به نام خدا

گزارش پژوهش۲ درس شناسایی الگو

یک الگوریتم ژنتیک خودآموز با استفاده از یادگیری تقویتی برای مسئله‌ی زمانبندی انعطاف‌پذیرکارها[۱]

استاد درس:

جناب دکتر کارشناس

نام و نام خانوادگی دانشجو:

امیررضا صدیقین

شماره دانشجویی:

۹۹۳۶۱۴۰۲۴

[۱] A self-learning genetic algorithm based on reinforcement learning for flexible job-shop scheduling problem

چکیده

مسئله‌ی زمانبندی انعطاف‌پذیر کارها (FJSP) یک مسئله‌ی پیچیده و از نوع مسائل NP-hard است. در این گزارش به بررسی یک الگوریتم ژنتیک خود آموز (SLGA) برای حل مسئله‌ی FJSP پرداخته شده است. در ابتدا به بررسی مدل خودآموز و در مرحله‌ی بعد به الگوریتم‌های SARSA و Q-learning پرداخته می‌شود. بعد از آن الگوریتم ارائه شده معرفی می‌گردد و در آخر آزمایش‌ها وپیاده‌سازی‌های انجام شده،شرح داده می‌شود.

۱-مقدمه

برنامه‌ریزی زمانی یکی از کارهای مهم در سیستم‌های تولیدی و صنعت است که باعث استفاده‌ی بهتر از منابع می‌شود. مسئله‌ی برنامه‌ریزی زمانی کارها[۱] (JSP) یک مسئله‌ی پیچیده و از نوع NP-hard است. مسائل JSP سنتی بیان می‌کند که مجموعه‌ای از کارها برای انجام شدن در نظر گرفته شده است. هر کار برای انجام شدن نیاز به یک سری عملگر[۲] دارند که این عملگرها، توسط یک سری ماشین قابل انجام هستند. ممکن است یک ماشین بیش‌تر از یک عملگر انجام دهد، که به آن FJSP (نسخه‌ی انعطاف‌پذی JSP) گویند. همچنین چندین ماشین می‌توانند عملگرهای یکسانی را انجام دهند. FJSP یک مسئله‌ی پیچیده‌تری نسبت به JSP است که هدف در آن، این است که برای هر کار، باید توالی عملگرها حفظ شود،همچنین این عملگرها به ماشین‌ها تخصیص داده شود و این روند در کوتاه ترین زمان همه‌ی کارها اتمام یابند.

با توجه به تحقیقات انجام شده، نشان داده شده است که الگوریتم‌های ژنتیک (GA) برای حل مسائل FJSP بسیار مناسب بوده است. نکته‌ی مهمی که مطرح است، آن است که مؤلفه‌های الگوریتم چگونه تنظیم شوند. دو تا از مؤلفه‌های اصلی الگوریتم‌های ژنتیک، احتمال تقطیع[۳] (P_c) و احتمال جهش[۴] (p_m) می‌باشد. اگر این دو مؤلفه خیلی بزرگ باشند، آن‌گاه به راحتی ممکن است افراد خوب جمعیت از دست بروند و اگر این دو مؤلفه خیلی کم باشند، آن‌گاه افراد جدید به سختی به وجود می‌آیند. تنظیم پارامترها کار سختی است و معمولا مؤلفه‌های الگوریتم‌های ژنتیک ارائه شده برای این مسئله، ثابت هستند یا رویکرد ثابتی دارند. در این گزارش یک الگوریتم خودآموز ژنتیک (SLGA) با روش یادگیری تقویتی (RL) برای مسئله‌ی FJSP ارائه شده است.

---

[۱] Job-shop scheduling problem

[۲] operation

[۳] Crossover probability

[۴] Mutation probability

۲-فرموله‌سازی مسئله

یک مسئله‌ی FJSP دارای مجموعه‌ی h تایی کارها $J = \{j_1, j_2, \dots, j_h\}$، مجموعه‌ی m تایی ماشین‌ها $M = \{m_1, m_2, \dots, m_m\}$ است و کار i ام دارای عملگرهایی است که هر کدام با o_{ij} نشان داده می‌شود که منظور از o_{ij}، j امین عملگر از کار i است. برای انجام هر عملگر o_{ij} توسط ماشین k زمانی نیاز است که آن را با t_{ijk} نشان می‌دهند. جدول ۱ یک مسئله‌ی با ۴ کار و ۳ ماشین را نشان می‌دهد که اعداد موجود t_{ijk} ها را نمایش می‌دهد.

Jobs	Operation	M_1	M_2	M_3
J_1	O_{11}	2	–	**5**
	O_{12}	–	**1**	4
J_2	O_{21}	**2**	3	–
	O_{22}	4	2	**1**
J_3	O_{31}	2	**3**	–
	O_{32}	**2**	–	5
J_4	O_{41}	**4**	–	2
	O_{42}	–	**3**	5

جدول۱ : نمونه‌ای از مسائل FJSP

یک سری فرضیات و محدودیت‌هایی برای مسائل FJSP در نظر گرفته می‌شودکه عبارت است از: ۱-کارها مستقل از هم هستند و t_{ijk} نیز ثابت هستند. ۲- باید توالی عملگرهای هر کار حفظ شوند.(به ترتیب باید باشند و نمی‌شود جابجا شوند). ۳-همه‌ی کارها و ماشین‌ها از همان لحظه‌ی اول در دسترس هستند. ۴- برای هر عملگر، حداقل یک ماشین وجