسبت به الگوریتم بهینه‌ساز گروه ذرات[3] معمولي، در محیط تک قله‌اي پویا، گزارش نموده‌اند.

در الگوریتم بهینه‌ساز گروه ذرات شبکه‌اي، هر ذره به یک نقطه از یک شبکه چهارخانه دو بعدي نگاشت مي‌شود. هر نقطه در شبکه یک همسایگي ون نیومن دارد که شامل چهار همسایگي مي‌شود (نقاطي از شبکه در چپ، راست، بالا و پایین). با استفاده از این ساختار ارتباط موجودات محدود مي‌شود و تنها همپوشاني همسایه‌ها تنها راه ضمني براي ارتباط بین آن‌ها و انتشار اطلاعات در شبکه معرفي مي‌باشد. در نتیجه، تاثیر یک عضو مي‌تواند به صورت تدریجي در شبکه انتشار یابد که این امر باعث مي‌شود که تنوع جمعیت نسبت به سایر مدل‌هاي استاندارد الگوریتم بهینه‌ساز گروه ذرات بهتر حفظ شود.

در الگوریتم بهینه‌ساز گروه ذرات سلسله مراتبي که یک نسخه بهبودیافته‌اي از آن نیز پیشنهاد شده است [۵۹]، از یک ساختار درختي و سلسله مراتبي استفاده شده است. در این روش پس از هر تغییر، نمودار درختي به نمودارهاي درختي فرعي قسمت مي‌شود و هر قسمت مي تواند به طور مستقل براي مدتي در جستجوي یک بهینه جدید باشد تا زماني که آن ها دوباره به هم بپیوندند.

۲-۱-۱۱الگوریتم بهینه‌ساز گروه ذرات تطبیقي[4]

در [۶۰] راهبرد تنوع بخشي مجدد به جمعیت بعد از بروز تغییر در محیط را مورد بررسي قرار دادند. این مقاله یکي از اولین پژوهش‌هاي انجام شده در زمینه بهینه‌سازي با استفاده از الگوریتم بهینه‌ساز گروه ذرات در محیط‌هاي پویا محسوب مي‌شود. در این روش، شناسایي بروز تغییر در محیط با محاسبه شایستگي در یک و یا چند نقطه از حافظه‌هاي ذرات انجام مي‌گیرد. پس از شناسایي تغییر، قسمتي از خاطرات و یا همه آن‌ها به صورت تصادفي جابه‌جا مي‌شوند. از آنجا که

[1] Fine Grained PSO (FGPSO)
[2] Hierarchical Particle Swarm Optimizer
[3] Particle Swarm Optimizer
[4] Adaptive Particle Swarm Optimization

تصادفي‌سازي باعث گم شدن اطلاعات مي‌شود، خطر از دست دادن بيش از حد اطلاعات وجود دارد. به گونه‌اي كه به نظر برسد كه الگوريتم دوباره اجرا شده است و گروه هيچ خاطره‌اي ندارد. از سوي ديگر، ميزان اندک تصادفي‌سازي ممكن است باعث شود تنوع كافي براي تعقيب بهينه ايجاد نشود و گروه بهينه را از دست بدهيم. براي تنظيم مقدار پارامتر ميزان تصادفي‌سازي، مي‌بايست در مورد محيط پويا و ميزان تغييرات آن، اطلاعاتي در دست داشت.

۲-۱-۱۲ مدل‌هاي اتمي[1]

با به كارگيري برخي روش‌ها براي اعمال دافعه متقابل ميان ذرات، گروه‌ها و يا بهينه‌هاي شناخته شده، مي‌توان يک مقدار ثابت و به اندازه كافي اميدوار كننده از تنوع را در طول فرآيند جستجو حفظ كرد. عمده اين روش‌ها از مدل‌هاي اتمي الهام گرفته‌اند. در نظريات كلاسيک، در هر اتم تعدادي الكترون وجود دارد كه در فاصله‌هاي مختلف و مشخص بر حول هسته مي‌چرخند. اين تصور در نظريه كوانتوم تا حدودي متفاوت است و الكترون‌ها ديگر در مسيرهاي قطعي و معين نمي‌چرخند بلكه در يک توده احتمالي در اطراف هسته، به صورت احتمالي توزيع مي شوند.

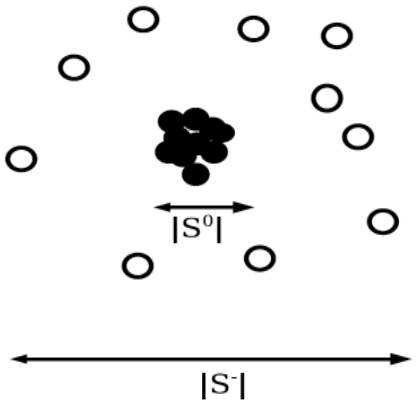
در روش‌هايي كه از اين گونه مدل‌هاي اتمي الهام گرفته شده است، هر گروه از موجودات، از يک زيرگروه باردار و يک زيرگروه خنثي تشكيل شده است كه مي‌توان آن را به صورت يک ابر از ذره‌هاي باردار كه در اطراف هسته خنثي دوران مي‌كنند نيز مجسم نمود (شكل ۲-۲). ذره‌هاي باردار باعث افزايش تنوع در اطراف زيرگروه همگرا شونده مي‌شوند، به طوري كه اگر يک بهينه در داخل اين ابر جابه‌جا شود قابل رديابي خواهد بود.

ذره‌هاي باردار مي‌توانند به هر دو صورت كلاسيک يا كوانتومي در نظر گرفته شوند. تفاوت آن‌ها در آن است كه ذرات كوانتومي بر خلاف ذرات كلاسيک از قوانين معمول جابه‌جايي ذرات استفاده نمي‌كنند بلكه به صورت تصادفي در اطراف هسته گروه در حال جستجو و حركت مي‌باشند. ذرات كوانتومي از نظر محاسباتي سريعتر هستند و حتي عملكرد بهتري نيز از خود نشان داده‌اند[۶۱]. كارايي بالاي اين مدل براي تعقيب بهينه در محيط‌هاي پوياي تک قله‌اي و با شدت تغييرات متفاوت در [۲۲, ۲۵] نشان داده شده است.

در شكل ۲-۲ يک گروه اتمي را نشان مي‌دهد كه در آن، زيرگروه ذره‌هاي خنثي (دايره‌هاي توپر)، به يک بهينه همگرا شده‌اند. قطر اين زيرگروه خنثي برابر $|S^0|$ مي‌باشد كه اندازه آن مي‌تواند

[1] Atomic Models

با یک ضریب مشخص در هر تکرار کاهش یابد. این زیرگروه با تعدادي ذره باردار احاطه شده است که شعاع این قسمت برابر $|S^-|$ است.



شکل ۲-۲) نمایش یک گروه در مدل اتمي.

ذراتي که عضو زیرگروه خنثي هستند، تحت قوانین بروزرساني متعارف، در حال حرکت مي‌باشند و اندازه هسته در اثر همگرایي ذارات به سمت بهینه به تدریج کاهش مي‌یابد. ذرات کوانتومي که عضو زیرگروه باردار هستند، از قوانین حرکتي متعارف استفاده نمي‌کنند بلکه بر طبق یک توزیع احتمال، حول هسته قرار مي‌گیرند. در نتیجه همگي آن‌ها به یك نقطه همگرا نمي‌شوند و سطح تنوع ثابتي را حفظ مي کنند. این توزیع احتمال که به مرکزیت بهترین نقطه یافت شده توسط آن گروه تعریف مي‌شود، به صورت‌هاي مختلفي قابل تعریف است از جمله توزیع گاوسي، توزیع حجم یکنواخت[۱] و توزیع حجم غیر یکنواخت[۲].

۲-۱-۱۳ الگوریتم بهینه‌ساز گروه ذرات سلولي[۳]

در این الگوریتم [۶۲] با استفاده از ایده تقسیم‌بندي فضاي جستجو و منطبق کردن یک اتوماتاي سلولي بر روي فضاي افراز شده سعي در حفظ تنوع و جستجوي مناسب در محیط‌هاي پویا شده است. هر سلول از این اتوماتاي سلولي بر اساس قوانیني که وضع شده است، وظیفه جستجو و کنترل قسمت متناظر با خود را برعهده دارد. موجودات بر حسب مکان خود در فضا، به حالت سلول‌هاي متناظر انتساب داده مي‌شوند و عمل جستجو در هر سلول و همسایگانش به صورت جداگانه و توسط الگوریتم بهینه‌ساز گروه ذرات انجام مي‌پذیرد. این جستجوي منطقه‌اي، موجب

[1] Uniform Volume Distribution (UVD)
[2] Non-Uniform Volume Distribution (NUVD)
[3] Cellular PSO

حفظ تنوع كافي براي جستجو و حفظ همزمان چندين بهينه مي‌شود. بين موجودات همسايه نيز تبادلاتي در مورد بهترين مكان‌هاي يافت شده انجام مي‌گيرد كه موجب استفاده مناسب‌تر از تجربيات سلول‌هاي همسايه و افزايش كارايي مي‌شود.

همچنين جهت جلوگيري از همگرايي بيش از حد موجودات به سمت يک بهينه، يک حد آستانه بر روي حداكثر تعداد موجودات مجاز در هر سلول گذاشته شده است و موجودات اضافي به يک سلول تصادفي در فضا ارسال مي‌شوند. حافظه‌هايي نيز براي نگهداري بهترين مكان يافت شده در هر سلول و همسايگانش تعريف شده است كه موجب انتقال تجربيات بين سلول‌ها و همچنين كاهش جستجوي‌هاي بي‌مورد و تكراري مي‌شود. اين الگوريتم كارايي نسبتا بالايي در تابع محک قله‌هاي متحرک از خود نشان داده است و مبناي روش‌هاي پيشنهادي در اين پايان‌نامه نيز مي‌باشد.

۲-۱-۱۴ روش ديده‌بانان خودسازمانده[1]

روش ديده‌بانان خودسازمانده [۴۱, ۴۲] از روش‌هاي چندجمعيتي[2] محسوب مي‌شود. ايده اصلي در اين روش اين است كه هنگامي‌كه يک بهينه محلي يافت مي‌شود جمعيت مي‌تواند به دو قسمت تقسيم شود، قسمت كوچكتر كه جمعيت فرزند نامیده مي‌شود و روي همان پاسخ بهينه مي‌ماند و به اصطلاح مي‌تواند از آن پاسخ بهينه نگهباني كند و جمعيت بزرگتر جمعيت مادر كه به جستجو براي يافتن بهينه‌هاي بعدي ادامه مي‌دهد. پس از مدتي مناطق نويد بخش توسط جمعيت‌هاي فرزند پوشش داده مي‌شوند در حالي كه جمعيت مادر به جستجوي خود براي قله‌هاي جديد ادامه مي‌دهد.

هنگامي‌كه قله‌هاي تحت نگهباني جا به جا مي‌شوند جمعيت‌هاي فرزند آن‌ها را در فضاي مساله دنبال مي‌كنند و حتي تقاضاي نيروي كمكي نيز مي‌كنند. بدين علت كه اندازه جمعيت محدود مي‌باشد، افراد به طور پيوسته بين آن جمعيت‌ها كه به نظر مي‌رسد بيشتر مورد نياز هستند تقسيم مي‌شوند و ممكن است قله‌هايي كه به نظر اميد بخش نباشند نيز رها گردد. اين روش در برخي از دسته مسايل كارايي بسيار بهتري نسبت به نسخه‌هاي مختلف الگوريتم‌هاي تكاملي از خود نشان داده است.

[1] Self Organizing Scouts (SOS)
[2] Multi-population

۲-۱-۱۵الگوريتم ژنتيک با تعادل انتقالي[1]

در اين روش [۱۹] سعي مي‌شود که با تقسيم جمعيت به يک جمعيت هسته[2] و چند کلوني[3] با جمعيت کوچکتر، قدرت اکتشاف الگوريتم حفظ شود. به نحوي که وظيفه جمعيت هسته يافتن و استخراج بهترين پاسخ بهينه است و ماموريت کلوني‌ها جستجوي نواحي ديگر فضاي مساله است که اين به قدرت اکتشاف الگوريتم کمک مي‌کند. هنگامي‌که يک کلوني خيلي به جمعيت هسته نزديک مي‌شود به وسيله انتخاب والديني با بيشترين فاصله با جمعيت هسته به سوي ديگري رانده مي‌شود. در فواصل معيني کلوني‌ها مهاجريني به جمعيت هسته مي‌فرستند و به اين وسيله آن را به روز مي‌کنند. اين روش از الگوريتم ژنتيک ساده در مسايل با تغييرات مکرر ولي با انتقال کوچک پهنه جستجو بهتر عمل مي‌کند.

۲-۱-۱۶الگوريتم ژنتيک چند مليتي[4]

در اين رويکرد [۲۰] دسته‌بندي افراد جمعيت در چندين زيرجمعيت بر اساس روندي که روند تشخيص دره[5] ناميده مي‌شود انجام مي‌گيرد. در اين روند براي دو نقطه در فضاي جستجو، شايستگي چند نقطه تصادفي از نقاط روي خط متصل کننده اين دو نقطه به يکديگر را محاسبه مي‌کنيم. يک دره در صورتي تشخيص داده مي‌شود اگر ميزان شايستگي در يکي از اين نقاط کمتر از دو نقطه پاياني باشد. توجه داشته باشيد که اين رويکرد نيازمند تعداد محاسبات شايستگي بيشتري تنها براي تشخيص دره مي‌باشد. يک ويژگي جالب استفاده از يک جهش مي‌باشد که با فاصله هر فرد از بهترين موجود در زيرجمعيت آن فرد، افزايش مي‌يابد. اين روش نسبت به روش‌هاي اشتراک‌گذاري[6] نيز کارايي بالاتري از خود نشان داده‌اند.

۲-۱-۱۷ الگوريتم بهينه‌سازي همکارانه تکاملي-بهينه‌ساز گروه ذرات[7]

اين الگوريتم [۲۰] از دو جمعيت با اندازه برابر براي شناسايي و دنبال کردن بهينه متحرک در محيط‌هاي پويا استفاده مي کند که يکي از اين جمعيت‌ها مسئوليت حفظ تنوع و ديگري وظيفه

---

[1] Shifting Balance GA
[2] Core Population
[3] Colony
[4] Multinational GA
[5] Hill-valley
[6] Sharing
[7] Collaborative Evolutionary-Swarm Optimization (CESO)

استخراج بهينه‌هاي يافت شده را بر عهده دارد. جمعيت اول از الگوريتم تكامل تفاضلي مبتني بر ازدحام[1] استفاده مي نمايد كه معمولا براي بهينه‌سازي در مسايل چند قله‌اي به كار گرفته مي‌شود. اين الگوريتم از تركيب الگوريتم تكامل تفاضلي و روش ازدحام[2] ساخته شده است و كارايي بالايي در پيدا كردن چندين بهينه به صورت همزمان در محيط ايستا از خود نشان داده است. تنها تغيير كوچكي در الگوريتم تكامل تفاضلي مورد استفاده در آن داده شده است، بدين صورت كه فرزند توليد شده الزاما جايگزين والد خود نمي‌شود بلكه جايگزين شبيه‌ترين موجود خواهد شد (البته بدين شرط كه از نظر شايستگي از آن موجود بهتر هم باشد). اين جمعيت بهينه‌هاي موجود در فضا را مي‌يابد و حفظ مي‌نمايد. در حالي كه جمعيت ديگر با استفاده از الگوريتم بهينه‌ساز گروه ذرات، اين بهينه‌هاي يافت شده را مورد استخراج قرار مي‌دهد.

در شروع هر مرحله يک آزمايش براي شناسايي تغيير در محيط صورت مي‌گيرد بدين صورت كه بهترين موقعيت يافت شده در جمعيت اول مورد ارزيابي مجدد قرار مي گيرد و اگر تغييري در ميزان شايستگي آن ايجاد شده باشد مشخص مي شود كه تغيير در محيط رخ داده است. در صورتيكه يک تغيير در محيط شناسايي شود يا فاصله بين حافظه‌هاي بهترين مكان دو جمعيت از يک حد آستانه كوچكتر شود، جمعيت تكامل تفاضلي به داخل جمعيت بهينه‌ساز گروه ذرات كپي مي‌شود و جمعيت تكامل تفاضلي از نو راه‌اندازي مي‌شود تا به دنبال بهينه‌هاي جديد در فضا بگردد. در پايان هر مرحله نيز حافظه بهترين مكان جمعيت بهينه‌ساز گروه ذرات در صورتيكه از شايستگي بالاتري برخوردار باشد، جايگزين حافظه بهترين مكان جمعيت تكامل تفاضلي مي‌شود.

۲-۱-۱۸ الگوريتم مدل همكارانه

در [۶۳] يک الگوريتم چند گروهي پيشنهاد شده است كه در آن از دو نوع گروه استفاده شده است. يكي، گروه والد كه تنوع را حفظ مي‌كند و به شناسايي نواحي نويد بخش با استفاده از يک الگورتيم برنامه‌ريزي تكاملي[3] در فضاي جستجو مي‌پردازد، و يک مجموعه از گروه‌هاي فرزند كه به اكتشاف مناطق محلي و براي يافتن بهينه‌هاي محلي به وسيله يک الگوريتم بهينه‌ساز گروه ذرات سريع[4] مي‌پردازد. اين راهبرد گروه‌هاي فرزند را بر روي چندين قله تا تعداد ممكن مي‌گستراند و همگرايي به يک بهينه محلي در يک زمان كوتاه را تضمين مي‌كند.

---

[1] Crowding Based Differential Evolution
[2] Crowding
[3] Evolutionary Programming Algorithm
[4] Fast PSO

۲-۱-۱۹الگوريتم تكامل تفاضلي پويا[1]

اين الگوريتم از جمله اولين تلاش‌هاي انجام شده براي استفاده از الگوريتم تكامل تفاضلي در محيط‌هاي پويا محسوب مي‌شود كه كارايي بالايي نيز در تابع قله‌هاي متحرک از خود نشان داده است [۲۸]. در اين روش جمعيت به چندين زيرجمعيت تقسيم مي‌شود كه هر كدام از آن‌ها با استفاه از الگوريتم تكامل تفاضلي به جستجوي يک بهينه مي‌پردازند. نيمي از جمعيت هر زيرجمعيت، به صورت كوانتومي در نظر گرفته مي‌شود و نيمه ديگر توسط روابط الگوريتم تكامل تفاضلي تكامل داده مي‌شوند.

جهت جلوگيري از همگرايي دو زيرجمعيت به يک بهينه مشترک، در هر مرحله فاصله دو به دوي تمامي آن‌ها با يكديگر سنجيده مي‌شود و اگر از يک حد آستانه مشخص كمتر بود، زيرجمعيت با شايستگي كمتر، در مرحله بعدي از نو راه‌اندازي مي‌شود. اين حد آستانه براي هر محيط پويا، بر اساس تعداد قله‌هاي موجود در فضا و تعداد ابعاد توسط يک رابطه ساده مشخص مي‌شود. پس از انجام تغيير در فضا نيز، شايستگي تمامي موجودات دوباره محاسبه خواهد شد. نسخه‌اي بهبوديافته از اين الگوريتم نيز در [۲۴] ارايه شده است كه در آن با افزودن روش‌هايي مانند مهاجرت تصادفي سعي در بهبود كارايي اين الگوريتم داشته ولي بر خلاف پيچيدگي‌هاي زيادي به آن اضافه نموده، قادر به بهبود محسوسي از نظر آماري نبوده است.

۲-۱-۲۰الگوريتم بهينه‌سازي چند گروهي خواب زمستاني[2]

در اين الگوريتم [۲۷]، جمعيت شامل يک گروه والد و تعدادي گروه فرزند مي باشد كه وظيفه گروه والد جستجوي سراسري در فضاي جستجو و پيدا كردن نواحي مطلوب در فضاي جستجو مي‌باشد در حالي كه وظيفه گروه‌هاي فرزند، جستجوي محلي براي يافتن بهينه هاي محلي مي باشد. تمامي اين گروه‌ها از الگوريتم بهينه‌ساز گروه ذرات جهت جستجو استفاده مي‌كنند. زماني كه يک ناحيه مطلوب توسط گروه والد در فضاي جستجو مورد شناسايي قرار گيرد، يک گروه فرزند به مركزيت اين موقعيت يافت شده ايجاد مي‌گردد.

همچنين گروه‌هايي كه بر روي بهينه‌هاي محلي همگرا شده‌اند را پس از مدتي غير فعال مي‌كند تا فرصت بيشتري را در اختيار ساير گروه‌ها قرار دهد تا بهينه‌هاي جديد را مورد شناسايي قرار دهند.

---

[1] DynDE
[2] Hibernating Multi-Swarm Optimization Algorithm (HmSO)

گروه‌هايي براي غيرفعال شدن انتخاب مي‌شوند كه شعاع همگرايي آن‌ها از يک حد آستانه كمتر شده باشد و بهترين گروه از نظر شايستگي در فضاي جستجو نيز نباشند. پس از روي دادن تغيير در محيط، اين گروه‌هاي غيرفعال دوباره فعال مي‌شوند و به جستجو مي‌پردازند. اين الگوريتم در محيط‌هاي توليد شده توسط تابع قله‌هاي پويا مورد آزمايش قرار گرفته است و كارايي بسيار خوبي نيز نسبت به روش‌هاي مود مقايسه از خود نشان داده است.

۲-۱-۲۱الگوريتم چند گروهي كوانتومي ذرات[1]

اين الگوريتم [۳۱] مزاياي روش دفع، چند جمعيتي و تصادفي‌سازي را با هم ادغام مي‌كند. در اين الگوريتم جمعيت به چند زير گروه تقسيم مي‌شود. هر زيرگروه شامل ذرات باردار كوانتومي و عملگرهاي دفع و ضد همگرايي است. اين ذرات باردار وظيفه حفظ حد مناسبي از تنوع حول بهينه‌هاي يافت شده در طول فرايند جستجو را به عهده دارند. عملگر دفع تعامل محلي ميان گروه‌ها را تنظيم مي‌كند، به اين صورت كه به محض آنكه فاصله دو گروه از حد آستانه معيني كمتر شد، گروه ضعيف‌تر به صورت تصادفي از نو راه‌اندازي مي‌شود. عملگر دفع نيز با ممانعت از همگرايي بيش از حد ذرات در يک نقطه به حفظ تنوع در طول فرايند بهينه سازي كمک مي‌كند.

علاوه بر اين با در نظر گرفتن اين نكته كه ممكن است تعداد قله‌هاي محيط بيش از تعداد گروه‌ها باشد، ضروري است كه تعدادي از ذرات مسئول گشت‌زني در محيط براي كشف قله‌هاي بهتر و جديدتر قرار باشند. براي اين منظور يک عملگر ضد همگرايي براي ذرات باردار در الگوريتم تعبيه شده است. در [۲۶, ۶۴, ۶۵] دو نسخه متعارف و خودتطبقي براي آن پيشنهاد شده است. در نسخه اول، با يک گروه تنها شروع مي‌كند و گروه‌هاي بعدي را بر حسب نياز و با توجه به تعداد معين و از پيش تعيين شده ذرات اضافه مي‌كند. در نسخه دوم، تعداد كلي ذرات ثابت است و ذرات به صورت پويا به زير گروه‌هايي تقسيم مي‌شوند. در واقع سعي شده است تعداد مطلوب گروه‌ها به صورت تطبيقي تنظيم شود. در هر دو روش، با كاهش تعداد ذرات كوانتوم و تبديل همه ذرات خنثي به ذرات كوانتوم براي يک تكرار بعد از بروز تغيير در محيط، سرعت همگرايي به سوي بهينه‌هاي محلي افزايش مي‌يابد.

[1] Multi Quantum Based Swarm Optimization (mQSO)

۲-۲ نتيجه‌گيري

در اين فصل مروري كلي بر روي روش‌هاي پيشنهاد شده تا كنون در زمينه بهينه‌سازي تكاملي براي محيط‌هاي پويا انجام شد. البته به علت افزايش بيش از حد روش‌هاي ارايه شده در سال‌هاي اخير، تنها بخشي از آن‌ها در حد توان مورد بررسي و شرح قرار گرفت كه كارايي بالايي از خود نشان داده‌اند و يا بيشتر مورد توجه قرار گرفته‌اند. اكثر روش‌هاي پيشنهادي تا دهه ۹۰ ميلادي، بر اساس دست‌كاري عملگرهاي جهش يا انتخاب، افزودن جمعيت‌هاي تصادفي و يا استفاده از حافظه بنا شده بودند. ولي اكثر روش‌هايي كه در سال‌هاي اخير و از سال ۲۰۰۰ به بعد پيشنهاد شده‌اند و كارايي مناسبي از خود نشان داده‌اند، از نوع الگوريتم‌هاي چند جمعيتي مي‌باشند كه در آن‌ها ايده‌هايي نظير مدل‌هاي اتمي، دافعه و تصادفي‌سازي از روش‌هاي محبوب مورد استفاده در اين الگوريتم‌ها جهت مقابله با همگرايي و حفظ تنوع محسوب مي‌شوند.

فصل سوم

الگوریتم تکامل تفاضلي سلولي

(CellularDE)

۳. الگوريتم تكامل تفاضلي سلولي[1]

تاكنون الگوريتم‌هاي تكاملي مختلفي براي بهينه‌سازي در محيط‌هاي پويا پيشنهاد شده است. در طراحي الگوريتم‌هاي مناسب براي محيط‌هاي پويا بايد دو موضوع را مورد توجه قرار داد. اول اينكه پس از مدتي الگوريتم‌هاي تكاملي معمول، دچار همگرايي شده و تنوع جمعيتي آن‌ها به شدت كاهش مي‌يابد. در صورتي كه، پس از روي دادن تغيير در محيط، نياز به تنوع كافي جهت جستجوي دوباره فضا مي‌باشد. موضوع دوم استفاده از نتايج جستجوهاي پيشين است. در محيط‌هاي پويا معمولا تغييرات بسيار زياد نيستند و مي‌توان از نتايجي كه تاكنون بدست‌آمده است براي تسريع يافتن بهينه‌هاي جديد استفاده نمود. به همين علت اكثر اوقات نياز به راه‌اندازي دوباره جستجو پس از هر تغيير نمي‌باشد.

مدل پيشنهادي ما يک الگوريتم تكاملي مبتني بر اتوماتاي سلولي براي محيط‌هاي پوياي پيوسته مي‌باشد. ايده اصلي ما براي حفظ تنوع و همچنين استفاده از نتايج پيشين جستجو، تقسيم‌بندي فضاي جستجو و انجام جستجوي محلي در اطراف هر قسمت به صورت جداگانه مي‌باشد. اين ايده در گذشته، در محيط‌هاي ايستا مورد استفاده قرار گرفته [۳۹, ۴۰] ولي در محيط‌هاي پويا، تنها در مقالات [۴۱, ۴۲] كه يک الگوريتم سلولي مبتني بر الگوريتم بهينه‌ساز گروه ذرات[2] پيشنهاد شده است، به كار رفته است.

تقسيم‌بندي فضا بدين صورت انجام مي‌گيرد كه هر كدام از ابعاد فضاي جستجو به قسمت‌هاي مساوي تقسيم مي‌شود و يک اتوماتاي سلولي بر روي اين محيط قسمت‌بندي شده، منطبق مي‌شود. به طوري كه هر سلول، وظيفه جستجو و كنترل قسمت مربوط به خود را برعهده مي‌گيرد. موجودات موجود در هر قسمت، جزيي از حالت سلول مربوطه در نظر گرفته مي‌شوند و جستجو به صورت محلي در اطراف هر سلول انجام مي‌گيرد. اين كار موجب حفظ تنوع و به نوعي تقسيم جمعيت به زيرجمعيت‌هاي كوچک‌تر مي‌شود. هر زيرجمعيت مي‌تواند به جستجو و استخراج يک بهينه جداگانه بپردازد. مزيت اين روش نسبت به اكثر روش‌هاي چند جمعيتي آنست كه در آن تعداد موجودات هر زيرجمعيت و همچنين موجودات عضو آن، از قبل تعيين شده و ثابت نمي‌باشد و زيرجمعيت‌ها بر حسب نياز و شرايط محيط شكل مي‌گيرند.

---

[1] CellularDE
[2] Particle Swarm Optimizer (PSO)

جستجو در هر سلول، با استفاده از یک الگوریتم تکاملي انجام مي‌پذيرد. با توجه به اينکه در مدل پيشنهاد شده، جستجو در هر سلول به صورت محلي انجام مي‌گيرد و نياز زيادي به اکتشاف محيط نمي‌باشد، همگرايي سريع به نزديک‌ترين بهينه از نيازهاي الگوريتم جستجوي مورد استفاده مي‌باشد. الگوريتم‌هاي تکاملي کلاسيک مبتني بر جهش و بازترکيبي به علت سرعت همگرايي پايين و قابليت جستجوي محلي کند، براي مدل پيشنهاد شده مناسب نيستند. اين امر در آزمايشات نيز مشاهده و مورد تاييد قرار گرفت. بدين منظور از الگوريتم تکامل تفاضلي استفاده شد که داراي سرعت همگرايي بالا و قدرت جستجوي محلي بسيار خوبي است. اين الگوريتم در بخش ۹-۱ توضيح داده شده است. البته تغييراتي در مدل تکاملي و همچنين روش ساخت بردار کانديد در مدل اصلي، داده شده است تا براي جستجوي محلي با تعداد موجودات کم، کارايي مناسبي داشته باشد.

علاوه بر آن براي هر سلول حافظه‌هايي تعريف مي‌شود که دربرگيرنده بهترين مکان‌هاي جستجو شده در آن سلول تا آن زمان مي‌باشد. اين حافظه‌ها به الگوريتم اجازه مي‌دهند تا بتواند از نتايج جستجوي‌هاي پيشين در هر سلول آگاه باشد و از آن‌ها استفاده نمايد. اين حافظه‌ها تا حدود زيادي از جستجوي تکراري جلوگيري به عمل مي‌آورد و همچنين اجازه مي‌دهد تا سلول‌هاي همسايه بتوانند از تجربيات جستجوهاي يکديگر استفاده کنند.

پس از هر تغيير در محيط، اين امکان وجود دارد مکان بهينه‌ها تغيير کند. براي يافتن دوباره اين بهينه‌هاي از دست رفته، از يک جستجوي محلي پس از هر تغيير استفاده مي‌شود. در الگوريتم تکامل تفاضلي سلولي اوليه از يک جستجوي تصادفي ساده در اطراف بهترين مکان‌هاي سلول،بهره برده شده است. براي بهبود اين جستجوي محلي، يک روش جستجوي محلي بهبوديافته نيز پيشنهاد شده است که تا حدود زيادي کارايي الگوريتم را بويژه در دوره‌هاي تغيير بالا، افزايش مي‌دهد.

در اين فصل مدل پايه و اوليه اين الگوريتم پيشنهادي، که نام آن را الگوريتم تکامل تفاضلي سلولي[1] مي‌ناميم، را معرفي خواهيم کرد. علاوه بر آن يک روش جستجوي بهبوديافته نيز براي آن پيشنهاد خواهيم کرد. پس از آن، آزمايشاتي جهت تنظيم و بررسي تاثير پارامترهاي الگوريتم پيشنهادي انجام خواهيم داد. در انتها کارايي روش‌هاي پيشنهاد شده را با چند الگوريتم مطرح ديگر، بر روي تابع محک قله‌هاي متحرک مورد آزمايش و مقايسه قرار مي‌دهيم. در آزمايشات مشاهده مي‌شود که روش‌هاي پيشنهادي در اکثر محيط‌هاي مورد آزمايش، از کارايي بالاتري نسبت به ساير روش‌هاي مورد مقايسه برخوردارند.

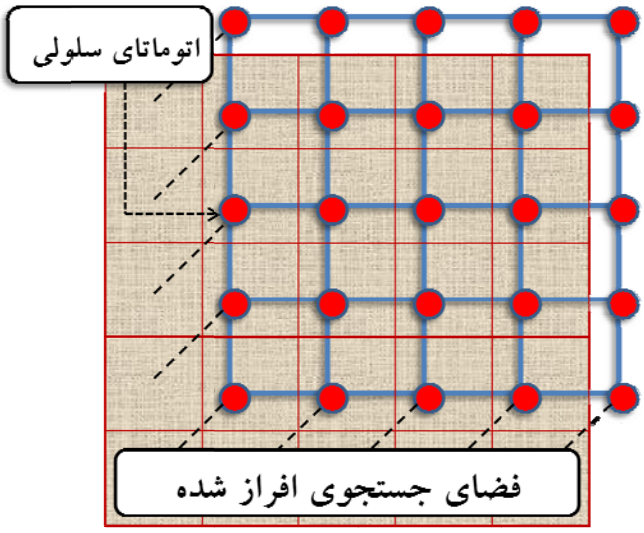
[1] CellularDE

۳-۱ روند الگوريتم

در اين بخش روند و مراحل الگوريتم پيشنهادي تكامل تفاضلي سلولي كه مدل اوليه و مبناي ساير روش‌هاي پيشنهاد شده مي‌باشد را به صورت كامل شرح و توضيح مي‌دهيم. در اين الگوريتم فضاي جستجو تقسيم‌بندي مي‌گردد و يک اتوماتاي سلولي همزمان بر آن طوري منطبق مي‌گردد كه هر سلول آن وظيفه نظارت و جستجو در يک قسمت فضا را بر عهده دارد.

- **راه‌اندازي اوليه**

اگر مساله مورد نظر d بعدي باشد و هر بعد نيز به N_p بخش مساوي و يكسان تقسيم شود، در آن صورت نياز به يک اتوماتاي سلولي d بعدي $CA=(Z^d,\varphi,N,F)$ با مجموعه سلول‌هاي $C=\{cell_i \mid 1\le i \le (N_p)^d\}$ مي‌باشد. در شكل ۳-۱، يک اتوماتاي سلولي ۵x۵ نشان داده شده است كه بر روي يک فضاي دو بعدي كه هر بعد آن به ۵ قسمت مساوي تقسيم‌بندي شده است منطبق گرديده است. به طور مثال براي يک محيط ۵ بعدي كه هر بعد آن به ۱۰ قسمت مساوي تقسيم شود، نياز به يک اتوماتاي سلولي ۵ بعدي با 10^5 سلول مي‌باشد. اين نمايي بودن تعداد سلول‌هاي مورد نياز نسبت به تعداد ابعاد محيط جستجو، مي‌تواند يكي از محدوديت‌هاي تقسيم‌بندي فضا مي‌باشد.



شكل ۳-۱) يک اتوماتاي سلولي منطبق شده بر روي يک فضاي دو بعدي تقسيم‌بندي شده

تابع Ψ كه آدرس چند بعدي يک سلول را محاسبه مي‌كند نيز بدين صورت تعريف مي‌گردد:

$$\Psi\{cell_k\} = (z_1, z_1, \cdots, z_d) \qquad (۱-۳)$$

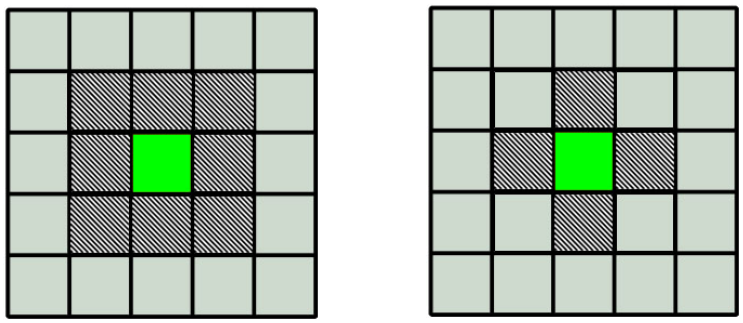
که در آن z_i بدين صورت محاسبه مي‌شوند:

$$z_1 = \lfloor k / (N_p)^{d-1} \rfloor$$
$$z_{i+1} = \left\lfloor (k - \sum_{j=1}^{i} (z_j * (N_p)^{d-j})) \Big/ (N_p)^{d-(i+1)} \right\rfloor \qquad (۲-۳)$$

تابع Ψ^{-1} نيز كه مكان خطي يک سلول را محاسبه مي‌نمايد بدين صورت تعريف مي‌شود:

$$\Psi^{-1}\{cell_{z_1, z_2, \cdots, z_d}\} = \sum_{i=1}^{d} (z_i - 1) * (N_p)^{d-i} \qquad (۳-۳)$$

براي هر سلول يک همسايگي تعريف مي‌شود كه دربرگيرنده سلول‌هايي است كه با آن سلول همسايه هستند و تمامي اعمال تكاملي بر روي آن‌ها به صورت محلي انجام مي‌پذيرد. دو نوع همسايگي متداول ون نيومن و مور مي‌باشد كه در شكل ۳-۲ براي يک فضاي دو بعدي نمونه نشان داده شده است. در اين شكل اندازه شعاع همسايگي يک در نظر گرفته شده است.



(ب) (الف)

شكل ۳-۲)انواع همسايگي‌هاي متداول (الف) همسايگي ون نيومن (ب) همسايگي مور

در الگوريتم جستجوي پيشنهادي از همسايگي مور استفاده شده است زيرا همسايگي ون نيومن بيش از حد فضاي محلي جستجو را محدود مي‌نمايد. اگر شعاع همسايگي نيز S_N در نظر گرفته شده باشد، بدين ترتيب براي سلول $cell_{z_1, z_2, \cdots, z_d}$، سلول‌هاي همسايه از رابطه (۳-۴) قابل محاسبه مي‌باشد.

$$Z(cell_{z_1,z_2,\cdots,z_d}) = \{cell_{m_1,m_2,\cdots,m_d} \mid m_k = z_1 \pm g, k = 1..d, -S_N \le g \le S_N, g \in Z\} \qquad (٤-٣)$$

در ابتدا جمعيت اوليه $P = \{ind_1, ind_2, ..., ind_M\}$ با اندازه M به صورت تصادفي و با احتمال يكنواخت توليد مي‌شود و موجودات توليد شده بر حسب مكانشان به حالت سلول‌هاي متناظرشان اضافه مي‌گردند. اگر مكان موجود iام را به صورت بردار $x_i = (x_{i1}, x_{i2}, \cdots, x_{id})$ در نظر بگيريم، در آن صورت آن را به سلول $cell_{z_1,z_2,\cdots,z_d}$ انتساب مي دهيم كه در آن مقادير z_i از رابطه زير بدست مي‌آيند:

$$z_i = \lfloor x_{ki} / N_p \rfloor + 1 \qquad (٥-٣)$$

اين موجودات بخشي از حالت سلول‌ها در نظر گرفته مي‌شوند و وظيفه جستجو در آن سلول و همسايگي آن را بر عهده دارند. تمامي روندهايي كه در مدل پيشنهادي انجام مي‌گيرد، از جمله خود الگوريتم تكامل تفاضلي نيز، همگي در قالب قوانين بروزرساني اتوماتاي سلولي تعريف مي‌شوند.

- **حافظه‌ها**

در هر گام از بروزرساني اتوماتاي سلولي، موجودات داخل هر سلول به صورت يک زيرجمعيت[1] مستقل در نظر گرفته مي‌شوند و با استفاده از الگوريتم تكامل تفاضلي كه در بخش ۹-۱ توضيح داده شد تكامل داده مي‌شوند. در مرحله انتخاب تصادفي موجودات جهت شركت در تكامل، تمامي موجودات سلول و سلول‌هاي همسايه در انتخاب شركت داده مي‌شوند. به علت خاصيت همزماني اتوماتاي سلولي همزمان، تكامل تمامي سلول‌ها به صورت موازي انجام مي‌پذيرد و موجودات حاصل از تكامل به عنوان حالت بعدي سلول‌ها در نظر گرفته مي‌شوند. جهت استفاده از نتايج جستجوهاي پيشين براي هر سلول يک حافظه نيز در نظر گرفته مي‌شود كه در آن مكان و شايستگي بهترين موجود در آن سلول از زمان آخرين تغيير تاكنون حفظ مي‌شود. اين حافظه كه براي سلول iام در مرحله tام بروزرساني ، $cMem_i^t$ ناميده مي‌شود تا حدود زيادي از جستجوهاي تكراري جلوگيري به عمل مي‌آورد كه خود اين امر موجب صرفه‌جويي در منابع مي‌گردد. اين حافظه از رابطه (۶-۳) قابل محاسبه مي‌باشد كه در آن f تابع شايستگي تابع بهينه‌سازي مي‌باشد.

$$cMem_i^t \leftarrow \underset{\forall k, ind_k \text{ is in } cell_i}{argmax} \left\{ f(ind_k), f(cMem_i^{t-1}) \right\} \qquad (٦-٣)$$

[1] Sub-population

با استفاده از اين حافظه‌ها، براي سلول iام در مرحله tام بروزرساني مقدار ديگري به نام $localBest_i^t$ بر اساس رابطه (۳-۷) تعريف مي‌شود كه نشاندهنده بهترين مكان پيدا شده در سلول iام و سلول‌هاي همسايه آن از زمان آخرين تغيير روي داده شده در محيط تاكنون است. در برخي از مدل‌هاي تكامل[1] مورد استفاده در الگوريتم تكامل تفاضلي كه بهترين مكان نيز نقش دارد از اين مكان به عنوان بهترين مكان استفاده مي‌شود.

$$localBest_i^t \leftarrow \underset{\forall j, cell_j \text{ is a neighbor of } cell_i}{argmax} \left\{ f\left(cMem_j^t\right) \right\} \qquad (۳-۷)$$

- **تكامل توسط الگوريتم تكامل تفاضلي**

براي انجام جستجو، موجودات داخل هر سلول با استفاده از الگوريتم تكامل تفاضلي تكامل داده مي‌شوند. در مرحله انتخاب موجودات براي ساخت بردار كانديد، تنها از موجودات داخل سلول استفاده مي‌شود. در الگوريتم تكامل تفاضلي استاندارد، در حد امكان از انتخاب موجود تكراري پرهيز مي‌شود ولي با توجه به محلي بودن جستجو و پخش شدن موجودات در فضا، اين امكان به علت كم بودن تعداد موجودات داخل هر سلول ميسر نمي‌باشد. براي شرح كامل الگوريتم تكامل تفاضلي استاندارد به پيوست ۹-۱ مراجعه شود.

در الگوريتم تكامل تفاضلي، تاكنون مدل‌هاي تكامل مختلفي براي توليد بردار مكان كانديد در مقالات پيشنهاد شده است كه هر كدام تاثيرات مختلفي بر روي سرعت و دقت همگرايي الگوريتم دارند. براي هر مساله، بر حسب نياز كاربر و ماهيت فضاي جستجو، يكي از اين مدل‌هاي تكامل به كار گرفته مي‌شود. با توجه به مدل پيشنهادي ما كه در آن موجودات داخل هر سلول و همسايگانش وظيفه جستجوي محلي در همان نواحي را بر عهده دارند، نياز زيادي به اكتشاف نيست و تمركز الگوريتم جستجو مي‌بايست بيشتر بر روي استخراج باشد. به همين علت از مدل‌هاي تكاملي‌اي استفاده مي‌شود كه از بهترين مكان نيز استفاده مي‌كنند. در اكثر آزمايشات براي سلول iام از يک مدل تكاملي پيشنهادي ((۳-۸)) استفاده شده است كه در آن $\overrightarrow{x_i}$ مكان موجود جاري، $\overrightarrow{x_{r1}}$، $\overrightarrow{x_{r2}}$ و $\overrightarrow{x_{r3}}$ مكان‌هاي سه موجود تصادفي از بين موجودات سلول iام و همسايگانش و λ_{rand} نيز يک عدد تصادفي مي‌باشد. اين مدل تكاملي، كه بر اساس رويه رايج براي نام‌گذاري مي‌توان آن را DE/best-to-current/1 ناميد، جستجو را حول بهترين مكان انجام مي‌دهد كه موجب بالا رفتن سرعت همگرايي الگوريتم مي‌شود. مزيت ديگر اين مدل تكاملي عدم نياز به پارامتر F مي‌باشد كه موجب كاهش تعداد پارامترهاي مورد نياز جهت تنظيم مي‌شود.

[1] Scheme

$$\vec{v_i} = localBest_j + \lambda_{rand1}.(\vec{x_i} - \vec{x_{r1}}) + \lambda_{rand2}.(\vec{x_{r2}} - \vec{x_{r3}}) \quad (۸-۳)$$

در برخي از آزمايشات بر حسب نياز، از ساير مدل‌هاي تكاملي نيز استفاده شده است. يكي از مشكلات اصلي استفاده از اين گونه از مدل‌هاي تكاملي، كه حول بهترين مكان جستجو را انجام مي‌دهند، اين است تنوع جمعيت به سرعت كاهش مي‌يابد و الگوريتم دچار همگرايي زودرس و يا سكون مي‌شود. براي كاهش اين اثر مضر، يک جهش نرمال با احتمال كوچک به الگوريتم اضافه مي‌كنيم تا با ايجاد كمي تنوع در موجودات، رخ دادن پديده‌هاي همگرايي زودرس و يا سكون را كاهش دهيم. اين جهش پس از ساختن بردار كانديد موجود جديد با استفاده از مدل تكاملي، با احتمال $\frac{1}{2}$ به يكي از ابعاد بردار كانديد به صورت تصادفي اضافه مي‌گردد. سيگماي اين جهش (δ_v) نيز يک در نظر گرفته مي‌شود.

- **جابه‌جايي بين سلول‌ها**

يكي ديگر از موضوعات، جابه‌جايي موجودات بين سلول‌ها مي‌باشد. در هنگام تكامل ممكن است مكان جديد موجودات در سلول‌هاي همسايه باشد كه در اين صورت مي‌بايست به حالت آن سلول جديد منتقل گردند. اما با توجه به محدوديت محلي بودن قوانين اتوماتاي سلولي، انتقال يک مرتبه و ناگهاني موجودات از يک سلول به سلول ديگر امكان ندارد. در مدل پيشنهاد شده براي رفع اين محدوديت، مكانيزمي پيشنهاد مي‌كنيم. مي‌بايست براي انتقال موجود از يک سلول به سلول ديگر، آن را بين سلول‌ها دست به دست كنيم تا به مقصد برسد. بدين منظور يک مجموعه قوانين به قوانين محلي هر سلول اضافه مي‌گردد تا اين عمل را پياده‌سازي نمايد. در طي اين جابه‌جايي موجودات مورد نظر به حالت غيرفعال در آورده مي‌شوند تا ديگر در تكامل شركت داده نشوند و پس از رسيدن به سلول مقصد دوباره فعال مي‌شوند.

جهت انتقال موجود در مسير مناسب به سمت سلول مقصد، براي هر موجود غيرفعال يک مشخصه به نام *des* در نظر گرفته مي‌شود كه حاوي آدرس چند بعدي سلول مقصد است. در هر بار بروز رساني، هر سلول مجموعه‌اي از تمامي موجودات غيرفعال در سلول‌هاي همسايه‌اش به غير از خودش تحت عنوان مجموعه Φ تشكيل مي‌دهد. نسخه برداري از تمامي موجودات اين مجموعه و اضافه كردن آن‌ها به موجودات غيرفعال سلول جاري، بسيار پر هزينه و بي مورد مي‌باشد. هر سلول مي‌بايست از بين اعضاي مجموعه Φ خود، موجوداتي را جهت نسخه‌برداري انتخاب كند كه تا حد امكان نسخه‌برداري‌هاي بي‌مورد كاهش يابد. يک شرط ساده و بسيار موثر اين است كه تنها موجوداتي انتخاب شوند كه در صورت اضافه شدن به سلول جاري از مقصد نهايي خود دورتر

نشوند. بدين منظور سلول $cell_{b_1,b_2,\cdots,b_d}$، موجود $Ind_j \in \Phi$ با مقصد $des_j = (c_1,c_2,\cdots,c_d)$ را كه هم اكنون در سلول همسايه $cell_{a_1,a_2,\cdots,a_d}$ است را در صورتي براي نسخه برداري انتخاب مي‌كند كه به ازاي تمامي مقادير صحيح $1 \le k \le d$، $a_k \le b_k \le c_k$ باشد. با اين عمل، نسخه‌هاي بسياري از موجود غيرفعال در فضاي اتوماتاي سلولي پخش مي‌شود و دست به دست مي‌شوند تا به سلول مقصد برسد. در هر بار بروزرساني موجودات غيرفعال داخل سلول بررسي مي‌شوند و در صورتي كه مقدار des آن‌ها با سلول فعلي يكسان باشد، فعال خواهند شد و سپس به مجموعه موجودات فعال سلول اضافه مي‌گردند تا در تكامل باز شركت داده شوند. در غير اين صورت تنها از مجموعه غيرفعال‌ها حذف مي‌شوند.

مشكل ديگر وجود نسخه‌هاي ديگري از اين موجود در ديگر سلول‌هاست كه براي حذف آن‌ها خصوصيت ديگري به نام Hop براي موجودات غيرفعال در نظر گرفته مي‌شود كه برابر با حداقل تعداد جابه‌جايي‌هايي است كه براي رسيدن موجود به مقصد نياز است. اين مقدار به شعاع همسايگي (S_N) و فاصله سلول مبدا ($cell_{s_1,s_2,\cdots,s_d}$) تا مقصد ($cell_{c_1,c_2,\cdots,c_d}$) بستگي دارد. براي موجود Ind_j، مقدار Hop_j به صورتي كه در رابطه (۳-۹) آمده است قابل محاسبه مي‌باشد.

$$Hop_j = \underset{k=1}{\overset{d}{Max}} \{|c_k - s_k|\} / S_N \qquad (۹-۳)$$

پس از نسخه برداري، يک واحد از مقدار Hop نسخه جديد كاسته مي‌شود و اگر اين مقدار منفي شد آن نسخه نابود مي‌شود. اين رويه، نسخه‌هاي اضافي پراكنده در فضاي اتوماتاي سلولي را پاک سازي خواهد كرد.

- **ظرفيت سلول**

يكي از مشكلاتي كه موجب به هدر رفتن منابع مي گردد تجمع بيش از اندازه موجودات در يک منطقه خاص است. در مدل پيشنهاد شده با توجه به ساختار محلي بودن جستجو نياز زيادي به تجمع بيش از اندازه موجودات در يک منطقه نيست. اين تجمع بي از حد نياز، موجب كاهش توانايي اكتشاف الگوريتم جستجو مي‌شود. براي كاهش اثر منفي اين پديده، بر روي تعداد موجودات فعال در هر سلول يک حد آستانه قرار داده مي‌شود. براي سلول $cell_i$ اين حد آستانه T_i ناميده مي‌شود كه در ابتداي كار الگوريتم براي تمامي سلول‌ها برابر با مقدار T_{init} قرار داده مي‌شود. اين مقدار اوليه معمولا بر اساس تعداد موجودات انتخاب مي‌گردد. اين مقدار مي‌بايست نه آنقدر كم انتخاب شود كه قدرت استخراج جستجو را كاهش دهد و نه آنقدر زياد كه موجب اتلاف منابع و در نتيجه كاهش قدرت اكتشاف الگوريتم جستجو شود. اين حد آستانه‌ها بر حسب الگوريتم مي تواند تا پايان جستجو ثابت

و يا حتي متغير در نظر گرفته شود. در الگوريتم تكامل تفاضلي سلولي، اين حد استانه براي همگي سلول‌ها ثابت و برابر در نظر گرفته مي‌شود.

در هر بار بروزرساني، هر سلول در ابتدا تعداد موجودات فعال در سلول را با حد آستانه خود مقايسه مي‌كند و در صورتي كه تعداد موجودات بيش از حد آستانه باشد مي‌بايست اين تعداد را كاهش دهد. بدين منظور از بين بدترين موجودات فعال، به تعداد موجودات اضافي حذف مي‌كند و به همان تعداد موجود جديد تصادفي توليد مي‌كند تا تعداد كل موجودات تغييري نداشته باشد. اين موجودات جديد در صورتي كه به سلول‌هاي ديگر تعلق داشته باشند، كه اكثرا هم اينگونه مي‌باشد، غيرفعال گرديده تا طي همان روند توضيح داده شده به سلول مقصد فرستاده شوند. توليد اين موجودات جديد قدرت اكتشاف الگوريتم جستجو را افزايش مي‌دهد و در نتيجه امكان يافتن مكان‌هاي جديد را بيشتر مي‌كند. مزيت ديگر اين اعمال اين محدوديت، حفظ تنوع جمعيت مي‌باشد.

- **تغيير در محيط**

ويژگي اصلي محيط‌هاي پويا تغيير مي‌باشد و موضوع مهمي كه در محيط‌هاي پويا مي‌بايست مورد توجه قرار گيرد، رويكرد مواجهه با تغييرات محيط مي‌باشد. پس از هر تغيير در محيط، پاسخ بهينه نيز ممكن است تغيير كرده باشد ولي با توجه به همگرايي جمعيت، قدرت اكتشاف جمعيت بسيار كاهش يافته و در نتيجه قادر به يافتن دوباره بهينه نمي‌باشد. تمركز عمده كارهاي پيشنهاد شده براي محيط‌هاي پويا بر روي رفع اين مشكل مي‌باشد.

هر سلول پس از تغيير در محيط، در ابتدا مقدار شايستگي موجودات داخل خود را باز محاسبه خواهد كرد. علاوه بر آن مقدار شايستگي حافظه خود را نيز بازمحاسبه خواهد كرد. البته با توجه به تعداد فوق‌العاده بالاي سلول‌ها، اين محاسبات شايستگي انجام شده براي حافظه، در تعداد محاسبات شايستگي الگوريتم شمرده نمي‌شود. البته مي توان پس از هر تغيير، حافظه را به طور كامل نابود كرد و يا تنها در سلول‌هايي كه موجودات فعال در آن‌ها وجود دارد، اين عمل بازمحاسبه را انجام داد كه اين امر موجب كاهش نسبي در كارايي الگوريتم مي‌شود.

علاوه بر آن، در هر سلول پس از شناسايي هر تغيير، يک جستجوي محلي حول حافظه $cMem$ انجام مي‌گيرد. اين جستجو در N_{LS} تكرار بعد از تغيير جهت يافتن دوباره بهينه محلي جابه‌جا شده انجام مي‌گيرد و بسيار موثر نيز واقع مي‌شود. جستجو بدين صورت انجام مي‌شود كه به ازاي هر موجود داخل سلول يک موجود تصادفي در يک ابركره[1] به مركز حافظه سلول $cMem$ و با

[1] Hyper Sphere

شعاع R_{LS} توليد مي‌شود و در صورت بهتر بودن از موجود متناظر خود جايگزين آن مي‌شود. مقدار شعاع جستجو و تعداد تكرارها بر حسب ميزان تغييرات در محيط قابل تنظيم مي‌باشد. اين جستجوي محلي اين امكان را براي موجودات اطراف يک بهينه فراهم مي‌كند كه بتوانند آن بهينه را دنبال كنند و مكان جديد آن را بيابند.

كار ديگري كه پس از هر تغيير انجام مي‌گيرد اينست كه در اولين بروزرساني پس از اتمام جستجوي محلي، مقدار ظرفيت تمامي سلول‌ها نصف مي‌گردد. اين كار بدين علت است كه جمعيت همگرا شده و اين كار با خارج كردن نيمي از موجودات داخل سلول امكان انجام يک جستجوي عمومي براي قله‌هاي جديد يا يافته نشده را فراهم مي‌كند. شبه كد الگوريتم پيشنهاد شده كه در آن ساختار كلي و روند آن را نشان مي‌دهد، در الگوريتم ۳-۱ درج شده است.

۳-۲ جستجوي محلي بهبوديافته

يكي از مراحل مهم و تاثيرگذار بر روي نتايج الگوريتم‌هاي تكاملي در محيط‌هاي پويا، روندي است كه بعد از وقوع تغيير در محيط در پيش گرفته مي‌شود. در الگوريتم پيشنهادي نيز پس از هر تغيير يک جستجوي محلي تصادفي انجام مي‌گيرد كه در افزايش كارايي الگوريتم بسيار موثر است. يكي از كارهايي كه در جهت بهبود كارايي مي‌توان انجام داد بهبود اين جستجوي محلي مي‌باشد. در اينجا يک روش جستجوي محلي ديگر پيشنهاد مي‌كنيم كه با وجودي كه به تعداد محاسبات شايستگي بالاتري نياز دارد ولي قادر است با دقت خيلي بيشتري بهينه جابه‌جا شده را بازيابد.

اين جستجوي محلي پيشنهادي، حول حافظه *cMem* هر سلول انجام مي‌پذيرد. روند كلي بدين گونه است كه براي هر بعد يک اندازه حركت و يک جهت حركت تعريف مي‌كنيم كه اندازه و جهت جستجو در آن بعد را مشخص مي‌كند. بر روي تمامي ابعاد به نوبت حركت كرده و در هر بعد در جهت مشخص شده و به اندازه مشخص شده براي آن بعد حركت مي‌كنيم. شايستگي موجود جديد بدست آمده را محاسبه كرده و در صورتي كه شايستگي بهتري داشت آن را جايگزين مكان فعلي مي‌كنيم. در غير اين صورت جهت حركت در آن بعد را معكوس كرده و در جهت جديد حركت مي‌كنيم. باز اگر شايستگي مكان جديد بهتر از فعلي بود جايگزيني انجام مي‌شود ولي در غير اين صورت اندازه حركت كوچک‌تر شده و به سراغ بعد بعدي مي‌رويم. در نتيجه در هر دور بررسي تمامي ابعاد، حداكثر نياز به محاسبات شايستگي برابر با دو برابر ابعاد مي باشد. اين روند بر روي تمامي ابعاد به دفعات انجام مي‌گيرد تا زماني كه در يک حركت، در هيچ بعدي بهبود حاصل نشود و همچنين در تمامي ابعاد به حداقل اندازه حركت رسيده باشيم.

اين جستجو تنها در داخل سلول‌هايي انجام مي‌گيرد كه در لحظه تغيير داراي موجودات فعال باشند. براي سلول $cell_i$ اين جستجوي محلي از مكان حافظه $cMem_i$ ($x_i = (x_{i1}, x_{i2}, \cdots, x_{id})$) آغاز مي‌شود. براي آن دو بردار d بعدي dir_i و sn_i تعريف مي كنيم كه به ترتيب جهت حركت و تعداد دفعات كاهش در هر بعد را مشخص مي‌كند. مقادير بردار dir_i عضو مجموعه $\{Up, Down\}$ مي‌باشند كه جهت حركت را مشخص مي كنند. در ابتدا جهت اوليه در هر بعد به صورت تصادفي در نظر گرفته مي‌شود و تعداد دفعات كاهش نيز قاعدتا برابر صفر مي‌باشد.

همچنين مقدار ss_i^k كه ميزان حركت در بعد kام مشخص مي‌كند، بر حسب تعداد دفعات كاهش، از طريق رابطه (۳-۱۰) محاسبه مي‌شود كه در آن ss_{init} مقدار اوليه اندازه حركت و a شتاب كاهش مي‌باشد. اين مقدار اوليه (ss_{init}) براي تمامي ابعاد در ابتدا يكسان در نظر گرفته مي‌شود.

$$ss_i^k = ss_{init} \times a^{sn_{ik}} \qquad (۳-۱۰)$$

بدين صورت در ابتدا با گام‌هاي بلند در يک جهت حركت مي‌كنيم تا زماني كه ديگر بهبودي حاصل نشود. در آن صورت جهت ديگر را آزمايش مي‌كنيم و اگر در آن جهت هم موفق نشديم مقدار تعداد دفعات كاهش براي آن بعد را يک واحد افزايش مي‌دهيم تا در دور بعدي حركت‌هاي كوچكتري انجام شود. براي اين تعداد كاهش‌ها در اندازه حركت مي بايست حداكثري تعيين شود تا روند پايان‌پذير باشد. اين حداكثر تعداد M_{sn} ناميده مي‌شود و در صورتي كه در هر بعد به اين مقدار برسيم ديگر حركتي در آن بعد انجام نمي‌شود. همچنين اگر در تمامي ابعاد به اين حداكثر برسيم و بهبودي در هيچ بعدي حاصل نشود، روند جستجو پايان پذيرفته است. هر چه تعداد كاهش‌هاي مجاز را بيشتر در نظر بگيريم دقت بهينه يافت شده بالاتر مي‌رود ولي از طرفي تعداد محاسبات شايستگي مورد نياز نيز به صورت نمايي افزايش مي‌يابد. معمولا به همين علت اين مقدار كم در نظر گرفته مي‌شود.

اگر مقدار ss_{init} بيش از حد بزرگ انتخاب شود، موجب بزرگ شدن گام‌هاي اوليه مي‌شود كه در نتيجه سرعت رسيدن به بهينه افزايش مي‌يابد. ولي در زمان‌هايي كه به بهينه نزديک هستيم بزرگ گرفتن آن موجب اتلاف محاسبات شايستگي مي شود. كوچک گرفتن بيش از حد آن نيز موجب كندي حركت و در نتيجه استفاده زياد از محاسبات شايستگي مي‌شود كه مسلما مطلوب نيست. شبه كد اين جستجوي محلي پيشنهادي براي سلول $cell_i$ در الگوريتم ۳-۲ درج شده است.

روش جستجوي پيشنهادي دقت دنبال كردن بهينه را بسيار افزايش مي‌دهد و حتي مي‌توان دقت بهينه دنبال شده را نيز با دقت خوبي تنظيم نمود. كارايي اين جستجو مخصوصا براي محيط‌هايي

که در آن‌ها بين ابعاد وابستگي وجود ندارد بسيار بالاست، زيرا اين روش از عدم وابستگي بين ابعاد بسيار سود مي‌برد. از معايب آن نيز مي‌توان به تعداد زياد محاسبات شايستگي مورد نياز براي انجام آن نام برد. نام نسخه الگوريتمي كه از اين جستجوي بهبود يافته براي دنبال كردن بهينه‌هاي در حال حركت استفاده مي‌كند را الگوريتم تكامل تفاضلي سلولي بهبود يافته (CellularDE+) مي‌ناميم.

الگوریتم ۳-۱) شبه کد الگوریتم CellularDE

Algorithm CellularDE

Initialize a cellular automaton with $(N_p)^d$ equal-sized cells
Initialize population of size M randomly in the cellular automaton
do
  for all $cell_i$ in cellular automaton **do**
    Update $cMem_i$ according to eq. (۶-۳)
    Calculate $localBest_i$ according to eq. (۷-۳)
    if there was a change in the past N_{LS} iterations **then**
      //Comment: Local Search Section
      for all $individual_m$ in $cell_i$ **do**
        Set $individual_m$ to a random position in a hyper-sphere with radius LS_r centered at $cMem_i$ if that random position has better fitness.
      end-for
    else
      Evolve active population in $cell_i$ by the modified version of DE
    end-if

    if it is the first iteration after the local search **then**
      $T = T_i / 2$
    else
      $T = T_i$
    end-if
    while population of $cell_i > T$
      Re-initialize the worst active individual to a random cell
    end-while

    if a change is detected in the environment **then**
      Re-evaluate $cMem_i$
      for all $individual_m$ in $cell_i$ **do**
        Re-evaluate $individual_m$
      end-for
    end-if
  end-for
until a termination condition is met

الگوريتم ۳-۲) شبه کد روش جستجوي محلي بهبوديافته

Algorithm Improved Local Search

$sn_i = (0,0,\ldots,0)_d$

$x_i =$ position of $cMem_i$

do

  for *k=1* **to** *d* **do**

    if $sn_i^k > M_{sn}$ **then**

      continue

    end-if

    $x_{tmp} = x_i$

    $ss_i^k = ss_{init} \times b^{sn_{ik}}$

    Move x_{tmp}^k in the direction according to dir_i^k with the size of ss_i^k

    if $fit(x_{tmp}) > fit(x_i)$ **then**

      $x_i = x_{tmp}$

    else

      $x_{tmp} = x_i$

      Move x_{tmp}^k in the opposite direction of dir_i^k with the size of ss_i^k

      if $fit(x_{tmp}) > fit(x_i)$ **then**

        $x_i = x_{tmp}$

        flip the direction of dir_i^k

      else

        sn_i^k ++

      end-if

    end-if

until $sn_i^k > M_{sn}$ and there was no improvement in any dimensions

۳-۳ آزمايشات

همان‌طور که در بخش ۱-۵ گفته شد براي محيط‌هاي پويا توابع محک متفاوتي تا کنون پيشنهاد شده است که همگي آن‌ها براي محيط‌هاي پيوسته طراحي نشده‌اند. از بين آن‌هايي که براي محيط‌هاي پيوسته طراحي شده‌اند، تابع محک قله‌هاي متحرک[1] از رايج‌ترين و کامل‌ترين آن‌هاست که مورد استقبال فراواني قرار گرفته است و اکثر کارهاي پيشنهاد شده بر روي اين تابع محک مورد آزمايش و ارزيابي قرار گرفته‌اند [۲۲-۳۴, ۳۷, ۳۸, ۴۱, ۴۲, ۵۰, ۶۴-۶۹].

[1] Moving Peaks Benchmark

همانطور که در بخش ۱-۵-۵ توضیح داده شده است این تابع شامل یک محیط چند بعدي است که داراي چندین قله مي‌باشد. هر قله مي‌تواند داراي مکان، ارتفاع، عرض تصادفي یا مشخص شده باشد و هنگامي که یک تغییر در محیط رخ دهد تمامي مشخصات قله‌ها قابل تغییر مي‌باشد. این تابع این قابلیت را دارد که ابعاد محیط، تعداد قله‌ها، شکل قله‌ها، میزان شدت تغییرات و دوره تغییرات تنظیم شود. این تابع داراي چند سناریوي از پیش تعریف شده و استاندارد مي‌باشد که در بین آن‌ها سناریوي دو از بقیه رایج‌تر است. تنظیمات و مشخصات محیط این سناریو که در آزمایشات به عنوان پیش فرض در نظر گرفته مي‌شود، در جدول ۳-۱ آمده است.

همچنین براي خود الگوریتم تکامل تفاضلي سلولي و جستجوي بهبودیافته نیز در تمامي آزمایشات پیش‌فرض‌هایي در نظر گرفته شده است که در جدول ۳-۲ درج شده‌اند. این مقادیر از طریق سعي و خطا براي محیط سناریو دو بدست آورده شده‌اند. در تمامي آزمایشات، میانگین خطاي آفلاین صد مرتبه اجرا گزارش شده است. در هر بار اجرا نیز تا ۵۰۰۰۰۰ بار محاسبه شایستگي الگوریتم‌ها اجرا خواهد شد.

جدول ۳-۱) مقادیر استاندارد پیش‌فرض براي تابع قله‌هاي متحرک (سناریوي دو)

پارامتر	مقدار
تعداد قله‌ها (m)	۱۰
دوره تغییرات (f_s)	۵۰۰۰
شدت تغییرات ارتفاع قله‌ها	۷
شدت تغییرات عرض قله‌ها	۱
شکل قله‌ها	مخروطي
حداکثر حرکت مکان قله‌ها	۱
ابعاد فضاي جستجو	۵
محدوده ارتفاع قله‌ها	[۳۰ ۷۰]
محدوده عرض قله‌ها	[۱ ۱۲]
ارتفاع استاندارد قله‌ها	۵۰
محدوده ي فضاي جستجو	[۰ ۱۰۰]

جدول ۳-۲) مقادير پيش‌فرض براي پارامترهاي الگوريتم پيشنهادي

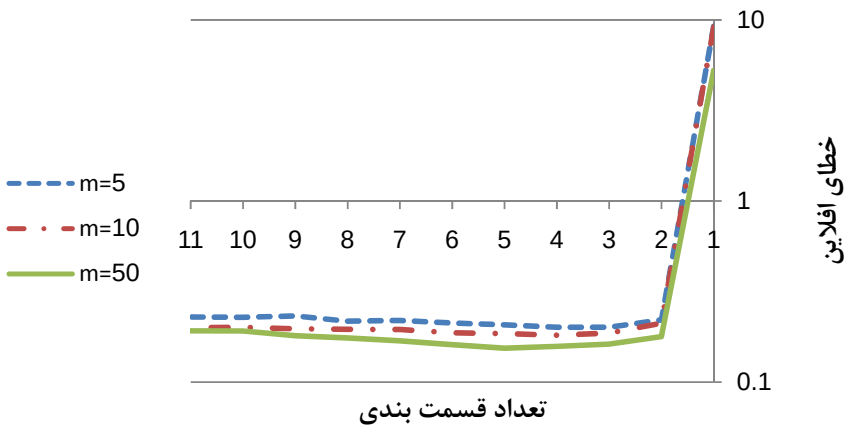
پارامتر	مقدار
اندازه جمعيت (M)	۱۰۰
تعداد تقسيم‌بندي ابعاد (N_p)	۱۰
نوع همسايگي	مور
شعاع همسايگي (S_N)	۲
ظرفيت اوليه سلول (T_{init})	۱۰
پارامتر F	0.2
پارامتر Cr	0.4
تعداد دفعات تكرار جستجوي محلي براي الگوريتم تكامل تفاضلي سلولي (N_{LS})	۳
تعداد دفعات تكرار جستجو براي جستجوي محلي بهبود يافته (N_{LS})	۱
شعاع جستجوي محلي(R_{LS})	۱
مقدار اوليه اندازه جابه‌جايي در روش جستجوي بهبود يافته (ss_{init})	۴
شتاب كاهش اندازه حركت در روش جستجوي بهبوديافته (a)	۰.۲
حداكثر تعداد دفعات كاهش در روش جستجوي بهبوديافته (M_{sn})	۴

۳-۳-۱ آزمايش تاثير تعداد تقسيم‌بندي فضا بر روي كارايي با اندازه همسايگي ۱

در اين آزمايش تاثير ميزان تعداد تقسيم‌بندي ابعاد محيط بر روي كارايي الگوريتم تكامل تفاضلي سلولي براي اندازه شعاع همسايگي ۱ مورد بررسي قرار داده مي‌شود. در تمامي آزمايشات از سناريوي دو تابع قله‌هاي متحرک با شعاع همسايگي يک، دوره تغيير ۵۰۰۰ و ابعاد مساله از يک تا پنج استفاده شده است. با توجه به احتمال وجود ارتباط بين تعداد قله‌ها و تعداد تقسيم‌بندي‌ها، در هر بعد آزمايش بر روي تعداد قله‌هاي ۵، ۱۰ و ۵۰ كه به ترتيب نماينده مسائل با تعداد قله‌هاي كم، متوسط و زياد هستند انجام شده است. نتايج اين آزمايشات به ترتيب براي ابعاد يک تا پنج در جدول ۳-۳ تا جدول ۳-۷ و شكل ۳-۳ تا شكل ۳-۷ درج شده است.

جدول ۳-۳) خطاي آفلاين به ازاي تعداد فلههاي مختلف و تعداد تقسيمبنديهاي مختلف براي محيط يک بعدي (اندازه همسايگي ۱)

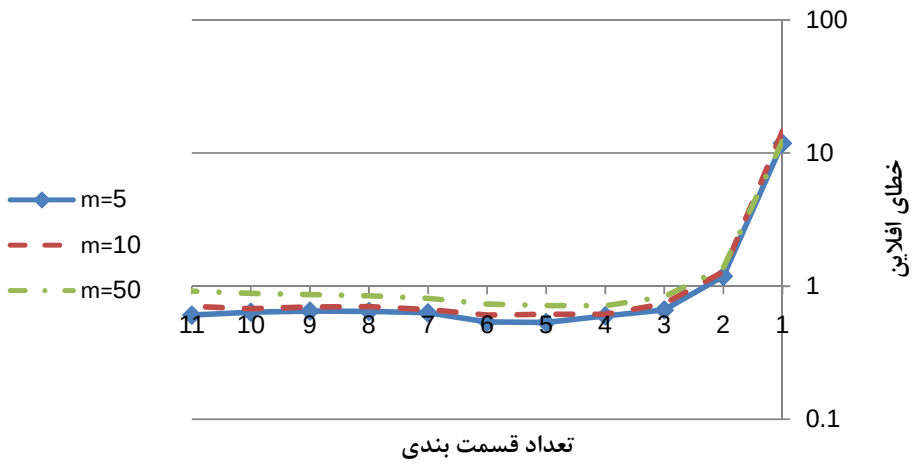
	تعداد فلهها		
تعداد تفسيمبندي	۵	۱۰	۵۰
۱	9.22	9.13	5.26
۲	0.22	0.21	0.18
۳	0.20	0.19	0.16
۴	0.20	0.18	0.16
۵	0.21	0.19	0.15
۶	0.21	0.19	0.16
۷	0.22	0.20	0.17
۸	0.22	0.20	0.18
۹	0.23	0.20	0.18
۱۰	0.23	0.20	0.19
۱۱	0.23	0.20	0.19



شکل ۳-۳) نمودار خطاي آفلاين به ازاي تعداد فلههاي مختلف و تعداد تقسيمبنديهاي مختلف براي محيط يک بعدي (اندازه همسايگي ۱)

جدول ۳-٤) خطاي آفلاين به ازاي تعداد فلههاي مختلف و تعداد تقسيمبنديهاي مختلف براي محيط دو بعدي (اندازه همسايگي ۱)

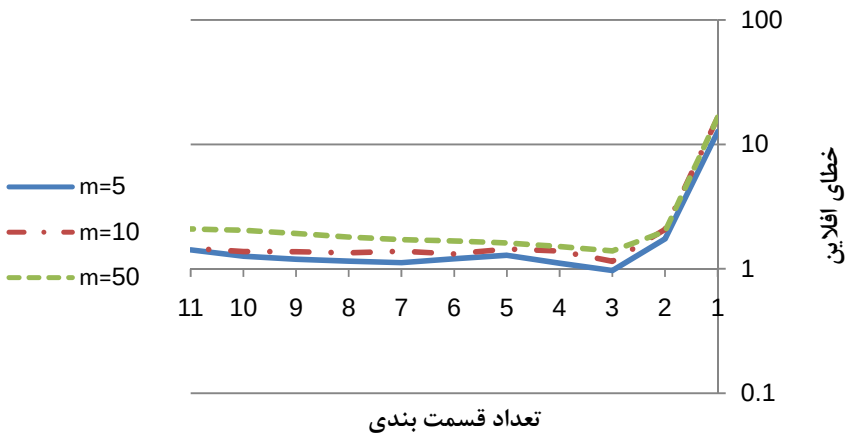
	تعداد فلهها		
تعداد تقسيمبندي	۵	۱۰	۵۰
۱	۱۱.۸۶	۱۴.۵۶	۱۲.۲۷
۲	۱.۱۹	۱.۲۹	۱.۳۷
۳	۰.۶۶	۰.۷۴	۰.۸۴
۴	۰.۶۰	۰.۶۲	۰.۷۱
۵	۰.۵۳	۰.۶۲	۰.۷۲
۶	۰.۵۴	۰.۶۱	۰.۷۴
۷	۰.۶۳	۰.۶۷	۰.۸۱
۸	۰.۶۵	۰.۷۰	۰.۸۵
۹	۰.۶۵	۰.۷۰	۰.۸۷
۱۰	۰.۶۴	۰.۶۸	۰.۸۸
۱۱	۰.۶۱	۰.۷۱	`.۰



شکل ۳-٤) نمودار خطاي آفلاين به ازاي تعداد فلههاي مختلف و تعداد تقسيمبنديهاي مختلف براي محيط دو بعدي (اندازه همسايگي ۱)

جدول ۳-۵) خطاي آفلاين به ازاي تعداد فلههاي مختلف و تعداد تقسيمبنديهاي مختلف براي محيط سه بعدي (اندازه همسايگي ۱)

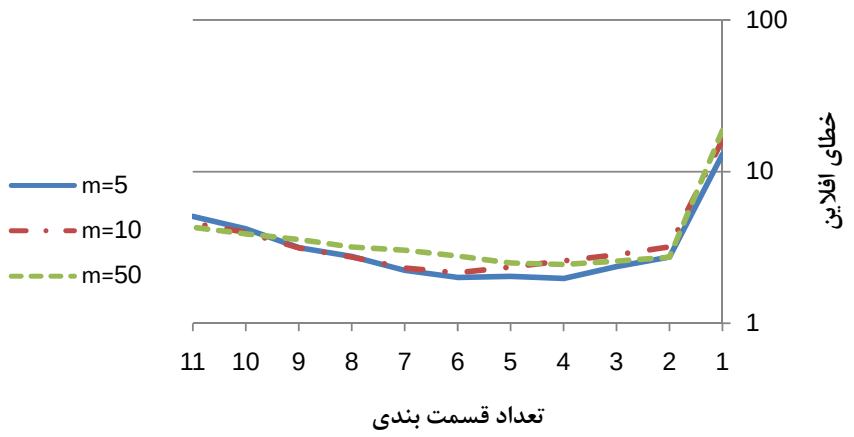
	تعداد فلهها		
تعداد تقسيمبندي	۵	۱۰	۵۰
۱	۱۲.۷۰	۱۶.۴۲	۱۶.۵۲
۲	۱.۷۴	۲.۰۸	۲.۰۰
۳	۰.۹۷	۱.۱۵	۱.۳۹
۴	۱.۱۱	۱.۳۹	۱.۵۱
۵	۱.۲۹	۱.۴۴	۱.۶۲
۶	۱.۲۱	۱.۳۲	۱.۶۷
۷	۱.۱۲	۱.۳۸	۱.۷۲
۸	۱.۱۵	۱.۳۵	۱.۸۰
۹	۱.۱۹	۱.۳۸	۱.۹۳
۱۰	۱.۲۶	۱.۳۸	۲.۰۴
۱۱	۱.۴۲	۱.۴۶	۲.۰۹



شکل ۳-۵) نمودار خطاي آفلاين به ازاي تعداد فلههاي مختلف و تعداد تقسيمبنديهاي مختلف براي محيط سه بعدي (اندازه همسايگي ۱)

جدول ۳-٦) خطاي آفلاين به ازاي تعداد فله‌هاي مختلف و تعداد تقسيم‌بندي‌هاي مختلف براي محيط چهار بعدي (اندازه همسايگي ۱)

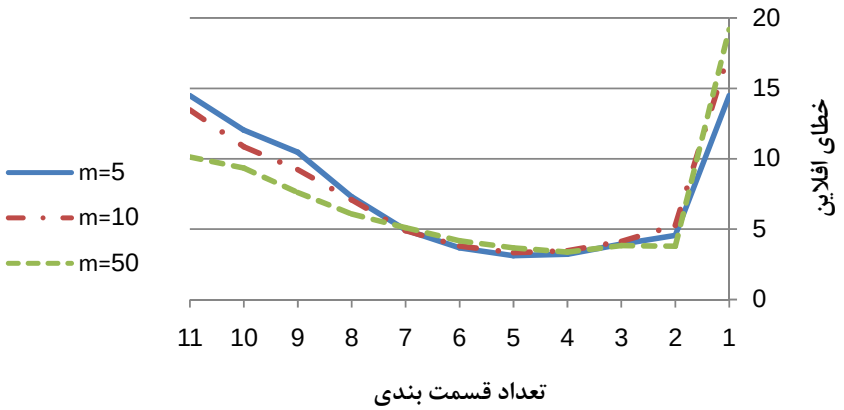
	تعداد فله‌ها		
تعداد تقسيم‌بندي	۵	۱۰	۵۰
۱	۱۲.۹۹	۱۶.۱۷	۱۸.۶۵
۲	۲.۷۳	۳.۲۰	۲.۷۳
۳	۲.۳۶	۲.۸۳	۲.۵۷
۴	۱.۹۷	۲.۵۸	۲.۴۴
۵	۲.۰۴	۲.۳۵	۲.۵۰
۶	۲.۰۰	۲.۱۵	۲.۷۸
۷	۲.۲۳	۲.۳۱	۳.۰۳
۸	۲.۷۵	۲.۷۳	۳.۱۸
۹	۳.۱۵	۳.۱۴	۳.۵۷
۱۰	۴.۱۹	۳.۹۷	۳.۸۸
۱۱	۵.۰۶	۴.۵۸	۴.۳۰



شكل ۳-٦) نمودار خطاي آفلاين به ازاي تعداد فله‌هاي مختلف و تعداد تقسيم‌بندي‌هاي مختلف براي محيط چهار بعدي (اندازه همسايگي ۱)

جدول ۳-۷) خطاي آفلاين به ازاي تعداد قله‌هاي مختلف و تعداد تقسيم‌بندي‌هاي مختلف براي محيط پنج بعدي (اندازه همسايگي ۱)

تعداد تقسيم‌بندي	۵	۱۰	۵۰
۱	۱۴.۴۹	۱۷.۵۹	۱۹.۱۹
۲	۴.۵۶	۵.۲۷	۳.۷۹
۳	۳.۹۴	۴.۱۳	۳.۸۳
۴	۳.۲۲	۳.۴۷	۳.۳۷
۵	۳.۱۲	۳.۳۳	۳.۶۷
۶	۳.۶۷	۳.۸۰	۴.۱۷
۷	۴.۹۵	۴.۸۹	۵.۱۰
۸	۷.۳۱	۷.۰۸	۶.۰۸
۹	۱۰.۴۶	۹.۲۳	۷.۶۰
۱۰	۱۲.۰۳	۱۰.۸۶	۹.۳۴
۱۱	۱۴.۴۸	۱۳.۴۶	۱۰.۱۳



شكل ۳-۷) نمودار خطاي آفلاين به ازاي تعداد قله‌هاي مختلف و تعداد تقسيم‌بندي‌هاي مختلف براي محيط پنج بعدي (اندازه همسايگي ۱)

همان طور كه در جداول و نمودارها قابل مشاهده است، تعداد تقسيم‌بندي بر روي كارايي الگوريتم تا حدود تاثيرگذار است و انتخاب مناسب آن براي هر مساله مي‌تواند بر روي نتايج موثر باشد. براي حالتي كه تقسيم‌بندي وجود ندارد، ميزان كاراي بسيار پايين است و اين امر با توجه به نياز زياد به حفظ تنوع طبيعي مي‌باشد. اين نتايج نشان از اهميت و تاثير بسزايي عمل تقسيم‌بندي بر روي كارايي مي‌باشد.

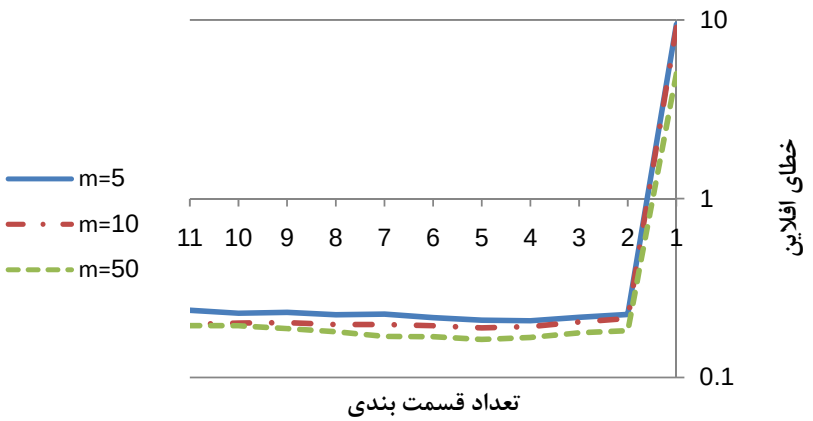
در تمامي نمودارها، براي تمامي ابعاد، در ابتدا با زياد شدن تعداد تقسيم‌بندي‌ها به طور كلي كارايي افزايش مي‌يابد ولي پس از آن با زياد شدن بيش از حد اين تعداد، افزايش كارايي متوقف و حتي تا حدودي كاهش نيز مي‌يابد. اين امر بدين علت است كه افزايش اوليه تعداد تقسيم‌بندي‌ها در ابتدا با كمک به حفظ تنوع موجب افزايش كارايي مي‌شود ولي افزايش بيش از حد اين تقسيم‌بندي موجب كوچكي بيش از حد محدوده جستجوي محلي و در نتيجه كند شدن فرآيند همگرايي الگوريتم مي‌شود. اين پديده كاهش كارايي با افزايش تعداد ابعاد بيشتر قابل مشاهده است زيرا با افزايش تعداد تقسيم‌بندي‌ها، تعداد سلول‌ها به صورت نمايي افزايش مي‌يابد كه اين امر در ابعاد بالاتر بيشتر خود را نشان مي‌دهد.

۳-۳-۲ آزمايش تاثير تعداد تقسيم‌بندي فضا بر روي كارايي با اندازه همسايگي ۲

در اين آزمايش تاثير ميزان تعداد تقسيم‌بندي ابعاد محيط بر روي كارايي الگوريتم تكامل تفاضلي سلولي براي اندازه شعاع همسايگي ۲ مورد بررسي قرار داده مي‌شود. در تمامي آزمايشات از سناريوي دو تابع قله‌هاي متحرک با شعاع همسايگي دو، دوره تغيير ۵۰۰۰ و ابعاد مساله از يک تا پنج استفاده شده است. با توجه به احتمال وجود ارتباط بين تعداد قله‌ها و تعداد تقسيم‌بندي‌ها، در هر بعد آزمايش بر روي تعداد قله‌هاي ۵، ۱۰ و ۵۰ كه به ترتيب نماينده مسائل با تعداد قله‌هاي كم، متوسط و زياد هستند انجام شده است. نتايج اين آزمايشات به ترتيب براي ابعاد يک تا پنج در جدول ۳-۸ تا جدول ۳-۱۲ و شكل ۳-۸ تا شكل ۳-۱۲ درج شده است.

جدول ۳-۸) خطاي آفلاين به ازاي تعداد قله‌هاي مختلف و تعداد تقسيم‌بندي‌هاي مختلف براي محيط يک بعدي (اندازه همسايگي ۲)

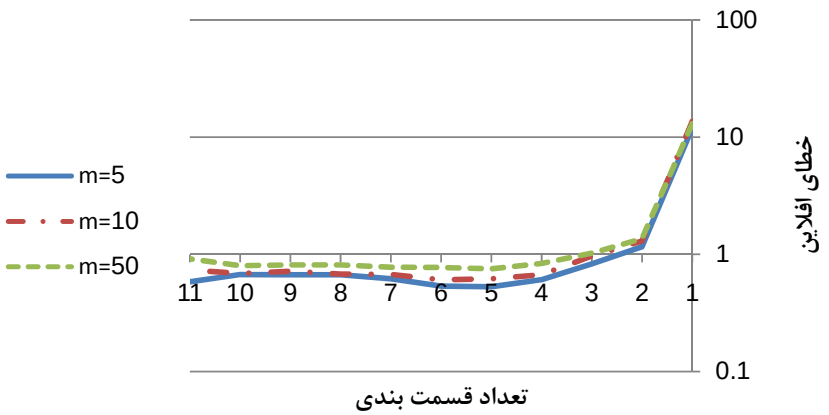
تعداد تقسيم‌بندي	۵	۱۰	۵۰
۱	۹.۴۴	۹.۰۹	۵.۰۱
۲	۰.۲۳	۰.۲۱	۰.۱۸
۳	۰.۲۲	۰.۲۰	۰.۱۸
۴	۰.۲۱	۰.۱۹	۰.۱۷
۵	۰.۲۱	۰.۱۹	۰.۱۶
۶	۰.۲۲	۰.۱۹	۰.۱۷
۷	۰.۲۳	۰.۲۰	۰.۱۷
۸	۰.۲۲	۰.۲۰	۰.۱۸
۹	۰.۲۳	۰.۲۰	۰.۱۹
۱۰	۰.۲۳	۰.۲۰	۰.۱۹
۱۱	۰.۲۴	۰.۲۰	۰.۱۹



شکل ۳-۸) نمودار خطاي آفلاين به ازاي تعداد قله‌هاي مختلف و تعداد تقسيم‌بندي‌هاي مختلف براي محيط يک بعدي (اندازه همسايگي ۲)

جدول ۳-۹) خطاي آفلاين به ازاي تعداد قله‌هاي مختلف و تعداد تقسيم‌بندي‌هاي مختلف براي محيط دو بعدي (اندازه همسايگي ۲)

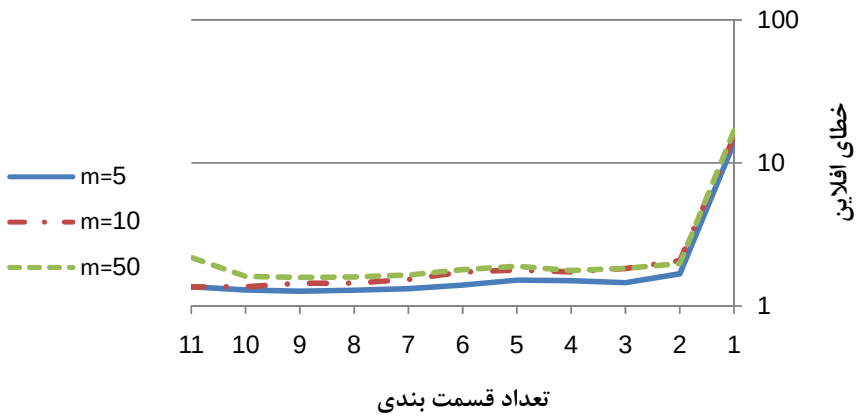
تعداد تقسيم‌بندي	۵	۱۰	۵۰
۱	۱۱.۹۵	۱۳.۶۷	۱۳.۰۲
۲	۱.۱۶	۱.۳۰	۱.۳۶
۳	۰.۸۳	۰.۹۷	۱.۰۲
۴	۰.۶۱	۰.۶۷	۰.۸۴
۵	۰.۵۳	۰.۶۲	۰.۷۵
۶	۰.۵۴	۰.۶۱	۰.۷۷
۷	۰.۶۲	۰.۶۸	۰.۷۸
۸	۰.۶۷	۰.۶۸	۰.۸۱
۹	۰.۶۷	۰.۷۲	۰.۸۱
۱۰	۰.۶۷	۰.۶۹	۰.۸۰
۱۱	۰.۵۹	۰.۷۴	۰.۹۱



شكل ۳-۹) نمودار خطاي آفلاين به ازاي تعداد قله‌هاي مختلف و تعداد تقسيم‌بندي‌هاي مختلف براي محيط دو بعدي (اندازه همسايگي ۲)

جدول ۳-۱۰) خطاي آفلاين به ازاي تعداد فله‌هاي مختلف و تعداد تقسيم‌بندي‌هاي مختلف براي محيط سه بعدي (اندازه همسايگي ۲)

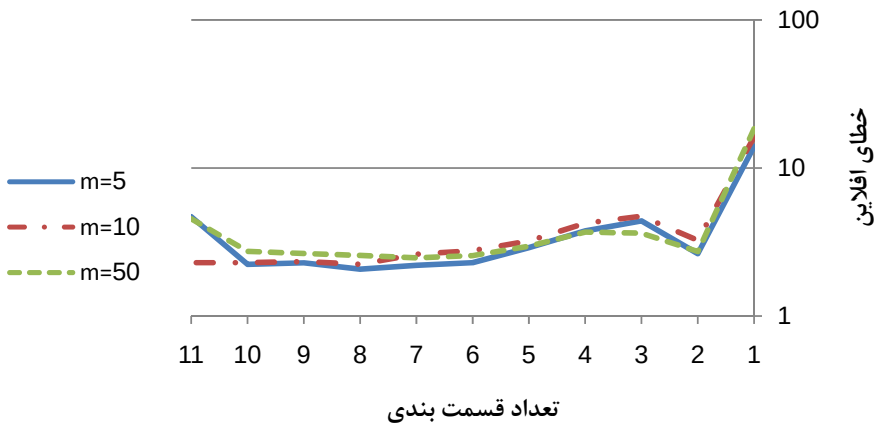
	تعداد فله‌ها		
تعداد تقسيم‌بندي	۵	۱۰	۵۰
۱	۱۳.۸۵	۱۵.۲۲	۱۶.۷۸
۲	۱.۶۸	۲.۰۷	۱.۹۸
۳	۱.۴۵	۱.۸۲	۱.۸۳
۴	۱.۵۰	۱.۷۳	۱.۷۷
۵	۱.۵۱	۱.۷۸	۱.۸۹
۶	۱.۴۰	۱.۷۳	۱.۸۰
۷	۱.۳۲	۱.۵۳	۱.۶۵
۸	۱.۲۹	۱.۴۴	۱.۵۹
۹	۱.۲۶	۱.۴۴	۱.۵۸
۱۰	۱.۲۹	۱.۳۶	۱.۶۱
۱۱	۱.۳۶	۱.۳۶	۲.۱۸



شكل ۳-۱۰) نمودار خطاي آفلاين به ازاي تعداد فله‌هاي مختلف و تعداد تقسيم‌بندي‌هاي مختلف براي محيط سه بعدي (اندازه همسايگي ۲)

جدول ۳-۱۱) خطاي آفلاين به ازاي تعداد فلههاي مختلف و تعداد تقسيمبنديهاي مختلف براي محيط چهار بعدي (اندازه همسايگي ۲)

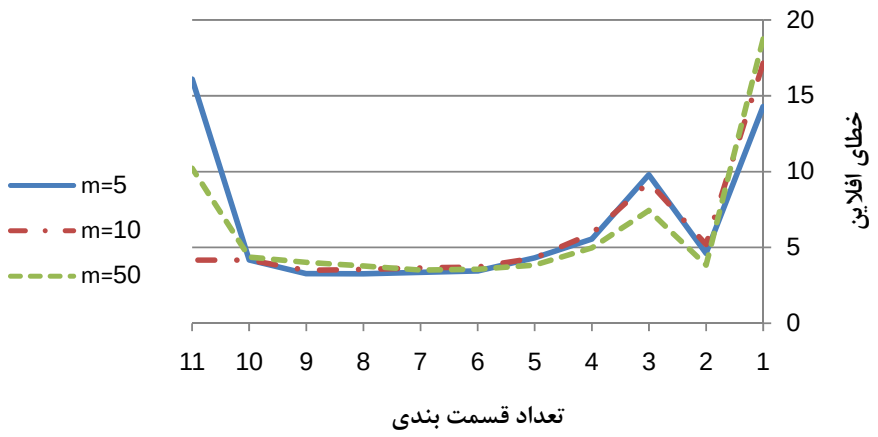
	تعداد فلهها		
تعداد تقسيمبندي	۵	۱۰	۵۰
۱	۱۳.۹۹	۱۶.۲۲	۱۸.۴۴
۲	۲.۶۴	۳.۲۴	۲.۷۲
۳	۴.۳۹	۴.۷۲	۳.۶۳
۴	۳.۷۶	۴.۲۳	۳.۶۸
۵	۲.۹۰	۳.۲۱	۲.۹۵
۶	۲.۳۰	۲.۷۷	۲.۵۶
۷	۲.۲۰	۲.۶۱	۲.۴۷
۸	۲.۰۷	۲.۲۴	۲.۵۶
۹	۲.۲۹	۲.۳۳	۲.۶۴
۱۰	۲.۲۳	۲.۲۹	۲.۷۳
۱۱	۴.۶۹	--	۴.۵۳



شکل ۳-۱۱) نمودار خطاي آفلاين به ازاي تعداد فلههاي مختلف و تعداد تقسيمبنديهاي مختلف براي محيط چهار بعدي (اندازه همسايگي ۲)

جدول ۳-۱۲) خطاي آفلاين به ازاي تعداد فله‌هاي مختلف و تعداد تقسيم‌بندي‌هاي مختلف براي محيط پنج بعدي (اندازه همسايگي ۲)

	تعداد فله‌ها		
تعداد تقسيم‌بندي	۵	۱۰	۵۰
۱	۱۴.۲۹	۱۷.۱۵	۱۸.۷۴
۲	۴.۶۱	۵.۱۹	۳.۸۳
۳	۹.۷۹	۹.۳۱	۷.۴۴
۴	۵.۵۶	۵.۹۱	۴.۹۷
۵	۴.۳۱	۴.۲۹	۳.۸۴
۶	۳.۴۴	۳.۷۰	۳.۵۶
۷	۳.۳۶	۳.۶۴	۳.۵۱
۸	۳.۲۶	۳.۵۴	۳.۷۷
۹	۳.۲۷	۳.۴۸	۴.۰۱
۱۰	۴.۱۷	۴.۱۶	۴.۳۶
۱۱	۱۶.۱۰	--	۱۰.۲۲



شكل ۳-۱۲) نمودار خطاي آفلاين به ازاي تعداد فله‌هاي مختلف و تعداد تقسيم‌بندي‌هاي مختلف براي محيط پنج بعدي (اندازه همسايگي ۲)

همان طور كه در جداول و نمودارها قابل مشاهده است، تعداد تقسيم‌بندي بر روي كارايي الگوريتم تا حدود تاثيرگذار است و انتخاب مناسب آن براي هر مساله مي‌تواند بر روي نتايج موثر باشد. براي حالتي كه تقسيم‌بندي وجود ندارد، ميزان كاراي بسيار پايين است و اين امر با توجه به نياز زياد به حفظ تنوع طبيعي مي‌باشد. اين نتايج نشان از اهميت و تاثير بسزايي عمل تقسيم‌بندي بر روي كارايي مي‌باشد.

در تمامي نمودارها، براي تمامي ابعاد، در ابتدا با زياد شدن تعداد تقسيم‌بندي‌ها به طور كلي كارايي افزايش مي‌يابد ولي پس از آن با زياد شدن بيش از حد اين تعداد، افزايش كارايي متوقف و حتي تا حدودي كاهش نيز مي‌يابد. اين امر بدين علت است كه افزايش اوليه تعداد تقسيم‌بندي‌ها در ابتدا با كمک به حفظ تنوع موجب افزايش كارايي مي‌شود ولي افزايش بيش از حد اين تقسيم‌بندي موجب كوچكي بيش از حد محدوده جستجوي محلي و در نتيجه كند شدن فرآيند همگرايي الگوريتم مي‌شود. اين پديده كاهش كارايي با افزايش تعداد ابعاد بيشتر قابل مشاهده است زيرا با افزايش تعداد تقسيم‌بندي‌ها، تعداد سلول‌ها به صورت نمايي افزايش مي‌يابد كه اين امر در ابعاد بالاتر بيشتر خود را نشان مي‌دهد. البته اين پديده نسبت به آزمايشاتي كه در آن‌ها اندازه همسايگي ۲ در نظر گرفته شده بود كمتر مشاهده مي‌شود. علت اين امر اينست كه با افزايش اندازه شعاع همسايگي، تاثير محلي بودن جستجو محلي كمتر مي‌شود. با توجه به نتايج بدست آمده در اين آزمايش، اگر بخواهيم براي اين مساله در تمامي اين محيط‌هاي مورد آزمايش براي همسايگي دو، يک مقدار يكسان انتخاب كنيم، مقدار ۸ به نظر مناسب مي‌رسد.

۳-۳-۳ آزمايش تاثير پارامترهاي F و Cr بر روي كارايي در مدل‌هاي تكاملي مختلف

در الگوريتم تكامل تفاضلي سلولي از الگوريتم تكامل تفاضلي جهت تكامل و جستجو استفاده شده است كه در آن از يک مدل تكاملي جهت ساخت بردارهاي كانديد استفاده مي‌شود. تاكنون در مدل‌هاي زيادي پيشنهاد شده است كه در جدول ۳-۱۳ چند مدل متداول از آن‌ها به همراه مدل تكاملي پيشنهادي ما ذكر شده است. در اين آزمايش تمامي اين چهار مدل تكاملي به ازاي مقادير مختلف F و Cr مورد آزمايش قرار گرفته‌اند تا بتوان بهترين مدل تكاملي به همراه مقادير مناسب براي پارامترهاي الگوريتم را پيدا نمود. البته در هنگام استفاده از مدل تكاملي پيشنهادي (شماره ۳) در الگوريتم اصلي، از پارامتر F استفاده نشده است و به جاي آن يک عدد تصادفي قرار مي‌گيرد. هدف از استفاده از پارامتر F در اين آزمايش نشان دادن عدم تاثير و وابستگي نتايج به آن مي‌باشد. به غير از مقادير پارامترهاي F و Cr، در تمامي آزمايشات از مقادير پيش‌فرض ذكر شده در جدول ۳-۲ و جدول ۳-۱ استفاده شده است. نتايج آزمايشات بر روي چهار مدل تكاملي به ترتيب در جدول ۳-۱۴ تا جدول ۳-۱۷ و شكل ۳-۱۳ تا شكل ۳-۱٦ درج شده است.

جدول ۳-۱۳) مدل‌هاي تكاملي مورد استفاده در الگوريتم تكامل تفاضلي

شماره	نام استاندارد	رابطه ساخت بردار كانديد
۱	DE/rand/1	$\overrightarrow{v_i} = \overrightarrow{x_{r1}} + F.(\overrightarrow{x_{r2}} - \overrightarrow{x_{r3}})$
۲	DE/rand-to-best/1	$\overrightarrow{v_i} = \overrightarrow{x_{r1}} + F.(localBest_j - \overrightarrow{x_{r2}})$
۳	DE/best-to-current/1	$\overrightarrow{v_i} = localBest_j + F.(\overrightarrow{x_i} - \overrightarrow{x_{r1}}) + \lambda_{rand}.(\overrightarrow{x_{r2}} - \overrightarrow{x_{r3}})$
۴	DE/current-to-best/1	$\overrightarrow{v_i} = \overrightarrow{x_i} + F.(localBest_j - \overrightarrow{x_i}) + \lambda_{rand}.(\overrightarrow{x_{r1}} - \overrightarrow{x_{r2}})$

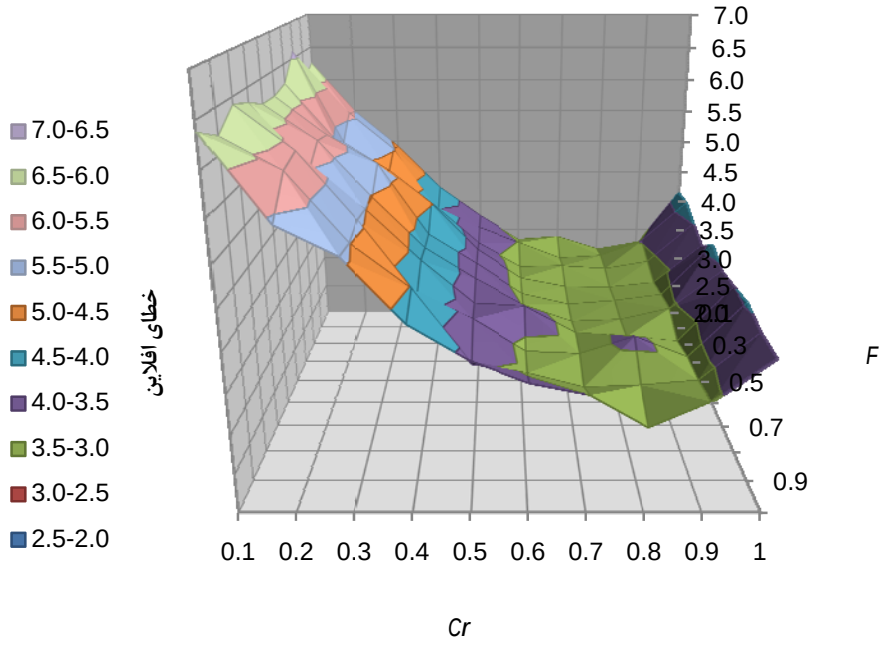
همان طور كه در جداول و نمودارها قابل مشاهده است، در تمامي مدل‌هاي تكاملي به غير از شماره ۳، در ابتدا با افزايش مقدار Cr نتايج بهبود مي‌يابد كه اين امر طبيعي است زيرا بدون استفاده از بازتركيبي الگوريتم دچار همگرايي و سكون زيادي مي‌شود. از طرف ديگر با افزايش بيش از حد آن اين كارايي رو به كاهش مي‌گذارد زيرا بالا بودن بيش از حد بازتركيبي نيز الگوريتم را باز دچار كندي و سكون مي‌كند. موضوع ديگري كه مشاهده مي‌شود نيز تاثيرگذاري بيشتر مقدار پارامتر Cr نسبت به F است كه اين امر به علت آنست كه در محيط‌هاي پويا تعداد محاسبات شايستگي مورد استفاده در آزمايشات بسيار زياد است و جمعيت پس از مدتي به مدت طولاني همگرا مي‌باشد و در اين زمان‌ها با توجه به شباهت موجودات به يكديگر، تاثير پارامتر F كم مي‌شود. البته همين تفاوت كوچک در محيط‌هاي پويا تاثيرگذار محسوب مي‌شود و به معناي ناديده گرفتن آن نيست.

اگر بخواهيم به طور كلي نتايج ۴ مدل را با يكديگر مقايسه كنيم، مي‌توان گفت مدل‌هاي۲ تا ۴ نسبت به مدل۱كارايي بالاتري از خود نشان مي‌دهند. علت آن اينست كه در مدل پيشنهادي ما نياز زيادي به همگرايي سريع وجود دارد كه وجود بردار بهترين موجود به اين امر كمک مي‌كند. از همين رو، مدل تكاملي ۳ نيز بالاترين كارايي كلي را نشان مي‌دهد زيرا بر پايه بهترين موجود است و در نتيجه داراي سرعت همگرايي بالاتري است.

موضوع ديگري كه در نمودار شكل ۳-۱۵ ديده مي‌شود عدم وابستگي زياد الگوريتم به مقدار پارامتر F و عدم نظم و رابطه منطقي بين روند تغييرات اين پارامتر و كارايي است. در نتيجه شايد بتوان اين پارامتر با حذف نمود و از يک عدد تصادفي استفاده نمود كه اين امر در آزمايشات تاييد شده است. به همين جهت در الگوريتم پيشنهادي در هنگام استفاده از مدل ۳ از پارامتر F استفاده نخواهد شد.

جدول ۳-۱٤) خطاي آفلاين به ازاي مقادير مختلف پارامترهاي F و Cr براي مدل تكاملي ۱

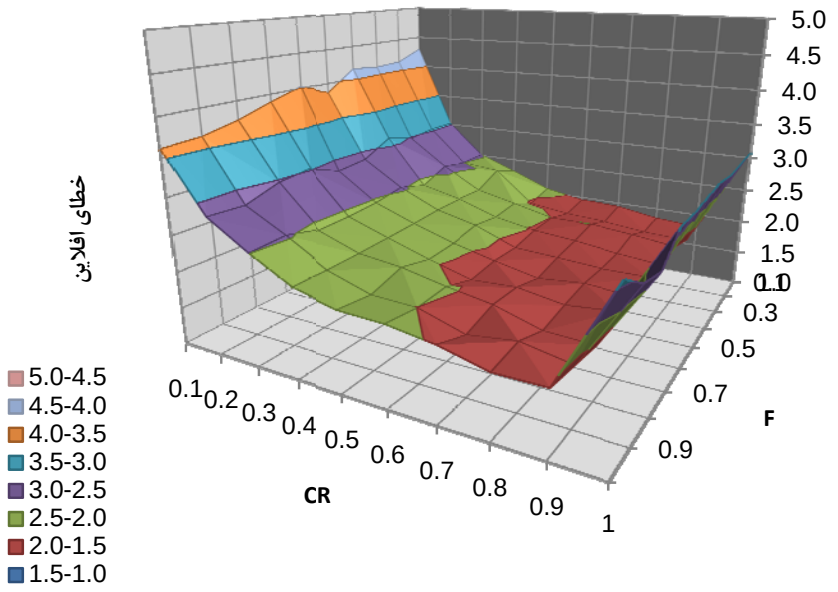
CR	F									
	۰.۱	۰.۲	۰.۳	۰.۴	۰.۵	۰.۶	۰.۷	۰.۸	۰.۹	۱.۰
۰.۱	۶.۲۵	۶.۰۹	۶.۶۰	۶.۲۴	۶.۱۷	۶.۱۹	۶.۳۲	۶.۴۲	۶.۲۰	۶.۳۸
۰.۲	۵.۴۸	۵.۵۹	۵.۲۷	۵.۴۵	۵.۶۶	۵.۷۲	۵.۴۹	۵.۸۹	۵.۸۲	۵.۴۳
۰.۳	۴.۹۲	۵.۰۲	۴.۸۴	۴.۹۷	۵.۲۶	۴.۹۸	۵.۰۱	۵.۰۷	۵.۰۱	۵.۰۸
۰.۴	۴.۲۳	۳.۹۳	۴.۳۰	۴.۵۲	۴.۲۷	۴.۲۹	۴.۳۵	۴.۱۴	۴.۴۸	۴.۳۳
۰.۵	۳.۷۴	۳.۷۴	۳.۷۰	۳.۸۲	۳.۸۳	۳.۸۵	۳.۷۷	۳.۸۳	۳.۵۵	۳.۹۱
۰.۶	۳.۳۲	۳.۴۸	۳.۳۱	۳.۳۷	۳.۳۸	۳.۴۸	۳.۷۳	۳.۴۰	۳.۴۳	۳.۶۳
۰.۷	۳.۳۷	۳.۲۲	۳.۳۳	۳.۲۳	۳.۲۲	۳.۲۳	۳.۳۶	۳.۳۷	۳.۲۵	۳.۴۹
۰.۸	۳.۱۶	۳.۲۷	۳.۲۸	۳.۲۳	۳.۲۷	۳.۲۶	۳.۳۰	۳.۶۰	۳.۲۸	۳.۰۹
۰.۹	۳.۲۱	۳.۵۲	۳.۴۰	۳.۳۰	۳.۳۷	۳.۴۲	۳.۲۳	۳.۳۲	۳.۳۴	۳.۴۱
۱.۰	۴.۱۴	۴.۱۵	۴.۰۱	۳.۸۷	۴.۱۱	۳.۹۶	۳.۹۹	۴.۰۴	۳.۹۴	۳.۹۳



شكل ۳-۱۳) نمودار خطاي آفلاين به ازاي مقادير مختلف پارامترهاي F و Cr براي مدل تكاملي ۱

جدول ۳-۱۵) خطاي آفلاين به ازاي مقادير مختلف پارامترهاي F و Cr براي مدل تكاملي ۲

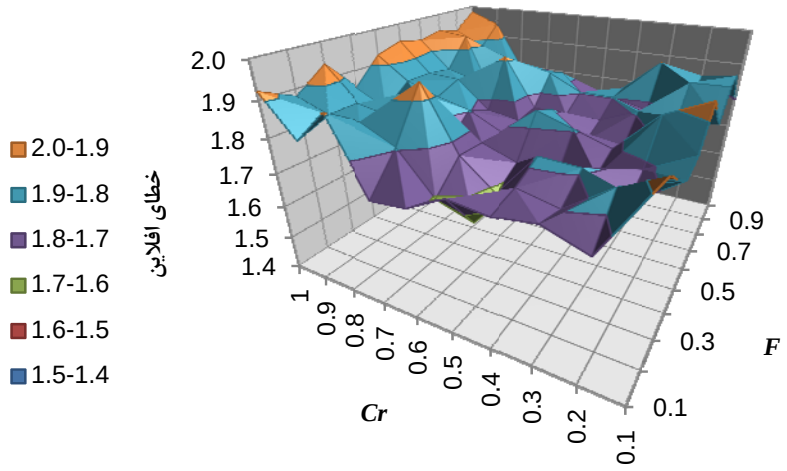
CR	F									
	۰.۱	۰.۲	۰.۳	۰.۴	۰.۵	۰.۶	۰.۷	۰.۸	۰.۹	۱.۰
۰.۱	۴.۲۹	۴.۱۳	۴.۱۱	۴.۱۵	۳.۸۶	۴.۰۲	۳.۸۹	۳.۷۴	۳.۶۴	۳.۶۱
۰.۲	۳.۰۲	۳.۰۳	۲.۹۸	۳.۱۰	۲.۸۹	۲.۸۵	۳.۰۲	۲.۹۷	۲.۸۲	۲.۸۵
۰.۳	۲.۵۱	۲.۵۰	۲.۵۸	۲.۴۱	۲.۵۳	۲.۴۶	۲.۴۳	۲.۴۲	۲.۵۳	۲.۵۰
۰.۴	۲.۲۹	۲.۴۵	۲.۲۶	۲.۲۸	۲.۲۸	۲.۲۵	۲.۲۳	۲.۲۱	۲.۲۱	۲.۲۱
۰.۵	۲.۰۶	۱.۹۶	۲.۱۳	۲.۰۹	۲.۱۴	۲.۰۷	۲.۰۴	۲.۱۵	۲.۰۲	۲.۰۸
۰.۶	۱.۹۶	۱.۸۹	۲.۰۰	۱.۹۸	۱.۹۱	۱.۹۳	۱.۸۸	۲.۰۳	۲.۰۰	۲.۰۶
۰.۷	۱.۹۴	۱.۸۶	۲.۰۰	۱.۹۳	۱.۸۸	۱.۸۰	۱.۹۱	۱.۹۳	۱.۸۱	۱.۹۸
۰.۸	۱.۸۴	۱.۸۰	۱.۹۵	۱.۸۶	۱.۸۵	۱.۸۷	۱.۸۲	۱.۷۴	۱.۹۰	۱.۸۳
۰.۹	۱.۸۱	۱.۷۸	۱.۸۸	۱.۸۱	۱.۸۶	۱.۸۳	۱.۸۸	۱.۷۲	۱.۷۸	۱.۸۴
۱.۰	۳.۰۷	۳.۰۴	۳.۰۲	۳.۰۸	۳.۰۶	۳.۰۳	۳.۰۴	۲.۷۴	۲.۸۴	۳.۱۲



شكل ۳-۱٤) نمودار خطاي آفلاين به ازاي مقادير مختلف پارامترهاي F و Cr براي مدل تكاملي ۲

جدول ۳-۱۶) خطاي آفلاين به ازاي مقادير مختلف پارامترهاي F و Cr براي مدل تكاملي ۳

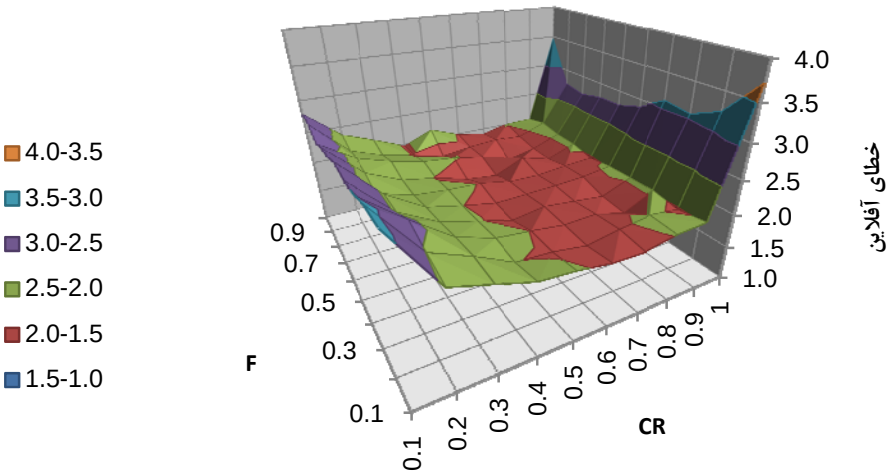
CR	F									
	۰.۱	۰.۲	۰.۳	۰.۴	۰.۵	۰.۶	۰.۷	۰.۸	۰.۹	۱.۰
۰.۱	۱.۹۴	۱.۸۸	۱.۸۳	۱.۸۴	۱.۹۴	۱.۸۷	۱.۸۱	۱.۸۸	۱.۸۹	۱.۷۹
۰.۲	۱.۷۴	۱.۸۴	۱.۷۵	۱.۷۹	۱.۸۸	۱.۸۵	۱.۷۹	۱.۸۸	۱.۸۱	۱.۸۵
۰.۳	۱.۷۸	۱.۷۶	۱.۸۱	۱.۷۴	۱.۷۷	۱.۷۴	۱.۸۱	۱.۷۸	۱.۹۰	۱.۷۴
۰.۴	۱.۷۷	۱.۷۸	۱.۸۱	۱.۶۹	۱.۸۱	۱.۷۳	۱.۷۶	۱.۸۰	۱.۸۱	۱.۷۶
۰.۵	۱.۷۸	۱.۶۶	۱.۶۹	۱.۷۳	۱.۸۴	۱.۸۱	۱.۷۶	۱.۸۰	۱.۷۳	۱.۷۵
۰.۶	۱.۷۲	۱.۶۹	۱.۷۷	۱.۷۶	۱.۷۶	۱.۷۶	۱.۷۷	۱.۸۵	۱.۸۳	۱.۷۶
۰.۷	۱.۷۱	۱.۸۱	۱.۹۴	۱.۷۴	۱.۸۰	۱.۷۷	۱.۹۱	۱.۷۸	۱.۸۲	۱.۸۰
۰.۸	۱.۹۱	۱.۸۵	۱.۸۳	۱.۸۷	۱.۸۳	۱.۸۵	۱.۸۲	۱.۷۴	۱.۸۵	۱.۸۰
۰.۹	۱.۸۲	۱.۸۵	۱.۸۵	۱.۸۰	۱.۸۶	۱.۸۶	۱.۸۵	۱.۹۴	۱.۸۸	۱.۹۵
۱.۰	۱.۹۲	۱.۸۸	۱.۹۴	۱.۸۵	۱.۹۳	۱.۹۵	۱.۹۳	۱.۹۵	۱.۹۳	۱.۹۸



شكل ۳-۱۵) نمودار خطاي آفلاين به ازاي مقادير مختلف پارامترهاي F و Cr براي مدل تكاملي ۳

جدول ۳-۱۷) خطاي آفلاين به ازاي مقادير مختلف پارامترهاي F و Cr براي مدل تكاملي ٤

CR	F									
	۰.۱	۰.۲	۰.۳	۰.۴	۰.۵	۰.۶	۰.۷	۰.۸	۰.۹	۱.۰
۰.۱	۳.۱۸	۳.۱۶	۳.۰۲	۲.۹۶	۲.۹۸	۲.۸۵	۲.۷۹	۲.۶۶	۲.۷۸	۲.۷۹
۰.۲	۲.۳۹	۲.۴۸	۲.۶۰	۲.۴۴	۲.۴۴	۲.۲۲	۲.۳۱	۲.۳۴	۲.۲۸	۲.۳۷
۰.۳	۲.۲۳	۲.۲۶	۲.۲۲	۲.۱۶	۲.۱۲	۲.۰۷	۲.۰۸	۲.۱۸	۲.۰۹	۲.۱۸
۰.۴	۲.۰۷	۲.۰۱	۲.۰۵	۲.۰۲	۲.۰۰	۲.۰۲	۱.۹۹	۲.۰۷	۱.۹۸	۱.۹۷
۰.۵	۲.۰۳	۱.۹۸	۲.۰۰	۱.۸۹	۱.۹۱	۲.۰۰	۱.۸۵	۲.۰۲	۱.۸۶	۲.۱۶
۰.۶	۱.۹۵	۱.۹۰	۱.۹۳	۱.۹۶	۱.۹۲	۱.۹۱	۱.۸۰	۱.۹۳	۱.۹۲	۱.۹۸
۰.۷	۱.۹۳	۱.۹۸	۱.۹۵	۱.۹۱	۱.۹۳	۱.۹۰	۲.۰۰	۱.۷۸	۱.۹۶	۱.۹۰
۰.۸	۲.۰۱	۲.۰۳	۱.۸۲	۱.۸۵	۱.۹۶	۱.۹۴	۱.۸۷	۱.۸۸	۱.۹۴	۲.۰۱
۰.۹	۲.۰۳	۱.۹۷	۲.۰۴	۲.۰۳	۲.۰۲	۱.۹۷	۲.۰۴	۲.۰۵	۲.۱۰	۲.۰۲
۱.۰	۳.۷۲	۳.۲۴	۳.۰۶	۳.۱۱	۲.۸۹	۲.۷۸	۲.۷۷	۲.۶۹	۲.۷۶	۳.۴۷



شكل ۳-۱٦) نمودار خطاي آفلاين به ازاي مقادير مختلف پارامترهاي F و Cr براي مدل تكاملي ٤

۳-۳-۴ آزمايش تاثير تعداد ظرفيت سلول بر روي كارايي به ازاي تعداد قلههاي مختلف

در اين آزمايش تاثير ميزان پارامتر ظرفيت سلول بر روي كارايي الگوريتم تكامل تفاضلي سلولي براي تعداد قلههاي مختلف مورد بررسي قرار داده ميشود. همچنين سعي ميشود تا اگر ارتباطي بين تعداد قلهها و ظرفيت سلول وجود دارد، آن را بررسي كنيم. به جز پارامترهاي مورد

آزمايش قرار گرفته، در تمامي محيط‌ها از مقادير پيش‌فرض ذكر شده در جدول ۳-۲ و جدول ۳-۱ براي پارامترهاي الگوريتم و محيط استفاده شده است. نتايج آزمايش بر حسب خطاي آفلاين در جدول ۳-۱۸ و نمودار سه بعدي آن در شكل ۳-۱۷ ترسيم شده است.

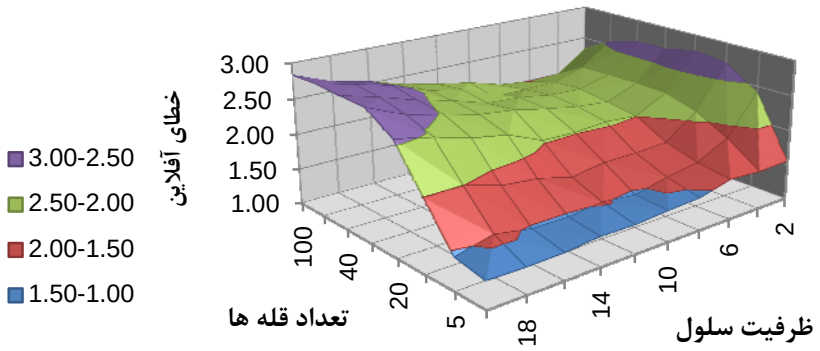
همان طور كه مشاهده مي‌شود مقدار ظرفيت سلول، يک پارامتر نسبتا تاثيرگذار در كارايي الگوريتم مي‌باشد و تنظيم مناسب آن به بهبود جستجو كمک مي‌كند. براي محيط تک قله‌اي ديده مي‌شود كه هر چه ظرفيت سلول بالاتر در نظر گرفته شود نتايج بهتر مي‌شود كه اين امر طبيعي است. زيرا در حالت تک قله نيازي به ايجاد محدوديت نمي‌باشد و هر چه موجودات بيشتري بر روي همان تک قله جستجو كنند مسلما نتايج بهتر است. در نتيجه در اين گونه محيط‌ها ايجاد محدوديت بر روي سلول‌ها موجب كاهش كارايي نيز حتي مي‌شود.

اما در محيط‌هاي اين گونه نمي‌باشد و انتخاب مناسب مقدار ظرفيت مي‌تواند به بهبود كارايي كمک كند. در محيط‌هاي ۵،۱۰ و ۲۰ قله‌اي رابطه‌اي ميان تعداد قله‌ها و بهترين مقدار براي ظرفيت سلول ديده مي‌شود. بدين شكل كه مقدار منايب برابر است با حاصل تقسيم اندازه جمعيت بر تعداد قله‌ها كه اين امر قابل پيش‌بيني بود. زيرا معمولا جمعيت هر چه تعداد قله‌هاي بيشتري را به طور همزمان پوشش دهد كارايي بالاتري بدست خواهد آمد و براي پوشش دادن تمامي قله‌ها، مي‌بايست اين جمعيت تقريبا به طور مساوي بين سلول‌هاي داراي قله تقسيم شده باشند. به طور مثال با توجه به اندازه جمعيت كه ۱۰۰ در نظر گرفته شده است، براي محييط ده قله‌اي، ظرفيت بهتر است برابر ۱۰ در نظر گرفته شود.

البته اين رابطه براي تعداد قله‌هاي بالاتر تا حدي تغيير مي‌كند زيرا ديگر جمعيت با توجه به تعداد زياد قله‌ها، قادر به حفظ هزمان همگي آن‌ها نيست. در ضمن براي جستجو و نگهداري هر قله به يک حداقل تعداد موجود نياز مي‌باشد. همان‌طور كه ديده مي‌شود حداقل براي هر قله تقريبا چهار موجود نياز مي‌باشد تا الگوريتم قادر به ادامه جستجو و حفظ آن قله باشد. به طور كلي مي‌توان بين تعداد قله‌ها در محيط و ظرفيت مناسب سلول رابطه وجود دارد.

جدول ۳-۱۸) خطاي آفلاين به ازاي مقادير مختلف ظرفيت سلول براي تعداد قله‌هاي مختلف

	تعداد قله‌ها							
T	۱	۵	۱۰	۲۰	۳۰	۴۰	۵۰	۱۰۰
۲	1.55	2.41	2.71	2.76	2.68	2.67	2.55	2.31
۴	1.54	1.86	2.10	2.24	2.25	2.25	2.24	1.99
۶	1.57	1.64	1.77	2.00	2.07	2.13	2.10	1.99
۸	1.46	1.57	1.74	2.04	2.12	2.15	2.10	2.04
۱۰	1.43	1.46	1.74	2.01	2.20	2.23	2.10	2.18
۱۲	1.40	1.51	1.68	2.03	2.30	2.26	2.32	2.27
۱۴	1.40	1.47	1.80	2.23	2.36	2.41	2.42	2.37
۱۶	1.36	1.47	1.85	2.32	2.48	2.54	2.60	2.52
۱۸	1.35	1.56	2.01	2.46	2.60	2.69	2.73	2.63
۲۰	1.36	1.44	1.99	2.56	2.81	2.82	2.85	2.83



شكل ۳-۱۷) نمودار خطاي آفلاين به ازاي مقادير مختلف ظرفيت سلول براي تعداد قله‌هاي مختلف

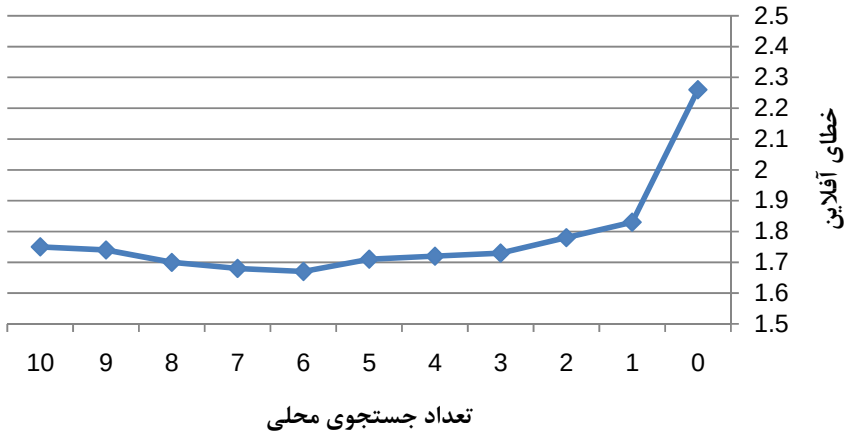
۳-۳-۵ آزمايش تاثير تعداد دفعات جستجوي محلي پس از هر تغيير بر روي كارايي

پس از هر تغيير در محيط اين امكان وجود دارد كه بهينه‌ها جابه‌جا شوند، لذا مي‌بايست بهينه‌هاي از دست رفته را دوباره يافت. براي اين كار، از جستجوي محلي استفاده شده است كه اين

عمل پس از هر تغيير، تا چند بار بروزرساني، به جاي جستجوي تكامل تفاضلي تكرار خواهد شد. در اين آزمايش ميزان تاثير اين جستجوي محلي بر رويي كارايي الگوريتم و همچنين مقدار مناسب تعداد دفعات تكرار آن را بررسي خواهيم كرد. در جدول ۳-۱۹ مقادير خطاي آفلاين به ازاي تعداد دفعات تكرار جستجوي محلي پس از هر تغيير درج شده است. در اين آزمايش از مقادير پيش‌فرض ذكر شده در جدول ۳-۲ و جدول ۳-۱ استفاده شده است.

جدول ۳-۱۹) خطاي آفلاين به ازاي تعداد دفعات جستجوي محلي متفاوت

	N_{LS}										
	۰	۱	۲	۳	۴	۵	۶	۷	۸	۹	۱۰
خطا	۲.۲۶	۱.۸۳	۱.۷۸	۱.۷۳	۱.۷۲	۱.۷۱	۱.۶۷	۱.۶۸	۱.۷۰	۱.۷۴	۱.۷۵



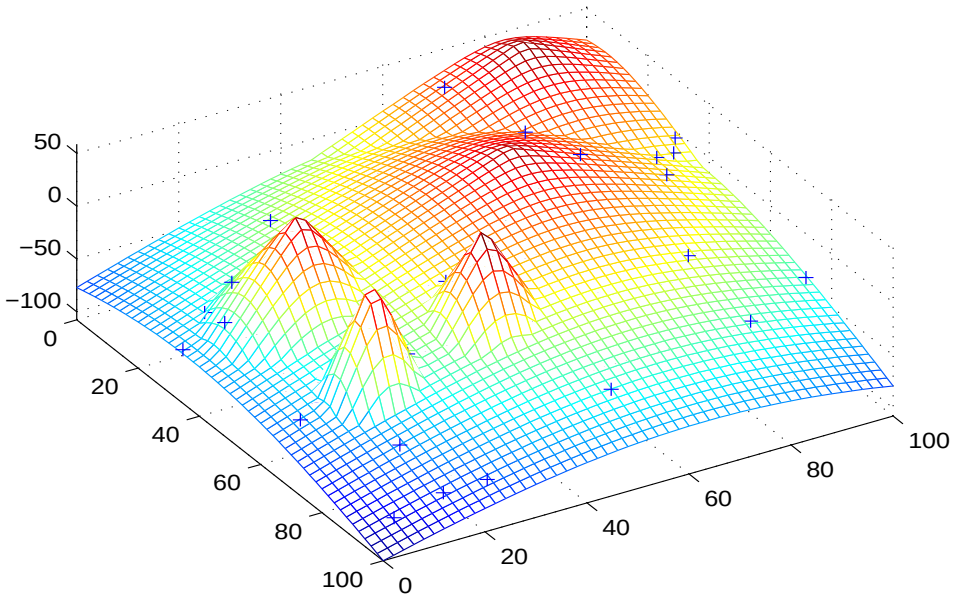
شكل ۳-۱۸) نمودار خطاي آفلاين به ازاي تعداد دفعات جستجوي محلي پس از هر تغيير

همان طور كه در نتايج مشاهده مي‌شود، حتي يک مرتبه جستجوي محلي بسيار تاثيرگذار مي‌باشد. اما با افزايش اين تعداد از ميزان تاثيرپذيري كاسته مي‌شود زيرا استفاده بيش از حد از اين جستجوي محلي غير ضروري است. اين بدين علت است كه جستجوي محلي وظيفه يافتن بهينه گم‌شده در اطراف مكان‌هاي فعلي را برعهده دارد و بيش از آن كاري از آن ساخته نيست. حتي تا حدودي شاهد كاهش كارايي نيز هستيم زيرا جستجوي محلي تعدادي از محاسبات شايستگي را به هدر مي‌دهد و به اندازه تكامل تفاضلي در جستجوي عمومي كارايي ندارد.

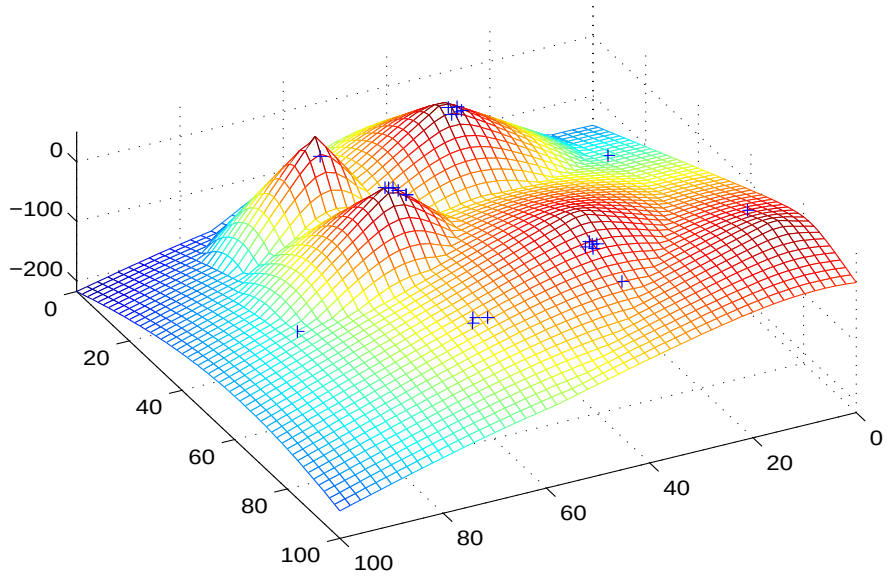
۳-۳-۶ ارزيابي كارايي الگوريتم تكامل تفاضلي سلولي و جستجوي بهبوديافته

در اين آزمايش سعي مي‌كنيم تا كارايي الگوريتم تكامل تفاضلي سلولي پيشنهادي (CellularDE) و همچنين نسخه همراه با جستجوي بهبود يافته (CellularDE+) را در محيط‌هاي پويا مورد ارزيابي و مقايسه قرار دهيم.

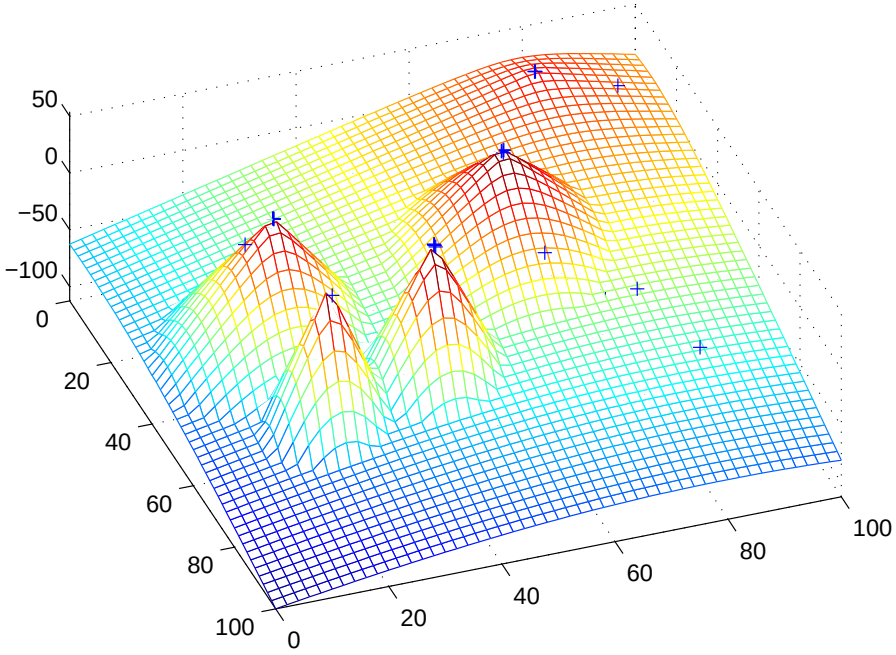
در ابتداي آزمايش، نمايشي از يک نمونه محيط پوياي دو بعدي به همراه تغييرات و جستجوهاي انجام شده در طي زمان را نشان خواهيم داد. در شكل ۳-۱۹ تا شكل ۳-۲۲ نماهايي از يک نمونه پهنه جستجوي دو بعدي در زمان‌هاي بروزرساني مختلف به همراه مكان موجودات ترسيم شده است. تعداد ۲۵ موجود در فضا وجود دارد كه براي تكامل آنان از الگوريتم تكامل تفاضلي سلولي استفاده شده است. دوره تغيير برابر ۵۰۰، تعداد تقسيم‌بندي هر بعد برابر ۱۰ و اندازه همسايگي ۲ در نظر گرفته شده است. همان طور كه در شكل ۳-۱۹ مشاهده مي‌شود، جمعيت اوليه در نقاط مختلف پهنه جستجو پراكنده‌اند و پنج قله با مشخصات مختلف در فضا وجود دارد. در شكل ۳-۲۰ نمايي از محيط پس از چندين بار بروزرساني نشان داده شده است كه محيط تغييراتي داشته است و موجودات توانسته‌اند تمامي پنج قله را بيابند و بر روي تمامي آنان موجوداتي قرار دارند. در شكل ۳-۲۱ كه مدتي پس از آن را نشان مي‌دهد، تغييراتي كه در محيط روي داده است ولي همچنان بر روي تمامي قله‌ها موجوداتي قرار دارند ولي چندين قله در نزديكي يكديگر قرار گرفته‌اند. اين نزديک سبب مي‌شود كه موجودات بر روي قله‌هاي با ارتفاع كمتر، به تدريج به سمت قله‌هاي اطراف كه بلندتر هستند بروند و در نتيجه همان طور كه در شكل ۳-۲۲ مشاهده مي‌شود پس از مدتي يكي از قله‌ها را از دست مي‌دهيم.



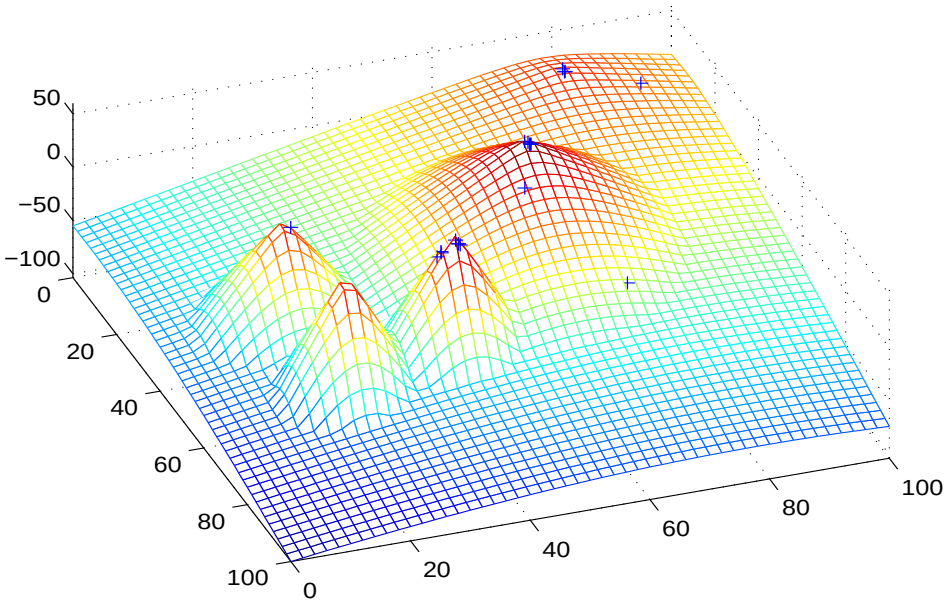
شكل ۳-۱۹) پهنه جستجوي دو بعدي حاوي ۵ قله به همراه مكان موجودات در اولين تكرار



شكل ۳-۲۰) پهنه جستجوي دو بعدي حاوي ۵ قله به همراه مكان موجودات در تكرار ٤٠ام



شكل ۳-۲۱) پهنه جستجوي دو بعدي حاوي ۵ قله به همراه مكان موجودات در تكرار ۱۵۷ام



شكل ۳-۲۲) پهنه جستجوي دو بعدي حاوي ۵ قله به همراه مكان موجودات در تكرار ۱۶۵ام

در ادامه كارايي الگوريتم تكامل تفاضلي سلولي پيشنهادي (CellularDE) و همچنين نسخه همراه با جستجوي بهبود يافته (CellularDE+) را با ساير الگوريتم‌ها در محيط‌هاي پويا مورد ارزيابي و مقايسه قرار دهيم. كارايي تمامي الگوريتم‌ها بر اساس خطاي آفلاين در محيط‌هاي پوياي با دوره تغيير ۵۰۰، ۱۰۰۰، ۲۵۰۰ و ۵۰۰۰ و به ازاي تعداد قله‌هاي مختلف ۱، ۵، ۱۰، ۲۰، ۳۰، ۴۰، ۵۰، ۱۰۰ و ۲۰۰ مورد مقايسه قرار گرفته است.

سالانه كارهاي زيادي در زمينه بهينه‌سازي تكاملي در محيط‌هاي پويا پيشنهاد مي‌شود كه اين روند در سال‌هاي اخير رشد زيادي نيز يافته است. اكثر كارهاي پيشنهاد شده براي محيط‌هاي پيوسته پويا، از تابع محک قله‌هاي متحرک (MPB) جهت بررسي كارايي روش‌هاي پيشنهادي خود استفاده مي‌كنند.اين تابع محک كه دربرگيرنده عمده محيط‌هاي توابع محک پيش از خود مي‌باشد، از سخت‌ترين محيط‌هاي محک پوياي طراحي شده مي‌باشد. ما نيز الگوريتم‌هاي پيشنهادي را بر روي مقادير پيش‌فرض سناريوي ۲ همين تابع محک كه در جدول ۳-۱ درج شده است، مورد ارزيابي قرار مي‌دهيم.

جهت مقايسه كارايي روش‌هاي پيشنهادي، تعدادي از اين الگوريتم‌ها كه بالاترين كارايي را بر روي تابع محک قله‌هاي متحرک از خود نشان داده‌اند انتخاب شده‌اند. از بين الگوريتم‌هاي مبتني بر تكامل تفاضلي پيشنهاد شده براي محيط‌هاي پويا [۲۴, ۲۷, ۶۷, ۷۰]، الگوريتم DynDE [۲۴] انتخاب شده است كه تاكنون بهترين نتايج را در بين اين دسته از الگوريتم‌ها، بويژه در تابع محک قله‌هاي متحرک، از خود نشان داده است [۳۴, ۵۲]. الگوريتم Cellular PSO [۴۱, ۴۲] نيز از جهت شباهت بسيار ساختاري و سلولي بودن انتخاب شده است تا مقايسه‌اي با ساير الگوريتم‌هاي سلولي نيز انجام شود. الگوريتم‌هاي mQSO [۲۵, ۶۵] ،Adaptive mQSO [۲۶] و HmSO [۳۱] نيز از الگوريتم‌هاي شاخص در بهينه‌سازي در محيط‌هاي پويا هستند كه نتايج بسيار خوبي نسبت به ساير الگوريتم‌ها تا كنون از خود نشان داده‌اند [۳۴, ۵۲]. البته بايد توجه داشت كه با توجه به اينكه روش‌هاي پيشنهادي براي بهينه‌سازي در محيط‌هاي پيوسته طراحي شده‌اند، اين الگوريتم‌ها از بين آن‌هايي انتخاب شده‌اند كه براي محيط‌هاي پيوسته هستند. براي الگوريتم‌هاي پيشنهادي، از مقادير پيش‌فرض جدول ۳-۲ و براي ساير روش‌ها از مقادير پيش‌فرض ذكر شده در خود مقالات پيشنهاد دهنده استفاده شده است.

نتايج بدست‌آمده از الگوريتم‌هاي مختلف براي دوره‌هاي تغيير مختلف در جدول ۳-۲۰ تا جدول ۳-۲۴ شده است. نتايج گزارش شده براي الگوريتم‌هاي HmSO، mQSO، Adaptive mQSO و Cellular PSO از مقاله [۳۱] استخراج شده است ولي الگوريتم DynDE پياده‌سازي شده است و نتايج بدست آمده از اجراي اين پياده‌سازي گزارش شده است. نتايج به صورت ميانگين خطاي آفلاين براي ۱۰۰ بار اجرا به همراه خطاي استاندارد[1] در جداول درج شده است.

علت درج خطاي استاندارد اين است كه در اين گونه روش‌ها كه محيط‌هاي مورد آزمايش متفاوت و پويا هستند، ميانگين نتيجه آزمايش به تنهايي نشاندهنده برتري روشي نيست، بلكه ميزان واريانس و گستردگي جواب‌ها نيز موثر است. در اين مواقع از تست‌هاي آماري براي نشان دادن وجود تفاوت آماري قابل توجه، بين نتايج دو آزمايش مي‌توان استفاده نمود. در اين آزمايش از روش تست آماري دانش‌آموز[2] با فاصله اطمينان[3] ۹۵% براي مقايسه نتايج الگوريتم‌ها استفاده شده است. در هر محيط آن روشي كه از نظر آماري بهتر از بقيه باشد را پررنگ كرده‌ايم و اگر چند روش نسبت به ديگران برتري داشته باشند، همه آن‌ها پررنگ شده‌اند.

جدول ۳-۲۰) خطاي آفلاين به همراه ميانگين خطاي استاندارد الگوريتم‌هاي مختلف به ازاي تعداد قله‌اي مختلف براي دوره تغيير ۵۰۰

	نام الگوريتم						
P	Adaptive mQSO	mQSO	HmSO	Cellular PSO	DynDE	CellularDE+	CellularDE
۱	۵.۰۸±۰.۲۷	36.52±3.2	8.53±0.49	22.37±3.8	27.68±35.57	12.18±0.27	8.20±0.19
۵	۵.۱۴±۰.۰۹	13.50±0.8	7.40±0.31	14.20±0.6	9.17±1.47	6.52±0.06	6.06±0.05
۱۰	6.20±0.11	11.18±0.4	7.56±0.27	13.55±0.5	6.59±0.12	6.26±0.04	۵.۹۳±۰.۰۴
۲۰	6.94±0.18	10.54±0.2	7.81±0.20	12.77±0.3	7.45±0.03	7.43±0.04	۵.۶۰±۰.۰۳
۳۰	7.23±0.16	10.37±0.2	8.33±0.18	12.55±0.4	7.97±0.03	7.80±0.04	۵.۵۶±۰.۰۳
۴۰	7.43±0.17	10.32±0.2	8.45±0.18	12.33±0.3	8.15±0.03	7.94±0.04	۵.۴۸±۰.۰۲
۵۰	7.49±0.09	10.33±0.2	8.83±0.17	12.19±0.3	8.35±0.03	8.03±0.04	۵.۴۷±۰.۰۲
۱۰۰	7.29±0.15	9.93±0.21	8.85±0.16	11.38±0.2	8.65±0.03	7.98±0.04	۵.۲۹±۰.۰۲
۲۰۰	6.82±0.14	9.67±0.20	8.85±0.16	11.34±0.2	8.75±0.04	7.59±0.03	۵.۰۷±۰.۰۲

[1] Standard Error
[2] Student's T-test
[3] Confidence Interval

جدول ۳-۲۱) خطاي آفلاين به همراه ميانگين خطاي استاندارد الگوريتم‌هاي مختلف به ازاي تعداد قله‌اي مختلف براي دوره تغيير ۱۰۰۰

نام الگوريتم							
CellularDE	CellularDE+	DynDE	Cellular PSO	HmSO	mQSO	Adaptive mQSO	P
4.70±0.10	6.64±0.19	16.84±9.39	7.90±0.47	4.46±0.26	19.1±1.5	۲.۶۸±۰.۱۴	۱
4.05±0.04	5.25±0.06	5.32±0.44	5.81±0.21	4.27±0.08	7.59±0.39	۳.۲۲±۰.۰۷	۵
۳.۹۵±۰.۰۳	4.29±0.04	4.25±0.05	5.70±0.15	4.61±0.07	6.33±0.23	4.11±0.08	۱۰
۴.۰۲±۰.۰۳	3.97±0.03	5.34±0.02	5.92±0.18	4.66±0.12	6.46±0.20	4.75±0.14	۲۰
۳.۹۶±۰.۰۳	4.28±0.03	5.80±0.04	6.07±0.16	4.83±0.09	6.51±0.16	4.98±0.10	۳۰
۳.۹۶±۰.۰۲	4.43±0.03	6.09±0.02	5.95±0.15	4.82±0.09	6.43±0.18	5.10±0.11	۴۰
۳.۹۷±۰.۰۲	4.47±0.03	6.22±0.02	6.01±0.15	4.96±0.03	6.67±0.16	5.12±0.05	۵۰
۳.۸۱±۰.۰۲	4.57±0.03	6.55±0.03	6.04±0.13	5.14±0.08	6.34±0.12	5.03±0.09	۱۰۰
۳.۵۸±۰.۰۱	4.39±0.02	6.49±0.02	5.90±0.13	5.25±0.08	6.13±0.12	4.65±0.09	۲۰۰

جدول ۳-۲۲) خطاي آفلاين به همراه ميانگين خطاي استاندارد الگوريتم‌هاي مختلف به ازاي تعداد قله‌اي مختلف براي دوره تغيير ۲۵۰۰

نام الگوريتم							
CellularDE	CellularDE+	DynDE	Cellular PSO	HmSO	mQSO	Adaptive mQSO	P
2.22±0.08	1.78±0.06	7.08±2.08	4.91±0.28	1.75±0.10	7.79±0.72	۱.۰۹±۰.۰۶	۱
2.36±0.05	2.45±0.06	3.14±0.11	2.95±0.20	1.92±0.11	3.53±0.18	۱.۵۸±۰.۱۳	۵
2.48±0.04	۲.۲۰±۰.۰۴	2.81±0.05	2.97±0.15	2.39±0.16	3.20±0.14	2.33±0.11	۱۰
2.70±0.04	۱.۸۸±۰.۰۲	3.83±0.05	3.51±0.14	2.46±0.09	3.83±0.12	2.84±0.09	۲۰
2.81±0.03	۱.۸۶±۰.۰۲	4.32±0.05	3.87±0.12	2.57±0.05	4.03±0.12	3.13±0.09	۳۰
2.80±0.03	۱.۸۵±۰.۰۲	4.54±0.05	3.89±0.10	2.56±0.06	3.90±0.11	3.23±0.08	۴۰
2.75±0.03	۱.۹۰±۰.۰۲	4.71±0.05	4.16±0.15	2.65±0.05	3.95±0.10	3.24±0.07	۵۰
2.76±0.02	۲.۰۷±۰.۰۲	4.90±0.05	4.18±0.11	2.72±0.04	3.81±0.10	3.20±0.06	۱۰۰
2.57±0.02	۲.۲۰±۰.۰۲	4.91±0.04	4.04±0.09	2.81±0.04	3.66±0.07	3.00±0.05	۲۰۰

جدول ۳-۲۳) خطاي آفلاين به همراه ميانگين خطاي استاندارد الگوريتم‌هاي مختلف به ازاي تعداد قله‌اي مختلف براي دوره تغيير ۵۰۰۰

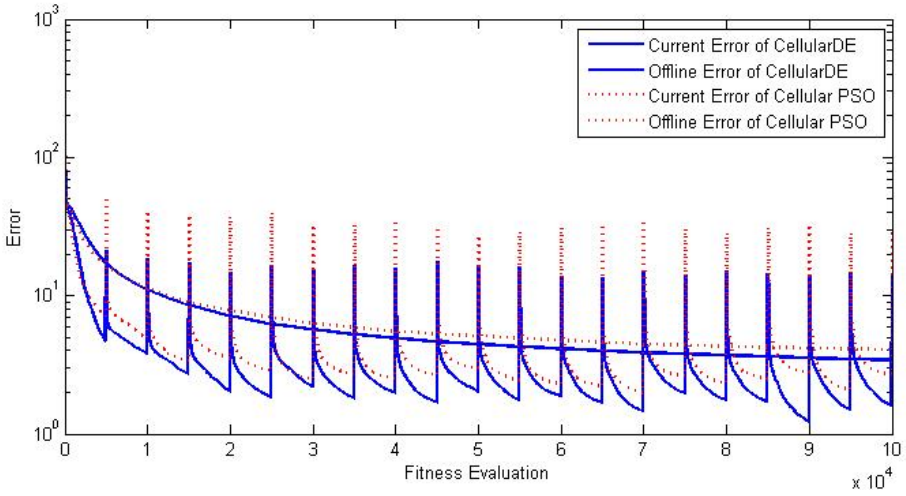
نام الگوريتم							
CellularDE	CellularDE+	DynDE	Cellular PSO	HmSO	mQSO	Adaptive mQSO	P
1.52±0.07	0.79±0.02	4.06±0.49	3.46±0.22	0.87±0.05	0.53±0.01	۰.۵۵±۰.۰۲	۱
1.53±0.06	1.57±0.06	2.22±0.15	1.79±0.12	1.18±0.04	1.05±0.06	۱.۰۰±۰.۰۴	۵
1.73±0.04	۱.۲۷±۰.۰۴	2.26±0.05	1.84±0.08	1.42±0.04	1.31±0.03	1.43±0.04	۱۰
2.04±0.04	۱.۲۴±۰.۰۳	3.14±0.07	2.63±0.11	1.50±0.06	1.69±0.05	1.95±0.05	۲۰
2.18±0.04	۱.۳۷±۰.۰۳	3.49±0.60	2.91±0.10	1.65±0.04	1.78±0.02	2.15±0.05	۳۰
2.22±0.04	۱.۳۸±۰.۰۲	3.82±0.09	3.16±0.11	1.65±0.05	1.86±0.02	2.28±0.04	۴۰
2.23±0.03	۱.۴۳±۰.۰۲	4.05±0.07	3.23±0.11	1.66±0.02	1.95±0.02	2.28±0.02	۵۰
2.24±0.02	۱.۴۲±۰.۰۲	4.18±0.09	3.43±0.10	1.68±0.03	1.95±0.01	2.31±0.03	۱۰۰
2.10±0.02	۱.۳۵±۰.۰۱	4.06±0.05	3.38±0.09	1.71±0.02	1.90±0.01	2.11±0.03	۲۰۰

همان‌طور كه مشاهده مي‌شود الگوريتم تكامل تفاضلي سلولي به طور كلي از كارايي مناسبي بويژه در دوره‌هاي تغيير بالا نسبت به ساير روش‌ها برخوردار است. در دوره‌هاي تغيير بالاتر كه امكان استفاده از تعداد محاسبات شايستگي بيشتري وجود دارد، الگوريتم همراه با جستجوي محلي بهبود يافته كارايي بالاتري از خود نشان داده است. زيرا اين روش از نظر استفاده زياد از محاسبات شايستگي ضعف دارد ولي در صورتي كه فرصت كافي داشته باشد قادر است نتايج بسيار خوبي توليد نمايد. در شكل ۳-۲۳، نمودار خطاي آفلاين و خطاي جاري براي دو الگوريتم تكامل تفاضلي سلولي و Cellular PSO براي محيطي با ۵۰ قله، دوره تغيير ۵۰۰۰ و ۱۰۰۰۰۰ مرتبه محاسبات شايستگي ترسيم شده است. نمودار ميانگين ۱۰۰ بار اجراي الگوريتم‌ها مي‌باشد. همان‌طور كه مشاهده مي‌شود الگوريتم پيشنهادي، نه تنها از نظر خطاي آفلاين بلكه از نظر دقت جواب نيز برتري دارد. اين امر نشان مي‌دهد كه الگوريتم پيشنهادي نه تنها از نظر جستجوي عمومي بلكه از نظر استخراج جواب نيز نسبت به Cellular PSO كارايي بالاتري دارد.

در تعداد قله‌هاي كم، الگوريتم‌هاي پيشنهادي نسبت به الگوريتم Adaptive mQSO تا حدي ضعيف‌تر عمل مي‌كنند كه اين امر به علت تطبيقي بودن اين روش از نظر اندازه جمعيت‌ها مي‌باشد كه با كوچكتر گرفتن اندازه جمعيت اين مشكل حل مي‌شود. به طور كلي، همانطور كه مشاهده مي‌شود الگوريتم‌هاي پيشنهادي به غير از محيط‌هايي با تعداد قله‌هاي كم كه به علت عدم تنظيم

مناسب اندازه جمعيت مي‌باشد، در ساير محيط‌هاي پويا كه درآن‌ها ارزيابي صورت گرفته از كارايي بسيار بالاتري برخوردارند.

براي نشان دادن توانايي الگوريتم‌هاي پيشنهادي در محيط هايي با تعداد قله‌هاي كم نيز، آزمايشاتي بر روي الگوريتم تكامل تفاضلي سلولي بهبود يافته با اندازه جمعيت متناسب با تعداد قله‌ها در محيط‌هايي با تعداد قله‌هاي ۱ و ۵ انجام شده است. در اين آزمايش، اندازه جمعيت برابر با حاضلصرب تعداد قله‌ها، m، در ظرفيت سلول، T_{init}=10، در نظر گرفته شده است. ساير پارامترها، همانند آزمايشات ذكر شده در بالا در نظر گرفته شده است.



شكل ۳-۲۳) نمودارهاي خطاي آفلاين و خطاي جاري الگوريتم تكامل تفاضلي سلولي و الگوريتم بهينه‌ساز گروه ذرات سلولي در حالت ۵۰ قله و فركانس ۵۰۰۰

همان طور كه در جدول ۳-۲۴ مشاهده مي‌شود، الگوريتم تكامل تفاضلي سلولي بهبود يافته از كارايي بسيار بالاتري نسبت به ساير روش‌هاي مورد مقايسه برخوردار است. اين امر نشان مي‌دهد كه الگوريتم پيشنهادي در صورت انتخاب اندازه جمعيت متناسب، در محيط‌هايي با تعداد قله‌هاي كم نيز از كارايي بالاتري برخوردار است.

جدول ۳-۲٤) خطاي آفلاين به همراه ميانگين خطاي استاندارد الگوريتم پيشنهادي با اندازه جمعيت مناسب به همراه نتايج ساير الگوريتم‌ها

دوره تغيير	p	Adaptive mQSO	mQSO	HmSO	CellularDE+ $M=100$	CellularDE+ $M=m*T_{init}$
۵۰۰	۱	5.08±0.27	36.52±3.2	8.53±0.49	12.18±0.27	**۰.۷۱**±۰.۰۱
	۵	5.14±0.09	13.50±0.8	7.40±0.31	6.52±0.06	**۴.۰۳**±۰.۰۴
۱۰۰۰	۱	2.68±0.14	19.1±1.5	4.46±0.26	6.64±0.19	۰.۳۷±۰.۰۱
	۵	3.22±0.07	7.59±0.39	4.27±0.08	5.25±0.06	**۰.۹۳**±۰.۰۴
۲۵۰۰	۱	1.09±0.06	7.79±0.72	1.75±0.10	1.78±0.06	۰.۱۸±۰.۰۲
	۵	1.58±0.13	3.53±0.18	1.92±0.11	2.45±0.06	۰.۹۷±۰.۰۴
۵۰۰۰	۱	0.55±0.02	0.53±0.01	0.87±0.05	0.79±0.02	۰.۱۱±۰.۰۱
	۵	1.00±0.04	1.05±0.06	1.18±0.04	1.57±0.06	۰.۷۶±۰.۰۵

٤-۳ نتيجه‌گيري

در اين فصل در ابتدا روند نسخه پايه‌اي الگوريتم پيشنهادي براي بهينه‌سازي در محيط‌هاي پويا به صورت كامل شرح داده شد. اين مدل پيشنهادي كه الگوريتم تكامل تفاضلي سلولي ناميده شد، بر اساس مدل اتوماتاي سلولي و با استفاده از تقسيم‌بندي فضاي جستجو طراحي شده است. در روش پيشنهادي، فضاي جستجو به قسمت‌هاي كوچكتر تقسيم‌بندي مي‌شود و يک اتوماتاي سلولي بر آن منطبق مي‌شود كه در آن هر سلول، وظيفه كنترل جستجو در هر قسمت را بر عهده دارد. جستجو در هر سلول و همسايگانش به صورت موازي و جداگانه، توسط الگوريتم تكامل تفاضلي انجام مي‌شود كه اين امر موجب حفظ تنوع مي‌شود.

با توجه به شرايط مدل پيشنهادي، يک مدل تكاملي نيز براي الگوريتم تكامل تفاضلي پيشنهاد شد كه علاوه بر دارا بودن كارايي بالاتر نسبت به مدل‌هاي متداول، موجي كاهش يكي از پارامترهاي الگوريتم نيز مي‌شود. همچنين يک روش جستجوي محلي با قابليت تنظيم دقت جواب پيشنهاد شد كه در محيط‌هايي با دوره تغيير بالا، كارايي را به صورت قابل توجهي بهبود مي‌دهد. در انتها، پس از شرح كامل روند روش‌هاي پيشنهادي، آزمايشاتي در محيط‌هاي پوياي مختلف، جهت ارزيابي و مقايسه كارايي آن‌ها با ساير روش‌هاي تكاملي طراحي شده براي محيط‌هاي پويا انجام گرفت. در اين آزمايشات نشان داديم كه در اكثر محيط‌ها، روش پيشنهادي از كارايي نسبي خوبي نسبت به ساير روش‌ها برخوردار است.

فصل چهارم

الگوریتم کوه‌نورد

(Alpinist CellularDE)

۴. الگوريتم تكامل تفاضلي سلولي كوه‌نورد[1]

يكي از مشكلات الگوريتم تكامل تفاضلي سلولي، كه در بيشتر الگوريتم‌هاي پيشنهاد شده براي محيط‌هاي پويا نيز مشاهده مي‌شود، اتلاف منابع (اتلاف تعداد محاسبات شايستگي) در اطراف مناطقي است كه يک بهينه محلي در آن يافت شده و نيازي به جستجوي عمومي در اطراف آن نمي‌باشد. در اين مناطق مي‌توان با كم كردن ظرفيت سلول از تجمع بي‌مورد در آن‌ها جلوگيري نمود. مشكل دوم اينست كه در سلول‌هايي كه حاوي يک قله مي‌باشند، پس از مدتي جمعيت دچار همگرايي بالايي شده است و ديگر تنوع كافي براي الگوريتم تكامل تفاضلي در آن وجود ندارد. اين كمبود تنوع روند استخراج را بسيار كند مي‌كند. در اين مناطق بهتر است از يک روش جستجوي محلي مانند الگوريتم تپه‌نوردي[2] به جاي تكامل تفاضلي استفاده كرد.

براي اين كار مي‌بايست در ابتدا بتوان سلول‌هايي را كه در آن‌ها بهينه‌اي وجود دارد را از سلول‌هاي دامنه مجاور قله و همچنين سلول‌هايي كه بسيار دور از بهينه هستند را از هم تفكيک نمود. در روش پيشنهاد شده سلول‌ها را به چهار گونه زير تقسيم‌بندي مي كنيم:

- سلول ناشناخته[3]: در ابتدا كه هنوز اطلاعاتي در مورد سلول‌ها وجود ندارد، تمامي سلول‌ها از اين گونه محسوب مي‌شوند.
- سلول قله[4]: سلولي كه تشخيص داده شود كه در آن حداقل يک بهينه محلي وجود دارد را سلول قله در نظر مي‌گيريم.
- سلول دامنه[5]: سلول‌هاي مجاور سلول قله كه در شيب به سمت آن بهينه قرار دارند را سلول دامنه در نظر مي گيريم.
- سلول كوهپايه[6]: سلول‌هاي مجاور سلول دامنه كه در پايين‌ترين منطقه نسبت به بهينه محلي قرار دارند را سلول كوهپايه در نظر مي گيريم.

[1] Alpinist CellularDE
[2] Hill Climbing
[3] Unknown Cell
[4] Foot Cell
[5] Footside Cell
[6] Hillside Cell

آستانه تعداد موجودات فعال در سلول و روش جستجو در آن، بر حسب گونه هر سلول تنظيم مي‌شود. نسخه پيشنهادي را الگوريتم تكامل تفاضل سلولي كوه‌نورد مي‌ناميم.

٤-١-١ روند الگوريتم تكامل تفاضلي سلولي كوه‌نورد

در الگوريتم كوه‌نورد براي هر سلول يک گونه به نام LT در نظر گرفته مي‌شود مشخص كننده نسبت آن سلول با يک بهينه محلي مي‌باشد. بر اساس اين گونه‌ها، ظرفيت سلول و الگوريتم جستجوي داخل آن سلول مشخص مي‌شود. در هر بار بروزرساني هر سلول گونه خود را مشخص كرده و بر اساس آن روند جستجو را انجام مي‌دهد.

با توجه به اينكه در ابتدا اطلاعاتي در مورد محيط و سلول‌ها در دست نداريم، تمامي سلول‌ها گونه ناشناخته در نظر گرفته مي‌شوند. براي اين گونه از سلول‌ها از تكامل تفاضلي به همان شكلي كه در الگوريتم تكامل تفاضلي سلولي وجود داشت، استفاده مي‌شود. ظرفيت اين گونه از سلول‌ها نيز برابر همان مقدار اوليه در نظر گرفته مي‌شود.

پس از مدتي از انجام جستجو، بهينه‌ها به تدريج پيدا خواهند شد. ساده‌ترين معياري كه براي تشخيص سلول‌هاي حاوي يک قله وجود دارد اينست كه روند بهبود شايستگي در يک سلول را طي يک زمان بررسي كنيم و در صورت عدم مشاهده بهبود مي توان حدس زد كه آنجا يک بهينه وجود دارد كه به حد كافي استخراج بر روي آن انجام پذيرفته است. همچنين حافظه سلول نيز مي‌بايست داخل خود آن سلول باشد، بدين معنا كه سلول حاضر در بين همسايگانش بهترين شايستگي را دارا باشد.

در روش پيشنهاد شده، شايستگي حافظه $cMem$ سلول در طي شش بار بروزرساني و تكامل در يک آرايه حافظه شش‌تايي به نام $cMemHis$ نگهداري مي‌شود و در صورتي كه ميزان بهبود كمتر از MI بود، آن سلول از گونه قله در نظر گرفته مي‌شود. بايد در نظر داشت كه در اين شش بار حتما بايد موجودي در سلول براي تكامل و جستجو وجود داشته باشد تا مقداري در اين حافظه ثبت گردد. همچنين پس از روي دادن هر تغيير در محيط، اين ليست حافظه خالي مي‌شود.

در سلول‌هاي قله، جستجو توسط الگوريتم تپه‌نوردي[1] انجام مي‌گيرد. زيرا در اين گونه سلول‌ها موجودات بيش از حد همگرا شده و دچار نوعي سكون گرديده‌اند و الگوريتم تكامل تفاضلي ديگر قادر به استخراج مناسب در آن منطقه نمي‌باشد. الگوريتم تكامل تفاضلي در اين حالات كارايي پاييني

[1] Hill Climbing

دارد و الگوريتم تپه نوردي روش مناسبي براي ادامه جستجو مي باشد. علت عدم توقف كامل جستجو در اين گونه سلول‌ها اينست كه ممكن است در تشخيص اشتباهي رخ داده باشد و با استفاده از اين جستجوي محلي ساده امكان تشخيص اشتباه و رفع آن نيز تا حدودي فراهم مي شود و ممكن است پس از مدتي سلول از اين حالت در آيد. الگوريتم تپه‌نوردي براي سلول‌هاي قله بدين گونه است كه يكي از ابعاد حافظه $cMem$ به صورت تصادفي يكنواخت با شعاع ۰.۲ جهش داده مي‌شود و در صورت بهتر شدن، جايگزين بدترين موجود مي‌شود.

سلول‌هايي كه در همسايگي يک سلول قله قرار دارند و مكان $localBest$ آن‌ها در اين سلول قله قرار دارد، سلول‌هايي هستند كه به احتمال زياد در دامنه قله قرار دارند. اين گونه از سلول‌ها نياز زيادي به ظرفيت بالا ندارند و ظرفيت آن‌ها به نصف مقدار اوليه كاهش داده مي‌شود. گونه اين سلول‌ها دامنه در نظر گرفته مي‌شود. سلول‌هايي وجود دارند كه حتي دورتر از سلوهاي دامنه نسبت به قله مي‌باشند. اين سلول‌ها در همسايگي يک سلول دامنه هستند و مكان $localBest$ آن‌ها در اين سلول دامنه همسايه قرار دارد. براي اين سلول‌ها نيز ظرفيت به نصف مقدار اوليه كاهش داده مي‌شود.

نكته ديگر، حذف مرحله نصف كردن ظرفيت سلول‌ها پس از انجام جستجوي محلي است. علت آن اينست كه اين مرحله با توجه به اضافه شدن روندهاي جديد در مورد اعمال محدوديت بر روي ظرفيت سلول‌ها، ديگر نيازي به آن نمي‌باشد و حتي كارايي را تا حد كمي كاهش مي‌دهد.

در نتيجه الگوريتم كوه‌نورد از طريق تشخيص مكان قله‌ها و دامنه‌هاي اطراف آن، ظرفيت اين سلول‌ها را به طور متناسب تنظيم مي نمايد تا بتواند از اتلاف منابع جلوگيري نمايد و تنوع و توزيع‌شدگي موجودات در فضا را نيز بالا مي‌برد كه اين امر موجب بهبود كارايي مي‌گردد. شبه كد تابع مشخص كننده گونه يک سلول براي سلول $cell_i$ در الگوريتم ۴-۱ درج شده است. همچنين ساختار كلي الگوريتم كوه‌نورد نيز كه از اين گونه‌بندي استفاده نموده است در الگوريتم ۴-۲ نشان داده شده است.

الگوریتم ٤-١) شبه کد روند مشخص کننده گونه یک سلول

Algorithm Landscape Type Detector

PROCEDURE setType

INPUT : $cell_i$

BEGIN

if $cMem_i$ is located in $cell_i$ and $\left|BestHis\left(Last\right) - BestHis\left(First\right)\right| < MI$ **then**

$LT_i = FOOT$

else-if $cBestMem_i$ is located in $cell_j$ **and** $i \neq j$ **and** $LT_j == FOOT$ **then**

$LT_i = FOOTSIDE$

$T_i = T_{init} / 2$

else-if $cMem_i$ is located in $cell_j$ **and** $i \neq j$ **and** $LT_j == FOOTSIDE$ **then**

$LT_i = HILLSIDE$

$T_i = T_{init} / 2$

else

$LT_i = UNKNOWN$

end-if

END

END-PROCEDURE

الگوریتم ٤-٢) شبه کد الگوریتم کوه‌نورد

Algorithm Alpinist CellularDE

Initialize a cellular automaton with $(N_p)^d$ equal-sized cells

Initialize population of size M randomly in the cellular automaton

do
  for all $cell_i$ in CA **do**
    Update $cMem_i$ according to eq. (٦-٣)
    Calculate $localBest_i$ according to eq. (٧-٣)
    If size of $cMemHis_i$>6 **then**
      Remove the first member of $cMemHis_i$
    end-if
    if there is any active population in $cell_i$ **then**
      Add $cMem_i$ to $cMemHis_i$
    end-if
    setType($cell_i$)
    if there was a change in the past N_{LS} iterations **then**
      Conduct a local search centered at $cMem_i$
    else
      if $LT_i == FOOT$ **then**
        Evolve active population in $cell_i$ by Hill Climbing
      else
        Evolve active population in $cell_i$ by the modified version of DE
      end-if
    end-if

    while population of $cell_i > T_{init}$
      Re-initialize the worst active individual to a random cell
    end-while

    if a change is detected in the environment **then**
      Re-evaluate $cMem_i$
      for all $individual_m$ in $cell_i$ **do**
        Re-evaluate $individual_m$
      end-for
      Clear $cMemHis_i$
    end-if
  end-for
until a termination condition is met

٤-٢ ارزيابي كارايي الگوريتم كوه‌نورد

در اين آزمايش سعي مي‌كنيم تا كارايي الگوريتم كوه‌نورد (Alpinist CellularDE) و همچنين جستجوي بهبود يافته را با ساير الگوريتم‌ها در محيط‌هاي پويا مورد ارزيابي و مقايسه قرار دهيم. كارايي تمامي الگوريتم‌ها بر اساس خطاي آفلاين در محيط‌هاي پوياي با دوره تغيير ۵۰۰، ۱۰۰۰، ۲۵۰۰ و ۵۰۰۰ و به ازاي تعداد قله‌هاي مختلف ۱، ۵، ۱۰، ۲۰، ۳۰، ۴۰، ۵۰، ۱۰۰ و ۲۰۰ مورد مقايسه قرار گرفته است. الگوريتم‌هاي پيشنهادي را بر روي مقادير پيش‌فرض سناريوي ۲ تابع محک قله‌هاي متحرک كه در جدول ۳-۱ درج شده است، مورد ارزيابي قرار مي‌دهيم.

جهت مقايسه كارايي روش‌هاي پيشنهادي، تعدادي از اين الگوريتم‌ها كه بالاترين كارايي را بر روي تابع محک قله‌هاي متحرک از خود نشان داده‌اند انتخاب شده‌اند. نيز از جهت شباهت بسيار ساختاري و سلولي بودن انتخاب شده است تا مقايسه‌اي با ساير الگوريتم‌هاي سلولي نيز انجام شود. الگوريتم‌هاي mQSO [۲۵] Adaptive mQSO، [۲۶] و HmSO [۳۱] نيز از الگوريتم‌هاي شاخص در بهينه‌سازي در محيط‌هاي پويا هستند كه نتايج بسيار خوبي نسبت به ساير الگوريتم‌ها تا كنون از خود نشان داده‌اند[۳۴, ۵۲]. البته بايد توجه داشت كه با توجه به اينكه روش‌هاي پيشنهادي براي بهينه‌سازي در محيط‌هاي پيوسته طراحي شده‌اند، اين الگوريتم‌ها از بين آن‌هايي انتخاب شده‌اند كه براي محيط‌هاي پيوسته هستند. براي الگوريتم‌هاي پيشنهادي، از مقادير پيش‌فرض جدول ۳-۲ و براي ساير روش‌ها از مقادير پيش‌فرض ذكر شده در خود مقالات پيشنهاد دهنده استفاده شده است. مقدار حد آستانه MI برابر ۰.۰۱ در نظر گرفته شده است.

نتايج بدست‌آمده از الگوريتم‌هاي مختلف براي دوره‌هاي تغيير مختلف در جدول ۴-۱ تا جدول ۴-۴ درج شده است. نتايج گزارش شده براي الگوريتم‌هاي HmSO، mQSO، Adaptive mQSO و Cellular PSO از مقاله [۳۱] استخراج شده است ولي الگوريتم DynDE پياده‌سازي شده است و نتايج بدست آمده از اجراي اين پياده‌سازي گزارش شده است. نتايج به صورت ميانگين خطاي آفلاين براي ۱۰۰ بار اجرا به همراه خطاي استاندارد در جداول درج شده است.

در اين آزمايش از روش تست آماري دانش‌آموز با فاصله اطمينان ۹۵% براي مقايسه نتايج الگوريتم‌ها استفاده شده است. در هر محيط آن روشي كه از نظر آماري بهتر از بقيه باشد را پررنگ كرده‌ايم و اگر چند روش نسبت به ديگران برتري داشته باشند همه آن‌ها پررنگ شده‌اند.

جدول ٤-١) خطاي آفلاين به همراه ميانگين خطاي استاندارد الگوريتم‌هاي مختلف به ازاي تعداد قله‌اي مختلف براي دوره تغيير ٥٠٠

نام الگوريتم							
CellularDE	Alpinist CellularDE	CellularDE+	Alpinist CelluarDE+	HmSO	mQSO	Adaptive mQSO	p
8.20±0.19	6.40±0.14	12.18±0.27	5.76±0.12	8.53±0.49	36.52±3.2	٥.٠٨±٠.٢٧	١
6.06±0.05	5.41±0.05	6.52±0.06	9.41±0.07	7.40±0.31	13.50±0.8	٥.١٤±٠.٠٩	٥
5.93±0.04	٥.٥٨±٠.٠٤	6.26±0.04	8.03±0.05	7.56±0.27	11.18±0.4	6.20±0.11	١٠
٥.٦٠±٠.٠٣	6.18±0.04	7.43±0.04	7.79±0.03	7.81±0.20	10.54±0.2	6.94±0.18	٢٠
٥.٥٦±٠.٠٣	6.43±0.03	7.80±0.04	8.20±0.02	8.33±0.18	10.37±0.2	7.23±0.16	٣٠
٥.٤٨±٠.٠٢	6.50±0.03	7.94±0.04	8.46±0.02	8.45±0.18	10.32±0.2	7.43±0.17	٤٠
٥.٤٧±٠.٠٢	6.50±0.03	8.03±0.04	8.53±0.02	8.83±0.17	10.33±0.2	7.49±0.09	٥٠
٥.٢٩±٠.٠٢	6.32±0.03	7.98±0.04	8.43±0.02	8.85±0.16	9.93±0.21	7.29±0.15	١٠٠
٥.٠٧±٠.٠٢	5.98±0.03	7.59±0.03	7.88±0.02	8.85±0.16	9.67±0.20	6.82±0.14	٢٠٠

جدول ٤-٢) خطاي آفلاين به همراه ميانگين خطاي استاندارد الگوريتم‌هاي مختلف به ازاي تعداد قله‌اي مختلف براي دوره تغيير ١٠٠٠

نام الگوريتم							
CellularDE	Alpinist CellularDE	CellularDE+	Alpinist CelluarDE+	HmSO	mQSO	Adaptive mQSO	p
4.70±0.10	3.96±0.09	6.64±0.19	3.02±0.06	4.46±0.26	19.1±1.5	٢.٦٨±٠.١٤	١
4.05±0.04	٣.٨٤±٠.٠٤	5.25±0.06	6.12±0.06	4.27±0.08	7.59±0.39	3.22±0.07	٥
3.95±0.03	٣.٥٩±٠.٠٣	4.29±0.04	4.37±0.06	4.61±0.07	6.33±0.23	4.11±0.08	١٠
4.02±0.03	٣.٩٠±٠.٠٣	3.97±0.03	4.10±0.03	4.66±0.12	6.46±0.20	4.75±0.14	٢٠
٣.٩٦±٠.٠٣	٤.٠٥±٠.٠٣	4.28±0.03	4.49±0.02	4.83±0.09	6.51±0.16	4.98±0.10	٣٠
٣.٩٦±٠.٠٢	٤.٠٩±٠.٠٣	4.43±0.03	4.62±0.02	4.82±0.09	6.43±0.18	5.10±0.11	٤٠
٣.٩٧±٠.٠٢	٤.٠٨±٠.٠٣	4.47±0.03	4.67±0.02	4.96±0.03	6.67±0.16	5.12±0.05	٥٠
٣.٨١±٠.٠٢	3.99±0.02	4.57±0.03	4.75±0.02	5.14±0.08	6.34±0.12	5.03±0.09	١٠٠
٣.٥٨±٠.٠١	3.77±0.02	4.39±0.02	4.56±0.02	5.25±0.08	6.13±0.12	4.65±0.09	٢٠٠

جدول ٤-٣) خطاي آفلاين به همراه ميانگين خطاي استاندارد الگوريتم‌هاي مختلف به ازاي تعداد قله‌اي مختلف براي دوره تغيير ٢٥٠٠

نام الگوريتم							
CellularDE	Alpinist CellularDE	CellularDE+	Alpinist CelluarDE+	HmSO	mQSO	Adaptive mQSO	p
2.22±0.08	2.22±0.06	1.78±0.06	2.83±0.05	1.75±0.10	7.79±0.72	**١.٠٩±٠.٠٦**	**١**
2.36±0.05	2.18±0.03	2.45±0.06	2.92±0.05	1.92±0.11	3.53±0.18	**١.٥٨±٠.١٣**	**٥**
2.48±0.04	**٢.١١±٠.٠٣**	**٢.٢٠±٠.٠٤**	2.39±0.04	2.39±0.16	3.20±0.14	2.33±0.11	**١٠**
2.70±0.04	2.35±0.03	**١.٨٨±٠.٠٢**	2.14±0.03	2.46±0.09	3.83±0.12	2.84±0.09	**٢٠**
2.81±0.03	2.58±0.04	**١.٨٦±٠.٠٢**	2.06±0.02	2.57±0.05	4.03±0.12	3.13±0.09	**٣٠**
2.80±0.03	2.65±0.03	**١.٨٥±٠.٠٢**	2.02±0.03	2.56±0.06	3.90±0.11	3.23±0.08	**٤٠**
2.75±0.03	2.65±0.03	**١.٩٠±٠.٠٢**	2.09±0.02	2.65±0.05	3.95±0.10	3.24±0.07	**٥٠**
2.76±0.02	2.65±0.03	**٢.٠٧±٠.٠٢**	**٢.٠١±٠.٠٢**	2.72±0.04	3.81±0.10	3.20±0.06	**١٠٠**
2.57±0.02	2.44±0.02	2.20±0.02	**١.٨٢±٠.٠١**	2.81±0.04	3.66±0.07	3.00±0.05	**٢٠٠**

جدول ٤-٤) خطاي آفلاين به همراه ميانگين خطاي استاندارد الگوريتم‌هاي مختلف به ازاي تعداد قله‌اي مختلف براي دوره تغيير ٥٠٠٠

نام الگوريتم							
CellularDE	Alpinist CellularDE	CellularDE+	Alpinist +CellularDE	HmSO	mQSO	Adaptive mQSO	p
1.52±0.07	1.51±0.06	0.79±0.02	0.65±0.02	0.87±0.05	**٠.٥٣±٠.٠١**	**٠.٥٥±٠.٠٢**	**١**
1.53±0.06	1.51±0.04	1.57±0.06	1.32±0.03	1.18±0.04	**١.٠٥±٠.٠٦**	**١.٠٠±٠.٠٤**	**٥**
1.73±0.04	1.50±0.04	1.27±0.04	**١.٠٥±٠.٠٣**	1.42±0.04	1.31±0.03	1.43±0.04	**١٠**
2.04±0.04	1.83±0.04	**١.٢٤±٠.٠٣**	1.46±0.04	1.50±0.06	1.69±0.05	1.95±0.05	**٢٠**
2.18±0.04	1.98±0.04	**١.٣٧±٠.٠٣**	1.63±0.03	1.65±0.04	1.78±0.02	2.15±0.05	**٣٠**
2.22±0.04	2.12±0.04	**١.٣٨±٠.٠٢**	1.72±0.03	1.65±0.05	1.86±0.02	2.28±0.04	**٤٠**
2.23±0.03	2.11±0.04	**١.٤٣±٠.٠٢**	1.73±0.03	1.66±0.02	1.95±0.02	2.28±0.02	**٥٠**
2.24±0.02	2.08±0.03	1.42±0.02	**١.٠٣±٠.٠١**	1.68±0.03	1.95±0.01	2.31±0.03	**١٠٠**
2.10±0.02	1.95±0.02	1.35±0.01	**١.٠٣±٠.٠١**	1.71±0.02	1.90±0.01	2.11±0.03	**٢٠٠**

همان‌طور كه مشاهده مي‌شود الگوريتم كوه‌نورد در دوره‌هاي تغيير بالا (دوره‌هاي ٥٠٠٠ و ٢٥٠٠) كارايي بالاتري نسبت به الگوريتم تكامل تفاضلي سلولي از خود نشان داده است كه علت آن بهبود استفاده از جمعيت موجود از طريق كنترل ظرفيت سلول‌ها و همچنين بهبود استخراج قله‌ها با

استفاده از الگوريتم جستجوي محلي تپه‌نورد بر روي قله‌هايي كه موجودات بر روي آن دچار كاهش تنوع شده‌اند، مي‌باشد.

ولي اين برتري در دوره‌هاي تغيير پايين‌تر (دوره‌هاي تغيير ۱۰۰۰ و ۵۰۰) وجود ندارد. يكي از علل آن مي‌تواند عدم وجود فرصت كافي موجودات براي همگرايي باشد زيرا فاصله بين تغييرات محيط كم است و قبل از اينكه موجودات دچار همگرايي بيش از حد بشوند و بتوان از طريق روند ذكر شده، وجود قله‌ها در يک سلول را تاييد نمود، تغيير بعدي روي مي‌دهد. علت ديگر آنست كه اين كاهش ظرفيت سلول‌ها موجب كاهش تعداد موجودات در اطراف قله ها مي‌شود و هنگام جابه‌جايي قله‌ها نياز به مدت بيشتري جهت يافتن دوباره قله‌ها مي‌باشد. اين امر در دوره‌هاي تغيير بالاتر نيز وجود دارد ولي در آنجا زمان بيشتري تا تغيير بعدي جهت يافتن قله‌ها در اختيار الگوريتم قرار دارد.

برخلاف الگوريتم تكامل تفاضلي سلولي كه تنها در دوره‌هاي تغيير پايين (۵۰۰، ۱۰۰۰ و ۲۵۰۰) داراي برتري قابل ملاحظه‌اي نسبت به الگوريتم Adaptive mQSO مي باشد و در دوره تغيير ۵۰۰۰ داراي كارايي تقريبا يكساني با آن مي‌باشد، در نسخه كوه‌نورد با افزايش كارايي در دوره‌هاي تغيير بالاتر اين برتري مشهودتر شده است. البته همچنان در محيط‌هايي با تعداد قله‌هاي پايين‌تر از ۱۰ اين برتري وجود ندارد كه علت آن همان‌طور كه قبلا هم گفته شد، اتلاف جمعيت در اثر عدم امكان همگرايي موجودات زياد بر روي يک قله مي‌باشد كه در نتيجه آن موجودات زيادي در حال بيرون افتادن از سلول‌ها و انجام جستجوي عمومي بي‌مورد هستند. در محيط‌هايي با تعداد قله‌هاي بالاتر به علت كم بودن نسبي اندازه جمعيت به تعداد قله‌ها، ظرفيت سلول امر بازدارنده‌اي در اين زمينه ايفا نمي‌كند. در صورتي كه الگوريتم Adaptive mQSO با تطبيقي‌سازي اندازه جمعيت از اتلاف بي مورد موجودات جلوگيري مي‌كند.

البته اين افزايش كارايي در دوره‌هاي تغيير بالا آن قدري نيست كه توانسته باشد كارايي بهتري از الگوريتم HmSO از خود نشان دهد و همانند نسخه پايه، در دوره‌هاي تغيير بالاتر از كارايي پايين‌تري نسبت به اين الگوريتم برخوردار است هر چند اين فاصله در كارايي تا حدي كاهش يافته است. همين روند تا حدودي بين نسخه كوه‌نورد با الگوريتم mQSO نيز وجود دارد.

همان‌طور كه در جداول نتايج مشخص است، نسخه كوه‌نورد با وجود افزايش كارايي در دوره تغيير ۵۰۰۰ و نزديک شدن به كارايي الگوريتم mQSO باز قادر به ايجاد برتري نسبت به اين الگوريتم نشده است. اين عدم برتري در اين دوره تغيير براي نسخه پايه نيز صادق بود و تنها توانستيم اين فاصله را در نسخه كوه‌نورد تا حدود زيادي كاهش دهيم.

همان‌طور که از نتايج مشخص است، نسخه کوه‌نورد بهبود يافته در اکثر محيط‌ها، نسبت به نسخه معمولي کوه‌نورد هيچ افزايش کارايي‌اي از خود نشان نمي‌دهد. علت اين امر آنست که تمهيداتي که در الگوريتم کوه‌نورد براي تنظيم ظرفيت سلول‌ها در نظر گرفته شده است موجب پخش شدن بيش‌تر موجودات و در نتيجه پوشش تعداد سلول‌هاي بيش‌تر مي‌شود. همان‌طور که قبلا توضيح داده شد الگوريتم جستجوي محلي بهبود يافته به تعداد زيادي محاسبات شايستگي نياز دارد که اين افزايش تعداد سلول‌هاي حاوي موجود فعال، موجب افزايش بيش از حد تعداد محاسبات شايستگي مصرف شده مي‌شود.

اين کاهش کارايي حتي در محيط‌هايي با تعداد قله‌هاي کم موجب ضعيف شدن نتايج حتي نسبت به الگوريتم تکامل تفاضلي سلولي شده است. زيرا در محيط‌هايي با تعداد قله‌هاي کم که در آن‌ها تعداد موجودات بيش از حد نياز است، اين پديده بيش‌تر قابل مشاهده است. در نتيجه مي‌توان گفت افزودن روش جستجوي بهبود يافته به نسخه کوه‌نورد نه تنها موجب بالارفتن کارايي نمي‌شود بلکه کارايي را کاهش نيز مي‌دهد و در نتيجه بي‌فايده است.

٤-٣ نتيجه‌گيري

يکي از مشکلات الگوريتم تکامل تفاضلي سلولي پايه اتلاف منابع در اطراف مناطقي است که يک بهينه محلي در آن يافت شده و نيازي به جستجوي عمومي در اطراف آن نمي‌باشد. براي کاهش اثر اين مشکل، در اين فصل نسخه‌اي از الگوريتم تکامل تفاضلي سلولي پيشنهاد شد که با گونه‌بندي سلول‌ها بر مبناي تشخيص سلول حاوي قله، سعي در بهبود کارايي و بازده موجودات فعلي دارد. نسخه پيشنهادي را الگوريتم تکامل تفاضل سلولي کوه‌نورد ناميديم.

پس از شرح الگوريتم پيشنهادي، آزمايشاتي بر روي آن در محيط‌هاي مختلف انجام گرفت و کارايي آن با الگوريتم ديگر مقايسه و بررسي شد. در آزمايشات مشاهده شد که الگوريتم پيشنهادي کوه‌نورد، در محيط‌هاي مورد آزمايش کارايي بالاتري نسبت به نسخه پايه دارد. البته مشاهده شد که افزودن جستجوي محلي بهبوديافته به اين نسخه، تنها در دوره‌هاي تغيير بالا موجب افزايش کارايي مي‌شود و استفاده از آن در ساير ميحط‌ها بي‌فايده مي‌باشد. علت آن افزايش تنوع و توزيع‌شدگي بيشتر موجودات در فضا مي‌باشد که با توجه به استفاده زياد محاسبات شايستگي توسط الگوريتم جستجوي محلي بهبود يافته، اين امر موجب کاهش کارايي مي‌شود.

فصل پنجم

الگوريتم‌هاي

تكامل تفاضلي سلولي تطبيقي

۵. الگوريتم‌هاي تكامل تفاضلي سلولي تطبيقي

الگوريتم تكامل تفاضلي به كار گرفته شده در مدل‌هاي سلولي پيشنهاد شده در فصول قبل داراي پارامترهايي از قبيل F و Cr است كه اين پارامترها تا كنون داراي يک مقدار ثابت در تمامي سلول‌ها بوده است. براي بدست آوردن حداكثر كارايي ممكن، مي‌بايست مقادير مناسب براي اين پارامترها را از طريق سعي و خطا بدست آورد كه اين مقادير براي محيط‌هاي مختلف متفاوت بدست مي‌آيند. از طرفي با توجه به تعداد زياد سلول‌ها و تغييرپذير بودن محيط با زمان، براي تمامي سلول‌ها مقادير ثابت و يكساني در نظر گرفته مي‌شود.

يكي از راه‌هاي كاهش وابستگي الگوريتم جستجو به تغييرات و تفاوت‌هاي محيط، تطبيقي نمودن پارامترهاي جستجو مي‌باشد. اين عمل حتي در بسياري از موارد مي‌تواند موجب بهبود نتايج نسبت به روش مقدار ثابت نيز گردد. به ويژه در محيط‌هاي پويا، پهنه جستجو مدام در حال تغيير مي‌باشد و نياز به اكتشاف و استخراج نه تنها در مكان‌هاي مختلف، بلكه در زمان‌هاي مختلف، متفاوت مي‌باشد. يكي از راه‌هاي كنترل ميزان توانايي استخراج و اكتشاف يک الگوريتم تكاملي، تنظيم پارامترهاي آن از قبيل احتمال جهش و بازتركيبي است كه در مدل پيشنهادي ما اين پارامترها F و Cr مي‌باشند.

۵-۱ الگوريتم تكامل تفاضلي يادگير سلولي[1]

يكي از روش‌هايي كه براي تنظيم خودكار مقدار يک پارامتر مي‌توان استفاده نمود كه ديگر نيازي به دخالت كاربر نباشد و همچنين امكان تطابق‌پذيري با زمان و مكان را نيز داشته باشد، استفاده از يادگيري تقويتي است. يادگيري تقويتي با استفاده از پاسخ‌هايي كه از محيط در ازاي اعمال خود دريافت مي‌كند قادر است مقدار مناسب يک پارامتر را بيابد.

در مدل سلولي پيشنهاد شده ما دو پارامتر F و Cr ،كه متعلق به الگوريتم تكامل تفاضلي هستند، وجود دارد كه مي‌توان مقدار مناسب آن‌ها را از طريق يادگيري بدست آورد. بدين منظور در داخل هر سلول، به ازاي هر پارامتر، يک اتوماتاي يادگير قرار مي‌دهيم كه وظيفه بدست آوردن مقادير

[1] Cellular Learning DE (CLDE)

بهينه آن پارامتر را بر عهده دارد که با تفاوت‌هاي جزيي، يک اتوماتاي يادگير سلولي محسوب مي‌شود. به همين علت نام مدل جديد را الگوريتم تکامل تفاضلي يادگير سلولي مي ناميم.

۵-۱-۱ روند الگوريتم پيشنهادي

در الگوريتم پيشنهادي (تکامل تفاضلي يادگير سلولي) به ازاي هر پارامتر در داخل هر سلول يک اتوماتاي يادگير قرار دارد (در سلول iام، اتوماتاي يادگير LAC_i براي پارامتر Cr و LAF_i براي پارامتر F). با توجه به اينکه پارامترهاي الگوريتم تکامل تفاضلي در بازه [0 1] در نظر گرفته شده‌اند، مجموعه $\alpha_{Cr} \equiv \{0.1, 0.2, ..., 0.9\}$ براي اتوماتاي يادگير پارامتر Cr و مجموعه $\alpha_F \equiv \{0.1, 0.2, ..., 0.9, 1.0\}$ براي اتوماتاي يادگير پارامتر F، به عنوان مجموعه اعمال در نظر گرفته مي‌شود. در هر سلول براي هر اتوماتاي يادگير، يک بردار احتمال متناظر و هم‌اندازه با مجموعه اعمالش تعريف مي‌شود که در ابتدا به صورت يکنواخت مقداردهي مي‌شوند.

در هر بار بروزرساني، اتوماتاي يادگير متناظر با هر پارامتر در داخل هر سلول، با توجه به بردار احتمال خود يکي از اعمال را انتخاب و از مقدار متناظر آن عمل براي آن پارامتر در عمل تکامل در آن سلول استفاده مي‌کند. اگر موجودي در آن سلول وجود داشته باشد و تکامل بر روي آن‌ها انجام پذيرد، يک سيگنال پاسخ توليد مي‌شود که مقدار آن از رابطه (۵-۱) محاسبه مي‌شود و در غير اين صورت صفر در نظر گرفته مي‌شود. در اين رابطه $\beta_i(t)$ پاسخ به عمل انجام شده توسط سلول iام، $SN_i(t)$ تعداد موجوداتي که شايستگي آن‌ها در طي تکامل بهبود يافته و $PN_i(t)$ تعداد کل موجودات تکامل يافته در بروزرساني tام مي‌باشد.

$$\beta_i(t) = 1 - \frac{SN_i(t)}{PN_i(t)} \qquad (۵-۱)$$

اين سيگنال پاسخ با استفاده از الگوريتم يادگيري L_{RI} در محيط مدل S به بردار احتمالات عمل‌ها اعمال مي‌شود. در اين روش، تنها از پاداش استفاده مي‌شود و تنبيه‌اي صورت نمي‌گيرد. علت آن اينست که بدتر شدن بردار کانديد ساخته شده توسط يک پارامتر نشانگر مناسبي از مناسب نبودن آن مقادير نمي‌باشد و اتفاقا در اکثر مواقع موجودات بهتر نمي‌شوند و سيگنال پاسخ عدد نزديک به يک نيست. هر چند اين موضوع در صورت موفقيت نيز صادق است ولي مقادير پارامترهايي مناسب با احتمال بيشتري موجب موفقيت مي‌شوند. از طرف ديگر اين نداشتن جريمه موجب کاهش تعداد پارامترهاي مورد نياز جهت تنظيم نيز مي‌شود. در نتيجه مقدار پارامتر پاداش a_L و پارامتر جريمه صفر در نظر گرفته مي‌شود.

با توجه به پويا بودن محيط و جابه‌جايي بهينه‌ها، اين امكان وجود دارد كه بهينه‌ها از داخل يک سلول به سلول ديگر بروند. با توجه به اينكه يادگيري هر سلول مختص به خود مي‌باشد. اين امر موجب از دست رفتن تجربيات سلول قبلي مي‌شود و مدتي زمان مي‌برد تا اين يادگيري در سلول جديد دوباره انجام گردد. علاوه بر آن با توجه به وجود شباهت مكاني و محلي استفاده از تجربيات سلول‌هاي همسايه مي‌تواند مفيد واقع گردد.

به همين جهت، علاوه بر سيگنال پاسخي كه از داخل خود سلول دريافت مي‌شود، سيگنال‌هاي پاسخ ديگري نيز از سلول‌هاي همسايه كه در آن‌ها طي بروزرساني قبلي تكاملي صورت گرفته نيز دريافت مي‌شود. در بروزرساني tام براي سلول iام كه در M سلول همسايه آن در بروزرساني قبلي موجودي براي تكامل وجود داشته، سيگنال پاسخ از سلول‌هاي همسايه از رابطه (۵-۲) بدست مي‌آيد.

$$\beta_i^N(t) = \frac{\sum_{k=1}^{M} \beta_k^N(t)}{M} \qquad (۵-۲)$$

كه در آن $\beta_k^N(t)$ دريافتي از سلول همسايه در صورتي برابر صفر است كه آن سلول همسايه در بروزرساني قبلي همان عمل سلول جاري را انجام داده باشد و در غير اين صورت يک در نظر گرفته مي‌شود. اين سيگنال تقويتي كه از اعمال سلول‌هاي همسايه توليد مي‌شود را نيز به بردار احتمال عمل‌ها، اعمال مي‌كنيم. براي اعمال اين سيگنال، از الگوريتم يادگيري L_{RP} استفاده مي‌كنيم. براي اين اعمال مقدار جريمه b_N و پاداش a_N در نظر گرفته مي‌شود كه قاعدتا مقادير برابري خواهند داشت.

دريافت سيگنال‌هاي تقويتي از سلول‌هاي همسايه كه در آن‌ها تكامل در حال انجام است موجب انتشار تجربيات بدست آمده از اين سلول‌ها به سلول‌هاي مجاور مي‌شود و در صورتي كه يک قله به سلول مجاور جابه‌جا شود، اين تجربيات سرعت يادگيري را بالا مي‌رود. از طرفي اين كار موجب تبادل تجربه بين سلول‌ها نيز مي‌گردد. شبه كد الگوريتم پيشنهاد شده كه الگوريتم تكامل تفاضلي يادگير سلولي نامگذاري شده است در الگوريتم ۵-۱ نشان داده شده است.

Algorithm Cellular Learning DE (CLDE)

Initialize a cellular automaton with $(N_p)^d$ equal-sized cells

Initialize population of size M randomly in the cellular automaton

do
  for all cell$_i$ in CA **do**
    Update $cMem_i$ according to eq. (٦-٣)
    Calculate $localBest_i$ according to eq. (٧-٣)
    if there was a change in the past N_{LS} iterations **then**
      for all $individual_m$ in cell$_i$ **do**
        Set $individual_m$ to a random position in a hyper-sphere with radius LS_r centered at $cMem_i$ if that random position has better fitness.
      end-for
    else
      The LAC_i choose an action according to his probability vector and F is set to that
      The LAF_i choose an action according to his probability vector and Cr is set to that
      Evolve active population in $cell_i$ by the modified version of DE with chosen parameters
      Calculate $\beta_i(t)$ and $\beta_i^N(t)$
      Update the probability vectors of LAC_i and LAF_i according to $\beta_i(t)$ and $\beta_i^N(t)$
    end-if

    if it is the first iteration after the local search **then**
      $T = T_i / 2$
    else
      $T = T_i$
    end-if

    while population of $cell_i > T_{init}$
      Re-initialize the worst active individual to a random cell
    end-while

    if a change is detected in the environment **then**
      Re-evaluate $cMem_i$
      for all $individual_m$ in $cell_i$ **do**
        Re-evaluate $individual_m$
      end-for
    end-if
  end-for
until a termination condition is met

۵-۲ الگوريتم تكامل تفاضلي فازي سلولي[1]

استفاده از سيستم‌هاي كنترل فازي يكي ديگر از روش‌هاي رايج تنظيم پارامتر الگوريتم‌هاي مختلف از جمله الگوريتم‌هاي تكاملي محسوب مي‌شود. اين گونه از سيستم‌هاي كنترل پارامتر فازي مبتني بر يک مجموعه قوانين فازي است كه توسط طراح الگوريتم معرفي شده است. اين مجموعه قوانين فازي بر حسب شرايط محيط، روند جستجو و جمعيت، قادر به انتخاب مناسب مقادير پارامترهاي مورد نياز الگوريتم مي‌باشد. استفاده از اين سيستم فازي، نياز به تنظيم پارامترهاي الگوريتم را از بين مي‌برد و همچنين اين امكان را فراهم مي كند كه مقادير اين پارامترها در طول زمان جستجو، تغيير و با وضعيت فعلي تطبيق يابند. اين تطبيق در طي زمان است كه امكان بهبود نتايج نسبت به حالت مقادير ثابت را فراهم مي‌كند. علاوه بر آن با توجه به سلولي بودن الگوريتم پيشنهادي، براي هر سلول مقادير اين پارامترها به صورت جداگانه و جزيي از حالت هر سلول در نظر گرفته مي‌شود. اين امر از جهتي با توجه به محلي بودن دسترسي به اطلاعات موجب كندي تنظيم مقادير خواهد شد ولي از طرف ديگر اين امكان را فراهم مي‌كند كه حتي در يک زمان واحد، در نقاط مختلف محيط بر حسب شرايط محيطي مقادير متفاوتي براي اين پارامترها در نظر گرفته شود. اين تطبيقي بودن محلي، براي محيط‌هايي كه پهنه آن‌ها در نقاط مختلف داراي تفاوت‌هاي زياد هستند بسيار مناسب به نظر مي‌رسد.

در مدل پيشنهاد شده ما، پارامترهاي تكاملي F و Cr مي‌باشند. براي هر سلول مقادير اين پارمترها به صورت جداگانه و با استفاده از يک مجموعه قوانين فازي محاسبه خواهد شد. اين مجموعه قوانين بر اساس دو معيار ميزان تغييرات در شايستگي و مكان موجودات مقادير پارامترهاي الگوريتم تكامل تفاضلي را تنظيم خواهد كرد. با توجه به محلي بودن مدل، قاعدتا تنها از اطلاعات موجودات داخل همسايگي هر سلول مي‌توان استفاده نمود. مدل پيشنهاد شده را مي‌توان نوعي از يک اتوماتاي سلولي فازي[2] در نظر گرفت. اين الگوريتم پيشنهادي مبتني بر قوانين فازي را الگوريتم تكامل تفاضلي فازي سلولي مي ناميم.

۵-۲-۱ روند الگوريتم پيشنهادي

در مدل پيشنهادي، در داخل هر سلول يک سيستم استنتاج فازي مبتني بر قوانين قرار داده مي‌شود كه وظيفه محاسبه مقادير پارامترهاي F و Cr را بر عهده دارد. اين سيستم فازي بر اساس

[1] Fuzzy CellularDE (FCDE)
[2] Fuzzy Cellular Automata (FCA)

تغییرات شایستگي و مکان موجودات داخل همسایگي تصمیم‌گیري مي‌نماید. بدین منظور براي هر سلول دو معیار C_f و C_ℓ که به ترتیب برابر با میانگین تغییرات شایستگي و مکان موجودات نسبت به بروزرساني قبلي مي‌باشد را تعریف مي‌کنیم. در هنگام بروزرساني n ام، این دو معیار براي سلول $cell_i$ که داراي M موجود فعال در خود و سلول‌هاي همسایه‌اش مي‌باشد، از طریق روابط (۵-۳) و (۵-۴) محاسبه مي‌شود. در این روابط، x_j^n و x_j^{n-1} به ترتیب مکان موجود j ام در بروزرساني n ام و n-1 ام را نشان مي‌دهد. در رابطه (۵-۳) d نیز تعداد ابعاد موجودات مي‌باشد.

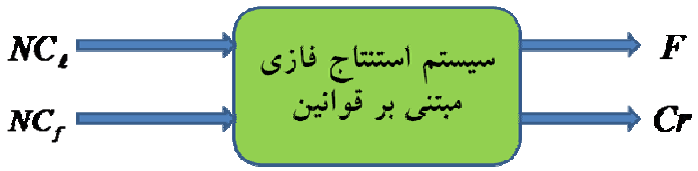
$$C_\ell^i = \sqrt{\frac{1}{M}\sum_{j=1}^{M}\sum_{k=1}^{d}\left(x_{jk}^{n} - x_{jk}^{n-1}\right)^2} \qquad (۵-۳)$$

$$C_f^i = \sqrt{\frac{1}{M}\sum_{j=1}^{M}\left(fit(x_j^n) - fit(x_j^{n-1})\right)^2} \qquad (۵-٤)$$

خروجي این دو معیار داراي محدوده مشخصي نمي‌باشند و بر حسب مساله و تابع شایستگي مي‌توانند داراي محدوده‌هاي متفاوتي باشند. اما ورودي‌هاي سیستم‌هاي فازي مبتني بر قوانین، مي‌بایست در یک محدوده مشخص باشند تا بتوان آن‌ها را فازي نمود. بدین علت این دو معیار را توسط روابط (۵-۵) و (۵-۶) نرمال‌سازي مي کنیم تا مقادیر بین [0 1] داشته باشند. این مقادیر نرمال شده را که NC_f و NC_ℓ مي‌نامیم را به عنوان ورودي‌هاي سیستم فازي در نظر مي‌گیریم. شماي کلي این سیستم فازي که داراي ورودي‌هاي NC_f و NC_ℓ و خروجي‌هاي F و Cr مي‌باشد در شکل ۵-۱ نمایش داده شده است.

$$NC_f^i = 1-(1+C_f^i)\times e^{-C_f^i} \qquad (۵-۵)$$

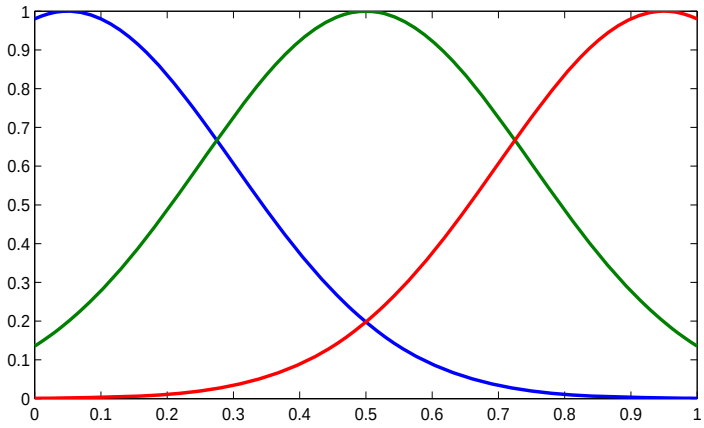
$$NC_\ell^i = 1-(1+C_\ell^i)\times e^{-C_\ell^i} \qquad (۵-۶)$$



شکل ۵-۱) نماي کلي سیستم کنترل فازي پارامترها

در هنگام ساخت قوانین فازي نیاز به تعریف مجموعه‌هاي فازي براي ورودي‌ها و همچنین خروجي‌هاي فازي مي‌باشد. بدین منظور براي ورودي‌ها و خروجي‌ها سه مجموعه فازي گوسي *SMALL*، *MEDIUM* و *BIG* را تعریف مي‌کنیم و مرکز و سیگماي این سه مجموعه را به ترتیب

برابر با (0.35 , 0.05)، (0.35 , 0.5) و (0.35,0.95) در نظر مي‌گيريم. اين سه مجموعه كه در شكل ۵-۲ نمايش داده شده‌اند، محدوده مورد نياز [0 1] را به صورت مناسب براي هر دو ورودي و خروجي‌ها پوشش مي‌دهند.



شكل ۵-۲) سه مجموعه فازي *SMALL*، *MEDIUM* و *BIG* (به ترتيب از چپ به راست)

به طور كلي زماني كه NC_ℓ بزرگ باشد بدين معناست كه در ابتداي جستجو هستيم و الگوريتم هنوز همگرا نشده است كه در نتيجه آن مكان موجودات تغييرات زيادي دارند. در اين مواقع مقادير پارامترها بزرگ در نظر گرفته مي‌شود. برعكس زماني كه اين مقدار كوچك باشد بدين معناست كه الگوريتم همگرا شده و پارامترها مي‌بايست كوچك‌تر انتخاب شوند. مقدار ورودي NC_f نيز هرگاه بزرگ باشد بدين معناست كه الگوريتم در حال پيشرفت و صعود به قله مي‌باشد و در نتيجه نياز به افزايش مقادير پارامترها مي‌باشد. بر اساس اين رويكرد كلي مي‌توان ۹ قانون براي تنظيم مقادير پارامترها پيشنهاد داد كه در شكل ۵-۳ ذكر شده‌اند.

شكل ۵-۳) مجموعه قوانين فازي جهت تنظيم مقادير پارامترها

FUZZY RULES *FCDE*
RULE 1: IF NC_ℓ **IS** *SMALL* **AND** NC_f **IS** *SMALL* **THEN** *F* **IS** *SMALL* **AND** *Cr* **IS** *SMALL*
RULE 2: IF NC_ℓ **IS** *SMALL* **AND** NC_f **IS** *MEDIUM* **THEN** *F* **IS** *MEDIUM* **AND** *Cr* **IS** *MEDIUM*
RULE 3: IF NC_ℓ **IS** *SMALL* **AND** NC_f **IS** *BIG* **THEN** *F* **IS** *BIG* **AND** *Cr* **IS** *BIG*
RULE 4: IF NC_ℓ **IS** *MEDIUM* **AND** NC_f **IS** *SMALL* **THEN** *F* **IS** *MEDIUM* **AND** *Cr* **IS** *MEDIUM*
RULE 5: IF NC_ℓ **IS** *MEDIUM* **AND** NC_f **IS** *MEDIUM* **THEN** *F* **IS** *MEDIUM* **AND** *Cr* **IS** *MEDIUM*
RULE 6: IF NC_ℓ **IS** *MEDIUM* **AND** NC_f **IS** *BIG* **THEN** *F* **IS** *BIG* **AND** *Cr* **IS** *BIG*
RULE 7: IF NC_ℓ **IS** *BIG* **AND** NC_f **IS** *SMALL* **THEN** *F* **IS** *BIG* **AND** *Cr* **IS** *BIG*
RULE 8: IF NC_ℓ **IS** *BIG* **AND** NC_f **IS** *MEDIUM* **THEN** *F* **IS** *BIG* **AND** *Cr* **IS** *BIG*
RULE 9: IF NC_ℓ **IS** *BIG* **AND** NC_f **IS** *BIG* **THEN** *F* **IS** *BIG* **AND** *Cr* **IS** *BIG*

در هنگام انجام عمل استنتاج، از عملگرهاي MIN و MAX به ترتيب در عمليات‌هاي AND و تجمع‌سازي[1] استفاده مي‌كنيم. همچنين براي ديفازي‌سازي[2] از روش ناحيه مركزي كه در آن مركز ثقل مجموعه‌هاي خروجي محاسبه و به عنوان خروجي غير فازي شده تحويل داده مي‌شود.

در هر بار بروزرساني در داخل هر سلولي كه موجودي نياز به تكامل تفاضلي داشته باشد، مقادير NC_f و NC_ℓ محاسبه و به سيستم فازي داده مي‌شود و مقادير پارامترها به عنوان خروجي گرفته مي‌شود. از اين مقادير در الگوريتم تكامل تفاضلي استفاده مي‌شود. شبه كد الگوريتم تكامل تفاضلي فازي سلولي در الگوريتم ۵-۲ نشان داده شده است.

[1] Accumulation
[2] Defuzzification

الگوریتم ۵-۲) شبه کد الگوریتم تکامل تفاضلي فازي سلولي

Algorithm Fuzzy CellularDE (FCDE)

Initialize a cellular automaton with $(N_p)^d$ equal-sized cells

Initialize population of size M randomly in the cellular automaton

do
  for all $cell_i$ in CA **do**
    Update $cMem_i$ according to eq. (۶-۳)

    Calculate $localBest_i$ according to eq. (۷-۳)

    if there was a change in the past N_{LS} iterations **then**
      for all $individual_m$ in $cell_i$ **do**
        Set $individual_m$ to a random position in a hyper-sphere with radius LS_r centered at $cMem_i$ if that random position has better fitness.
      end-for
    else
      Calculate NC_f^i according to eq. (۵-۵)

      Calculate NC_ℓ^i according to eq. (۶-۵)

      Calculate the values of the parameters F and Cr
        by using the Rule Based Fuzzy Inference System

      Evolve active population in $cell_i$ by the modified version of DE
        with the new calculated F and Cr
    end-if

    if it is the first iteration after the local search **then**
      $T = T_i / 2$
    else
      $T = T_i$
    end-if

    while population of $cell_i > T_{init}$
      Re-initialize the worst active individual to a random cell
    end-while

    if a change is detected in the environment **then**
      Re-evaluate $cMem_i$
      for all $individual_m$ in $cell_i$ **do**
        Re-evaluate $individual_m$
      end-for
    end-if
  end-for
until a termination condition is met

۵-۳ بررسي ميزان تاثير مقادير پارامترهاي F و Cr در محيط‌هاي پوياي مختلف

در اين آزمايش ميزان تاثير مقدار پارامترهاي F و Cr بر روي كارايي الگوريتم تكامل تفاضلي سلولي را در محيط‌هايي با دوره تغيير متفاوت و تعداد قله‌هاي مختلف مورد بررسي قرار خواهيم داد. هدف از اين آزمايش آنست كه نشان دهيم كه نياز به تطبيقي كردن مقادير اين پارامترها وجود دارد. بدين منظور الگوريتم تكامل تفاضلي سلولي را به ازاي مقادير مختلف براي پارامترهاي F و Cr بر روي محيط‌هاي پوياي مختلف مورد اجرا قرار داده‌ايم. از اين طريق متوجه خواهيم شد كه اولا در يک محيط پوياي مشخص، مقدار اين پارامترها تا چه حدي در كارايي الگوريتم مي‌تواند تاثيرگذار باشد و دوما بررسي خواهيم كرد كه آيا مقدار بهينه براي اين پارامترها در محيط‌هاي مختلف يكسان است يا خير. اين آزمايش ميزان نياز به تطبيقي شدن اين پارامترها را نشان خواهد داد.

مقادير پارامترهاي الگوريتم از مجموعه $\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$ در نظر گرفته شده‌اند و براي آزمايش نيز محيط‌هايي با دوره تغيير ۱۰۰۰ و ۵۰۰۰ به ازاي تعداد قله‌هاي مختلف ۱، ۱۰، ۳۰ و ۱۰۰ انتخاب شده‌اند. در تمامي آزمايشات، براي ساير پارامترهاي محيط از مقادير پيش‌فرض سناريوي ۲ كه در جدول ۳-۱ ذكر شده است استفاده است و همچنين براي ساير پارامترهاي الگوريتم نيز از مقادير درج شده در جدول ۵-۱ استفاده نموده‌ايم. تمامي نتايج نيز به صورت ميانگين خطاي آفلاين حاصل از ۱۰۰ بار اجرا گزارش شده است. در تمامي حالات، الگوريتم براي ۲۰۰۰۰۰ بار محاسبه شايستگي اجرا شده است.

نتايج آزمايش بر روي پارامتر F براي دوره‌هاي تغيير ۱۰۰۰ و ۵۰۰۰ به ترتيب در جدول ۵-۲ و جدول ۵-۳ درج شده است. همچنين در هر محيط، به ازاي مقادير مختلف پارامتر مورد آزمايش، كمترين و بيشترين خطاي آفلاين بدست آمده به همراه بهترين مقدار بدست آمده براي آن گزارش شده است. همچنين براي نشان دادن روند كلي تغيير كارايي الگوريتم به ازاي مقادير مختلف پارامتر F، نمودار خطاي آفلاين الگوريتم به ازاي مقادير مختلف پارامتر در محيط‌هايي با تعداد قله‌هاي ۱۰،۳۰ و ۱۰۰ با دوره تغيير ۱۰۰۰ و ۵۰۰۰ به ترتيب در نمودارهاي شكل ۵-۴ و شكل ۵-۵ نشان داده شده است.

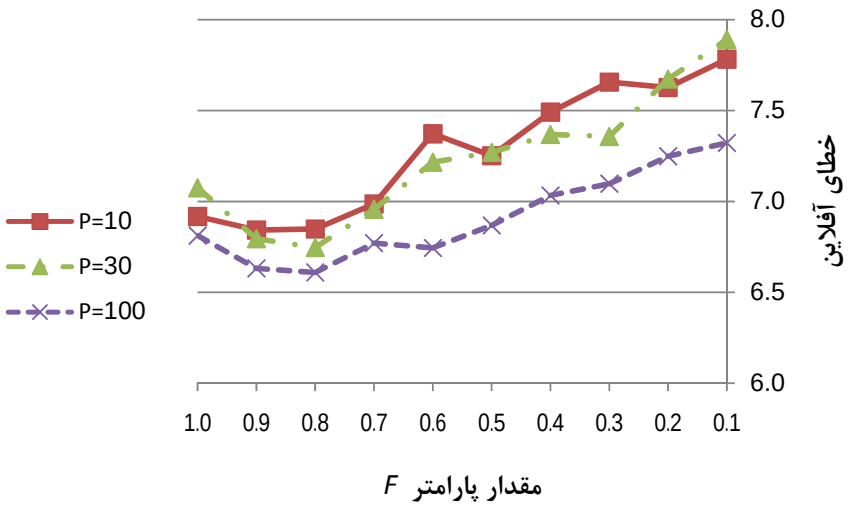
جدول ۵-۱) مقادير پيش‌فرض براي پارامترهاي الگوريتم‌هاي تطبيقي

پارامتر	مقدار
اندازه جمعيت (M)	۱۰۰
تعداد تقسيم‌بندي ابعاد (N_p)	۶
نوع همسايگي	مور
شعاع همسايگي (S_N)	۱
ظرفيت اوليه سلول (T_{init})	۱۰
تعداد دفعات تكرار جستجوي محلي براي الگوريتم تكامل تفاضلي سلولي (N_{LS})	۳
شعاع جستجوي محلي(R_{LS})	۱
ضرايب پاداش (a_N , a_L)	۰.۱
ضرايب تنبيه (b_N , b_L)	۰.۱
مقدار پيش‌فرض پارامتر F	۰.۵
مقدار پيش‌فرض پارامتر Cr	۰.۵

تمامي آزمايشات انجام شده بر روي پارامتر F به همان صورت و در همان محيط‌هاي ذكر شده بر روي پارامتر Cr نيز انجام گرفته است. نتايج آزمايش بر روي مقادير مختلف براي آن پارامتر در محيط‌هايي با دوره تغيير ۱۰۰۰ و ۵۰۰۰ به ترتيب در جدول ۵-۴ و جدول ۵-۵ درج شده است. همچنين نمودارهاي اين تغييرات نيز به ترتيب در شكل ۵-۶ و شكل ۵-۷ براي محيط‌هايي با دوره‌هاي تغيير ۱۰۰۰ و ۵۰۰۰ ترسيم شده است.

جدول ۵-۲) خطاي آفلاين به ازاي مقادير مختلف پارامتر F براي محيط‌هايي با تعداد فله‌هاي مختلف و دوره تغيير ۱۰۰۰

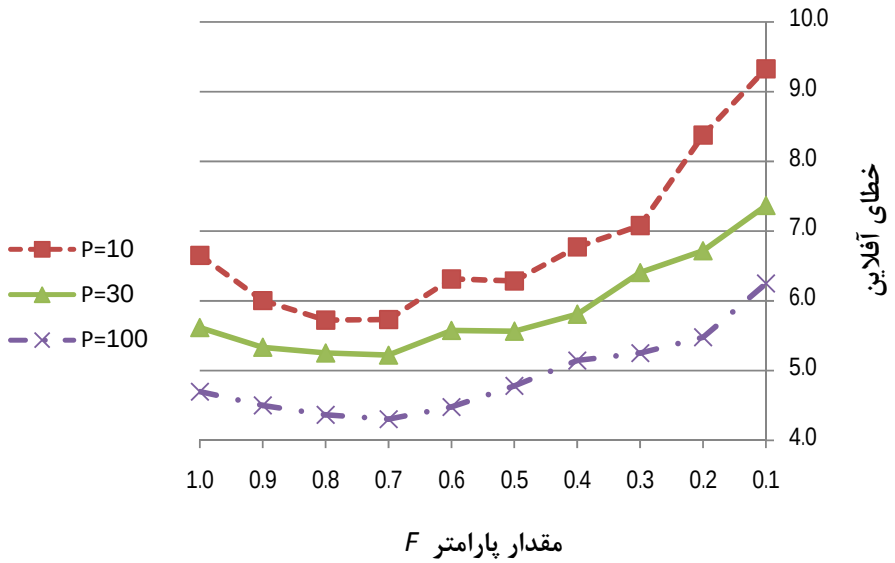
	تعداد فله‌هاي محيط			
F	۱	۱۰	۳۰	۱۰۰
۰.۱	18.88	7.78	7.89	7.32
۰.۲	18.98	7.63	7.67	7.25
۰.۳	19.02	7.66	7.36	7.10
۰.۴	17.67	7.49	7.37	7.03
۰.۵	17.88	7.25	7.27	6.87
۰.۶	17.66	7.37	7.22	6.74
۰.۷	16.65	6.99	6.96	6.77
۰.۸	16.18	6.85	**۶.۷۵**	**۶.۶۱**
۰.۹	**۱۶.۱۲**	**۶.۸۴**	6.80	6.63
۱.۰	16.33	6.92	7.08	6.81
کمترين	16.12	6.84	6.75	6.61
بيشترين	19.02	7.78	7.89	7.32
مقدار بهينه	0.9	0.9	0.8	0.8



شكل ۵-٤) نمودار خطاي آفلاين به ازاي مقادير مختلف پارامتر F در محيط‌هايي با تعداد فله‌هاي متفاوت براي دوره تغيير ۱۰۰۰

جدول ۵-۳) خطاي آفلاين به ازاي مقادير مختلف پارامتر F براي محيط‌هايي با تعداد فله‌هاي مختلف و دوره تغيير ۵۰۰۰

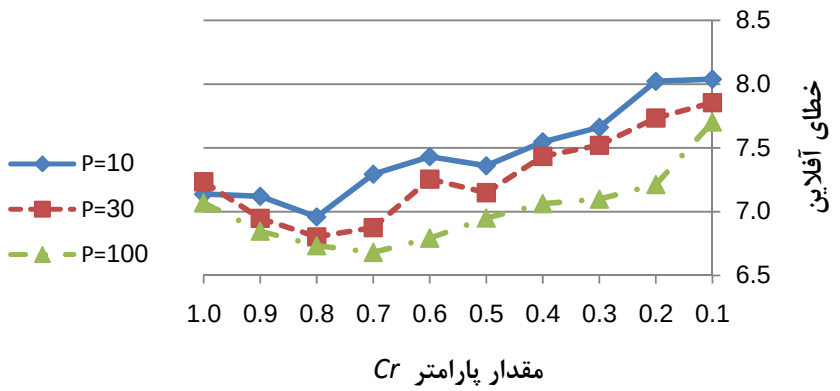
	تعداد فله‌هاي محيط			
F	۱	۱۰	۳۰	۱۰۰
۰.۱	18.55	9.33	7.37	6.25
۰.۲	16.65	8.38	6.72	5.48
۰.۳	14.48	7.08	6.41	5.25
۰.۴	12.33	6.77	5.81	5.14
۰.۵	12.65	6.29	5.56	4.78
۰.۶	10.65	6.31	5.57	4.48
۰.۷	10.88	5.73	**۵.۲۲**	**۴.۳۰**
۰.۸	**۹.۶۵**	**۵.۷۲**	5.25	4.37
۰.۹	10.06	6.00	5.33	4.50
۱.۰	18.59	6.65	5.62	4.70
کمترین	9.65	5.72	5.22	4.30
بیشترین	18.59	9.33	7.37	6.25
مقدار بهینه	0.8	0.8	0.7	0.7



شکل ۵-۵) نمودار خطاي آفلاين به ازاي مقادير مختلف پارامتر F در محيط‌هايي با تعداد فله‌هاي متفاوت براي دوره تغيير ۵۰۰۰

جدول ٥-٤) خطاي آفلاين به ازاي مقادير مختلف پارامتر Cr براي محيطهايي با تعداد فلههاي مختلف و دوره تغيير ١٠٠٠

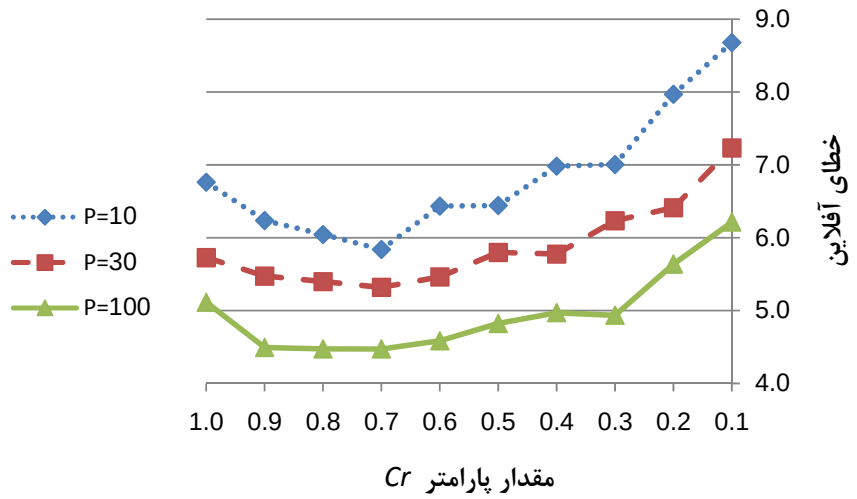
	تعداد فلههاي محيط			
Cr	١	١٠	٣٠	١٠٠
٠.١	20.34	8.04	7.86	7.70
٠.٢	17.44	8.02	7.74	7.21
٠.٣	19.15	7.66	7.52	7.10
٠.۴	18.85	7.55	7.43	7.06
٠.۵	18.22	7.36	7.15	6.95
٠.۶	15.50	7.43	7.26	6.79
٠.٧	16.15	7.29	6.88	۶.۶۸
٠.٨	17.16	۶.۹۶	۶.٨٠	6.73
٠.٩	١۴.٣٩	7.12	6.95	6.85
١.٠	14.43	7.14	7.24	7.07
کمترين	14.39	6.96	6.80	6.68
بيشترين	20.34	8.04	7.86	7.70
مقدار بهينه	0.9	0.8	0.8	0.7



شكل ٥-٦) نمودار خطاي آفلاين به ازاي مقادير مختلف پارامتر Cr در محيطهايي با تعداد فلههاي متفاوت و دوره تغيير ١٠٠٠

جدول ۵-۵) خطاي آفلاين به ازاي مقادير مختلف پارامتر Cr براي محيطهايي با تعداد قلهاي مختلف و دوره تغيير ۵۰۰۰

	تعداد قلههاي محيط			
Cr	۱	۱۰	۳۰	۱۰۰
۰.۱	16.21	8.68	7.23	6.21
۰.۲	15.54	7.97	6.41	5.64
۰.۳	12.72	7.01	6.23	4.94
۰.۴	12.76	6.98	5.78	4.97
۰.۵	12.72	6.44	5.80	4.82
۰.۶	11.17	6.43	5.46	4.59
۰.۷	10.11	**۵.۸۴**	**۵.۳۲**	**۴.۴۷**
۰.۸	10.35	6.04	5.40	4.47
۰.۹	**۹.۵۰**	6.24	5.47	4.49
۱.۰	9.51	6.76	5.73	5.12
کمترين	9.50	5.84	5.32	4.47
بيشترين	16.21	8.68	7.23	6.21
مقدار بهينه	0.90	0.70	0.70	0.70



شكل ۵-۷) نمودار خطاي آفلاين به ازاي مقادير مختلف پارامتر Cr در محيطهايي با تعداد قلههاي متفاوت دوره تغيير ۵۰۰۰

همانطور كه مشاهده ميشود، مقدار پارامترها در هر تمامي محيطها بر روي كارايي الگوريتم تاثيرگذار ميباشد. اين تاثير در محيطهايي با دوره تغيير كم و تعداد قلههاي كم بيشتر قابل مشاهده است. همچنين در محيطهاي مختلف، مقدار بهينه براي اين پارامترها با يكديگر متفاوت است. در

محيط‌هاي با دوره تغيير كمتر مقدار بهينه پارامترها بالاتر است زيرا در اين محيط‌ها با توجه به فرصت كمي كه براي جستجو بين هر دو تغيير وجود دارد، نياز به همگرايي سريع‌تري مي‌باشد. براي همگرايي سريع‌تر مقدار اين پارامترها مي‌بايست بزرگتر انتخاب شوند. علاوه بر آن با افزايش تعداد قله‌ها در محيط، مشاهده مي‌شود كه اين مقدار بهينه كاهش مي‌يابد. علت آن نيز اينست كه با افزايش تعداد قله‌ها، نياز بيشتري به جستجوي عمومي مي‌باشد و در نتيجه مي‌بايست با كم كردن مقادير پارامترها از سرعت همگرايي تا حدي كاست تا الگوريتم قادر به جستجوي عمومي بيشتري باشد.

در نتيجه همان‌طور كه شرح داده شد، مقدار پارامترهاي مورد آزمايش، در كارايي الگوريتم موثر است و همچنين مقدار بهينه آن در محيط‌هاي مختلف با يكديگر متفاوت است. پس نياز به تطبيقي كردن اين پارامترها جهت تطبيق با شرايط مختلف محيطي وجود دارد.

۴-۵ ارزيابي كارايي الگوريتم‌هاي تطبيقي

در اين فصل دو الگوريتم مختلف جهت تنظيم تطبيقي پارامترهاي الگوريتم تكامل تفاضلي سلولي كه شامل پارامترهاي F و Cr مي‌باشد، شرح داده شد. الگوريتم اول را كه بر مبناي اتوماتاي سلول يادگير و با استفاده از الگوريتم‌هاي يادگير تقويتي بنا شده است را الگوريتم تكامل تفاضلي يادگير سلولي (CLDE) ناميده‌ايم و الگوريتم ديگر كه از سيستم استنتاج فازي جهت تنظيم مناسب پارامترها استفاده مي‌كند را الگوريتم تكامل تفاضلي فازي سلولي (FCDE) ناميده‌ايم.

در اين آزمايش سعي مي‌كنيم تا كارايي الگوريتم‌هاي تطبيقي پيشنهاد شده در اين فصل را در محيط‌هاي پوياي مختلف مورد بررسي قرار دهيم. الگوريتم تكامل تفاضلي يادگير سلولي را در محيط‌هايي با دوره تغيير ۵۰۰، ۱۰۰۰، ۲۵۰۰ و ۵۰۰۰ و به ازاي تعداد قله‌هاي مختلف مورد آزمايش قرار داديم. اما الگوريتم تكامل تفاضلي فازي سلولي را به جهت زمان‌بر بودن بيش از حد، تنها در محيط‌هايي با دوره تغيير ۱۰۰۰ و ۵۰۰۰ و تنها به ازاي تعداد قله‌هاي ۱، ۱۰ و ۴۰ آزمايش كرده‌ايم. در هر محيط، الگوريتم را در سه حالت تنظيم تطبيقي پارامتر *F*، *تنظيم تطبيقي پارامتر Cr* و تنظيم تطبيقي هر دو پارامتر مورد آزمايش قرار گرفته است. در دو حالت اول، مقادير پارامترهايي كه تطبيقي نيستند را ثابت و برابر مقدار پيش‌فرض در نظر مي‌گيريم.

در تمامي آزمايشات، براي ساير پارامترهاي محيط از مقادير پيش‌فرض سناريوي دو كه در جدول ۳-۱ ذكر شده است استفاده است و همچنين براي ساير پارامترهاي الگوريتم نيز از مقادير درج شده در جدول ۵-۱ استفاده نموده‌ايم. تمامي نتايج نيز به صورت ميانگين خطاي آفلاين حاصل از

۱۰۰ بار اجرا گزارش شده است. در تمامي محيطها، الگوريتم تكامل تفاضلي يادگير سلولي را براي ۵۰۰۰۰۰ مرتبه محاسبه شايستگي و الگوريتم تكامل تفاضلي فازي سلولي را براي ۲۰۰۰۰۰ بار محاسبه شايستگي اجرا نموده‌ايم. تمامي نتايج نيز به صورت ميانگين خطاي آفلاين ۱۰۰ مرتبه اجرا گزارش شده‌اند.

جدول ۵-٦) خطاي آفلاين الگوريتم تكامل تفاضلي يادگير سلولي به ازاي تعداد فله‌اي مختلف براي دوره تغيير ۵۰۰

	پارامتر تطبيقي شده		
p	CLDE *(CR)*	CLDE *(F)*	CLDE *(F,CR)*
۱	12.50	12.55	13.09
۵	7.08	6.99	7.06
۱۰	6.24	6.14	6.15
۲۰	6.75	6.69	6.74
۳۰	6.93	6.96	6.89
۴۰	7.08	7.03	7.03
۵۰	7.08	7.09	7.07
۱۰۰	6.95	6.99	6.90
۲۰۰	6.56	6.59	6.63

جدول ۵-۷) خطاي آفلاين الگوريتم تكامل تفاضلي يادگير سلولي به ازاي تعداد قله‌اي مختلف براي دوره تغيير ۱۰۰۰

پارامتر تطبيقي شده			
CLDE *(F,CR)*	CLDE *(F)*	CLDE *(CR)*	*p*
13.02	12.90	12.83	۱
5.50	5.61	5.54	۵
5.02	4.91	5.02	۱۰
5.53	5.50	5.64	۲۰
5.82	5.77	5.88	۳۰
5.86	5.86	5.92	۴۰
5.99	5.90	5.94	۵۰
5.86	5.85	5.82	۱۰۰
5.43	5.43	5.49	۲۰۰

جدول ۵-۸) خطاي آفلاين الگوريتم تكامل تفاضلي يادگير سلولي به ازاي تعداد قله‌اي مختلف براي دوره تغيير ۲۵۰۰

پارامتر تطبيقي شده			
CLDE *(F,CR)*	CLDE *(F)*	CLDE *(CR)*	*p*
8.22	7.34	7.88	۱
4.29	4.08	4.13	۵
4.05	4.08	4.13	۱۰
4.63	4.53	4.63	۲۰
4.82	4.68	4.67	۳۰
4.83	4.79	4.76	۴۰
4.84	4.75	4.78	۵۰
4.70	4.67	4.65	۱۰۰
4.35	4.29	4.28	۲۰۰

جدول ۵-۹) خطاي آفلاين الگوريتم تكامل تفاضلي يادگير سلولي به ازاي تعداد فلهاي مختلف براي دوره تغيير ۵۰۰۰

پارامتر تطبيقي شده			
CLDE *(F,CR)*	CLDE *(F)*	CLDE *(CR)*	*p*
5.81	5.71	5.72	۱
3.80	3.59	3.91	۵
3.75	3.67	3.74	۱۰
4.13	4.01	4.15	۲۰
4.09	4.01	3.99	۳۰
4.21	4.05	4.08	۴۰
4.21	4.15	4.05	۵۰
3.97	3.85	3.90	۱۰۰
3.60	3.61	3.57	۲۰۰

جدول ۵-۱۰) خطاي آفلاين الگوريتم تكامل تفاضلي فازي سلولي به ازاي تعداد فلهاي مختلف براي دوره تغيير ۱۰۰۰

پارامتر تطبيقي شده			
FCDE *(F,CR)*	FCDE *(F)*	FCDE *(CR)*	*p*
17.40	16.65	17.25	۱
7.38	6.01	7.29	۱۰
7.56	7.30	7.48	۴۰

جدول ۵-۱۱) خطاي آفلاين الگوريتم تكامل تفاضلي فازي سلولي به ازاي تعداد فلهاي مختلف براي دوره تغيير ۵۰۰۰

پارامتر تطبيقي شده			
FCDE *(F,CR)*	FCDE *(F)*	FCDE *(CR)*	*p*
10.67	10.32	10.54	۱
6.60	6.23	6.42	۱۰
5.60	5.17	5.21	۴۰

همان‌طور كه مشاهده مي‌شود هر دو الگوريتم با توجه به حداقل و حداكثر كارايي ممكن، كارايي قابل قبولي در تمامي حالات از خود نشان داده‌اند. البته اين كارايي در حد بهترين حالت مقدار

ثابت نمي‌باشد كه اين امر با توجه به كندي فرآيند يادگيري و همچنين نسبي بودن درستي قوانين فازي امري دور از انتظار نبود.

همچنين در اكثر حالات نتايج تطبيقي سازي هر دو پارامتر به صورت همزمان نسبت به حالت يک پارامتري، از نظر آماري برابر و يا تا حدود بسيار كمي ضعيف‌تر است. اين امر نيز با توجه تطبيقي سازي هر دو پارامتر به صورت همزمان دور از ذهن نبود. در الگوريتم يادگيري بر اساس نتايج حاصل از تكامل، سيگنال‌هاي يادگيري ساخته مي‌شوند و اين سيگنال‌ها براي هر دو پارامتر يكسان است. در صورتي كه نتيجه تكامل حاصل و برآيند انتخاب هر دو پارامتر مي‌باشد و اين وابستگي در آموزش در نظر گرفته مي‌شود. به بيان ساده‌تر مشخص نيست كه اين بهبود يا عدم بهبود در تكامل، حاصل انتخاب مناسب كدام پارامتر است. در الگوريتم فازي نيز بر اساس قوانين طرح شده هر دو پارامتر به صورت همزمان كاهش و يا افزايش داده مي‌شوند كه اين امر موجب مي‌شود شاهد كارايي پايين‌تري نسبت به بهترين حالت مقدار ثابت باشيم.

در هر دو الگوريتم، به طور كلي مي توان گفت هر چند در يک محيط مشخص قادر به افزايش كارايي نسبت به بهترين حالت مقدار ثابت نشده‌ايم ولي توانسته‌ايم با تطبيقي‌سازي پارامترها، اولا نياز به انجام آزمايشات زياد براي يافتن مقدار ثابت بهينه و تنظيم مقدار هر پارامتر را از بين ببريم و به نوعي تنظيم اين پارامترها را از حالت دستي و توسط كاربر به حالت خودكار تبديل كنيم. دوما اينكه اين تطبيقي‌سازي، موجب بالا رفتن قابليت تطبيق الگوريتم با شرايط محيطي مختلف مي‌شود. بدين معنا كه الگوريتم در حالت تطبيقي به صورت ميانگين كارايي بالاتري از حالتي كه تنها يک مقدار ثابت براي تمامي حالات در نظر گرفته شود، از خود نشان مي‌دهد.

۵-۵ نتيجه‌گيري

الگوريتم تكامل تفاضلي به كار گرفته شده در مدل‌هاي سلولي پيشنهاد شده در فصول قبل داراي دو F و Cr بود. براي بدست آوردن حداكثر كارايي ممكن، مي‌بايست مقادير مناسب براي اين پارامترها را از طريق سعي و خطا بدست آورد كه اين مقادير براي محيط‌هاي مختلف نيز حتي متفاوت بدست مي‌آيند. در اين فصل دو روش جهت تطبيقي‌سازي اين پارامترها پيشنهاد شد. در الگوريتم اول، يک الگوريتم بر مبناي مدل اتوماتاي يادگير سلولي پيشنهاد شد كه در آن از يادگيري تقويتي جهت يادگيري مقدار مناسب هر پارامتر استفاده مي‌شود و به همين جهت اين الگوريتم را الگوريتم تكامل تفاضلي يادگير سلولي ناميديم. الگوريتم دوم كه الگوريتم تكامل تفاضلي فازي

سلولي مي‌ناميم، با استفاده از قوانين فازي و سيستم استنتاج فازي سعي در تنظيم مقدار مناسب مقدار پارامترها را داشت.

پس از شرح روند دو الگوريتم، با انجام يک سري آزمايشات در ابتدا نشان داديم که نياز به تطبيقي سازي مقدار اين پارامترها مي‌باشد و سپس با انجام آزمايشاتي کارايي اين دو الگوريتم را در محيط‌هاي پوياي مختلف مورد بررسي قرار داديم. در آزمايشات مشاهده شد که روش‌هاي پيشنهاد شده از کارايي نسبتا قابل قبولي برخوردارند و قادر هستند تا قابليت تطبيق الگوريتم با شرايط محيط‌هاي مختلف را بالا ببرند.

فصل ششم

نتيجه‌گيري و كارهاي آينده

۶. نتيجه‌گيري و كارهاي آينده

۶-۱ نتيجه‌گيري

محيط‌هايي كه در آن‌ها تابع شايستگي به زمان وابسته باشد را محيط‌هاي پويا گوييم كه يكي از روش‌هاي متداول بهينه‌سازي و جستجو در آن‌ها، روش‌هاي تكاملي مي‌باشد. در اين پايان‌نامه علاوه بر بررسي و معرفي محيط‌هاي پويا و كارهاي گذشته انجام شده در آن‌ها، الگوريتم‌هاي تكاملي مختلفي مبتني بر مدل‌هاي اتوماتاي سلولي براي بهينه‌سازي در اين گونه محيط‌ها پيشنهاد و مورد آزمايش و بررسي قرار گرفت. دو چالش عمده در طراحي الگوريتم‌هاي تكاملي براي محيط‌هاي پويا، حفظ كافي تنوع جمعيت و استفاده مناسب از تجربيات جستجوهاي پيشين است كه در روش‌هاي پيشنهاد شده تا كنون به طور كامل حل نشده‌اند. در روش‌هاي پيشنهاد شده سعي شد تا با اين دو چالش تا حد امكان مقابله شود.

در فصل دوم به معرفي و شرح الگوريتم پيشنهادي اوليه پرداختيم كه بر مبناي مدل اتوماتاي سلولي و با استفاده از ايده تقسيم‌بندي فضاي جستجو طراحي شده است و مبنا و پايه روش‌هاي پيشنهاد شده در فصول بعدي نيز مي‌باشد. در الگوريتم پيشنهادي با تقسيم‌بندي فضا و انجام جستجو و تكامل به صورت جداگانه در هر قسمت توسط الگوريتم تكامل تفاضلي، ضمن حفظ تنوع، امكان جستجوي محلي سريع و مناسب آن منطقه نيز هست. براي بهبود جستجوي منطقه‌اي نيز، از يک مدل تكاملي پيشنهادي جهت ساخت بردار كانديد استفاده شد كه در عين جستجوي محلي مناسب در منطقه، داراي سرعت همگرايي بسيار بالايي نيز بود. مزيت ديگر اين مدل تكاملي، حذف يكي از پارامترهاي الگوريتم تكامل تفاضلي بود. اين مدل تكاملي كارايي بسيار بالاتري در مدل پيشنهاد شده، نسبت به مدل‌هاي تكاملي رايج براي الگوريتم تكامل تفاضلي از خود نشان داد. اين الگوريتم را الگوريتم تكامل تفاضلي سلولي (CellularDE) ناميديم.

پس از آن براي بهبود روند بازيابي قله‌هاي گمشده پس از تغيير، يک روش جستجوي محلي مناسب پيشنهاد شد كه كارايي الگوريتم را بويژه در دوره‌هاي تغيير بالا افزايش مي‌داد. تنها ضعف اين روش مصرف بالاي تعداد محاسبات شايستگي بود كه آن را بيشتر براي محيط‌هاي با دوره تغيير زياد مناسب مي‌كند. پس از شرح و توضيح كامل روند روش‌هاي پيشنهاد شده، آزمايشاتي جهت تنظيم مقادير پارامترهاي الگوريتم و همچنين بررسي تاثير اين پارامترها و ارتباط بين آن‌ها انجام شد. در آزمايش نهايي نيز كارايي روش‌هاي پيشنهادي را با چندين روش ديگر كه بر روي تابع محک قله‌هاي

متحرک کارايي بالايي از خود نشان داده‌اند مورد ارزيابي قرار داديم. در آزمايشات مشاهده شد که در اکثر محيط‌هاي مورد آزمايش قرار گرفته، روش‌هاي پيشنهادي از کارايي بالاتري نسبت به ساير روش‌ها برخوردارند. در دوره‌هاي تغيير پايين، روش جستجوي بهبود يافته و در دوره‌هاي تغيير بالا، الگوريتم تکامل تفاضلي سلولي از برتري نسبي برخوردارند.

در فصل سوم با انجام تغييراتي در الگوريتم تکامل تفاضلي سلولي که در فصل دوم پيشنهاد شده بود، نسخه ديگري از آن به نام الگوريتم تکامل تفاضلي سلولي کوه‌نورد[1] پيشنهاد شد که با دستکاري متناسب ظرفيت سلول‌ها و تغيير به موقع روش جستجو در برخي سلول‌ها سعي در استفاده بهينه از منابع و بهبود جستجو داشت. يکي از مشکلات الگوريتم تکامل تفاضلي سلولي اين بود که در مناطق داراي قله، موجودات همگرا شده و ديگر نياز زيادي به جستجوي عمومي نبود و بهتر بود تا منابع را بر روي استخراج همان بهينه يافت شده متمرکز کنيم. مشکل ديگر تجمع و جستجوي بي‌مورد در سلول‌هاي همسايه يک سلول داراي قله بود و مي‌بايست با محدود کردن ظرفيت سلول‌هاي اطراف از اين اتفاق جلوگيري مي‌کرديم.

بدين منظور در ابتدا سلول‌هايي که احتمال وجود قله در آن‌ها بالا است را تشخيص مي‌دهيم، سپس در ضمن کاهش ظرفيت آن، از الگوريتم تپه‌نوردي براي جستجو در آن سلول استفاده مي‌کنيم. زيرا جمعيت در اين سلول‌ها از تنوع کمي برخوردار است و همچنين نياز زيادي به جستجوي عمومي نيست، به همين جهت در اين گونه از سلول‌ها يک جستجوي محلي بازده بالاتري دارد. همچنين در سلول‌هاي اطراف قله که دامنه آن محسوب مي‌شوند، ظرفيت سلول کاهش به شدت داده مي‌شود تا از تجمع بي‌مورد موجودات در آن‌ها جلوگيري شود. در انتها نيز، در آزمايشاتي کارايي اين نسخه پيشنهادي با نسخه‌هاي قبلي و ساير الگوريتم‌ها مورد ارزيابي و مقايسه قرار داده شد. در آزمايشات نشان داده شد که در تمامي محيط‌هاي مورد آزمايش نسخه کوه‌نورد کارايي بالاتري نسبت به نسخه الگوريتم تکامل تفاضلي سلولي از خود نشان مي‌دهد.

در فصل چهارم، سعي کرديم با استفاده از دو روش يادگيري تقويتي و همچنين استنتاج فازي پارامترهاي الگوريتم تکامل تفاضلي سلولي را به صورت خودکار کنترل و تنظيم نماييم. اين امر براي محيط‌هاي پويا که مدام در حال تغيير هستند و همچنين حالت‌ها و سناريوهاي مختلفي دارند مي‌تواند بسيار کمک‌کننده باشد. مزيت ديگر اين روش‌ها عدم نياز به تنظيم پارامترهاي الگوريتم تکاملي از طريق سعي و آزمايش براي هر محيط به صورت جداگانه است.

[1] Alpinist CellularDE

در اين فصل دو الگوريتم با نام‌هاي الگوريتم تكامل تفاصلي يادگير سلولي[1] و الگوريتم تكامل تفاصلي فازي سلولي[2] پيشنهاد شد. الگوريتم اول بر مبناي مدل اتوماتاي يادگير سلولي طراحي شده است و با استفاده از يادگيري تقويتي سعي در يافتن مقدار مناسب براي پارامترها دارد و الگوريتم ديگر از سيستم اسستنتاج فازي جهت تنظيم مقدار پارامترها استفاده مي‌كند. در انتهاي فصل نيز آزمايشاتي جهت بررسي كارايي هر دو الگوريتم تطبيقي پيشنهاد شده، انجام شد. نتايج آزمايشات نشان داد كه روش‌هاي پيشنهاد شده از كارايي نسبتا قابل قبولي در محيط‌هاي مختلف برخوردارند و قابليت تطبيق‌پذيري الگوريتم را نسبت به محيط‌هاي پوياي مختلف افزايش مي‌دهند.

۲-٦ پيشنهاداتي براي كارهاي آينده

با توجه به پژوهش‌ها و كارهايي كه در اين پايان‌نامه بر روي محيط‌هاي پويا و روش‌هاي بهينه‌سازي تكاملي در آن‌ها انجام شد مي‌توان موضوعات پژوهشي زير را به عنوان موضوعات كارهاي آينده پيشنهاد داد:

- **طراحي توابع محک مناسب:** همان‌طور كه در فصل اول بيان شد، تاكنون توابع محک بسياري جهت ارزيابي كارايي روش‌هاي پيشنهادي در محيط‌هاي پويا پيشنهاد شده است كه از بين آن‌ها تابع محک قله‌هاي متحرک، كه در اين پايان‌نامه نيز از آن استفاده شد، رايج‌ترين و جامع‌ترين تابع محک در محيط‌هاي پيوسته مي‌باشد. اما دو موضوع مهم در اين رابطه وجود دارد.

  اول اينكه در اين تابع محک بين ابعاد وابستگي وجود ندارد و پهنه جستجو به صورت محلي بسيار ساده است كه اين موضوع موجب سوء استفاده از اين نقطه ضعف مي‌شود. به طور مثال در تعدادي از روش‌هاي پيشنهاد شده، از الگوريتم‌هاي جستجوي محلي خاصي استفاده مي‌شود كه تنها در محيط‌هاي ساده با عدم وابستگي بين ابعاد كارايي بسيار بالايي دارند و يا در اكثر روش‌هاي موجود از اطلاعات موجود در مورد ساختار محيط از جمله چند قله‌اي بودن محيط در طراحي الگوريتم استفاده مي‌شود. در نتيجه نياز به طراحي محيطي مي‌باشد كه پهنه جستجوي سخت‌تر و پيچيده‌تري داشته باشد.

[1] Cellular Learning DE (CLDE)
[2] Fuzzy CellularDE (FCDE)

دوم اينكه هنوز مشخص نيست كه اين توابع محک تا چه حدي به محيط‌هاي پوياي واقعي شباهت دارند و آيا اينكه اين توابع معيار مناسبي جهت نشان دادن كارايي در محيط‌هاي واقعي نيز هستند يا خير. به همين دليل نياز به انجام پژوهش‌هاي بيشتري بر روي محيط‌هاي پوياي واقعي پيوسته مي‌باشد تا بتوان محيطي كه بازتاب‌دهنده مشخصات محيط‌هاي پويا در دنياي واقعي است، طراحي نمود.

- **بهينه‌سازي در محيط‌هايي با ابعاد بسيار بالا:** يكي از محدوديت‌هاي روش‌هاي پيشنهادي كه همگي بر اساس تقسيم‌بندي فضا بنا شده بودند، نمايي بودن تعداد همسايه‌ها نسبت به تعداد ابعاد محيط جستجو مي‌باشد كه موجب زمان‌بر شدن محاسبه همسايگي در ابعاد بسيار بالا مي‌شود. براي رفع اين مشكل مي‌توان از روش‌هاي تكاملي همكارانه[1] استفاده نمود. بدين صورت در محيط‌هايي با ابعاد بالا، براي هر چند بعد يک اتوماتاي سلولي جداگانه در نظر گرفته شود و بين اين اتوماتاهاي سلولي روابطي براي انجام جستجو به صورت همكارانه تعريف شود.
- **تطبيقي‌سازي اندازه جمعيت:** يكي از مشكلات روش‌هاي پيشنهاد شده در اين پايان‌نامه عدم تطبيق‌پذيري اندازه جمعيت با مشخصات محيط بود. اين امر در محيط‌هاي غير پويا زياد حياتي نيست اما در براي الگوريتم‌هايي كه براي محيط‌هاي پويا طراحي مي‌شوند اين موضوع از اهميت بيشتري برخوردار است. زيرا از اين الگوريتم‌ها انتظار مي‌رود كه با تغييرات محيطي نيز خود را وفق دهند. كاهش كارايي روش‌هاي پيشنهادي در محيط‌هاي مختلف نيز تا حدودي از اين موضوع نشات مي‌گرفت زيرا اندازه جمعيت تنها براي حالت سناريوي دوم مناسب تشخيص داده شده بود.

براي انجام اين كار به طور مثال مي‌توان در ابتدا جمعيت اوليه نسبتا زيادي توليد كرد تا محيط را به طور كامل جستجو كنند، سپس از طريق زير نظر گرفتن ورود و خروج موجودات اضافي در سلول‌ها متوجه ازدياد موجودات در فضاي جستجو شد و آن‌ها را به جاي بيرون انداختن، نابود ساخت و يا به طور موقت از كار انداخت. و يا مي‌توان از طريق كم بودن تعداد موجودات بر روي يک قله متوجه كمبود جمعيت شد و به موجودات آن سلول اضافه كرد.

[1] Cooperative Evolutionary

- **تطبيقي‌سازي اندازه ظرفيت سلول:** يكي ديگر از كارهايي كه مي‌تواند علاوه بر افزايش كارايي روش‌هاي پيشنهادي موجب افزايش تطبيق‌پذيري آن‌ها با محيط‌هاي پوياي مختلف شود، تطبيقي سازي و تنظيم مناسب اندازه ظرفيت هر سلول به صورت جداگانه مي‌باشد. به طور مثال در برخي سلول‌ها كه نياز به موجودات بيشتري جهت جستجو و يا استخراج مي‌باشد ظرفيت را افزايش دهيم و در مناطقي كه جستجو قبلا انجام پذيرفته و يا ديگر نيازي به جستجو نيست اين مقدار را افزايش دهيم.

فصل هفتم

مراجع

۷. مراجع

[1] Kalamanje, K., "**Micro-genetic algorithms for stationary and non-stationary function optimization**," *SPIE: Intelligent Control and Adaptive Systems*, vol. 1196, 1989, pp. 289-296.

[2] Ramsey, C.L. and Grefenstette, J.J., "**Case-Based Initialization of Genetic Algorithms**," *Morgan Kaufmann Publishers Inc.*, 1993, pp. 84-91.

[3] Cobb, H.G. and Grefenstette, J.J., "**Genetic algorithms for tracking changing environments**," Proceedings of the 5th International Conference of Genetic Algorithms, 1993, pp. 523-530.

[4] Grefenstette, J.J., "**Genetic algorithms for changing environments**," *Parallel Problem Solving from Nature*, vol. 2, 1992, pp. 137-144.

[5] Morrison, R. and De Jong, K., "**Triggered hypermutation revisited**," Proceedings of the IEEE Congress on Evolutionary Computation (CEC), 2002, pp. 1025-1032.

[6] Vavak, F., Fogarty, T. and Jukes, K., "**A genetic algorithm with variable range of local search for tracking changing environments**," *Parallel Problem Solving from Nature*, 1996, pp. 376-385.

[7] Branke, J., "**Memory enhanced evolutionary algorithms for changing optimization problems**," Proceedings of the IEEE Congress on Evolutionary Computation (CEC), 1999, pp. 1875–1882.

[8] Eggermont, J. and Lenaerts, T., "**Dynamic optimization using evolutionary algorithms with a case-based memory**," Proceedings of the Benelux Conference on Artificial Intelligence, 2002, pp. 107-114.

[9] Louis, S.J. and Johnson, J., "**Solving similar problems using genetic algorithms and case-based memory**," Proceedings of the 7th International Conference of Genetic Algorithms, 1997, pp. 101-127.

[10] Ng, K.P. and Wong, K.C., "**A new diploid scheme and dominance change mechanism for non-stationary function optimization**," *Morgan Kaufmann Publishers Inc.*, 1995, pp. 159-166.

[11] Simões, A. and Costa, E., "**Variable-size memory evolutionary algorithm to deal with dynamic environments**," *Applications of Evolutionary Computing*, 2007, pp. 617-626.

[12] Simões, A. and Costa, E., "**Improving memory's usage in evolutionary algorithms for changing environments**," Proceedings of the IEEE Congress on Evolutionary Computation (CEC), 2007, pp. 276-283.

[13] Yang, S., "**Non-stationary problem optimization using the primal-dual genetic algorithm**," Proceedings of the Congress on Evolutionary Computation (CEC), 2004, pp. 2246-2253.

[14] Yang, S., "**Memory-Based Immigrants for Genetic Algorithms in Dynamic Environments**," Proceedings of the Genetic and Evolutionary Computation Conference (GECCO), 2005, pp. 1115-1122.

[15] Yang, S., "**Explicit memory schemes for evolutionary algorithms in dynamic environments**," *Evolutionary Computation in Dynamic and Uncertain Environments*, vol. 51, 2007, pp. 3-28.

[16] Yang, S., "**Genetic algorithms with memory and elitism-based immigrants in dynamic environments**," *Evolutionary Computation*, vol. 16, no. 3, 2008, pp. 385-416.

[17] Markon, S., Arnold, D.V., Back, T., Beielstein, T. and Beyer, H.G., "**Thresholding-a selection operator for noisy ES**," Proceedings of the Congress on Evolutionary Computation (CEC), 2001, pp. 465-472.

[18] Hughes, E., "**Evolutionary Multi-objective Ranking with Uncertainty and Noise**," Lecture Notes in Computer Science 1993, Springer Berlin / Heidelberg, 2001, pp. 329-343.

[19] Oppacher, F. and Wineberg, M., "**The shifting balance genetic algorithm: Improving the GA in a dynamic environment**," Proceedings of the Genetic and Evolutionary Computation Conference (GECCO), 1999, pp. 504-510.

[20] Ursem, R.K., "**Multinational GAs: Multimodal Optimization Techniques in Dynamic Environments**," Proceedings of the Genetic and Evolutionary Computation Conference (GECCO), 2000, pp. 19-26.

[21] Branke, J. and Schmeck, H., "**Designing evolutionary algorithms for dynamic optimization problems**," *Advances in evolutionary computing: theory and applications*, 2002, pp. 239-262.

[22] Blackwell, T. and Branke, J., "**Multi-swarm optimization in dynamic environments**," *Applications of Evolutionary Computing*, 2004, pp. 489-500.

[23] Blackwell, T.M. and Branke, J., "**Multi-swarm, exlusion and anti-convergence in dynamic environments**," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 4, 2004, pp. 459-472.

[24] Mendes, R. and Mohais, A., "**DynDE: a differential evolution for dynamic optimization problems**," Proceedings of the IEEE Congress on Evolutionary Computation (CEC), 2005, pp. 2808-2815.

[25] Blackwell, T. and Branke, J., "**Multiswarms, exclusion, and anti-convergence in dynamic environments**," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 4, 2006, pp. 459-472.

[26] Blackwell, T., Branke, J. and Li, X., "**Particle swarms for dynamic optimization problems**," *Swarm Intelligence*, 2008, pp. 193-217.

[27] du Plessis, M. and Engelbrecht, A., "**Improved differential evolution for dynamic optimization problems**," Proceedings of the IEEE Congress on Evolutionary Computation (CEC), 2008, pp. 229-234.

[28] Li, C. and Yang, S., "**Fast multi-swarm optimization for dynamic optimization problems**," Proceedings of the Fourth International Conference on Natural Computation, 2008, pp. 624-628.

[29] Wang, H., Wang, N. and Wang, D., "**Multi-swarm optimization algorithm for dynamic optimization problems using forking**," Proceedings of the Control and Decision Conference (CCDC), 2008, pp. 2415-2419.

[30] Kamosi, M., Hashemi, A. and Meybodi, M., "**A New Particle Swarm Optimization Algorithm for Dynamic Environments**," *Swarm, Evolutionary, and Memetic Computing*, 2010, pp. 129-138.

[31] Kamosi, M., Hashemi, A.B. and Meybodi, M., "**A hibernating multi-swarm optimization algorithm for dynamic environments**," Proceedings of the Second World Congress on Nature and Biologically Inspired Computing (NaBIC), 2010, pp. 363-369.

[32] Liu, L. and Ranji Ranjithan, S., "**An adaptive optimization technique for dynamic environments**," *Engineering Applications of Artificial Intelligence*, vol. 23, no. 5, 2010, pp. 772-779.

[33] Liu, L., Yang, S. and Wang, D., "**Particle swarm optimization with composite particles in dynamic environments**," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 40, no. 6, 2010, pp. 1634-1648.

[34] Moser, I. and Chiong, R., "**Dynamic function optimisation with hybridised extremal dynamics**," *Memetic Computing*, vol. 2, no. 2, 2010, pp. 137-148.

[35] Novoa-Hernández, P., Pelta, D. and Corona, C., "**Improvement Strategies for Multi-swarm PSO in Dynamic Environments**," *Nature Inspired Cooperative Strategies for Optimization (NICSO)*, 2010, pp. 371-383.

[36] González, J.R., Masegosa, A.D. and García, I.J., "**A cooperative strategy for solving dynamic optimization problems**," *Memetic Computing*, vol. 3, no. 1, 2011, pp. 3-14.

[37] Parvin, H., Minaei, B. and Ghatei, S., "**A New Particle Swarm Optimization for Dynamic Environments**," *Computational Intelligence in Security for Information Systems*, 2011, pp. 293-300.

[38] Rezazadeh, I., Meybodi, M. and Naebi, A., "**Adaptive Particle Swarm Optimization Algorithm for Dynamic Environments**," *Advances in Swarm Intelligence*, 2011, pp. 120-129.

[39] Garai, G. and Chaudhuri, B., "**A distributed hierarchical genetic algorithm for efficient optimization and pattern matching**," *Pattern recognition*, vol. 40, no. 1, 2007, pp. 212-228.

[40] Garai, G. and Chaudhuri, B., "**A cascaded genetic algorithm for efficient optimization and pattern matching**," *Image and Vision Computing*, vol. 20, no. 4, 2002, pp. 265-277.

[41] Hashemi, A. and Meybodi, M., "**A multi-role cellular PSO for dynamic environments**," Proceedings of the 14th International CSI Computer Conference, 2009, pp. 412-417.

[42] Hashemi, A. and Meybodi, M., "**Cellular Pso: A Pso for Dynamic Environments**," Proceedings of the Advances in Computation and Intelligence, Springer, 2009, pp. 422-433.

[43] Rastegar, R., Meybodi, M.R. and Hariri, A., "**A new fine-grained evolutionary algorithm based on cellular learning automata**," *International Journal of Hybrid Intelligent Systems*, vol. 3, no. 2, 2006, pp. 83-98.

[44] Jafarpour, B. and Meybodi, M., "**Recombinative CLA-EC**," Proceedings of the 19th IEEE International Conference on Tools with Artificial Intelligence, 2007, pp. 415-422.

[45] Tomassini, M., "**The parallel genetic cellular automata: Application to global function optimization**," Proceedings of the International Conference of Artificial Neural Network and Genetic Algorithms, 1993, pp. 385–391.

[46] Tomassini, M., "**Parallel and distributed evolutionary algorithms: A review**," *Evolutionary Algorithms in Engineering and Computer Science*, 1999, pp. 113–133.

[47] Vafashoar, R., Meybodi, M. and Momeni Azandaryani, A., "**CLA-DE: a hybrid model based on cellular learning automata for numerical optimization**," *Applied Intelligence*, 2011, pp. 1-14.

[48] Zanganeh, S., Meybodi, M.R. and Sedehi, M.H., "**Continuous CLA-EC**," Proceedings of the Fourth International Conference on Genetic and Evolutionary Computing (ICGEC), 2010, pp. 186-189.

[49] Branke, J., ***Evolutionary optimization in dynamic environments.***, Kluwer Academic Publishers, Norwell, 2001.

[50] Branke, J., "**Memory enhanced evolutionary algorithms for changing optimization problems**," Proceedings of the IEEE Congress on Evolutionary Computation (CEC), 2002.

[51] Jin, Y. and Branke, J., "**Evolutionary optimization in uncertain environments-a survey**," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 3, 2005, pp. 303-317.

[52] Cruz, C., González, J.R. and Pelta, D.A., "**Optimization in dynamic environments: a survey on problems, methods and measures**," *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, 2011, pp. 1-22.

[53] Krishnakumar, K., "**Microgenetic algorithms for stationary and nonstationary function optimization**," *SPIE: Intelligent Control and Adaptive Systems*, vol. 1196, 1990, pp. 289-296.

[54] Smith, R.E., "**Diploid genetic algorithms for search in time varying environments**," Proceedings of the Annual Southeast Regional Conference of the ACM, 1987, pp. 175-179.

[55] Munetomi, M., Takai, Y. and Sato, Y., "**StGA: An application of a genetic algorithm to stochastic learning automata**," *Systems and computers in Japan*, vol. 27, no. 10, 1996, pp. 68-78.

[56] Babu, G.P., "**Clustering in non-stationary environments using a clan-based evolutionary approach**," *Biological cybernetics*, vol. 73, no. 4, 1995, pp. 367-374.

[57] Hart, E. and Ross, P., "**The evolution and analysis of a potential antibody library for use in job-shop scheduling**," *New Ideas in Optimisation*, 1999, pp. 185–202.

[58] Li, X. and Dam, K.H., "**Comparing particle swarms for tracking extrema in dynamic environments**," Proceedings of the IEEE Congress on Evolutionary Computation (CEC), 2003, pp. 1772-1779.

[59] Janson, S. and Middendorf, M., "**A hierarchical particle swarm optimizer for dynamic optimization problems**," *Applications of Evolutionary Computing*, 2004, pp. 513-524.

[60] Janson, S. and Middendorf, M., "**A hierarchical particle swarm optimizer and its adaptive variant**," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 35, no. 6, 2005, pp. 1272-1282.

[61] Hu, X. and Eberhart, R.C., "**Adaptive particle swarm optimization: detection and response to dynamic systems**," Proceedings of the IEEE Congress on Evolutionary Computation (CEC), 2002, pp. 1666-1670.

[62] Blackwell, T.M., "**Particle swarms and population diversity**," *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, vol. 9, no. 11, 2005, pp. 793-802.

[63] Lung, R.I. and Dumitrescu, D., "**A collaborative model for tracking optima in dynamic environments**," Proceedings of the IEEE Congress on Evolutionary Computation (CEC), 2007, pp. 564-567.

[64] Blackwell, T. and Branke, J., "**Multi-swarm Optimization in Dynamic Environments, Applications of Evolutionary Computing**," Lecture Notes in Computer Science 3005, Springer, 2004, pp. 489-500.

[65] Blackwell, T., "**Particle swarm optimization in dynamic environments**," *Evolutionary Computation in Dynamic and Uncertain Environments*, 2007, pp. 29-49.

[66] Blum, C. and Merkle, D., "**Particle swarms for dynamic optimization problems**," *Swarm Intelligence*, vol. 2, 2008, pp. 123-138.

[67] Brest, J., Zamuda, A., Boskovic, B., Maucec, M. and Zumer, V., "**Dynamic optimization using self-adaptive differential evolution**," Proceedings of the IEEE Congress on Evolutionary Computation (CEC), 2009, pp. 415-422.

[68] Li, C. and Yang, S., "**A clustering particle swarm optimizer for dynamic optimization**," Proceedings of the IEEE Congress on Evolutionary Computation, 2009, pp. 439-446.

[69] Lung, R.I. and Dumitrescu, D., "**Evolutionary swarm cooperative optimization in dynamic environments**," *Natural Computing*, vol. 9, no. 1, 2010, pp. 83-94.

[70] Kanlikilicer, A., Keles, A. and Uyar, A., "**Experimental analysis of binary differential evolution in dynamic environments**," Proceedings of the IEEE Congress on Evolutionary Computation (CEC), 2007, pp. 2509-2514.

[71] Price, K., Storn, R. and Lampinen, J., ***Differential evolution: a practical approach to global optimization***, Springer Verlag, 2005.

[72] Price, K.V., "**Differential evolution: a fast and simple numerical optimizer**," Proceedings of the Biennial Conference of the North American (NAFIPS), 1996, pp. 524-527.

[73] Storn, R. and Price, K., "**Minimizing the real functions of the ICEC'96 contest by differential evolution**," Proceedings of the IEEE Congress on Evolutionary Computation (CEC), 1996, pp. 842-844.

[74] Storn, R. and Price, K., "**Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces**," *Journal of global optimization*, vol. 11, no. 4, 1997, pp. 341-359.

[75] Narendra, K.S. and Thathachar, M., "**Learning Automata: Survey**," *IEEE Transactions on Systems, Man and Cybernetics*, no. 4, 1974, pp. 323-334.

[76] Narendra, K.S. and Thathachar, M.A.L., ***Learning automata: an introduction***, Prentice-Hall, Inc. Upper Saddle River, NJ, USA, 1989.

[77] Wolfram, S., "**Universality and complexity in cellular automata**," *Physica D: Nonlinear Phenomena*, vol. 10, no. 1-2, 1984, pp. 1-35.

[78] Harmon, M.E. and Harmon, S.S., ***Reinforcement learning: a tutorial***, Wright State University, 1996.

[79] Sutton, R.S. and Barto, A.G., ***Reinforcement learning***, Cambridge,MA: MIT Press, 1998.

[80] Tsetlin, M., ***Automation theory and modeling of biological systems***, Academic Press, New York, NY, 1994.

[81] Varshavskii, V. and Vorontsova, I., "**On the behavior of stochastic automata with variable structure**," *Automation and Remote Control*, vol. 24, no. 3, 1963.

[82] Fu, K., "**Stochastic automata as models of learning systems**," *Adaptive, Learning and Pattern Recognition Systems: Theory and Applications*, 1968, pp. 393-430.

[83] Fu, K. and Li, T., "**Formulation of learning automata and automata games+**," *Information Sciences*, vol. 1, no. 3, 1969, pp. 237-256.

[84] Fu, T., "**On stochastic automata and languages**," *Information Sciences*, vol. 1, no. 4, 1969, pp. 403-419.

[85] Mars, P., Chen, J., Nambiar, R. and Fidler, J., ***Learning Algorithms: Theory and Applications in Signal Processing***, CRC Press, Inc. Boca Raton, FL, USA, 1996.

[86] Oommen, B. and Ma, D., "**Fast automata solutions to the equal partitioning problem**," *The Institute of Electrical and Electronics Engineers*, 1986, pp. 358.

[87] Oommen, B.J. and Croix, D.S., "**String taxonomy using learning automata**," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 27, no. 2, 1997, pp. 354-365.

[88] Oommen, B.J., "**Graph partitioning using learning automata**," *IEEE Transactions on Computers*, vol. 45, no. 2, 1996, pp. 195-208.

[89] Tsoularis, A., Kambhampati, C. and Warwick, K., "**Path planning of robots in noisy workspaces using learning automata**," Proceedings of the IEEE International Symposium on Intelligent Control, 1993, pp. 560-564.

[90] Farajzadeh, N. and Meybodi, M.R., "**Learning automata-based clustering algorithm for sensor networks**," Proceedings of the 13th International CSI Computer Conference, 2007, pp. 780-787.

[91] Gholipour, M. and Meybodi, M., "**LA-mobicast: A learning automata based mobicast routing protocol for wireless sensor networks**," *Sensor Letters*, vol. 6, no. 2, 2008, pp. 305-311.

[92] Lakshmivarahan, S. and Thathachar, M., "**Absolutely expedient learning algorithms for stochastic automata**," *IEEE Transactions on Systems, Man, and Cybernetics*, no. 3, 1973, pp. 281-286.

[93] Lakshmivarahan, S. and Thathachar, M., "**Optimal non-linear reinforcement schemes for stochastic automata**," *Information Sciences*, vol. 4, no. 2, 1972, pp. 121-128.

[94] Lakshmivarahan, S. and Thathachar, M., "**Absolute expediency of Q-and S-model learning algorithms**," *IEEE Transactions on Systems, Man and Cybernetics*, no. 3, 1976, pp. 222-226.

[95] Viswanathan, R. and Narendra, K., ***Expedient and Optimal Variable-Structure Stochastic Automata***, Dunham Lab., Yale University, 1970.

[96] Mason, L., "**An optimal learning algorithm for S-model environments**," *IEEE Transactions on Automatic Control*, vol. 18, no. 5, 1973, pp. 493-496.

[97] Meybodi, M., Beigy, H. and Taherkhani, M., "**Cellular learning automata and its applications**," *Sharif Journal of Science and Technology*, vol. 19, no. 25, 2003, pp. 54–77.

[98] Beigy, H. and Meybodi, M., "**A mathematical framework for cellular learning automata**," *Advances on Complex Systems*, vol. 7, no. 4, 2004, pp. 295-319.

فصل هشتم

واژه‌نامه

۸. واژه‌نامه

۸-۱ واژه‌نامه انگليسي به فارسي

English phrase	معادل فارسي
Accumulation	تجمع‌سازي
Actions Probability Vector	بردار احتمال اعمال
Adaptive Particle Swarm Optimization	الگوريتم بهينه‌ساز گروه ذرات تطبيقي
Allele	آلل
Alpanist CellularDE	الگوريتم تكامل تفاضلي سلولي كوه‌نورد
Asynchronous	غير همگام
Atomic Models	مدل‌هاي اتمي
Averaging over Space	ميانگين‌گيري زماني
Averaging over Time	ميانگين‌گيري مكاني
Benchmark	مسائل محک
Blind Search	كوركورانه
Boltzmann	تابع بولتزمن
Bounded	محدود شده
Breeder Genetic Algorithm	الگوريتم ژنتيک زاينده
Candidate Vector	بردار كانديد
Cauchy	توزيع كوشي
Cellular Automata	اتوماتاي سلولي
Cellular Learning Automata	اتوماتاي يادگير سلولي
Cellular Learning DE (CLDE)	الگوريتم تكامل تفاضلي يادگير سلولي
Cellular PSO	الگوريتم بهينه‌سازگروه ذرات سلولي

CellularDE	الگوريتم تكامل تفاضلي سلولي
Chromosome	كروموزوم
Class	كلاس
Colony	كلوني
Confidence Interval	فاصله اطمينان
Configuration	پيكربندي
Cooperative Evolutionary	تكاملي همكارانه
Core population	جمعيت هسته
Crowding	ازدحام
Crowding Based Differential Evolution	الگوريتم تكامل تفاضلي مبتني بر ازدحام
Defuzzification	ديفازي‌سازي
Deterministic Automata	اتوماتاي قطعي
Differential Evolution	الگوريتم تكامل تفاصلي
Diffusion	پراكنده
Dominant	غالب
Dynamic Benchmark	تابع محک پويا
Dynamic Bit Matching	تطبيق بيت‌هاي پويا
Dynamic Environments	محيط‌هاي پويا
DynDE	الگوريتم تكامل تفاضلي پويا
Elite	برگزيده
Evolution	تكامل
Evolution Strategy	استراتژي تكاملي
Evolutionary Algorithms	الگوريتم‌هاي تكاملي
Evolutionary Programming Algorithm	الگورتيم برنامه‌ريزي تكاملي

Explicit Averaging	ميانگين‌گيري آشكار
Exploitation	استخراج
Exploration	اكتشاف
Fast PSO	الگوريتم بهينه ساز گروه ذرات سريع
Feedback	بازخورد
Fine Grained PSO (FGPSO)	الگوريتم بهينه‌ساز گروه ذرات شبكه‌اي
Fitness Evaluation	ارزيابي شايستگي
Fixed Structure	ساختار ثابت
Foot Cell	سلول قله
Footside Cell	سلول دامنه
Fu	فو
Fuzzy Cellular Automata	اتوماتاي سلولي فازي
Fuzzy CellularDE (FCDE)	الگوريتم تكامل تفاضلي فازي سلولي
Fuzzy Induction System	سيستم استنتاج فازي
Game Theory	تئوري بازي
General	عمومي
General Behavior	رفتار عمومي
Genetic Algorithm	الگوريتم ژنتيک
Genetic Programming	برنامه‌نويسي ژنتيک
Genotype	ژنوتيپ
Graph Partitioning	جداسازي گراف
Hamming Distance	فاصله همينگ
Heuristic Search	جستجوي مكاشفه‌اي
Hibernating Multi-Swarm Optimization Algorithm (HmSO)	الگوريتم بهينه‌سازي چند گروهي خواب زمستاني

Hill Climbing	الگوريتم تپه‌نوردي
Hillside Cell	سلول کوه‌پايه
Hill-valley	دره
Homogeneous	يکنواخت
Hyper Mutation	ابر جهش
Hyper Sphere	ابرکره
Implicit	تلويحي
Implicit Averaging	ميانگين‌گيري ضمني
Iteration	گام
Iterative Arrays	آرايه‌هاي تکرار شونده
JAVA	جاوا
Job Shop Scheduling	زمانبندي توليد کارگاهي
John Holland	جان هولند
John Koza	جان کوزا
Landscape	پهنه جستجو
LISP	ليسپ
Log-normal Self Adaptation	خود تطبيقي لگاريتمي
Meta Model	ابر مدل
Method	روند
Monte Carlo	مونت کارلو
Moving Parabola	سهمي متحرک
Moving Peaks Function	تابع محک قله‌هاي متحرک
Multi Population	چند جمعيتي
Multi Quantum Based Swarm Optimization (mQSO)	الگوريتم چند گروهي کوانتومي ذرات

Multinational GA	الگوريتم ژنتيک چند مليتي
Multiplicative Self Adaptation	خود تطبيقي افزاينده
Multiploidy	چند رشته‌اي
Najim	نجيم
Narendra	نارندرا
Nearest Neighbor Analysis	آناليز همسايه هاي مجاور
NetBeans	نت بينز
Non-stationary	غير ايستا
Non-Uniform Volume Distribution (NUVD)	توزيع حجم غير يکنواخت
Offline Performance	کارايي آفلاين
ONEMAX	مساله ايستاي بيشترين يک
Online Performance	کارايي آنلاين
Optima	بهينه
Orthogonal	متعامد
Oscillating Peaks	قله‌هاي نوسان‌گر
Outer Totalistic	کلي خارجي
Package	بسته
Parameter Estimation	تخمين پارامترها
Partial Solution	جواب‌هاي جزئي
Particle Swarm Optimizer (PSO)	الگوريتم بهينه‌ساز گروه ذرات
Path Planning	برنامه‌ريزي مسير
Pattern Recognition	تشخيص الگو
Poznyak	پوزنياک
Punishment Parameter	پارامتر تنبيه

Random Immigrants	مهاجرت تصادفي
Recessive	مغلوب
Recombination	بازترکیبي
Reinforcement Learning	یادگیري تقویتي
Restart/Reinitialization	شروع از نو
Reward Parameter	پارامتر پاداش
Robust	مقاوم
Robustness	پایداري و مقاومت
Roulete Wheel	چرخ رولت
Rule Based Fuzzy Induction System	سیستم استنتاج فازي مبتني بر قوانین
Scheme	مدل تکاملي
Self Organizing Scouts (SOS)	دیده‌بانان خودسازمانده
Sharing	اشتراک‌گذاري
Shifting Balance GA	الگوریتم ژنتیک با تعادل انتقالي
Standard Error	خطاي استاندارد
Stationary	ایستا
Stochastic Genetic Algorithm (StGA)	الگوریتم ژنتیک تصادفي
Stochastic Learning Automata	اتوماتاي یادگیر تصادفي
Stochastic Universal Sampling (SUS)	نمونه‌گیري فراگیر احتمالي
Student’s T-test	تست آماري دانش‌آموز
Sub-optimal	زیر بهینه
Sub-population	زیرجمعیت
Suger-beet-presses	پرس چغندر قند
Supervised Learning	یادگیري با نظارت

Synchronous	همگام
Thermodynamical Genetic Algorithm (TDGA)	الگوريتم ژنتيک ترموديناميکي
Topological Dynamics	توپولوژيکي
Totalistic	کلي
Trade off	تعادل
Triallelic	سه مقداري
Triggered Hypermutation	ابر جهش محرکي
Tsetlin	ستلين
Tsypkin	سيپکين
Ulam	اولام
Uncertain Environments	محيطهاي غير قطعي
Uniform Volume Distribution (UVD)	توزيع حجم يکنواخت
Unknown Cell	سلول ناشناخته
Variable Local Search (VLS)	جستجوي محلي متغير
Variable Structure	ساختار متغير
Varshavski	وارشاوسکي
Viswanatan	ويسواناتان
Von Neumann	ون‌نيومن
Vorontsova	وورونتسوا
Wind Shear	چاله هوايي
Wolfram Numbers	اعداد ولفرام
Wrap Around	دور چرخشي

۲-۸ واژه‌نامه فارسي به انگليسي

عبارت فارسي	English Equivalence
ابر جهش	Hyper Mutation
ابر جهش محركي	Triggered Hypermutation
ابركره	Hyper Sphere
ابرمدل	Meta Model
اتوماتاي سلولي	Cellular Automata
اتوماتاي سلولي فازي	Fuzzy Cellular Automata
اتوماتاي قطعي	Deterministic Automata
اتوماتاي يادگير تصادفي	Stochastic Learning Automata
اتوماتاي يادگير سلولي	Cellular Learning Automata
ارزيابي شايستگي	Fitness Evaluation
ازدحام	Crowding
استخراج	Exploitation
استراتژي تكاملي	Evolution Strategy
اشتراک‌گذاري	Sharing
اكتشاف	Exploration
الگورتيم برنامه‌ريزي تكاملي	Evolutionary Programming Algorithm
الگوريتم بهينه ساز گروه ذرات سريع	Fast PSO
الگوريتم بهينه‌سازي چند گروهي خواب زمستاني	Hibernating Multi-Swarm Optimization Algorithm (HmSO)
الگوريتم تپه‌نوردي	Hill Climbing
الگوريتم تكامل تفاصلي	Differential Evolution
الگوريتم تكامل تفاضلي پويا	DynDE

الگوريتم تكامل تفاضلي سلولي	CellularDE
الگوريتم تكامل تفاضلي سلولي كوه‌نورد	Alpanist CellularDE
الگوريتم تكامل تفاضلي فازي سلولي	Fuzzy CellularDE (FCDE)
الگوريتم تكامل تفاضلي مبتني بر ازدحام	Crowding Based Differential Evolution
الگوريتم تكامل تفاضلي يادگير سلولي	Cellular Learning DE (CLDE)
الگوريتم چند گروهي كوانتومي ذرات	Multi Quantum Based Swarm Optimization (mQSO)
الگوريتم ژنتيک	Genetic Algorithm
الگوريتم ژنتيک با تعادل انتقالي	Shifting Balance GA
الگوريتم ژنتيک ترموديناميكي	Thermodynamical Genetic Algorithm (TDGA)
الگوريتم ژنتيک تصادفي	Stochastic Genetic Algorithm (StGA)
الگوريتم ژنتيک چند مليتي	Multinational GA
الگوريتم ژنتيک زاينده	Breeder Genetic Algorithm
الگوريتم بهينه‌ساز گروه ذرات	Particle Swarm Optimizer (PSO)
الگوريتم بهينه‌ساز گروه ذرات تطبيقي	Adaptive Particle Swarm Optimization
الگوريتم بهينه‌ساز گروه ذرات سلولي	Cellular PSO
الگوريتم بهينه‌ساز گروه ذرات شبكه‌اي	Fine Grained PSO (FGPSO)
الگوريتم‌هاي تكاملي	Evolutionary Algorithms
اولام	Ulam
ايستا	Stationary
آرايه‌هاي تكرار شونده	Iterative Arrays
آلل	Allele
آناليز همسايه هاي مجاور	Nearest Neighbor Analysis
بازتركيبي	Recombination
بازخورد	Feedback

Actions Probability Vector	بردار احتمال اعمال
Candidate Vector	بردار كانديد
Elite	برگزيده
Path Planning	برنامه‌ريزي مسير
Genetic Programming	برنامه‌نويسي ژنتيک
Package	بسته
Optima	بهينه
Local Optima	بهينه محلي
Reward Parameter	پارامتر پاداش
Punishment Parameter	پارامتر تنبيه
Robustness	پايداري و مقاومت
Diffusion	پراکنده
Suger-beet-presses	پرس چغندر قند
Landscape	پهنه جستجو
Poznyak	پوژنياک
Configuration	پيکربندي
Boltzmann	تابع بولتزمن
Dynamic Benchmark	تابع محک پويا
Moving Peaks Function	تابع محک قله‌هاي متحرک
Accumulation	تجمع‌سازي
Parameter Estimation	تخمين پارامترها
Student’s T-test	تست آماري دانش‌آموز
Pattern Recognition	تشخيص الگو
Dynamic Bit Matching	تطبيق بيت‌هاي پويا

تعادل	Trade off
تكامل	Evolution
تكاملي همكارانه	Cooperative Evolutionary
تلويحي	Implicit
توپولوژيكي	Topological Dynamics
توزيع حجم غير يكنواخت	Non-Uniform Volume Distribution (NUVD)
توزيع حجم يكنواخت	Uniform Volume Distribution (UVD)
توزيع كوشي	Cauchy
تئوري بازي	Game Theory
جان كوزا	John Koza
جان هولند	John Holland
جاوا	JAVA
جداسازي گراف	Graph Partitioning
جستجوي محلي متغير	Variable Local Search (VLS)
جستجوي مكاشفه‌اي	Heuristic Search
جمعيت هسته	Core population
جواب‌هاي جزئي	Partial Solution
فاصله همينگ	Hamming Distance
چند جمعيتي	Multi Population
چند رشته‌اي	Multiploidy
خطاي استاندارد	Standard Error
خود تطبيقي افزاينده	Multiplicative Self Adaptation
خود تطبيقي لگاريتمي	Log-normal Self Adaptation
دره	Hill-valley

Wrap Around	دور چرخشي
Self Organizing Scouts (SOS)	ديده‌بانان خودسازمانده
Defuzzification	ديفازي‌سازي
General Behavior	رفتار عمومي
Method	روند
Job Shop Scheduling	زمانبندي توليد كارگاهي
Sub-optimal	زير بهينه
Sub-population	زيرجمعيت
Genotype	ژنوتيپ
Fixed Structure	ساختار ثابت
Variable Structure	ساختار متغير
Tsetlin	ستلين
Footside Cell	سلول دامنه
Foot Cell	سلول قله
Hillside Cell	سلول كوه‌پايه
Unknown Cell	سلول ناشناخته
Triallelic	سه مقداري
Moving Parabola	سهمي متحرک
Tsypkin	سيپكين
Fuzzy Induction System	سيستم استنتاج فازي
Rule Based Fuzzy Induction System	سيستم استنتاج فازي مبتني بر قوانين
Restart/Reinitialization	شروع از نو
General	عمومي
Dominant	غالب

Non-stationary	غير ايستا
Asynchronous	غير همگام
Confidence Interval	فاصله اطمينان
Wind Shear	چاله هوايي
Fu	فو
Oscillating Peaks	قله‌هاي نوسان‌گر
Blind Search	كوركورانه
Offline Performance	كارايي آفلاين
Online Performance	كارايي آنلاين
Chromosome	كروموزوم
Class	كلاس
Colony	كلوني
Totalistic	كلي
Outer Totalistic	كلي خارجي
Iteration	گام
LISP	ليسپ
Orthogonal	متعامد
Bounded	محدود شده
Dynamic Environments	محيط‌هاي پويا
Uncertain Environments	محيط‌هاي غير قطعي
Scheme	مدل تكاملي
Atomic Models	مدل‌هاي اتمي
ONEMAX	مساله ايستاي بيشترين يک
Benchmark	مسائل محک

Recessive	مغلوب
Robust	مقاوم
Random Immigrants	مهاجرت تصادفي
Monte Carlo	مونت كارلو
Explicit Averaging	ميانگين‌گيري آشكار
Averaging over Space	ميانگين‌گيري زماني
Implicit Averaging	ميانگين‌گيري ضمني
Averaging over Time	ميانگين‌گيري مكاني
Narendra	نارندرا
NetBeans	نت بينز
Najim	نجيم
Stochastic Universal Sampling (SUS)	نمونه‌گيري فراگير احتمالي
Synchronous	همگام
Varshavski	وارشاوسكي
Wolfram Numbers	اعداد ولفرام
Von Neumann	ون‌نيومن
Vorontsova	وورونتسوا
Viswanatan	ويسواناتان
Supervised Learning	يادگيري با نظارت
Reinforcement Learning	يادگيري تقويتي
Homogeneous	يكنواخت

فصل نهم

پیوست‌ها

۹. پیوست‌ها

۹-۱ الگوریتم تکامل‌تفاضلي

در سال ۱۹۹۵، و با تلاش‌هاي پرایس و استورن، الگوریتم تکاملي جدیدي به نام تکامل تفاضلي (DE) معرفي شد. این دو در مقاله‌هاي بعدي خود، نشان دادند این الگوریتم توانایي خوبي در بهینه‌سازي توابع غیر خطي مشتق‌ناپذیر دارد [۷۱-۷۴]. این الگوریتم به سادگي، با جمع کردن تفاضل وزن‌دار دو بردار از جمعیت با یک بردار دیگر، جمعیت اولیه را به سوي بهترین راه‌حل سوق مي‌دهد. بعدها ابتکارات جدیدي در این الگوریتم اعمال شد، به نحوي که بهبودهایي در آن ایجاد شد. الگوریتم DE نسخه‌هاي گوناگوني دارد که با نمادگذاري DE/x/y/z مشخص مي‌شوند. این نمادگذاري در ادامه همین بخش معرفي مي‌شود. دقت شود که توضیحات زیر مربوط به چگونگي عملکرد نسخه DE/rand/1/bin است.

۹-۱-۱ مراحل الگوریتم تکامل تفاضلي

به عنوان یک الگوریتم تکاملي، اولین گام الگوریتم تکامل تفاضلي، مقداردهي اولیه مي‌باشد که طي آن، از یک توزیع تصادفي یکنواخت براي ایجاد جمعیت اولیه استفاده مي‌گردد. سپس در حلقه‌اي با یک یا چند شرط مشخص، عملیات انتخاب والدین، جهش و بازترکیبي به نحوي که در ادامه مي‌آید انجام مي‌پذیرند.

- جهش

این عملگر براي تمامي موجودات درون جمعیت اعمال مي‌گردد. مطابق فرمول (۹-۱) ، سه بردار از جمعیت به صورت تصادفي انتخاب مي‌گردند و تفاضل وزن‌داري از دو بردار اول با بردار سوم جمع شده و بردار جهش‌یافته را ایجاد مي‌کنند.

$$v_i = x_{r1} + F \times (x_{r2} - x_{r3}) \qquad (۹-۱)$$

در فرمول فوق، پارامتر F، وزن جهش مي‌باشد و تنظیم آن از اهمیت بالایي برخوردار است، چرا که مقدار آن تأثیر فوق‌العاده برجسته‌اي بر عملکرد الگوریتم در حل مسائل مختلف مي‌گذارد. همانطور که مشاهده مي‌شود، نحوه انجام جهش در اینجا با استراتژي تکامل تفاوت دارد که از توزیع-

هاي از پيش معين‌شده‌اي مانند گاوسي استفاده مي‌شود. در اينجا، الگوريتم از موجودات درون خود جمعيت براي سوق دادن ديگر موجودات به سمت نقاط بهينه استفاده مي‌كند.

- بازتركيبي

براي افزايش تنوع بردارهاي درون جمعيت، عملگر بازتركيبي به صورت فرمول (۹-۲) تعريف مي‌گردد.

$$u_i = \{u_1, u_2, ..., u_D\} | \quad (۹-۲)$$

كه در فرمول فوق:

$$u_{ji} = \begin{cases} v_{ji} & \text{if } \mathrm{U}(0,1) \le Cr \text{ or } j == r_j \\ \mathrm{x}_{ji} & \text{otherwise} \end{cases} \quad (۹-۳)$$

اين فرآيند، برداري را ايجاد مي‌كند كه بردار كانديدا ناميده مي‌شود. در فرمول فوق، پارامتر Cr احتمال وارد كردن عضوي از بردار جهش‌يافته به بردار كانديدا مي‌باشد كه مقدار آن تأثير زيادي بر عملكرد الگوريتم در مواجهه با مسائل هدف مختلف خواهد داشت. تنظيم اين پارامتر مانند پارامتر وزن تفاضل از اهميت بالايي برخوردار است.

مهم‌ترين ويژگي‌هاي الگوريتم تكامل تفاضلي، سادگي و مقاوم بودن آن است. اين روش تنها با تنظيم سه پارامتر شروع به كار مي‌كند. پارامتر Cr احتمال انجام بازتركيبي و پارامتر F وزن جهش است كه در تفاضل دو بردار ضرب مي‌شود و به بردار سوم اضافه مي‌شود. به گفته استورن و پرايس، پارامتر F معمولا بين ۰ تا ۱ تنظيم مي‌شود. اين الگوريتم، ابتدا جمعيتي با مقادير تصادفي در محدوده مقادير مسئله مربوطه ايجاد مي‌كند. سپس براي هر بردار عضو جمعيت، يک بردار جهش‌يافته به بدين صورت ساخته مي‌شود كه، سه بردار به صورت تصادفي انتخاب مي‌شوند، به شكلي كه هيچ يک يكسان نباشند. تفاضل دو بردار اول در پارامتر F ضرب مي‌شود و حاصل آن با بردار سوم جمع مي شود. هر يک از مؤلفه‌هاي اين بردار جهش‌يافته با احتمال Cr به بردار كانديدا منتقل مي‌شود و در غير اين صورت، مؤلفه معادل در بردار اصلي جايگزين مي‌شود. البته براي تضمين اينكه حداقل يک مؤلفه به بردار كانديدا منتقل مي‌شود، يک مؤلفه به صورت اتفاقي و بدون در نظر گرفتن Cr از بردار جهش‌يافته به بردار كانديدا منتقل مي‌شود. براي تمامي بردارهاي كانديدا، تابع ارزيابي فراخواني مي‌شود. به منظور انتخاب بازماندگان از روش حريصانه استفاده مي‌شود و هر بردار با بردار جهش يافته مربوطه مقايسه مي‌شود، هر كدام كه شايستگي بيشتري داشته باشد به نسل بعدي منتقل مي‌شود.

اين مراحل، تا رسيدن به تعداد نسل‌هاي مورد نظر، يا رسيدن به كمينه عمومي، ادامه مي‌يابند. شكل ۹-۱ ساختار كلي DE/rand/1/bin را نمايش مي‌دهد و شبه كد اين الگوريتم نيز در

الگوريتم ۹-۱ درج شده است.

الگوريتم ۹-۱) شبه كد الگوريتم تكامل تفاضلي

Algorithm Differential Evolution

Initialize population with uniform distribution

do

for each individual $\vec{x}_i$ in the population **do in parallel**

Select three different individuals $\vec{x}_1$,$\vec{x}_2$, and $\vec{x}_3$ from the current population randomly

Generate a trial vector $\vec{v}_i$ using eq. (۱-۹)

Create a new vector $\vec{u}_i$ using binomial crossover according to eq. (۲-۹) and eq. (۳-۹)

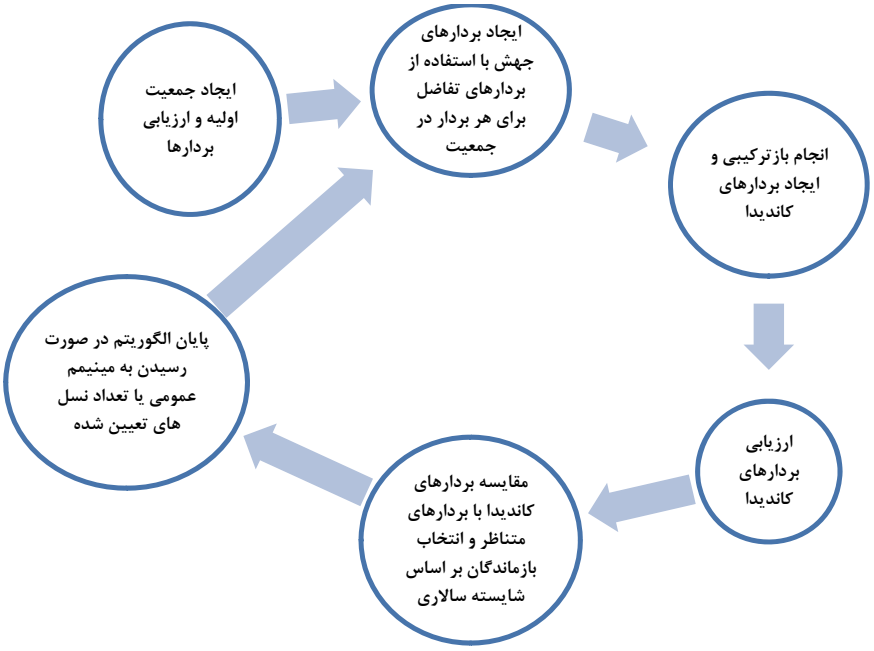
Evaluate the candidate$\vec{u}_i$

if fitness($\vec{u}_i$)>fitness($\vec{x}_i$) **then**

Replace individual $\vec{x}_i$ with $\vec{u}_i$

end-if

until a termination condition is met



شکل ۹-۱) نمودار الگوریتم تکامل تفاضلي

۹-۱-۲ نمادگذاري الگوريتم تكامل تفاضلي

الگوريتم تكامل تفاضلي نسخه‌هاي گوناگوني دارد كه بوسيله نمادسازي زير مشخص مي‌شوند. بسته به كاربرد، مي‌توان از هريک از اين نسخه‌ها براي بهينه‌سازي استفاده كرد. مرور كاملي بر توانايي‌هاي اين نسخه‌ها در بهينه‌سازي توابع مختلف در [۷۱] موجود است. نمادسازي الگوريتم تكامل تفاضلي به صورت DE/x/y/z مي‌باشد. كه در آن :

x، بردار جهش‌يافته مي‌باشد، كه مي‌تواند تصادفي يا براساس شايستگي انتخاب شود. براي اين نماد، دو گزينه *rand* و *best* پيش‌بيني شده‌اند. در صورت انتخاب گزينه *best*، همواره بهترين بردار عضو جمعيت براي انجام جهش انتخاب مي‌شود. و در صورت انتخاب *rand*، براي هر جهش يک بردار به صورت تصادفي انتخاب مي‌شود. y، تعداد بردارهاي تفاضل مورد استفاده است. z، نوع بازتركيبي مورد استفاده است.

۲-۹ اتوماتاي سلولي

اتوماتاي سلولي[1](CA) در اواخر دهه ۱۹۴۰ توسط ونيومن[2] مطرح و پس از او توسط رياضيداني بنام اولام[3] به عنوان مدلي براي بررسي رفتار سيستم‌هاي پيچيده پيشنهاد شد. اتوماتاي سلولي در حقيقت سيستم‌هاي ديناميكي گسسته‌اي هستند كه رفتارشان كاملاً بر اساس ارتباط محلي استوار است. اتوماتاي سلولي چندين بار و هر بار تحت نام مختلفي ابداع شده است. از ديدگاه رياضيات محض آنها را مي‌توان شاخه‌اي از ديناميک توپولوژيكي[4] و از ديدگاه مهندسي برق، آرايه‌هاي تكرار شونده[5] دانست [۷۵-۷۷].

در اتوماتاي سلولي، فضا به صورت يک شبكه تعريف مي‌گردد كه به هر خانه آن يک سلول گفته مي‌شود. زمان به صورت گسسته پيش مي‌رود و قوانين آن به صورت سرتاسري است كه از طريق آن در هر مرحله هر سلول، وضعيت جديد خود را با در نظر گرفتن همسايه‌هاي مجاور خود بدست مي‌آورد. اتوماتاي سلولي را مي‌توان به عنوان سيستم‌هاي محاسباتي نيز در نظر گرفت كه اطلاعات كد شده در خودشان را پردازش مي‌كنند. همچنين يک اتوماتاي سلولي به همراه واحد كنترل آن را مي‌توان به عنوان يک ماشين SIMD[6] تعبير نمود. قوانين اتوماتاي سلولي، نحوه تأثير پذيرفتن سلول از سلول‌هاي همسايه خود را مشخص مي‌كند. يک سلول را همسايه سلول ديگر گوئيم هرگاه بتواند آن را در يک مرحله و براساس قانون حاكم تحت تأثير قرار دهد. مي‌توان در بعضي شرايط براي سلول‌هاي واقع در مرزها، سلول‌هاي مرزهاي مقابل را به عنوان همسايه در نظر گرفت. در صورتيكه همسايگي بدين صورت در نظر گرفته شود، دور چرخشي[7] و در غير اينصورت محدود شده[8] گفته مي‌شود. در بدست آوردن وضعيت كنوني سلول علاوه بر وضعيت قبلي سلول‌هاي همسايه، مي‌توان وضعيت قبلي خود سلول را نيز دخالت داد. معمولاً قوانين اتوماتاي سلولي توسط كاربر طراحي مي‌شوند. تعريف رسمي اتوماتاي سلولي به صورت زير است :

تعريف - اتوماتاي سلولي d بعدي يک چندتايي $CA=(Z^d,\phi,N,F)$ است به طوريكه:

- Z^d يک شبكه از d تايي‌هاي مرتب از اعداد صحيح مي‌باشد. اين شبكه مي‌تواند يک شبكه متناهي، نيمه متناهي يا نامتناهي باشد.

---

[1] Cellular Automata
[2] Von Neumann
[3] Ulam
[4] Topological Dynamics
[5] Iterative Arrays
[6] Single Instruction-Multiple Data
[7] Wrap Around
[8] Bounded

- $\phi = \{1,...,m\}$ يک مجموعه متناهي از حالت‌ها مي‌باشد.

- $N = \{\bar{x}_1,...,\bar{x}_{\bar{m}}\}$ ، $\bar{x}_i \in Z^d$ ، يک زير مجموعه متناهي از Z^d مي‌باشد که بردار همسايگي خوانده مي‌شود. بردار همسايگي، موقعيت نسبي همسايگان را براي هر سلول u در شبکه سلولي به صورت زير مشخص مي‌کند:

$$N(u) = \{u + \bar{x}_i \mid i = 1,...,\bar{m}\} \qquad (٤-٩)$$

تابع $N(u)$ دو شرط زير را ارضا مي‌کند:

$$\begin{aligned} &\forall u \in Z^d \Rightarrow u \in N(u) \\ &\forall u,v \in Z^d \Rightarrow u \in N(v) \wedge v \in N(u) \end{aligned} \qquad (٥-٩)$$

براي نمونه در شکل ٩-٢ به ترتيب همسايگي ون‌نيومن، $N = \{(0,0),(1,0),(0,1),(0,-1),(-1,0)\}$ و مور، $N = \{(0,0),(1,0),(0,1),(-1,0),(0,-1),(-1,-1),(1,-1),(-1,1)\}$ ، در فضاي دو بعدي مي‌باشند نشان داده شده‌اند.

- $F : \underline{\varphi}^{\bar{m}} \to \underline{\varphi}$ قانون محلي اتوماتاي سلولي مي‌باشد که به سه دسته تقسيم مي‌شود:

- قانون عمومي[1]

در اين قانون مقدار يک سلول در مرحله بعدي، به مقدار تک تک سلول‌هاي همسايه در حالت فعلي وابسته است.

- قانون کلي[2]

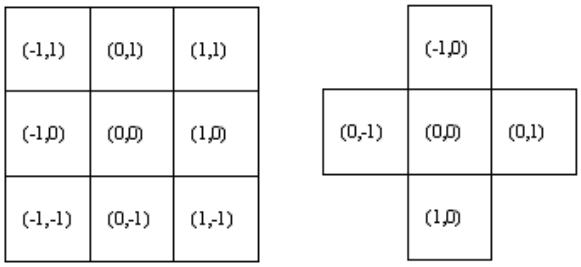
در اين قانون مقدار يک سلول در مرحله بعدي، به تعداد سلول‌هاي همسايه که در حالت‌هاي مختلف مي‌باشند، وابسته است. در اين نوع قانون برخلاف قانون عمومي، توجه‌اي به تک تک سلول‌ها نمي‌شود.

- قانون کلي خارجي[3]

تنها تفاوتي که اين قانون با قوانين کلي دارد در اين است که در تعيين حالت بعدي سلول، حالت فعلي نيز موثر است.

---

[1] General
[2] Totalistic
[3] Outer Totalistic



(ب) **(الف)**

شكل ۹-۲) (الف) همسايگي ون نيومن (ب) همسايگي مور براي اتوماتاي سلولي

ويژگي‌هاي اساسي اتوماتاي سلولي عبارتند از:

آ- فضايي گسسته دارند.

ب- زمان به صورت گسسته پيش مي‌رود.

پ- هر سلولي تعداد محدودي از وضعيت‌هاي ممكن را اختيار مي‌كند.

ت- تمام سلول‌ها يكسان مي‌باشند.

ث- عمل به‌روز در آوردن سلول‌ها به صورت همگام مي‌باشد.

ج- قوانين تصادفي نبوده و بطور قطعي اعمال مي‌شوند.

چ- قانون در هر سايت فقط بستگي به مقادير همسايه‌هاي اطراف آن دارد.

ح- قانون براي مقدار جديد هر سايت فقط بستگي به مقادير تعداد محدودي از مراحل قبل دارد.

براي پيچيده‌تر كردن مدل، مي‌توان تغييرات زير را بر آن اعمال نمود:

آ- افزايش تعداد ابعاد محيط

ب- افزايش تعداد وضعيت‌هاي هر سلول

پ- افزايش طول همسايگي

ت- متغير نمودن همسايگي در طي زمان

ث- متغير نمودن قانون در طي زمان

ج- تغيير در شرايط مرزي

چ- تبديل قانون اتوماتاي سلولي از حالت قطعي به حالت احتمالي

ح- استفاده از شبكه‌هاي با توپولوژي‌هاي متفاوت مانند شبكه‌هاي مثلثي و ۶ضلعي

يكي از اشكالات اتوماتاي سلولي، تعيين فرم قطعي قوانين مورد نياز براي يک كاربرد خاص است. براي مثال تشخيص اين كه براي رسيدن به يک هدف خاص در سمت دوم قانون چه حالتي قرار گيرد، سخت مي‌باشد. از طرفي تمام قوانين مطرح شده در اتوماتاي سلولي مجاز نمي‌باشند و بررسي خود اين شرط نيز در بعضي موارد طاقت‌فرسا مي‌باشد. دليل ديگر نامناسب بودن اين اتوماتاي سلولي، محدوديت آن به مدل كردن سيستم‌هاي قطعي مي‌باشد. از طرفي اغلب سيستم‌هايي كه توسط اين ابزار مدل مي‌شوند، سيستم‌هاي پيچيده‌اي هستند كه دو ويژگي عمده در آنها به چشم مي‌خورد. يكي نويزهايي كه در سيستم وارد مي‌شود و دوم عدم قطعيت و احتمالي بودن سيستم. بدين ترتيب براي چنين سيستم‌هايي وضع قوانين به صورت قطعي، منطقي به نظر نمي‌رسد.

راه‌حل‌هاي متفاوتي در برخورد با اين مشكلات به نظر مي‌رسد. يكي از اين راه‌حل‌ها احتمالاتي كردن قوانين مي‌باشند. به اين ترتيب كه تمام قانون‌هاي امكان‌پذير را در نظر بگيريم و براي فعال شدن آنها احتمالي در نظر بگيريم. اما مشكلي كه باز گريبان‌گير ما خواهد شد آن است كه شناسايي همين احتمال‌ها در سيستم‌هاي ناشناخته عملي نمي‌باشد. پس بايد به سمتي حركت كنيم كه به نحوي خود ابزار با گذشت زمان بتواند قوانين مناسب را استخراج كند. بدين منظور چندين رهيافت را مي‌توان مورد بررسي قرار داد.

رهيافت اول آنست كه تمام مجموعه قوانين را براي اتوماتاي سلولي توليد كنيم و سپس با استفاده از روشهاي مكاشفه‌اي مانند الگوريتم‌هاي ژنتيكي به جستجو برروي اين مجموعه قوانين بپردازيم. بدين ترتيب كه هر سلول با توجه به وضعيت خود در جهان سلولي، مجموعه‌اي از قوانين را مورد پذيرش قرار دهد. البته اين كار اصل همگني را براي اتوماتاي سلولي از بين خواهد برد. در صورتيكه بخواهيم همچنان پايبند اصل همگني باشيم، مي‌توانيم الگوريتم جستجو را براي كل ابزار پياده‌سازي كنيم و قانون‌هاي پذيرفته شده را براي همه سلول‌ها بكار ببريم. مشكلي كه اين مدل ايجاد خواهد كرد اينست كه در مواردي كه فضاي جستجو بسيار بزرگ است، همگرايي اتوماتاي سلولي بسيار كند خواهد بود و از طرفي براي بسياري از كاربردها اين مدل توجيه كننده نمي‌باشد. همين طور ممكن است نتوانيم مبناي خوبي براي همگرا شدن سيستم بيابيم.

رهيافتي كه در برابر تمام ايده‌هاي گذشته به ذهن مي‌رسد، اين است كه به گونه‌اي، هوشمندي را به سلول‌هاي اتوماتاي سلولي بافزاييم و بدين وسيله، تحمل مدل در برابر فرآيندهاي احتمالي و ايجاد نويز در سيستم را افزايش دهيم. به اين ترتيب، استفاده از الگوريتم‌هاي يادگيري ماشين در ساختار سلول مي‌تواند به منظور حل مشكلات مفيد واقع شود.

۳-۹ اتوماتاي يادگير

فرآيند يادگيري موجودات زنده يكي از موضوعات تحقيقاتي جديد بشمار مي‌آيد. اين تحقيقات به دو دسته كلي تقسيم مي‌شوند. دسته نخست به شناخت اصول يادگيري موجودات زنده و مراحل آن مي‌پردازند و دسته دوم بدنبال ارائه يک متدلوژي براي قرار دادن اين اصول در يک ماشين مي‌باشند. يادگيري به صورت تغييرات ايجاد شده در كارايي يک سيستم بر اساس تجربه‌هاي گذشته تعريف مي‌شود [۷۵]. يک ويژگي مهم سيستم‌هاي يادگير، توانايي بهبود كارايي خود با گذشت زمان است. به بيان رياضي مي‌توان اينطور عنوان كرد كه هدف يک سيستم يادگير، بهينه‌سازي وظيفه‌اي است كه كاملا شناخته شده نيست [۷۵, ۷۶]. بنابراين يک رويكرد به اين مسأله، كاهش اهداف سيستم يادگير به يک مسأله بهينه‌سازي است كه بر روي مجموعه‌اي از پارامترها تعريف مي‌شود و هدف آن پيدا كردن مجموعه پارامترهاي بهينه مي‌باشد.

در بسياري از مسائل مطرح شده، اطلاعي از پاسخ‌هاي صحيح مسأله (كه يادگيري با نظارت[1] به آنها نياز دارد) در دست نيست. به همين علت استفاده از يک روش يادگيري بنام يادگيري تقويتي[2] مورد توجه قرار گرفته است. يادگيري تقويتي نه زير مجموعه شبكه‌هاي عصبي است و نه انتخابي بجاي آنها محسوب مي‌شود. بلكه رويكردي متعامد[3] براي حل مسائل متفاوت و مشكل‌تر بشمار مي‌رود. يادگيري تقويتي، از تركيب برنامه‌نويسي پويا و يادگيري نظارتي براي دستيابي به يک سيستم قدرتمند يادگيري ماشين استفاده مي‌كند. در يادگيري تقويتي هدفي براي عامل يادگير مشخص مي‌شود تا به آن دست يابد. آنگاه عامل مذكور ياد مي‌گيرد كه چگونه با آزمايش‌هاي صحيح و خطا با محيط خود، به هدف تعيين شده برسد [۷۸].

در يادگيري تقويتي يک عامل يادگيرنده در طي يادگيري با فعل و انفعالات مكرر با محيط، به يک سياست كنترل بهينه مي‌رسد. كارايي اين فعل و انفعالات با محيط بوسيله بيشينه (كمينه) بودن

[1] Supervised Learning
[2] Reinforcement Learning
[3] Orthogonal

پاداش (جريمه) عددي كه از محيط گرفته مي‌شود، ارزيابي مي‌گردد. علاوه بر اين روش‌هاي يادگيري تقويتي، اولا استفاده از يادگيري به روشي ساده، سيستماتيک و واقعي براي رسيدن به يک جواب تقريباً بهينه را بيان مي‌کنند (پيدا کردن اين جواب بهينه با استفاده از روش‌هاي سنتي بسيار مشکل است). ثانياً، از آنجايي که اين روش يادگيري در محيطي بلادرنگ در حال انجام است، مي‌توان آنرا همزمان با فعاليت محيط (مانند شبکه حسگر) انجام داد. که در اين حالت با تمام رخدادهاي پيش‌بيني نشده به صورت يک تجربه جديد برخورد مي‌شود که مي‌توان از آنها براي بهبود کيفيت يادگيري استفاده کرد [۷۹].

مزيت اصلي يادگيري تقويتي نسبت به ساير روش‌هاي يادگيري عدم نياز به هيچگونه اطلاعاتي از محيط (بجز سيگنال تقويتي) است. يکي از روش‌هاي يادگيري تقويتي، اتوماتاي يادگير تصادفي[1] است که در اين پايان‌نامه از آن به عنوان مکانيزم يادگيري استفاده شده است. اتوماتاي تصادفي بدون هيچگونه اطلاعاتي درباره عمل بهينه سعي در يافتن پاسخ مسأله دارد. يک عمل اتوماتا به صورت تصادفي انتخاب شده و در محيط اعمال مي‌گردد. سپس پاسخ محيط دريافت شده و احتمال عمل‌ها بر طبق الگوريتم يادگيري به‌روز مي‌شوند و روال فوق تکرار مي‌گردد. اتوماتاي تصادفي که به صورت فوق در جهت افزايش کارايي خود عمل کند، يک اتوماتاي يادگير تصادفي گفته مي‌شود.

۹-۳-۱ تاريخچه اتوماتاي يادگير

در دهه ۱۹۶۰ ميلادي، سيپکين[2] روشي براي ساده کردن مسائل فوق به مسأله‌اي براي تشخيص پارامترهاي بهينه و بکاربردن روش‌هاي تپه نوردي براي حل آن معرفي کرد. ستلين[3] و همکارانش در همان زمان کار بر روي اتوماتاي يادگير را آغاز کردند. مفهوم اتوماتاي يادگير نخستين بار توسط وي مطرح شد [۸۰]. ستلين به مدل‌سازي رفتارهاي سيستم‌هاي بيولوژيکي علاقمند بود و يک اتوماتاي قطعي که در محيطي تصادفي فعاليت مي‌کرد را به عنوان مدلي براي يادگيري معرفي نمود. رويکرد ديگري که توسط نارندرا[4] و ويسواناتان[5] ارائه شد [۷۶]، در نظر گرفتن مسأله به صورت پيدا کردن عمل بهينه از ميان مجموعه‌اي از عمل‌هاي مجاز يک اتوماتاي تصادفي بود. تفاوت دو روش اخير در اينست که در روش اول در هر تکرار، فضاي پارامترها به‌روز مي‌شود اما در روش دوم فضاي احتمال

[1] Stochastic Learning Automata
[2] Tsypkin
[3] Tsetlin
[4] Narendra
[5] Viswanatan

به‌روز مي‌گردد. پس از آن اكثر كارهاي انجام شده در تئوري اتوماتاي يادگير، در ادامه مسير معرفي شده توسط ستلين انجام شدند. وارشاوسكي[1] و وورونتسوا[2] اتوماتاي يادگير با ساختار متغير را مطرح كردند كه بردار احتمالات عمل‌هاي[3] خود را بروز مي‌كرد و در نتيجه باعث كاهش تعداد حالات نسبت به اتوماتاي قطعي مي‌شد [۸۱].

نخستين تلاش‌ها براي استفاده از اتوماتاي يادگير در كاربردهاي كنترلي توسط فو[4] و همكارانش انجام شد. از جمله اين كارها مي‌توان به كاربردهايي از اتوماتاي يادگير در تخمين پارامترها[5]، تشخيص الگو[6] و تئوري بازي[7] اشاره كرد [۸۲-۸۴]. مثال‌ها و كاربردهاي متعدد ديگري از اتوماتاي يادگير در كتابي توسط نجيم[8] و پوژنياک[9] ارائه شده است [۸۵].

نتايج اين تحقيقات به استفاده از اتوماتا به عنوان مدلي از يادگيري با كاربردهايي در مسيريابي شبكه‌هاي تلفن، تشخيص الگو و كنترل تطبيق‌پذير منجر شد [۷۵, ۸۲, ۸۵, ۸۶]. از ديگر كاربردهاي اتوماتاي يادگير در مسائل واقعي مي‌توان به تشخيص الگو [۸۷]، جداسازي گراف[10] [۸۸]، برنامه‌ريزي مسير[11] [۸۹] اشاره كرد. اخيرا اتوماتاهاي يادگير در شبكه‌هاي حسگر بي‌سيم نيز مورد استفاده قرار گرفته است [۹۰, ۹۱].

يک اتوماتاي يادگير را مي‌توان به صورت يک شئ مجرد كه داراي تعداد متناهي عمل است، در نظر گرفت. اتوماتاي يادگير با انتخاب يک عمل از مجموعه عمل‌هاي خود و اعمال آن بر محيط، عمل مي‌كند. عمل مذكور توسط يک محيط تصادفي ارزيابي مي‌شود و اتوماتا از پاسخ محيط براي انتخاب عمل بعدي خود استفاده مي‌كند. در طي اين فرايند اتوماتا ياد مي‌گيرد كه عمل بهينه را انتخاب نمايد. نحوه استفاده از پاسخ محيط به عمل انتخابي اتوماتا كه در جهت انتخاب عمل بعدي اتوماتا استفاده مي‌شود، توسط الگوريتم يادگيري اتوماتا مشخص مي‌گردد.

۹-۳-۲ تعريف اتوماتاي يادگير

يک اتوماتاي يادگير از دو قسمت اصلي تشكيل شده است:

---

[1] Varshavski
[2] Vorontsova
[3] Action Probability
[4] Fu
[5] Parameter Estimation
[6] Pattern Recognition
[7] Game Theory
[8] Najim
[9] Poznyak
[10] Graph Partitioning
[11] Path Planning

۱) یک اتوماتاي تصادفي با تعداد محدودي عمل و یک محیط تصادفي که اتوماتا با آن در ارتباط است.
۲) الگوریتم یادگیري که اتوماتا با استفاده از آن عمل بهینه را یاد مي‌گیرد.

۹-۳-۲-۱ اتوماتاي تصادفي

یک اتوماتاي تصادفي به صورت پنج‌تایي $SA \equiv \{\alpha, \beta, F, G, \phi\}$ تعریف مي‌شود که $\alpha \equiv \{\alpha_1, \alpha_2, ..., \alpha_r\}$ مجموعه عمل‌هاي اتوماتا (r تعداد عمل‌هاي اتوماتا)، $\beta \equiv \{\beta_1, \beta_2, ..., \beta_m\}$ مجموعه ورودي‌هاي اتوماتا، $F \equiv \phi \times \beta \to \phi$ تابع تولید وضعیت جدید، $G \equiv \phi \to \alpha$ تابع خروجي که وضعیت فعلي را به خروجي بعدي نگاشت مي‌کند و $\phi(n) \equiv \{\phi_1, \phi_2, ..., \phi_k\}$ مجموعه وضعیت‌هاي داخلي اتوماتا در لحظه n، مي‌باشند.

مجموعه α شامل خروجي‌هاي (عمل‌هاي) اتوماتا است که اتوماتا در هر گام یک عمل از r عمل این مجموعه را براي اعمال بر محیط انتخاب مي‌نماید. مجموعه ورودي‌ها (β) ورودي‌هاي اتوماتا را مشخص مي‌کند. توابع F و G وضعیت فعلي ورودي را به خروجي بعدي (عمل بعدي) اتوماتا نگاشت مي‌کنند. اگر نگاشت‌هاي F و G قطعي باشند، اتوماتا یک اتوماتاي قطعي[1] نامیده مي‌شود. در حالي که نگاشت‌هاي F و G تصادفي باشند، اتوماتا یک *اتوماتاي تصادفي* نامیده مي‌شود.

اتوماتاي یادگیر به دو گروه اتوماتاي با ساختار ثابت[2] و اتوماتاي با ساختار متغیر[3] تقسیم بندي مي‌گردند. در اتوماتاي تصادفي با ساختار ثابت احتمال عمل‌هاي اتوماتا ثابت هستند. در حالي که در اتوماتاي تصادفي با ساختار متغیر احتمالات عمل‌هاي اتوماتا در هر تکرار به‌روز مي‌شوند. در اتوماتاي یادگیر با ساختار متغیر، تغییر احتمال‌هاي عمل‌ها بر اساس الگوریتم یادگیري انجام مي‌شود. همچنین در اتوماتاي یادگیر با ساختار متغیر، وضعیت داخلي اتوماتا ϕ توسط احتمالات عمل‌هاي اتوماتا بازنمایي مي‌شوند. در واقع اتوماتا به صورت یک state-output automata در نظر گرفته مي‌شود که خروجي آن معادل با وضعیت داخلي آن مي‌باشد. وضعیت داخلي اتوماتا $\phi(n)$ در لحظه n را با بردار احتمال عمل‌هاي اتوماتا $P(n)$ که در زیر آمده است، نشان داده مي‌شود:

$$P(n) \equiv \{p_1(n), p_2(n), ..., p_r(n)\} \qquad (۶-۹)$$

بطوریکه

---

[1] Deterministic Automata
[2] Fixed Structure Automata
[3] Variable Structure Automata

$$\sum_{i=1}^{r} p_i(n) = 1 \,, \quad \forall n \quad , \quad p_i(n) = \text{Prob}\,[\alpha(n) = \alpha_i\,] \qquad (۷-۹)$$

در آغاز فعاليت اتوماتا، احتمال عمل‌هاي آن با هم برابر و مساوي $\frac{1}{r}$ مي‌باشند (كه r تعداد عمل‌هاي اتوماتا مي‌باشد).

- محيط

محيط را مي‌توان توسط سه‌تايي $E \equiv \{\alpha, \beta, c\}$ نشان داد كه در آن $\alpha \equiv \{\alpha_1, \alpha_2, ..., \alpha_r\}$ مجموعه ورودي‌هاي محيط، $\beta \equiv \{\beta_1, \beta_2, ..., \beta_m\}$ مجموعه خروجي‌هاي محيط و $c \equiv \{c_1, c_2, ..., c_r\}$ مجموعه احتمال‌هاي جريمه مي‌باشند.

ورودي محيط يكي از r عمل انتخاب شده اتوماتا است. خروجي (پاسخ) محيط به هر عمل i توسط β_i مشخص مي‌شود. اگر β_i يک پاسخ دودويي باشد، محيط مدل P ناميده مي‌شود. در چنين محيطي $\beta_i(n) = 1$ به عنوان پاسخ نامطلوب يا شكست و $\beta_i(n) = 0$ به عنوان پاسخ مطلوب يا موفقيت در نظر گرفته مي‌شوند. در محيط مدل Q، $\beta_i(n)$ شامل تعداد محدودي از مقادير قرار گرفته در بازه [۰،۱] مي‌باشد. درحالي كه در محيط مدلS مقادير $\beta_i(n)$ يک متغير تصادفي در بازه [۰،۱] مي‌باشد ($\beta_i(n) \in [0,1]$). مجموعه C احتمالات جريمه (شكست) پاسخ‌هاي محيط را مشخص مي‌كند و به صورت زير تعريف مي‌شود:

$$c_i = \text{Prob}\{\beta(n) = 1 \,|\, \alpha(n) = \alpha_i\}, \quad i = \{1, 2, ..., r\} \qquad (۸-۹)$$

كه احتمال اينكه عمل α_i پاسخ نامطلوبي از محيط دريافت كند را نشان مي‌دهد. مقادير α_iها نامشخص هستند و فرض مي‌شود كه c_iها يک حداقل يكتا دارند. به همين صورت مي‌توان محيط را توسط مجموعه احتمالات پاداش (موفقيت) $\{d_i\}$ نشان داد كه در اين حالت d_i نشان‌دهنده احتمال دريافت پاسخ مطلوب به عمل α_i مي‌باشد. در محيط‌هاي ايستا[1] مقادير احتمال جريمه (α_iها) ثابت هستند. در حالي كه در محيط‌هاي غير ايستا[2] احتمالات جريمه در طول زمان تغيير مي‌كند.

ارتباط اتوماتاي تصادفي با محيط در شكل ۹-۳ نشان داده شده است. از اين مجموعه به همراه الگوريتم يادگيري تحت عنوان اتوماتاي يادگير تصادفي نام برده مي‌شود. به همين ترتيب اتوماتاي

[1] Stationary
[2] Non- Stationary

يادگير تصادفي را مي‌توان با چهارتايي $LA \equiv \{\alpha, \beta, p, T\}$ نشان داد كه $\alpha \equiv \{\alpha_1, \alpha_2, ..., \alpha_r\}$ مجموعه عمل‌هاي اتوماتا (r تعداد عمل‌هاي اتوماتا)، $\beta \equiv \{\beta_1, \beta_2, ..., \beta_r\}$ مجموعه ورودي‌هاي اتوماتا، $p \equiv \{p_1, p_2, ..., p_r\}$ بردار احتمال عمل‌هاي اتوماتا و $T \equiv p(n+1) = T[\alpha(n), \beta(n), p(n)]$ الگوريتم يادگيري مي‌باشد.

شكل ۹-۳) اتوماتاي يادگير تصادفي

۹-۳-۲-۲ الگوريتم‌هاي يادگيري استاندارد

الگوريتم يادگيري T به صورت $P(n+1) = T(p(n), \alpha(n), \beta(n))$ نشان داده مي‌شود. اگر T يک عملگر خطي باشد، الگوريتم يادگيري تقويتي، خطي ناميده مي‌شود. در غير اينصورت الگوريتم يادگيري غير خطي ناميده مي‌شود. ايده اصلي تمام الگوريتم‌هاي يادگيري به صورت زير است:

اگر اتوماتاي يادگير در تكرار nام، يک عمل خود مانند α_i را انتخاب كند و يک پاسخ مطلوب از محيط دريافت نمايد، $p_i(n)$ (احتمال عمل α_i) افزايش و احتمال ساير عمل‌ها كاهش مي‌يابد. بالعكس، در صورت نامطلوب بودن پاسخ دريافتي از محيط، احتمال عمل α_i كاهش و احتمال ساير عمل‌هاي اتوماتا افزايش مي‌يابد. در هر حال، تغييرات به گونه‌اي صورت مي‌گيرد تا حاصل جمع $p_i(n)$ ها همواره ثابت و مساوي يک باقي بماند. تغيير احتمال عمل‌ها به صورت زير مي‌باشد.

- پاسخ مطلوب از محيط

$$p_i(n+1) = p_i(n) + \sum_{\substack{j=1 \\ j \neq i}}^{r} f_j[p_j(n)]$$
$$p_j(n+1) = p_j(n) - f_j[p_j(n)] \qquad \forall j, \ j \neq i \qquad (۹-۹)$$

- پاسخ نامطلوب از محيط

$$p_i(n+1) = p_i(n) - \sum_{\substack{j=1 \\ j \neq i}}^{r} g_j[p_j(n)]$$
$$p_j(n+1) = p_j(n) - g_j[p_j(n)] \qquad \forall j,\ j \neq i \tag{۹-۱۰}$$

توابع f_j و g_j دو تابع غير منفي هستند كه به ترتيب توابع پاداش و جريمه ناميده مي‌شوند. همانطور كه در روابط فوق مشاهده مي‌شود، اين روابط صحت رابطه (۱-۱۳) را حفظ مي‌كنند. از آنجايي كه الگوريتم‌هاي يادگيري خطي از لحاظ رياضي ساده‌تر مي‌باشند، بررسي‌هاي زيادي بر روي آن‌ها انجام شده است. در الگوريتم يادگيري تقويتي خطي (در اتوماتايي با چند عمل) توابع f_j و g_j به صورت زير تعريف شده‌اند:

$$f_j[p_j(n)] = ap_j(n) \qquad 0 < a < 1 \tag{۹-۱۱}$$

$$g_j[p_j(n)] = \frac{b}{r-1} - bp_j(n) \qquad 0 \leq b < 1 \tag{۹-۱۲}$$

كه r تعداد عمل‌هاي اتوماتا، a پارامتر پاداش و b پارامتر جريمه مي‌باشند. با استفاده از روابط (۹-۶) و (۹-۷) شكل عمومي الگوريتم يادگيري به صورت زير است. اگر در گام nام عمل α_i انتخاب شده باشد، سپس در گام n+۱ام خواهيم داشت:

- پاسخ مطلوب از محيط

$$p_i(n+1) = p_i(n) + a[1 - p_i(n)]$$
$$p_j(n+1) = (1-a)p_j(n) \qquad \forall j,\ j \neq i \tag{۹-۱۳}$$

- پاسخ نامطلوب از محيط

$$p_i(n+1) = (1-b)p_i(n)$$
$$p_j(n+1) = \frac{b}{r-1} + (1-b)p_j(n) \qquad \forall j,\ j \neq i \tag{۹-۱۴}$$

با توجه به مقادير a و b در روابط فوق، سه حالت را مي‌توان در نظر گرفت. اگر مقادير a و b برابر باشند، اتوماتاي يادگير L_{RP}[1] ناميده مي‌شود. زمانيكه b مساوي با صفر باشد اتوماتاي يادگير

[1] Linear Reward Penalty

L_{RI} [1] ناميده مي‌شود. اگر $b<<a$ باشد، اتوماتاي يادگير $L_{R\varepsilon P}$ [2] ناميده مي‌شود. اتوماتاي L_{RP} رفتاري *expedient* از خود نشان مي‌دهد. در حاليكه دو اتوماتاي L_{RI} و $L_{R\varepsilon P}$ رفتاري زير بهينه دارند. روش‌هاي بروزرساني غير خطي نيز معرفي شده‌اند ([۸۵]) كه بهبود محسوسي نسبت به روش‌هاي خطي فوق نشان نمي‌دهند.

يک عامل تعيين كننده كه استفاده از اتوماتاي يادگير تصادفي را در حل مسائل محدود مي‌كند، نرخ همگرايي كند اتوماتاي يادگير است. اين عامل با افزايش تعداد عمل‌هاي اتوماتا تشديد مي‌شود. هرچند استفاده از برخي از روش‌هاي يادگيري غير خطي باعث افزايش سرعت همگرايي اتوماتا مي‌گردد.

۹-۳-۲-۳ الگوريتم‌هاي يادگيري مدل S

در محيط‌هاي S پاسخ محيط به عمل‌هاي اتوماتاي يادگير، يک متغير تصادفي در بازه [۰،۱] مي‌باشد. بنابراين خروجي محيط (ورودي اتوماتا) به صورت زير تغيير مي‌كند:

$$\beta(n) \equiv \{\beta_1, \beta_2, ..., \beta_r\} \equiv \beta_i \in [0,1]; \quad \forall i \qquad (۹-۱۵)$$

از آنجاييكه پاسخ محيط از نوع S يک متغير تصادفي در بازه [۰،۱] است، استفاده از مدلS براي سيستم‌هاي يادگير نياز به اطلاعات اوليه‌اي از كران بالا و پايين شاخص‌هاي كارايي سيستم دارد تا بتواند پاسخ‌هاي محيط را در مقياس [۰،۱] تنظيم نمايد. كارايي مناسب مدلS در [۸۵, ۹۲, ۹۳] نشان داده شده است. در [۹۴] يک الگوريتم غير خطي اما بهينه براي يک اتوماتا با دو عمل در محيط مدل S معرفي شده است. يک روش زير بهينه براي يک اتوماتا با چند عمل در محيط مدل S نيز در [۹۵] معرفي شده است.

- الگوريتم $S-L_{RI}$

در محيط مدل P محيط توسط احتمال جريمه‌ها تعريف مي‌شود. براي هر عمل مانند α_i، محيط پاسخي با يک مقدار تصادفي $[\beta_i(n) \mid \alpha_i]$ به اتوماتا مي‌دهد كه ورودي اتوماتا را تشكيل

[1] Linear Reward Inaction
[2] Linear Reward Epsilon Penalty

مي‌دهد. در مدل P، پاسخ $\beta_i(n)$ با احتمال c_i برابر ۱ (پاسخ نامطلوب) و با احتمال c_i-۱ صفر (پاسخ مطلوب) در نظر گرفته مي‌شود. اما در محيط مدل S، محيط به صورت زير تعريف مي‌شود:

$$E \equiv \{\alpha, \beta, s\} \tag{۹-۱۶}$$

كه

$$s(n) \equiv \{s_1, s_2, ..., s_r\}; \quad s_i = E\{\beta_i(n) \mid \alpha_i\}; \forall i \tag{۹-۱۷}$$

s_i مقدار ميانگين پاسخ β_i براي عمل α_i مي‌باشد و در واقع s_iها به عنوان شدت جريمه‌ها محسوب مي‌شوند. فرض كنيد در گام nام عمل α_i انتخاب شده باشد و پاسخ محيط به آن $\beta_i(n)$ بوده باشد. در چنين حالتي احتمال عمل‌هاي اتوماتاي $S-L_{RI}$ به صورت زير به‌روز مي‌شوند [۹۶]:

$$\begin{aligned} p_i(n+1) &= p_i(n) + a(1-\beta_i(n))(1-p_i(n)) \\ p_j(n+1) &= p_j(n) - a(1-\beta_i(n))p_j(n) \qquad \forall j, \ j \neq i \end{aligned} \tag{۹-۱۸}$$

كه a پارامتر يادگيري و ۰<a<۱ مي‌باشد.

- الگوريتم $S-L_{RP}$

الگوريتم يادگيري تقويتي اتوماتاي يادگير در مدل $S-L_{RP}$ به صورت زير بردار احتمالات عمل‌هاي اتوماتا را به‌روز مي‌كند [۹۱]. در اتوماتايي با r عمل، اگر در تكرار nام عمل α_i انتخاب شده باشد و پاسخ محيط به آن $\beta_i(n)$ باشد، بردار احتمال‌هاي اتوماتا طبق رابطه (۹-۱۹) به‌روز مي‌شود.

$$\begin{aligned} p_i(n+1) &= p_i(n) + a(1-\beta_i(n))(1-p_i(n)) \\ p_j(n+1) &= p_j(n) - a(1-\beta_i(n))p_j(n) \qquad \forall j, \ j \neq i \end{aligned} \tag{۹-۱۹}$$

- الگوريتم $S-L_{R\varepsilon P}$

اتوماتاي يادگيري $S-L_{R\varepsilon P}$ با r عمل و پارامتر پاداش a و پارامتر جريمه b، به صورت زير بردار عمل‌هاي خود را به‌روز مي‌كند. اگر در تكرار nام عمل α_i انتخاب شده باشد و پاسخ محيط به آن $\beta_i(n)$ باشد، بردار احتمال‌هاي اتوماتا طبق رابطه (۹-۲۰) بروز مي‌شود.

$$\begin{aligned} p_i(n+1) &= p_i(n) + a.(1-\beta_i(n)).(1-p_i(n)) - b.\beta_i(n).p_i(n) \\ p_j(n+1) &= p_j(n) - a.(1-\beta_i(n)).p_j(n) + b.\beta_i(n).\left[\frac{1}{r-1} - p_j(n)\right] - a.(1-\beta_i(n)).p_j(n) \quad \forall j \quad j \neq i \end{aligned} \tag{۹-۲۰}$$

۹-۳-۳ ويژگي‌هاي اتوماتاهاي يادگير

براي استفاده از اتوماتاي يادگير در يک کاربرد بايد ويژگي‌هاي بنيادي اتوماتاهاي يادگير و طبيعت کاربرد را با هم در نظر گرفت. اتوماتاهاي يادگير با دارا بودن ويژگي‌هاي زير به عنوان ابزار قدرتمندي در حل بسياري از مسائل استفاده گرديده است.

۱. اتوماتاهاي يادگير در شرايطي که هيچگونه اطلاعاتي در دسترس نيست به خوبي عمل مي‌کنند.
۲. اتوماتاهاي يادگير در شرايطي که عدم اطمينان وجود دارد داراي عملکرد خوبي مي‌باشند.
۳. اتوماتاهاي يادگير عمل جستجو را در فضاي احتمال انجام مي‌دهند.
۴. اتوماتاهاي يادگير براي بهبود وضعيت خود در هر مرحله به يک بازخورد[1] ساده از محيط نياز دارند.
۵. اتوماتاهاي يادگير به عنوان مدلي براي يادگيري در محيط‌هاي توزيع‌شده و چند عامله با ارتباطات محدود و اطلاعات ناقص بسيار کارساز هستند.
۶. اتوماتاهاي يادگير ساختاري ساده دارند و به همين دليل به سادگي در نرم‌افزار يا سخت‌افزار قابل پياده‌سازي هستند.
۷. اتوماتاهاي يادگير در کاربردهاي بهينه‌سازي نياز به استفاده از معيار کارايي نظري و مشتق‌پذير ندارند.
۸. اتوماتاهاي يادگير داراي بار محاسباتي بسيار کم مي‌باشند و به همين دليل به سادگي در کاربردهاي بلادرنگ قابل استفاده مي‌باشند.
۹. روش‌هاي تحليل رياضي قدرتمندي براي تجزيه و تحليل اتوماتاهاي يادگير وجود دارد.

علاوه بر ويژگي‌هاي فوق، استفاده از اتوماتاهاي يادگير در کاربردهايي مفيد است که يک يا چند خصوصيت زير را دارا باشند.

۱. کاربرد به اندازه‌ي کافي پيچيده و غيرقطعي باشد به طوريکه يک مدل رياضي براي آن موجود نباشد.
۲. کاربرد قابليت کنترل توزيعي و مدل‌سازي توسط مجموعه‌اي از عامل‌هاي خودمختار را داشته باشد.

---

[1] Feedback

۳. سیگنال تقویتي یک متغیر تصادفي باشد و بر اساس معیار کارایي تولید گردد.
۴. بهبود کمي در کارایي مورد نظر صرفه‌ي اقتصادي بالایي داشته باشد.
۵. الگوریتم قطعي براي کاربرد مورد نظر وجود نداشته باشد.

البته اتوماتاهاي یادگیر داراي محدودیت‌هایي نیز مي‌باشند که استفاده از آنها را در بسیاري از کاربردها با مشکلاتي مواجه مي‌کند. این محدودیت‌ها عبارتند از:

۱. اتوماتاهاي یادگیر از اطلاعات اولیه‌ي کمي استفاده مي‌کنند و اطلاعات اضافي دربـاره‌ي محیط همیشه در اتوماتاهاي یادگیر قابل استفاده نمي‌باشد.
۲. نرخ همگرایي اتوماتاهاي یادگیر براي بسیار از کاربردها پایین مي‌باشد.

اتوماتاهاي یادگیر مدل‌هاي غیر متداعي مي‌باشند.

٤-۹ اتوماتاي یادگیر سلولي[1]

اتوماتاي یادگیر سلولي، یک مدل ریاضي براي سیستم‌هایي با اجزاي ساده است، بطوریکه رفتار هر جزء بر اساس رفتار همسایگانش و نیز تجربیات گذشته‌اش تعیین و اصلاح مي‌شود. اجزاء ساده تشکیل دهنده این مدل، از طریق کنش و واکنش با یکدیگر رفتار پیچیده‌اي از خود نشان مي‌دهند، بنابراین از آن مي‌توان در مدل‌سازي بسیاري از مسائل بهره برد. این مدل اولین بار در [۹۷] معرفي شد. سرانجام در [۹۸] بیگي و میبدي، این مدل را به صورت ریاضي مورد تحلیل قرار داده و رفتار همگرایي آن را مطالعه نمودند که نتیجه قضایاي مهمي در باب قابلیت همگرایي این مدل به پاسخ بهینه محلي خود، مي‌باشد.

هر اتوماتاي یادگیر سلولي، از یک اتوماتاي سلولي تشکیل شده است که هر سلول آن به یک یا چند اتوماتاي یادگیر مجهز مي‌باشد که حالت این سلول را مشخص مي‌سازد. مانند اتوماتاي سلولي، قانون محلي در محیط حاکم است و این قانون تعیین مي‌کند که آیا عمل انتخاب شده توسط یک اتوماتا در سلول باید پاداش داده شود و یا اینکه جریمه شود. دادن پاداش و یا جریمه باعث بروز درآورده شدن ساختار اتوماتاي یادگیر سلولي بمنظور نیل به یک هدف مشخص مي‌گردد. ایده اصلي اتوماتاي یادگیر سلولي، که زیر مجموعه‌اي از اتوماتاي یادگیر سلولي تصادفي محسوب مي‌شود، استفاده از اتوماتاي یادگیر براي محاسبه احتمال انتقال حالت در اتوماتاي سلولي تصادفي مي‌باشد.

[1] Cellular Learning Automata

اتوماتاي يادگير سلولي را مي‌توان به دو دسته غير همگام[1] و همگام[2] تقسيم كرد. در مدل همگام، تمام سلول‌ها با يک ساعت سراسري هماهنگ شده و به طور همزمان اجرا مي‌شوند.

تعريف- اتوماتاي يادگير سلولي: اتوماتاي يادگير سلولي d-بعدي يک چندتايي $CLA = (Z^d, \phi, A, N, F)$ است به طوريكه:

- Z^d يک شبكه از d-تايي‌هاي مرتب از اعداد صحيح مي‌باشد. اين شبكه مي‌تواند يک شبكه متناهي، نيمه متناهي يا متناهي باشد.

- φ يک مجموعه متناهي از حالت‌ها مي‌باشد.

- A، يک مجموعه از اتوماتاهاي يادگير (LA) است كه هر اتوماتاي يادگير به يک سلول نسبت داده مي‌شود.

- $N = \{\bar{x}_1, ..., \bar{x}_{\bar{m}}\}$ يک زير مجموعه متناهي از Z^d مي‌باشد كه بردار همسايگي خوانده مي‌شود.

- $F : \underline{\phi}^{\bar{m}} \to \underline{\beta}$ قانون محلي CLA مي‌باشد، بطوريكه $\underline{\beta}$ مجموعه مقاديري است كه مي‌تواند به عنوان سيگنال تقويتي پذيرفته شود.

تعريف- پيكربند ي[3] : پيكربندي اتوماتاي يادگير سلولي در زمان k، يک ماتريس $\underline{p}(k) = (\underline{p'}_1(k), ..., \underline{p'_n}(k))$ مي‌باشد كه $\underline{p'}_i(k)$ بردار احتمال اتوماتاي يادگير سلول iام مي‌باشد.

اعمال قانون محلي حاكم بر اتوماتاي يادگير سلولي به هر سلول سبب مي‌شود تا پيكربندي فعلي اتوماتاي سلولي به يک پيكربندي ديگر تبديل شود.

تعريف- پيكربندي قطعي و احتمالاتي: يک پيكربندي $\underline{p}$ را در صورتي كه بردار احتمال هر كدام از اتوماتاها يک بردار يكه باشد، يک پيكربندي قطعي مي‌گوييم. در غير اين صورت آن را احتمالاتي مي‌گوييم.

تعريف- رفتار عمومي[4]: رفتار عمومي‌اتوماتاي يادگير سلولي نگاشت $\Lambda : K \to K$ مي‌باشد كه رفتار ديناميكي اتوماتاي يادگير سلولي را توصيف مي‌كند. K همان مجموعه فضاي پيكربندي اتوماتاي يادگير سلولي است.

---

[1] Asynchronous
[2] Synchronous
[3] Configuration
[4] General Behavior

تعريف- تكامل[1]: تكامل اتوماتاي يادگير سلولي، دنباله اي از پيكربنديهاي متوالي $\{\underline{p}(k)\}_{k\geq 0}$ مي‌باشد. به طوريكه:

$$\underline{p}(k+1)=\Lambda(\underline{p}(k)) \qquad (۲۱-۹)$$

تعريف- اتوماتاي يادگير سلولي يكنواخت[2]: اتوماتاي يادگير سلولي را يكنواخت مي‌گوييم، اگر براي تمام سلول‌ها، تابع همسايگي، قانون محلي و اتوماتاهاي يادگير يكسان باشند.

تعريف- حالت عمومي: حالت اتوماتاي يادگير سلولي يک نگاشت $S:\prod_i \phi \to Z$ مي‌باشد كه به هر الگوي تشكيل شده توسط اتوماتاي يادگير سلولي عددي را نسبت مي‌دهد. و به صورت زير قابل محاسبه است:

$$S(CLA)=(y_1 \ldots y_N)_m \qquad (۲۲-۹)$$

كه $(.)_m$ نمايش عدد در مبناي m، N تعداد سلول‌هاي اتوماتاي يادگير سلولي و $(y_1,\ldots,y_N)$ ترتيب مشخصي از سلول‌ها مي‌باشد. براي مثال براي يک اتوماتاي يادگير سلولي خطي با ۵ سلول و هر سلول با دو حالت ۰ و ۱ كه در حال حاضر الگوي 10011 را توليد كرده باشد، حالت $(10011)_2=19$ مي‌باشد به اين ترتيب آشكار است كه تعداد حالاتي كه يک اتوماتاي يادگير سلولي مي‌تواند داشته باشد، m^N، است.

تعريف- اعداد ولفرام[3]: با فرض اينكه $\underline{\beta}=\{0,1\}$ و در هر سلول يک اتوماتاي يادگير با m عمل $\{0,\ldots,m-1\}$ قرار گرفته باشد، براي توصيف يک قانون خاص مي‌توانيم از روش بازنمايي قانون با استفاده از اعداد ولفرام (كه در بازنمايي قوانين در اتوماتاي سلولي استفاده مي‌شود) استفاده كنيم. ابتدا تمام $m^{\underline{m}}$ عضو دامنه قانون را به صورت خطي مرتب مي‌كنيم. سپس دنباله‌اي از نتايج مربوط به هر كدام را، بازنمايي يک عدد در مبناي m در نظر مي‌گيريم. به اين ترتيب عدد ولفرام قانون مورد نظر معادل دهدهي آن است. براي مثال براي يک اتوماتاي يادگير سلولي خطي با $\phi=\{0,1\}$ و $\underline{\beta}=\{0,1\}$، قانون شكل، قانون شماره ۲(۰۰۱۱۰۱۱۰)=۵۴ است.

[1] Evolution
[2] Homogeneous
[3] Wolfram Numbers

۱۱۱	۱۱۰	۱۰۱	۱۰۰	۰۱۱	۰۱۰	۰۰۱	۰۰۰
۰۰	۰۰	۱۱	۱۱	۰۰	۱۱	۱۱	۰۰

شکل ۹-٤) قانون ٥٤

۹-۵ شبیه‌سازي روش‌هاي پیشنهادي

تمامي شبیه‌سازي‌هاي انجام شده براي این پایان‌نامه با استفاده از زبان جاوا[1] و در محیط برنامه‌نویسي نت بینز نسخه ۷.۰[2] انجام شده است. در این بخش ضمن ارایه منابع کد این شبیه‌سازي‌ها، توضیحاتي نیز در مورد روندها[3] و کلاس‌هاي[4] پیاده‌سازي شده داده خواهد شد.

بسته[5] پروژه پیاده‌سازي روش‌هاي پیشنهادي که cellularevolution نام دارد، شامل کلاس‌هاي زیر است:

- کلاس `CellularEvolution` : این کلاس اصلي پروژه براي اجراي شبیه‌سازي‌ها مي‌باشد که روند اصلي نیز در آن قرار دارد.
- کلاس `Benchmark` : این کلاس مشخص‌کننده قالب کلي تمامي توابع محکي است که برنامه قابلیت اجرا و آزمایش بر روي آن‌ها را دارد. براي هر تابع محک، توابع این کلاس به صورت جداگانه پیاده‌سازي خواهند شد.
- کلاس `MPBenchmark` : این کلاس، پیاده‌سازي کلاس `Benchmark` براي تابع محک قله‌هاي متحرک است.
- کلاس `Parameters` : این کلاس حاوي تمامي پارامترهاي مورد نیاز اجراي برنامه مي‌باشد.
- کلاس `EvolutionUtils` : این کلاس شامل روندهاي کمکي و ابزاري مختلف در زمینه‌هاي گوناگون است که در قسمت‌هاي مختلف پیاده‌سازي از آن‌ها استفاده شده است.

[1] JAVA
[2] NetBeans IDE 7.0
[3] Method
[4] Class
[5] Package

- کلاس Chromosome : اين کلاس موجودات مورد نياز الگوريتم تکاملي را تعريف و مشخص مي‌کند.
- کلاس ChromosomeMem : اين کلاس به عنوان حافظه‌هاي مورد نياز در سلول‌ها مورد استفاده قرار مي‌گيرند.
- کلاس CA : اين کلاس پياده‌سازي ساختار کلي يک اتوماتاي سلولي است.
- کلاس Cell : اين کلاس مشخص‌کننده سلول‌هاي داخل اتوماتاي سلولي است.
- کلاس CellSate : اين کلاس حالت هر سلول را تعريف مي‌کند.

۹-۵-۱ کلاس Parameters

در اين کلاس مقادير اوليه تمامي پارامترهاي مورد نياز در قسمت‌هاي مختلف را تعريف و مقداردهي مي‌کند. علاوه بر آن فايل myRules.fcl را بارگذاري مي‌کند که حاوي قوانين فازي مورد نياز براي موتور استنتاج فازي در الگوريتم تکامل تفاضلي فازي است. محتويات اين فايل که شامل تعاريف متغيرهاي فازي، روش استنتاج و خود قوانين در جدول ۹-۱ درج شده است و موتور فازي از اطلاعات و تنظيمات بار شده از اين فايل جهت استنتاج استفاده خواهد کرد.

جدول ۹-۱) محتویات فایل تنظیمات سیستم استنتاج فازي

myRules.fcl

```
FUNCTION_BLOCK FuzzyCellularDE

  VAR_INPUT
      ncl : REAL;
      ncf : REAL;
  END_VAR

  VAR_OUTPUT
      F : REAL;
      Cr : REAL;
  END_VAR

  FUZZIFY ncl
     TERM small := gauss 0.05 0.35;
     TERM medium := gauss 0.5 0.35;
     TERM big := gauss 0.95 0.35;
  END_FUZZIFY

  FUZZIFY ncf
      TERM small := gauss 0.05 0.35;
      TERM medium := gauss 0.5 0.35;
      TERM big := gauss 0.95 0.35;
  END_FUZZIFY

  DEFUZZIFY F
   TERM small := gauss 0.05 0.35;
      TERM medium := gauss 0.5 0.35;
      TERM big := gauss 0.95 0.35;
      METHOD : COG;
      DEFAULT := 0;
  END_DEFUZZIFY

  DEFUZZIFY Cr
      TERM small := gauss 0.05 0.35;
      TERM medium := gauss 0.5 0.35;
      TERM big := gauss 0.95 0.35;
      METHOD : COG;
      DEFAULT := 0;
  END_DEFUZZIFY

  RULEBLOCK main
      AND : MIN;
      // Use 'min' activation method
      ACT : MIN;
      // Use 'max' accumulation method
      ACCU : MAX;

     RULE 1 : IF ncf IS small AND ncl IS small
                  THEN F IS small, Cr IS small;

     RULE 2 : IF ncf IS small AND ncl IS medium
                  THEN F IS medium, Cr IS medium;
```

```
      RULE 3 : IF ncf IS small AND ncl IS big
                  THEN F IS big, Cr IS big;

      RULE 4 : IF ncf IS medium AND ncl IS small
                  THEN F IS medium, Cr IS medium;

      RULE 5 : IF ncf IS medium AND ncl IS medium
                  THEN F IS medium, Cr IS medium;

      RULE 6 : IF ncf IS medium AND ncl IS big
                  THEN F IS big, Cr IS big;

      RULE 7 : IF ncf IS big AND ncl IS small
                  THEN F IS big, Cr IS big;

      RULE 8 : IF ncf IS big AND ncl IS medium
                  THEN F IS big, Cr IS big;

      RULE 9 : IF ncf IS big AND ncl IS big
                  THEN F IS big, Cr IS big;

  END_RULEBLOCK

END_FUNCTION_BLOCK
```

کد این کلاس به صورت زیر است:

```
package cellularevolution;
import net.sourceforge.jFuzzyLogic.*;

public class Parameters {
    public FIS myFIS;

    public long evalsMax;
    public int populationSize;
    public int partitionsNum;
    public int thetaInit;
    public int localSearchNum;
    public int neighborhoodSize;

    public double FInit;
    public double CrInit;
    public int DEScheme;
    public double mutProb;

    public double hillMut;

    public double ssInit;
    public double stepMax;
    public double acc;

    public boolean alpinistEnable;
    public int localSearchType;
    public boolean FLearningEnable;
    public boolean CrLearningEnable;
    public boolean FNLearningEnable;
    public boolean CrNLearningEnable;
    public boolean FFuzzyEnable;
    public boolean CrFuzzyEnable;

    public double aF;
```

```
public double bF;
public double aFN;
public double bFN;
public double aCr;
public double bCr;
public double aCrN;
public double bCrN;

public int dimensionsNum;
public int peaksNum;
public int changePeriod;
public double moveSeverity;
public double heightSeverity;
public double widthSeverity;
public double minWidth;
public double maxWidth;
public double standardWidth;
public double lambda;
public int peaksType;

public Parameters(){
    myFIS = FIS.load("FuzzyRules.fcl",true);

    dimensionsNum=5;
    peaksNum=10;
    changePeriod=5000;
    populationSize=100;
    partitionsNum=10;
    thetaInit=10;
    neighborhoodSize=2;
    localSearchType=1;
    localSearchNum=3;
    evalsMax=500000;

    alpinistEnable=false;
    FFuzzyEnable=false;
    CrFuzzyEnable=false;

    DEScheme=3;
    FInit=0.5;
    CrInit=0.5;
    mutProb=0.5;

    hillMut=0.2;

    ssInit=5;
    stepMax=4;
    acc=0.2;

    FLearningEnable=false;
    aF=0.1;
    bF=0.0;
    FNLearningEnable=false;
    aFN=0.1;
    bFN=0.1;

    CrLearningEnable=false;
    aCr=0.1;
    bCr=0.0;
    CrNLearningEnable=false;
    aCrN=0.1;
    bCrN=0.1;

    moveSeverity=1.0;
    heightSeverity=7.0;
    widthSeverity=1.0;
    minWidth=1.0;
    maxWidth=12.0;
    standardWidth=0.0;
    lambda=0.0;
    peaksType=1;
}
```

- روند printParams : با اجراي اين روند تمامي مقادير جاري براي پارامترهاي روش‌هاي پيشنهادي و همچنين محيط محک چاپ خواهد شد.

```
    public void printParams(){
        System.out.println("Population Size="+populationSize+", "+
                           "Number of Dimensions="+dimensionsNum+", "+
                           "Number of Peaks="+peaksNum+", "+
                           "Change Period="+changePeriod+", "+
                           "Number of Partitions="+partitionsNum+", "+
                           "Initial Theta="+thetaInit+", "+
                           "Neighborhood Size="+neighborhoodSize+", "+
                           "Number of LocalSearch="+localSearchNum+", "+
                           "DE Scheme="+DEScheme+", "+
                           "Mutation Probability="+mutProb+", "+
                           "Initial F="+FInit+", "+
                           "initial Cr="+CrInit+", "+
                           "F Learning="+FLearningEnable+", "+
                           "Cr Learning="+CrLearningEnable+", "+
                           "F Neighbor Learning="+FNLearningEnable+", "+
                           "Cr Neighbor Learning="+CrNLearningEnable+", "+
                           "F Fuzzy="+FFuzzyEnable+", "+
                           "Cr Fuzzy="+CrFuzzyEnable+", "+
                           "Move Severity="+moveSeverity+", "+
                           "Alpinist="+alpinistEnable+", "+
                           "Type of Local Search="+localSearchType+", "+
                           "Number of Local Search="+localSearchType+", "+
                           "Maximum Number of Evaluations="+evalsMax);
    }
}
```

۹-۵-۲ کلاس Benchmark

در اين كلاس هيچ كدام از روندها پياده‌سازي نشده‌اند و تنها قالب كلي توابع محک قابل استفاده توسط برنامه را مشخص مي‌كند. در آن خصوصياتي نيز براي نگهداري پارامترها و وضعيت جاري محيط پوياي محک تعريف شده است از قبيل تعداد ابعاد، تعداد قله‌ها، دوره تغيير، حداقل و حداكثر دامنه محيط در هر بعد، دانه اوليه اعداد تصادفي، شماره دوره فعلي كه محيط در آن قرار دارد، تعداد دفعات تكامل كه از آخرين تغيير تاكنون انجام شده و همچنين فلگ مشخص كننده وقوع تغيير در محيط.

```
package cellularevolution;

public abstract class Benchmark {
    public boolean isChanged;

    public int dimensionsNum;
    public int peaksNum;

    public int changePeriod;
    public double[] minCoordinate;
    public double[] maxCoordinate;
    public long mainSeed;

    public int periodNum;
    public int itSinceLastChange;

    public abstract void  init();

    public abstract long evalsNum();
```

```
    public abstract void printPeaksData();
    public abstract void printErrors();

    public abstract double eval(Chromosome ch);
    public abstract double eval(double[] gens);

    public abstract double evalDummy(Chromosome ch);
    public abstract double evalDummy(double[] gens);

    public abstract int genNum();
    public abstract double offlineError();
    public abstract double currentError();
}
```

۳-۵-۹ کلاس MPBBenchmark

این کلاس یکي از پیاده‌سازي‌هاي کلاس Benchmark براي تابع محک قله‌هاي متحرک است. در این کلاس نمونه‌اي از تابع قله‌هاي محرک ساخته شده و پارامترهاي آن مقداردهي اولیه مي‌شوند. روندهایي نیز براي محاسبه شایستگي موجودات، بازگرداندن مقدار خطاي آفلاین و یا خطاي جاري تا آن لحظه در آن پیاده‌سازي شده‌اند.

```
package cellularevolution;
import peaks.*;
import java.util.*;

public class MPBenchmark extends Benchmark{
    movpeaks function;
    double moveSeverity;
    double heightSeverity;
    double widthSeverity;
    double minWidth;
    double maxWidth;
    double standardWidth;
    double lambda;
    int peakType;
    long MPBSeed;
```

- روند `init` : با اجراي این روند یک نمونه محیط پویاي قله‌هاي متحرک را بر اساس مقادیر مشخص شده در خصوصیات همین کلاس ساخته مي‌شود و متغیرهاي مشخص کننده حالت‌هاي جاري نیز مقداردهي اولیه مي‌شوند.

```
    public void init(){
        dimensionsNum=EvolutionUtils.pars.dimensionsNum;
        peaksNum=EvolutionUtils.pars.peaksNum;
        changePeriod=EvolutionUtils.pars.changePeriod;

        minCoordinate=new double[dimensionsNum];
        maxCoordinate=new double[dimensionsNum];
        for(int i=0;i<dimensionsNum;i++){
            minCoordinate[i]=0.0;
            maxCoordinate[i]=100.0;
        }

        moveSeverity=EvolutionUtils.pars.moveSeverity;
        heightSeverity=EvolutionUtils.pars.heightSeverity;
        widthSeverity=EvolutionUtils.pars.widthSeverity;
        minWidth=EvolutionUtils.pars.minWidth;
        maxWidth=EvolutionUtils.pars.maxWidth;
        standardWidth=EvolutionUtils.pars.standardWidth;
        lambda=EvolutionUtils.pars.lambda;
        peakType=EvolutionUtils.pars.peaksType;
```

```
        periodNum=1;
        itSinceLastChange=0;
        isChanged=false;

        EvolutionUtils.initRandom((new Random()).nextLong());
        MPBSeed=EvolutionUtils.rnd.nextLong();

        function=new movpeaks(peaksNum,
                              dimensionsNum,
                              changePeriod,
                              moveSeverity,
                              minWidth,
                              maxWidth,
                              standardWidth,
                              lambda,
                              heightSeverity,
                              widthSeverity,
                              peakType,
                              MPBSeed);

        function.mincoordinate=minCoordinate[0];
        function.maxcoordinate=maxCoordinate[0];
        function.init_peaks();
    }
```

- روند `evalsNum` : اين روند تعداد محاسبات شايستگي كه تاكنون انجام گرفته را برمي‌گرداند.

```
    public long evalsNum(){
        return function.get_number_of_evals();
    }
```

- روند `printPeaksData` : اين روند با اجراي تابع اصلي كه در كلاس تابع محک پوياي تعريف شده است، مشخصات جاري قله‌هاي محيط از قبيل مكان، اندازه و ارتفاع را چاپ مي‌كند.

```
    public void printPeaksData(){
        function.printPeakData();
    }
```

- روند `printErrors` : با اجراي اين روند، خلاصه‌اي از خطاي آفلاين جاري، بهترين خطا و همچنين بهترين قله موجود در محيط به همراه اكسترمم عمومي بودن بهترين قله يافت شده چاپ خواهد شد.

```
    public void printErrors(){
        System.out.println("Offline Error:"+function.get_offline_error());
        System.out.println("Best Error:"+function.get_current_error());
        System.out.println("Current
Peak:+"+function.getCurrentPeak()+","+function.get_right_peak());
        System.out.println("-----------------------------------------");
    }
```

- روند `eval` : اين دو روند يک موجود و يا مكان يک موجود را به عنوان ورودي گرفته و شايستگي آن را محاسبه مي‌كنند.

```
    public double eval(Chromosome ch){
        double tmp;
        tmp=eval(ch.gens);
        return tmp;
    }

    public double eval(double[] gens){
        double tmp;
        tmp=function.eval_movpeaks(gens);
        if(function.get_number_of_evals()%changePeriod==0){
```

```
            isChanged=true;
            EvolutionUtils.benchmark.periodNum=EvolutionUtils.benchmark.periodNum+1;
        }
       return tmp;
    }
```

- روند evalDummy : اين دو روند نيز با دريافت يک موجود و يا مکان آن، شايستگي فعلي آن را محاسبه مي‌کنند ولي اين محاسبه شايستگي در تعداد دفعات شايستگي منظور نخواهد شد.

```
    public double evalDummy(Chromosome ch){
        return function.dummy_eval(ch.gens);
    }

    public double evalDummy(double[] gens){
        return function.dummy_eval(gens);
    }
```

- روند genNum : اين روند اندازه ابعاد محيط را بر مي‌گرداند.

```
    public int genNum(){
        return function.geno_size;
    }
```

- روند offlineError : اين روند خطاي آفلاين تا اين لحظه را بر مي‌گرداند.

```
    public double offlineError(){
        return function.get_offline_error();
    }
```

- روند getCurrentError : اين روند خطاي جاري (کمترين خطا از آخرين تغيير تاکنون) را بر مي‌گرداند.

```
    public double currentError(){
        return function.get_current_error();
    }
}
```

۴-۵-۹ کلاس EvolutionUtils

در اين کلاس روندهاي مختلفي وجود دارد که در کلاس‌ها و قسمت‌هاي مختلف و جداگانه کاربرد دارند. اين روندها در اين کلاس قرار داده شده‌اند تا همگي به آن‌ها دسترسي داشته باشند. مقادير پارامترها، منبع مشترک توليد اعداد تصادفي، تابع محک، مجموعه موجودات غيرفعال، مجموعه موجودات کانديد براي بروزرساني و متغيرهاي خروجي همگي در اين کلاس تعريف شده‌اند.

```
package cellularevolution;
import java.util.*;

public class EvolutionUtils {
    public static Parameters pars=new Parameters();;
    public static Random rnd;
    public static Benchmark benchmark;
    public static ArrayList<Chromosome> inactivePopulation;
    public static TreeSet<Cell> updateCandidates;
    public final static double bigMin=-100000000;
```

- روند chooseAction : اين روند با گرفتن يک بردار احتمال اعمال، يکي از اعمال را بر حسب احتمال انتخاب مي ند و به عنوان خروجي برمي‌گرداند.

```
    public static int chooseAction(double[] actionProps){
        double rn;
        double sum;
        int Ind=-1;
        rn=EvolutionUtils.rnd.nextDouble();
        sum=0;
        for(int k=0;k<actionProps.length;k++){
            if(rn<sum+actionProps[k]){
                Ind=k;
                break;
            }
            sum+=actionProps[k];
        }
        return Ind;
    }
```

- روند SLRI : این روند با دریافت بردار احتمال اعمال، شماره عمل انجام شده، پارامترهاي پاداش و تنبیه و همچنین پاسخ محیط، بردار احتمال را بر حسب آن‌ها و با استفاده از الگوریتم یادگیري تقویتي SLRI که یک مدل S است، بروزرساني مي‌کند.

```
    public static void SLRI(double[] actionProps,int actionNum,double a,double B){
        actionProps[actionNum]=actionProps[actionNum]+a*(1-B)*(1-actionProps[actionNum]);
        for(int k=0;k<actionProps.length;k++){
            if(k==actionNum)
                continue;
            actionProps[k]=actionProps[k]-a*(1-B)*actionProps[k];
        }
    }
```

- روند SLRP : این روند با دریافت بردار احتمال اعمال، شماره عمل انجام شده، پارامترهاي پاداش و تنبیه ، بردار احتمال را بر حسب آن‌ها و با استفاده از الگوریتم یادگیري تقویتي SLRP که یک مدل S است، بروزرساني مي‌کند.

```
    public static void SLRP(double[] actionProps,int actionNum,double a,double b,double
B){
        actionProps[actionNum]=actionProps[actionNum]-B*b*actionProps[actionNum]+(1-
B)*a*(1-actionProps[actionNum]);
        for(int k=0;k<actionProps.length;k++){
            if(k==actionNum)
                continue;
            actionProps[k]=actionProps[k]+B*((b/(actionProps.length-1))-
b*actionProps[k])-(1-B)*a*actionProps[k];
        }
    }
```

- روند TLRPReward : این روند با دریافت بردار احتمال اعمال، عمل انجام شده را به میزان پاداش مشخص شده در ورودي، با مدل T مورد پاداش قرار مي‌دهد.

```
    public static void TLRPReward(double[] actionProps,int actionNum,double reward){
        actionProps[actionNum]=actionProps[actionNum]+reward*(1-actionProps[actionNum]);
        for(int k=0;k<actionProps.length;k++){
            if(k==actionNum)
                continue;
            actionProps[k]=(1-reward)*actionProps[k];
        }
    }
```

- روند TLRPPunish : این روند با دریافت بردار احتمال اعمال، عمل انجام شده را به میزان تنبیه مشخص شده در ورودي، با مدل T مورد تنبیه قرار مي‌دهد.

```
    public static void TLRPPunish(double[] actionProps,int actionNum,double punish){
        actionProps[actionNum]=(1-punish)*actionProps[actionNum];
        for(int k=0;k<actionProps.length;k++){
            if(k==actionNum)
                continue;
```

```
            actionProps[k]=(punish/(actionProps.length-1))+(1-punish)*actionProps[k];
        }
    }
```

- روند `initRandom` : اين روند يک دانه اوليه را به عنوان ورودي گرفته و در ضمن ساختن يک توليدکننده اعداد تصادفي جديد، اين مقدار به عنوان دانه اوليه آن قرار مي‌دهد.

```
    public static void initRandom(long seed){
        rnd=new Random(seed);
    }
```

- روند `getLinearPos` : اين روند با گرفتن مکان يک سلول به صورت چند بعدي، مکان خطي آن را محاسبه و بازمي‌گرداند.

```
    public static int getLinearPos(int[] partition,Benchmark bm){
        int pos=0;
        int order=1;
        for(int i=partition.length-1;i>=0;i--){
            if(partition[i]<0 || partition[i]>=pars.partitionsNum)
                return -1;
            pos=pos+order*partition[i];
            order=order*pars.partitionsNum;
        }
        return pos;
    }
```

- روند `samePartition` : اين روند يا دريافت بردار مکان دو سلول، مشخص مي کند که آيا اين دو سلول يکي هستند يا خير.

```
    public static boolean samePartition(int[] p1,int[] p2){
        for(int i=0;i<p1.length;i++){
            if(p1[i]!=p2[i])
                return false;
        }
        return true;
    }
```

- روند `toBound` : اين روند عدد ورودي را در صورت خارج بودن از محدوده مشخص شده در ورودي، برابر با مقادير لبه قرار مي‌دهد.

```
    public static double toBound(double num,double min,double max){
        return (num*(max-min))+min;
    }
```

- روند `getPartition` : اين روند آدرس چند بعدي سلول حاوي مکان يک موجود را برمي‌گرداند.

```
    public static int[] getPartition(Chromosome ch,Benchmark bm){
        int[] partition=new int[ch.gens.length];
        for(int i=0;i<partition.length;i++){
            partition[i]=(int) ((ch.gens[i]-bm.minCoordinate[i]) /
((bm.maxCoordinate[i]-bm.minCoordinate[i])/(double)pars.partitionsNum));
            if(partition[i]>=pars.partitionsNum)
                partition[i]=pars.partitionsNum-1;
        }
        return partition;
    }
}
```

۹-۵-۵ کلاس Chromosome

در این کلاس موجود مورد استفاده در الگوریتم تکاملي تعریف شده است که در آن مکان، شایستگي، سلول مقصد و فاصله تا سلول مقصد را نگهداري مي‌کنیم.

```
package cellularevolution;

public class Chromosome implements Comparable{
    public double[] gens;
    public double fitness;
    public int desCell;
    public int hop;
```

- روند compareTo : این روند موجود جاري را با موجود ورودي بر حسب شایستگي مقایسه مي‌کند. از این روند جهت مرتب‌سازي موجودات بر حسب شایستگي استفاده مي‌شود.

```
    public int compareTo(Object e){
        if(((Chromosome)e).fitness<this.fitness)
            return -1;
        if(((Chromosome)e).fitness>this.fitness)
            return 1;
        return 0;
    }
```

- روند Chromosome : این‌ها روندهاي سازنده کلاس جاري هستند که خصوصیات موجود را مقداردهي اولیه و آماده استفاده مي‌کنند.

```
    public Chromosome(int numberOfDimensions){
        gens=new double[numberOfDimensions];
        fitness=EvolutionUtils.bigMin;
        hop=-1;
        desCell=-1;
    }

    public Chromosome(int numberOfDimensions,double[] min,double[] max){
        this(numberOfDimensions);
        for(int j=0;j<numberOfDimensions;j++)

gens[j]=EvolutionUtils.toBound(EvolutionUtils.rnd.nextDouble(),min[j],max[j]);

    }
}
```

۹-۵-۶ کلاس ChromosomeMem

این کلاس حافظه‌هاي مورد استفاده در سلول‌ها براي نگهداري مکان بهترین موجودات را تعریف مي‌کند. علاوه بر نگهداري مقادیر شایستگي و مکان موجود، دوره‌اي که این حافظه در آن معتبر است را نیز مشخص مي‌کنیم.

```
package cellularevolution;

public class ChromosomeMem{
    public double[] gens;
    public double fitness;
    boolean isEmpty=true;
    public int periodNum;
```

- روند ChromosomeMem : اين روند سازنده كلاس جاري مي‌باشد و وظيفه آن مقداردهي اوليه خصوصيات كلاس مي‌باشد.

```
    public ChromosomeMem(){
        isEmpty=true;
        fitness=EvolutionUtils.bigMin;
        periodNum=EvolutionUtils.benchmark.periodNum;
        gens=null;
    }
```

- روند isEmpty : اين روند مشخص مي كند كه آيا اين حافظه خلي است يا خير.

```
    public boolean isEmpty(){
        return isEmpty;
    }
```

- روند update : اين دو روند يک موجود و يا حافظه ديگر را به عنوان ورودي دريافت مي‌كند و مقادير حفظ شده در آن را به داخل خود انتقال مي‌دهد.

```
    public void update(ChromosomeMem mem){
        if(mem==null || mem.isEmpty()){
            gens=null;
            fitness=EvolutionUtils.bigMin;
            periodNum=EvolutionUtils.benchmark.periodNum;
            isEmpty=true;
            return;
        }
        if(gens==null)
            gens=new double[mem.gens.length];
        System.arraycopy(mem.gens, 0, gens, 0, mem.gens.length);
        fitness=mem.fitness;
        periodNum=mem.periodNum;
        isEmpty=false;
    }

    public void update(Chromosome ch){
        periodNum=EvolutionUtils.benchmark.periodNum;
        if(ch==null){
            gens=null;
            fitness=EvolutionUtils.bigMin;
            isEmpty=true;
            return;
        }
        if(gens==null)
            gens=new double[ch.gens.length];
        System.arraycopy(ch.gens, 0, gens, 0, ch.gens.length);

        fitness=ch.fitness;
        isEmpty=false;
    }
```

- روند getAliveClone : اين روند يک موجود تكاملي بر اساس داده‌هاي موجود در حافظه جاري مي‌سازد.

```
    public Chromosome getAliveClone(){
        Chromosome ch=new Chromosome(gens.length);
        System.arraycopy(gens, 0, ch.gens, 0, gens.length);
        ch.fitness=fitness;
        return ch;
    }
}
```

۹-۵-۷ کلاس CellSate

این کلاس حالت سلول‌ها در مدل اتوماتاي سلولي را به همراه اطلاعاتي که جزيي از حافظه محسوب مي‌شوند را تعريف مي‌کند. اطلاعات حالت سلول شامل موجودات داخل سلول، حد آستانه ظرفيت سلول، حافظه بهترين موجود، مقدار پارامترهاي تکاملي انتخاب شده و همچنين بردار احتمالات اعمال مربوط به اين پارامترها مي‌باشد.

```
package cellularevolution;
import java.util.*;

public class CellState{
    public ArrayList<Chromosome> population;
    public int theta;

    public ChromosomeMem cellMem;

    public double F;
    public double Cr;

    public double[] FActionProp;
    public double[] CrActionProp;
```

- روند `addIndivisual` : اين روند يک موجود را به ليست موجودات اين حافظه اضافه مي‌کند.

```
    public void addIndivisual(Chromosome ind){
        if(!population.contains(ind))
            population.add(ind);
    }
```

- روند `CellState` : اين روند سازنده کلاس جاري مي‌باشد و وظيفه آن مقداردهي اوليه خصوصيات حافظه و آماده‌سازي آن‌ها براي استفاده مي‌باشد.

```
    public CellState(){
        theta=-1;
        population=new ArrayList<Chromosome>();

        cellMem=new ChromosomeMem();
        F=-1;
        Cr=-1;
        FActionProp=new double[10];
        for(int i=0;i<FActionProp.length;i++){
            FActionProp[i]=0;
        }
        CrActionProp=new double[9];
        for(int i=0;i<CrActionProp.length;i++){
            CrActionProp[i]=0;
        }
    }
}
```

۹-۵-۸ کلاس Cell

اين کلاس تعريف‌کننده يک سلول در مدل اتوماتاي سلولي است. تمامي خصوصيات و تواناييهاي يک سلول در اين کلاس تعريف و پياده‌سازي شده است، از قبيل: حالت فعلي، حالت

بعدي، ـدرس مكان سلول، همسايگان با شعاع يک و بيشتر، آخرين مرتبه‌اي که موجودي در اين سلول تکامل يافته، تعداد دفعاتي که موجودات در اين سلول تکامل يافته‌اند پس از آخرين تغيير، نوع سلول، ليست تاريخچه بهترين موجود، آخرين اعمال انتخاب شده براي پارامترهاي تکاملي، حافظه‌هاي بهترين موجود محلي و بهترين در همسايگي.

```
package cellularevolution;
import java.util.*;

public class Cell implements Comparable{
    public CellState currentState;
    public CellState nextState;

    public int[] partition;
    public Cell[] neighbors;
    public Cell[] adjacentNeighbors;

    int lastHistAdd;

    int landscapeType;
    LinkedList<Double> cMemHist;

    ChromosomeMem currentBest;
    ChromosomeMem localBest;
    Cell bestNCell;

    int FInd;
    int CrInd;

    double Cl=0.5;
    double Cf=0.5;
```

- روند `Cell` : اين روند سازنده کلاس جاري مي‌باشد و با دريافت مکان چند بعدي سلول فعلي، آن آدرس را به خصوصيت خود انتقال مي‌دهد.

```
    public Cell(int[] partition){
        this.partition=partition;
    }
```

- روند `init` : با اجراي اين روند خصوصيات سلول جاري براي شروع استفاده مقداردهي اوليه مي‌شوند.

```
    public void init(){
        currentState=new CellState();
        nextState=null;
        landscapeType=0;
        cMemHist=new LinkedList<Double>();
        lastHistAdd=0;
        FInd=-1;
        CrInd=-1;
        currentBest=null;
        localBest=null;
        Cl=0.5;
        Cf=0.5;
        bestNCell=null;
    }
```

- روند `initNextState` : اين روند آماده‌سازي حالت آينده سلول را براي استفاده انجام مي‌دهد که شامل انتقال بخشي از اطلاعات از حالت فعلي نيز مي‌شود.

```
    public void initNextState(){
        nextState=new CellState();

        if(EvolutionUtils.pars.FLearningEnable)
```

```
            System.arraycopy(currentState.FActionProp, 0, nextState.FActionProp, 0,
currentState.FActionProp.length);
        if(EvolutionUtils.pars.CrLearningEnable)
            System.arraycopy(currentState.CrActionProp, 0, nextState.CrActionProp, 0,
currentState.CrActionProp.length);
        nextState.F=currentState.F;
        nextState.Cr=currentState.Cr;
        nextState.theta=currentState.theta;
        nextState.cellMem.update(currentState.cellMem);
    }
```

- روند update : اين روند براي بروزرساني سلول توسط كلاس اتوماتاي سلولي اجرا مي‌شود. در اين روند، در صورت وجود موجودات فعال در سلول، روند مربوط به اجراي عمليات تكامل و كنترل جمعيت اجرا مي‌شود.

```
    public void update(){
        initNextState();
        if(currentState.population.size()>0){
            refreshMems();
            setCurrentBest();
            setLocalBest();
            if (EvolutionUtils.pars.alpinistEnable)
                landscapeTypeDetect();
            else
                landscapeType=0;
            evolve();
            controlDensity();
        }
    }
```

- روند evolve : با اجراي اين روند، روند مناسب جهت تكامل موجودات داخل سلول انتخاب و اجرا مي‌شود..

```
    public void evolve(){
        if
(EvolutionUtils.benchmark.itSinceLastChange<=EvolutionUtils.pars.localSearchNum){
            if(EvolutionUtils.pars.localSearchType==1)
                randomLocalSearch();
            else

ImprovedLocalSearch(EvolutionUtils.pars.ssInit,EvolutionUtils.pars.stepMax,EvolutionUtil
s.pars.acc);
        }else{
            if(landscapeType==0)
                DEEvolve();
            if(landscapeType==1)
                hillClimb();
            if(landscapeType==2)
                DEEvolve();
            if(landscapeType==3)
                DEEvolve();
        }
    }
```

- روند DEEvolve : با اجراي اين روند موجودات فعال داخل سلول توسط الگوريتم تكامل تفاضلي تكامل داده مي‌شوند.

```
    public void DEEvolve(){
        double F=currentState.F;
        double Cr=currentState.Cr;

        ArrayList<Chromosome> popPool=new ArrayList<Chromosome>();
        popPool.addAll(currentState.population);

        if(EvolutionUtils.pars.CrFuzzyEnable || EvolutionUtils.pars.FFuzzyEnable){
            double NCl=0,NCf=0;
            NCl=1-(1+Cl)*Math.exp(-Cl);
            NCf=1-(1+Cf)*Math.exp(-Cf);
```

```
            EvolutionUtils.pars.myFIS.setVariable("d21", NCl);
            EvolutionUtils.pars.myFIS.setVariable("d22", NCf);
            EvolutionUtils.pars.myFIS.evaluate();
            if(EvolutionUtils.pars.CrFuzzyEnable)
                Cr=EvolutionUtils.pars.myFIS.getVariable("Cr").defuzzify();
            if(EvolutionUtils.pars.FFuzzyEnable)
                F=EvolutionUtils.pars.myFIS.getVariable("F").defuzzify();
        }

        Cl=0;
        Cf=0;
        if(EvolutionUtils.pars.CrLearningEnable){
            CrInd=EvolutionUtils.chooseAction(nextState.CrActionProp);
            Cr=(CrInd+1)*0.1;
        }
        if(EvolutionUtils.pars.FLearningEnable){
            FInd=EvolutionUtils.chooseAction(nextState.FActionProp);
            F=(FInd+1)*0.1;
        }

        int winNum=0;
        for(int i=0;i<currentState.population.size();i++){
            Chromosome x1,x2,x3,x4,xb,res,currentCh;
            currentCh=popPool.get(i);
            x1=currentCh;
            x2=popPool.get(EvolutionUtils.rnd.nextInt(popPool.size()));
            x3=popPool.get(EvolutionUtils.rnd.nextInt(popPool.size()));
            x4=popPool.get(EvolutionUtils.rnd.nextInt(popPool.size()));

            xb=localBest.getAliveClone();
            res=DEOperator(F,xb,x1,x2,x3,x4);
            int jRand;
            jRand=EvolutionUtils.rnd.nextInt(res.gens.length);

            for(int j=0;j<res.gens.length;j++){
                if(EvolutionUtils.rnd.nextDouble()>Cr && j!=jRand){
                    res.gens[j]=currentCh.gens[j];
                }
            }


clipToBound(res,EvolutionUtils.benchmark.minCoordinate,EvolutionUtils.benchmark.maxCoord
inate);
            res.fitness=eval(res);
            if(res.fitness>=currentCh.fitness){
                if(EvolutionUtils.pars.CrFuzzyEnable ||
EvolutionUtils.pars.FFuzzyEnable){
                    Cf+=(res.fitness-currentCh.fitness)*(res.fitness-currentCh.fitness);
                    for(int k=0;k<res.gens.length;k++)
                        Cl+=(res.gens[k]-currentCh.gens[k])*(res.gens[k]-
currentCh.gens[k]);
                }
                addNewInd(res);
            }else{
                addInd(currentCh,false);
            }

            if(res.fitness>currentCh.fitness)
                winNum++;
        }

        if(EvolutionUtils.pars.CrFuzzyEnable || EvolutionUtils.pars.FFuzzyEnable){
            Cf=Math.sqrt(Cf/currentState.population.size());
            Cl=Math.sqrt(Cl/currentState.population.size());
        }

        if(EvolutionUtils.pars.CrLearningEnable)
            EvolutionUtils.SLRI(nextState.CrActionProp,CrInd, EvolutionUtils.pars.aCr,
1-(((double)winNum)/(double)currentState.population.size()));
        if(EvolutionUtils.pars.FLearningEnable)
            EvolutionUtils.SLRI(nextState.FActionProp,FInd, EvolutionUtils.pars.aF, 1-
(((double)winNum)/(double)currentState.population.size()));
```

```
    }
```

- روند DEOperator : اين روند در الگوريتم تكامل تفاضلي مورد استفاده قرار مي‌گيرد و وظيفه آن توليد بردار كانديد است. اين روند با دريافت موجودات انتخاب شده، بردار كانديد را بر اساس مدل تكاملي مشخص شده مي‌سازد.

```
    public Chromosome DEOperator(double F,Chromosome xb,Chromosome x1,Chromosome
x2,Chromosome x3,Chromosome x4){
        Chromosome newCh=new Chromosome(xb.gens.length);
        for(int i=0;i<newCh.gens.length;i++){
            if(EvolutionUtils.pars.DEScheme==1){
                newCh.gens[i]=x1.gens[i]+F*(x2.gens[i]-x3.gens[i]);
            }else{
            if(EvolutionUtils.pars.DEScheme==2){
                newCh.gens[i]=x1.gens[i]+F*(xb.gens[i]-x2.gens[i]);
            }else{
            if(EvolutionUtils.pars.DEScheme==3){
                newCh.gens[i]=xb.gens[i]+EvolutionUtils.rnd.nextDouble()*(x1.gens[i]-
x3.gens[i])+EvolutionUtils.rnd.nextDouble()*(x2.gens[i]-x4.gens[i]);
            }else{
            if(EvolutionUtils.pars.DEScheme==4){
                newCh.gens[i]=x1.gens[i]+F*(xb.gens[i]-
x1.gens[i])+EvolutionUtils.rnd.nextDouble()*(x2.gens[i]-x3.gens[i]);
            }
            }
            }
            }

        }
        if(EvolutionUtils.rnd.nextDouble()<EvolutionUtils.pars.mutProb){
            int ttt=EvolutionUtils.rnd.nextInt(newCh.gens.length);
            newCh.gens[ttt]=newCh.gens[ttt]+EvolutionUtils.rnd.nextGaussian();
        }
        return newCh;
    }
```

- روند ImprovedLocalSearch : با اجراي اين روند، جستجوي محلي بهبوديافته حول حافظه بهترين موجود انجام مي‌گيرد.

```
    public void ImprovedLocalSearch(double sinit,double maxStep,double acc){

        for(int i=0;i<currentState.population.size()-1;i++)
            addInd(currentState.population.get(i),false);

        Chromosome currentCh;
        currentCh=currentState.cellMem.getAliveClone();
        Chromosome newCh;
        newCh=new Chromosome(currentCh.gens.length);

        int[] mutDir;
        int[] mutStepNum;
        double[] mutStep;

        mutDir=new int[currentCh.gens.length];
        mutStep=new double[currentCh.gens.length];
        mutStepNum=new int[currentCh.gens.length];
        for(int k=0;k<mutDir.length;k++){
            if(EvolutionUtils.rnd.nextBoolean())
                mutDir[k]=-1;
            else
                mutDir[k]=1;
            mutStep[k]=sinit;
            mutStepNum[k]=1;
            newCh.gens[k]=currentCh.gens[k];
        }
        currentCh.fitness=EvolutionUtils.benchmark.evalDummy(currentCh);
        newCh.fitness=currentCh.fitness;
        boolean isBetter;
```

```
        boolean changeStep;
        do{
            isBetter=false;
            changeStep=false;
            for(int k=0;k<newCh.gens.length;k++){
                double tmpFitness;
                if(mutStepNum[k]>maxStep)
                    continue;
                newCh.gens[k]=newCh.gens[k]+mutDir[k]*mutStep[k];
                tmpFitness=eval(newCh);
                if(tmpFitness<newCh.fitness){
                    newCh.gens[k]=newCh.gens[k]-mutDir[k]*mutStep[k];
                    mutDir[k]=mutDir[k]*-1;
                    newCh.gens[k]=newCh.gens[k]+mutDir[k]*mutStep[k];
                    tmpFitness=eval(newCh);
                    if(tmpFitness<newCh.fitness){
                        newCh.gens[k]=newCh.gens[k]-mutDir[k]*mutStep[k];
                        mutDir[k]=mutDir[k]*-1;
                        if(mutStepNum[k]<=maxStep){
                            mutStep[k]=sinit*Math.pow(acc,mutStepNum[k]);
                            changeStep=true;
                            mutStepNum[k]=mutStepNum[k]+1;
                        }
                    }else{
                        newCh.fitness=tmpFitness;
                        isBetter=true;
                    }
                }else{
                    newCh.fitness=tmpFitness;
                    isBetter=true;
                }
            }
        }while(isBetter || changeStep);

clipToBound(newCh,EvolutionUtils.benchmark.minCoordinate,EvolutionUtils.benchmark.maxCoo
rdinate);
        newCh.fitness=eval(newCh);
        if(newCh.fitness>=currentCh.fitness){
            addNewInd(newCh);
        }else{
            addInd(currentState.population.get(currentState.population.size()-1),false);
        }
    }
```

- روند randomLocalSearch : با اجراي اين روند، جستجوي محلي ساده و تصادفي در اطراف حافظه بهترين موجود انجام مي‌گيرد.

```
    public void randomLocalSearch(){
        ArrayList<Chromosome> popPool=new ArrayList<Chromosome>();
        popPool.addAll(currentState.population);
        for(int i=0;i<currentState.population.size();i++){
            Chromosome currentCh;
            currentCh=popPool.get(i);
            double sum;
            Chromosome newCh;
            newCh=new Chromosome(currentCh.gens.length);
            sum=0;
            for(int j=0;j<newCh.gens.length;j++){
                newCh.gens[j]=EvolutionUtils.rnd.nextDouble()-0.5;
                sum=sum+newCh.gens[j]*newCh.gens[j];
            }
            sum=Math.sqrt(sum);
            double r;
            r=EvolutionUtils.rnd.nextDouble();
            for(int j=0;j<currentCh.gens.length;j++){
                newCh.gens[j]=(newCh.gens[j]/sum)*r+localBest.gens[j];
            }

clipToBound(newCh,EvolutionUtils.benchmark.minCoordinate,EvolutionUtils.benchmark.maxCoo
rdinate);
            newCh.fitness=eval(newCh);
            if(currentCh.fitness>=newCh.fitness)
                addInd(currentCh,false);
```

```
            else
                addNewInd(newCh);
        }
    }
```

- روند `hillClimb` : با اجراي اين روند تكامل با استفاده از الگوريتم تپه‌نوردي انجام خواهد شد.

```
    public void hillClimb(){
        double mut=EvolutionUtils.pars.hillMut;
        for(int i=0;i<currentState.population.size()-1;i++)
            addInd(currentState.population.get(i),false);
        if(currentState.population.size()>0){
            Chromosome newCh;
            newCh=new Chromosome(currentState.cellMem.gens.length);
            System.arraycopy(currentState.cellMem.gens, 0, newCh.gens, 0,
newCh.gens.length);
            int mGen=EvolutionUtils.rnd.nextInt(newCh.gens.length);
            newCh.gens[mGen]=newCh.gens[mGen]+EvolutionUtils.rnd.nextGaussian()*mut;

clipToBound(newCh,EvolutionUtils.benchmark.minCoordinate,EvolutionUtils.benchmark.maxCoo
rdinate);
            newCh.fitness=eval(newCh);
            if(newCh.fitness>=currentState.cellMem.fitness){
                addNewInd(newCh);
            }else{
                addInd(currentState.population.get(currentState.population.size()-
1),false);
            }
        }
    }
```

- روند `landscapeTypeDetect` : اين روند نوع سلول را براي استفاده در نسخه كوه‌نورد مشخص مي‌كند.

```
    public void landscapeTypeDetect(){
        int obsRange=6;
        nextState.theta=EvolutionUtils.pars.thetaInit;

if(EvolutionUtils.benchmark.itSinceLastChange>EvolutionUtils.pars.localSearchNum){
            if(currentState.population.isEmpty())
                return;

            if(lastHistAdd!=EvolutionUtils.benchmark.itSinceLastChange-1)
                cMemHist.clear();

            if(cMemHist.size()==obsRange)
                cMemHist.removeFirst();

            cMemHist.add(currentBest.fitness);
            lastHistAdd=EvolutionUtils.benchmark.itSinceLastChange;


            if((cMemHist.size()==obsRange && bestNCell==this && cMemHist.getLast()-
cMemHist.getFirst()<0.1)){
                landscapeType=1;
            }else{
                if(bestNCell!=this && bestNCell.landscapeType==1){
                    landscapeType=2;
                    nextState.theta=EvolutionUtils.pars.thetaInit/2;
                }else{
                    if(bestNCell!=this && bestNCell.landscapeType==2){
                        landscapeType=3;
                        nextState.theta=EvolutionUtils.pars.thetaInit/2;
                    }
                }
            }
        }
    }
```

- روند controlDensity : با اجراي اين روند، با توجه به آستانه ظرفيت سلول جاري، موجودات اضافي به يک محل تصادفي در فضا منتقل مي‌شوند. اين موجودات به غير فعال شده تا زماني که به مقصد برسند.

```
    public void controlDensity(){
        int theta=currentState.theta;

        if(EvolutionUtils.pars.alpinistEnable==false &&
EvolutionUtils.benchmark.itSinceLastChange>=EvolutionUtils.pars.localSearchNum &&
EvolutionUtils.benchmark.itSinceLastChange<=EvolutionUtils.pars.localSearchNum)
            theta=EvolutionUtils.pars.thetaInit/2;


        Collections.sort((List)nextState.population);
        if(nextState.population.size()>theta){
            int nOD=EvolutionUtils.benchmark.genNum();
            while(nextState.population.size()>theta){
                Chromosome ch;
                ch=new
Chromosome(nOD,EvolutionUtils.benchmark.minCoordinate,EvolutionUtils.benchmark.maxCoordi
nate);
                ch.fitness=eval(ch);
                addNewInd(ch);
                nextState.population.remove(nextState.population.size()-1);
            }
        }
    }
```

- روند refreshMems : با اجراي اين روند، حافظه‌هاي سلول در صورت نياز بروزرساني مي‌شوند.

```
    public void refreshMems(){
        if(!currentState.cellMem.isEmpty())

currentState.cellMem.fitness=EvolutionUtils.benchmark.evalDummy(currentState.cellMem.gen
s);
        if(currentState.cellMem.periodNum!=EvolutionUtils.benchmark.periodNum){
            if(!currentState.cellMem.isEmpty())

currentState.cellMem.fitness=EvolutionUtils.benchmark.evalDummy(currentState.cellMem.gen
s);
        }
        currentState.cellMem.periodNum=EvolutionUtils.benchmark.periodNum;
    }
```

- روند setCurrentBest : با اجراي اين روند، بهترين موجود فعال در سلول مشخص مي‌شود و در خصوصيت مربوطه نگهداري مي‌شود.

```
    public void setCurrentBest(){
        Chromosome best=null;
        for(int i=0;i<currentState.population.size();i++){
            Chromosome tmp;
            tmp=currentState.population.get(i);
            if(best==null || tmp.fitness>=best.fitness)
                best=tmp;
        }
        if(currentBest==null)
            currentBest=new ChromosomeMem();
        currentBest.update(best);
    }
```

- روند cMemUpdate : با اجراي اين روند، بهترين موجود جاري با حافظه بهترين موجود مقايسه و در صورت برتري، جايگزين آن مي‌شود.

```
    public void cMemUpdate(){
        refreshMems();
```

```
        if(currentBest!=null){
            if(currentState.cellMem.isEmpty()){
                currentState.cellMem.update(currentBest);
            }else{
                if(currentBest.fitness>=currentState.cellMem.fitness)
                    currentState.cellMem.update(currentBest);
            }
        }
    }
```

- روند setLocalBest : با اجراي اين روند، بهترين موجود در همسايگي محاسبه و در خصوصيت مربوطه نگهداري مي‌شود.

```
    public void setLocalBest(){
        bestNCell=this;
        if(localBest==null)
            localBest=new ChromosomeMem();
        localBest.update(currentState.cellMem);

        for(int i=0;i<neighbors.length;i++){
            if(!neighbors[i].currentState.cellMem.isEmpty()){

if(neighbors[i].currentState.cellMem.periodNum!=EvolutionUtils.benchmark.periodNum){

neighbors[i].currentState.cellMem.fitness=EvolutionUtils.benchmark.evalDummy(neighbors[i
].currentState.cellMem.gens);

neighbors[i].currentState.cellMem.periodNum=EvolutionUtils.benchmark.periodNum;
                }

                if(localBest.isEmpty()){
                    localBest.update(neighbors[i].currentState.cellMem);
                    bestNCell=neighbors[i];
                }else{
                    if(neighbors[i].currentState.cellMem.fitness>=localBest.fitness){
                        localBest.update(neighbors[i].currentState.cellMem);
                        bestNCell=neighbors[i];
                    }
                }
            }
        }
    }
```

- روند addNewInd : از طريق اين روند مي‌توان موجود جديدي را به سلول اضافه نمود. اگر مكان اين موجود در سلول جاري نباشد، مقدمات انتقال آن به سلول مقصد را نيز فراهم مي‌كند.

```
    public void addNewInd(Chromosome ind){

clipToBound(ind,EvolutionUtils.benchmark.minCoordinate,EvolutionUtils.benchmark.maxCoord
inate);
        int[] indPartition=EvolutionUtils.getPartition(ind,EvolutionUtils.benchmark);
        if(EvolutionUtils.samePartition(indPartition,partition)){
            addInd(ind,false);
        }else{

ind.desCell=EvolutionUtils.getLinearPos(indPartition,EvolutionUtils.benchmark);
            ind.hop=getDist(partition,indPartition);
            addInd(ind,false);
        }
    }
```

- روند addInd : با استفاده از اين روند مي‌توان موجودي را كه مطمئن هستيم كه متعلق به سلول جاري است را به آن اضافه كنيم.

```
    public void addInd(Chromosome ind,boolean isCurrent){
        if(ind.hop>0){
            EvolutionUtils.inactivePopulation.add(ind);
        }else{
            EvolutionUtils.updateCandidates.add(this);
```

```
            if(isCurrent)
                currentState.addIndivisual(ind);
            else{
                nextState.addIndivisual(ind);
            }
        }
    }
```

- روند `eval` : با استفاده از اين روند شايستگي يک موجود يا يک مکان را محاسبه مي‌کنيم.

```
    public double eval(Chromosome ch){
        double tmp;
        tmp=eval(ch.gens);
        return tmp;
    }

    public double eval(double[] gens){
        double tmp;
        tmp=EvolutionUtils.benchmark.eval(gens);
        return tmp;
    }
```

- روند `getDist` : اين روند با دريافت مکان دو سلول مبدا و مقصد، فاصله اين دو را با توجه به اندازه همسايگي برمي‌گرداند.

```
    public int getDist(int[] src,int[] des){
        int max=0;
        for(int i=0;i<src.length;i++){
            if(Math.abs(src[i]-des[i])>max)
                max=Math.abs(src[i]-des[i]);
        }
        return (int)Math.ceil(((double)max)/EvolutionUtils.pars.neighborhoodSize);
    }
```

- روند `clipToBound` : اين روند با دريافت يک موجود، تمامي ابعاد آن را که از محدوده مجاز خارج شده‌اند را به مقادير لبه برمي‌گرداند.

```
    public void clipToBound(Chromosome ch,double[] min,double[] max){
        for(int i=0;i<ch.gens.length;i++){
            if(ch.gens[i]<min[i])
                ch.gens[i]=min[i];
            if(ch.gens[i]>max[i])
                ch.gens[i]=max[i];
        }
    }
```

- روند `compareTo` : اين روند با دريافت يک سلول ديگر و مقايسه آن با سلول جاري، نسبت اين دو سلول به هم را مشخص مي‌کند. اين مقايسه از روي آدرس دو سلول انجام مي‌گيرد.

```
    public int compareTo(Object ob1){
        for(int i=0;i<((Cell)ob1).partition.length;i++){
            if(((Cell)ob1).partition[i]<this.partition[i] )
                return 1;
            if(((Cell)ob1).partition[i]>this.partition[i] )
                return -1;
        }
        return 0;
    }
```

- روند `equals` : اين روند با دريافت يک سلول ديگر و مقايسه آن با سلول جاري، مشخص مي‌کند که آيا اين دو سلول يکي هستند يا خير. اين مقايسه از روي آدرس دو سلول انجام مي‌گيرد.

```
    public boolean equals(Object ob1){
        for(int i=0;i<((Cell)ob1).partition.length;i++){
```

```
            if(((Cell)ob1).partition[i]!=this.partition[i] )
                return false;
        }
        return true;
    }
}
```

۹-۵-۹ کلاس CA

این کلاس اصلي پيادهسازي مدل اتوماتاي سلولي است و آرايه سلولها نيز در همين کلاس نگهداري ميشود. تمامي روندهاي کلي اتوماتاي سلولي در اين کلاس تعريف شده است. مجموعه موجودات غيرفعال و مجموعه سلولهايي که در آنها موجود فعال وجود دارد را نيز در خصوصيات اين کلاس نگهداري ميشوند.

```
package cellularevolution;
import java.util.*;

public class CA {
    public Cell[] cells;
    public int numberOfCells;
    public ArrayList<Chromosome> inactivePopulation;
    TreeSet<Cell> updateCandidates;
```

- روند CA : اين روند سازنده کلاس جاري ميباشد و وظيفه آن مقداردهي اوليه و آمادهسازي خصوصيات کلاس ميباشد. در اين روند مجموعه سلولها ساخته و راه اندازي اوليه ميشوند. همسايگي تمامي سلولها محاسبه ميشود و اتوماتاي سلولي را آماده به کار ميکند.

```
    public CA(){
        updateCandidates=new TreeSet<Cell>();
        inactivePopulation=new ArrayList<Chromosome>();
        EvolutionUtils.updateCandidates=updateCandidates;
        EvolutionUtils.inactivePopulation=inactivePopulation;

        EvolutionUtils.benchmark.itSinceLastChange=0;
        EvolutionUtils.benchmark.isChanged=false;
        EvolutionUtils.benchmark.periodNum=0;


numberOfCells=(int)Math.pow(EvolutionUtils.pars.partitionsNum,EvolutionUtils.pars.dimens
ionsNum);
        cells=new Cell[numberOfCells];

        for(int i=0;i<numberOfCells;i++)
            cells[i]=new Cell(getPartition(i));

        for(int i=0;i<numberOfCells;i++){
            cells[i].neighbors=getNeighbors(i,EvolutionUtils.pars.neighborhoodSize);
        }

        if(EvolutionUtils.pars.CrNLearningEnable
||EvolutionUtils.pars.FNLearningEnable){
            for(int i=0;i<numberOfCells;i++){
                cells[i].adjacentNeighbors=getNeighbors(i,1);
            }
        }

        for(int i=0;i<numberOfCells;i++){
            cells[i].init();
        }
```

```
        initEvolution();
    }
```

- روند initEvolution : اين روند، چند روند ديگر را كه براي توليد اوليه جمعيت، پارامترها و حافظه‌هاي بهترين موجودات هستند را اجرا مي‌نمايد.

```
    public void initEvolution(){
        initPars();
        initPopulation();
        initBests();
    }
```

- روند reInit : با اجراي اين روند، تمامي مقادير به حالت اوليه در مي‌آيند و اتوماتاي سلولي از نو راه‌اندازي مي‌شود.

```
    public void reInit(){
        updateCandidates.clear();
        inactivePopulation.clear();
        EvolutionUtils.benchmark.itSinceLastChange=0;
        EvolutionUtils.benchmark.isChanged=false;
        EvolutionUtils.benchmark.periodNum=0;

numberOfCells=(int)Math.pow(EvolutionUtils.pars.partitionsNum,EvolutionUtils.pars.dimens
ionsNum);
        for(int i=0;i<numberOfCells;i++)
            cells[i].init();
        initEvolution();
    }
```

- روند initPopulation : با اجراي اين روند، موجوداتي به صورت تصادفي ساخته مي‌شوند و به سلول مربوطه اضافه مي‌گردند.

```
    public void initPopulation(){
        for(int i=0;i<EvolutionUtils.pars.populationSize;i++){
            Chromosome ch=new Chromosome(EvolutionUtils.benchmark.dimensionsNum);
            for(int j=0;j<EvolutionUtils.benchmark.dimensionsNum;j++){

ch.gens[j]=EvolutionUtils.toBound(EvolutionUtils.rnd.nextDouble(),EvolutionUtils.benchma
rk.minCoordinate[j],EvolutionUtils.benchmark.maxCoordinate[j]);
            }
            ch.fitness=EvolutionUtils.benchmark.eval(ch);

cells[EvolutionUtils.getLinearPos(EvolutionUtils.getPartition(ch,EvolutionUtils.benchmar
k),EvolutionUtils.benchmark)].addInd(ch,true);
        }
    }
```

- روند initPars : با اجراي اين روند، مقادير پارامترهاي سلول‌ها و مجموعه احتمال اعمال همگي مقداردهي اوليه مي‌شوند.

```
    public void initPars(){
        for(int i=0;i<cells.length;i++){
            cells[i].currentState.theta=EvolutionUtils.pars.thetaInit;
            cells[i].currentState.F=EvolutionUtils.pars.FInit;
            cells[i].currentState.Cr=EvolutionUtils.pars.CrInit;
            for(int j=0;j<cells[i].currentState.FActionProp.length;j++)

cells[i].currentState.FActionProp[j]=1.0/cells[i].currentState.FActionProp.length;
            for(int j=0;j<cells[i].currentState.CrActionProp.length;j++)

cells[i].currentState.CrActionProp[j]=1.0/cells[i].currentState.CrActionProp.length;
        }
    }
```

- روند initBests : با اجراي اين روند، مقادير حافظه‌هاي بهترين موجود محاسبه و مقداردهي مي‌شود.

```
    public void initBests(){
        Iterator<Cell> it;
        it=updateCandidates.iterator();
        while(it.hasNext()){
            Cell c;
            c=it.next();
            c.setCurrentBest();
            c.currentState.cellMem.update(c.currentBest);
        }
        it=updateCandidates.iterator();
        while(it.hasNext()){
            Cell c;
            c=it.next();
            c.setLocalBest();
        }
    }
```

- روند update : با اجراي اين روند، يک مرتبه بروزرساني کل اتوماتاي سلولي به صورت همزمان انجام مي‌شود. در صورت تشخيص وقوع تغيير در محيط نيز، کارهاي لازم از قبيل بروزرساني مقادير شايستگي ها و حافظه‌ها انجام مي‌گيرد.

```
    public void update(){

EvolutionUtils.benchmark.itSinceLastChange=EvolutionUtils.benchmark.itSinceLastChange+1;
        Iterator<Cell> it;
        ArrayList<Cell> actionUpdateCands=new ArrayList<Cell>();


        TreeSet<Cell> cellForUpdate=(TreeSet<Cell>)updateCandidates.clone();
        TreeSet<Cell> cellForNeighborUpdate=(TreeSet<Cell>)updateCandidates.clone();
        updateCandidates.clear();


        it=cellForUpdate.iterator();
        while(it.hasNext()){
            Cell c;
            c=it.next();
            updateCandidates.add(c);
            if(EvolutionUtils.benchmark.isChanged){
                c.nextState=c.currentState;
            }else{
                c.update();
            }
            actionUpdateCands.add(c);
        }


        movInactives(cellForUpdate);


        it=cellForUpdate.iterator();
        while(it.hasNext()){
            Cell c;
            c=it.next();
            c.currentState=c.nextState;
            c.nextState=null;
            if(c.currentState.population.isEmpty()){
                updateCandidates.remove(c);
            }
        }


        if(EvolutionUtils.benchmark.isChanged){
            for(int j=0;j<inactivePopulation.size();j++){
                Chromosome ch;
                ch=inactivePopulation.get(j);
                ch.fitness=EvolutionUtils.benchmark.eval(ch.gens);
            }
            it=updateCandidates.iterator();
```

```
            while(it.hasNext()){
                Cell c;
                c=it.next();
                for(int j=0;j<c.currentState.population.size();j++){
                    Chromosome ch;
                    ch=c.currentState.population.get(j);
                    ch.fitness=EvolutionUtils.benchmark.eval(ch.gens);
                }
            }
        }

        it=cellForUpdate.iterator();
        while(it.hasNext()){
            Cell c;
            c=it.next();
            c.setCurrentBest();
            c.cMemUpdate();
        }

        if(EvolutionUtils.benchmark.isChanged){
            EvolutionUtils.benchmark.itSinceLastChange=0;
            EvolutionUtils.benchmark.isChanged=false;
        }

        if(EvolutionUtils.pars.CrNLearningEnable||EvolutionUtils.pars.FNLearningEnable){
            it=cellForNeighborUpdate.iterator();
            while(it.hasNext()){
                Cell myCell=it.next();
                Cell[] nn1=myCell.adjacentNeighbors;
                for(int q=0;q<nn1.length;q++) {
                    Cell cc=nn1[q];
                    if(EvolutionUtils.pars.FNLearningEnable){
                        if(cc.FInd==-1)

cc.FInd=EvolutionUtils.chooseAction(cc.currentState.FActionProp);
                        if(cc.FInd==myCell.FInd)

EvolutionUtils.TLRPReward(cc.currentState.FActionProp,cc.FInd,EvolutionUtils.pars.aFN);
                        else

EvolutionUtils.TLRPPunish(cc.currentState.FActionProp,cc.FInd,EvolutionUtils.pars.bFN);
                        cc.FInd=EvolutionUtils.chooseAction(cc.currentState.FActionProp);
                    }

                    if(EvolutionUtils.pars.CrNLearningEnable){
                        if(cc.CrInd==-1)

cc.CrInd=EvolutionUtils.chooseAction(cc.currentState.CrActionProp);
                        if(cc.CrInd==myCell.CrInd)

EvolutionUtils.TLRPReward(cc.currentState.CrActionProp,cc.CrInd,EvolutionUtils.pars.aCrN
);
                        else

EvolutionUtils.TLRPPunish(cc.currentState.CrActionProp,cc.CrInd,EvolutionUtils.pars.bCrN
);

cc.CrInd=EvolutionUtils.chooseAction(cc.currentState.CrActionProp);
                    }
                }
            }
        }
    }
```

- روند movInactives : با اجراي اين روند، مجموعه موجودات غيرفعال بررسي مي‌شوند و موجوداتي كه به مقصد رسيده‌اند را فعال مي‌كند. اين روند، در هر بار بروزرساني كل اتوماتاي سلولي يک بار اجرا مي‌شود.

```
    public void movInactives(TreeSet<Cell> cellForUpdate){
        for(int i=0;i<inactivePopulation.size();i++){
```

```
            Chromosome tmpCh=inactivePopulation.get(i);
            if(tmpCh.hop==1){
                tmpCh.hop=-1;
                if(cells[tmpCh.desCell].nextState==null){
                    cells[tmpCh.desCell].initNextState();
                }
                cells[tmpCh.desCell].addInd(tmpCh, false);
                cellForUpdate.add(cells[tmpCh.desCell]);
                tmpCh.desCell=-1;
                inactivePopulation.remove(i);
            }else{
                tmpCh.hop=tmpCh.hop-1;
            }
        }
    }
```

- روند boroBinim : از اين روند بازگشتي جهت محاسبه همسايگي هاي يک سلول استفاده مي‌شود.

```
    public void boroBinim(int i,ArrayList<Cell> l,int cellNum,int neighborhoodSize){
        int tmp;
        if(i>=cells[cellNum].partition.length){

tmp=EvolutionUtils.getLinearPos(cells[cellNum].partition,EvolutionUtils.benchmark);
            if(tmp>=0 && tmp!=cellNum){
                l.add(cells[tmp]);
            }
            return;
        }
        for(int k=-neighborhoodSize;k<=neighborhoodSize;k++){
            if(cells[cellNum].partition[i]+k<EvolutionUtils.pars.partitionsNum &&
cells[cellNum].partition[i]+k>=0){
                cells[cellNum].partition[i]=cells[cellNum].partition[i]+k;
                boroBinim(i+1,l,cellNum,neighborhoodSize);
                cells[cellNum].partition[i]=cells[cellNum].partition[i]-k;
            }
        }
    }
```

- روند getNeighbors : با اجراي اين روند، سلول‌هاي همسايه سلول ورودي محاسبه و نگهداري مي‌شوند.

```
    public Cell[] getNeighbors(int cellNum,int neighborhoodSize){
        ArrayList<Cell> nTmp =new
ArrayList<Cell>((int)Math.pow(neighborhoodSize*2+1,EvolutionUtils.benchmark.dimensionsNu
m));
        boroBinim(0,nTmp,cellNum,neighborhoodSize);
        Cell[] neighbors=new Cell[nTmp.size()];
        return nTmp.toArray(neighbors);
    }
```

- روند getPartition : اين روند، آدرس چند بعدي يک سلول را محاسبه مي‌کند و به عنوان خروجي برمي‌گرداند.

```
    public int[] getPartition(int cellNum){
        int[] partition=new int[EvolutionUtils.benchmark.dimensionsNum];
        int
order=(int)Math.pow(EvolutionUtils.pars.partitionsNum,EvolutionUtils.benchmark.dimension
sNum-1);
        for(int i=0;i<EvolutionUtils.benchmark.dimensionsNum;i++){
            partition[i]=cellNum/order;
            cellNum=cellNum%order;
            order=order/EvolutionUtils.pars.partitionsNum;
        }
        return partition;
    }
}
```

۹-۵-۱۰ کلاس CellularEvolution

این کلاس اصلي پروژه مي‌باشد که از طريق آن شبيه‌سازي‌ها اجرا مي‌شوند. تابع اصلي نيز در اين کلاس قرار دارد. انجام اجراهاي متوالي نيز بر عهده اين کلاس است.

```
package cellularevolution;

public class CellularEvolution {
    public CA myCA;
```

- روند main : اين روند اصلي کلاس مي‌باشد که براي اجراي پروژه، فراخواني مي‌شود. اين روند اتوماتاي سلولي و تابع محک را به تعداد دفعات تنظيم شده مي‌سازد و اجرا مي نمايد. فايل‌هاي خروجي نيز در اينجا ساخته و تنظيم مي‌شوند.

```
    public static void main(String[] args) {
        int runsNum=100;
        Benchmark benchmark;
        CellularEvolution testCA=null;
        double sum=0;
        for(int i=1;i<=runsNum;i++){
            double err;
            benchmark=new MPBenchmark();
            benchmark.init();
            EvolutionUtils.benchmark=benchmark;
            if(i==1){
                EvolutionUtils.pars.printParams();
                testCA=new CellularEvolution();
            }else{
                testCA.myCA.reInit();
            }
            err=testCA.run(EvolutionUtils.pars.evalsMax);
            System.gc();
            sum=sum+err;
            System.out.println(i+" "+err+" "+sum/i);
        }
    }
```

- روند run : روند main براي اجراي اتوماتاي سلولي ساخته شده به تعداد دفعات شايستگي تعيين شده، از اين روند استفاده مي‌کند.

```
    public double run(long maxEval){
        while(EvolutionUtils.benchmark.evalsNum()<maxEval){
            myCA.update();
        }
        return EvolutionUtils.benchmark.offlineError();
    }
```

- روند CellularEvolution : اين روند سازنده کلاس جاري مي‌باشد که در آن در ابتدا يک اتوماتاي سلولي جديد ساخته مي‌شود.

```
    public CellularEvolution(){
        myCA=new CA();
    }
}
```

Abstract

Many real world optimization problems are dynamic in which the landscape is time dependent and the optimums may change over time such as dynamic economic modeling, dynamic resource scheduling, dynamic vehicle routing. Several evolutionary algorithms, based on Cellular Automata (CA), Cellular Learning Automata (CLA) and Fuzzy Cellular Automata (FCA), are proposed for dynamic environments. All proposed models are based on the idea of search space partitioning.

The first proposed algorithm (CellularDE) which is based on cellular automata uses Differential Evolution (DE) to search each partition and its neighborhood separately. A new scheme for generating candidate vector in DE is proposed which shows superior results in the proposed model compared to conventional schemes. It also eliminates one of the parameters of DE. In order to increase the efficiency of tracking the moving optimums, an improved local search is also proposed which makes a significant improvement in environments with low rate of change frequency. Proposed algorithms are compared with some of the other evolutionary algorithms which have showed the most outstanding results on Moving Peaks Benchmark (MPB). In most dynamic environments, proposed algorithms show superior results compared to other evolutionary algorithms.

Also a modified version of the original algorithm is proposed which is named Alpinist CellularDE. In this algorithm, the search efficiency and usage of resources is improved by detecting the location of peaks and their hillsides. It showed superior results compared to the original version in most experiments.

Moreover, two other versions of the original algorithm are proposed in order to increase the adaptivity of the proposed algorithm to different dynamic environments and also to obviate the necessity for configuration of the parameters of DE. These adaptive versions, which are based on CLA and FCA, configure the values of DE parameters using Reinforcement Learning (RL) and Rule Based Fuzzy Induction System (RBFIS). These algorithms have acceptable performance in adapting to different dynamic environments compared to non-adaptive version.

Keywords: Dynamic Environments, Evolutionary Algorithms, Differential Evolution, Cellular Automata, Cellular Learning Automata, Fuzzy Cellular Automata

Amirkabir University of Technology
(Tehran Polytechnic)

Computer Engineering and Information Technology Department

Submitted in partial fulfillment of the requirements
for the Master of Science degree in
Artificial Intelligence and Robotics

Designing of Evolutionary Algorithms based on Cellular Automata for Dynamic Environments

By
Vahid Noroozi

Supervisor
Professor Mohammad Reza Meybodi

September 2011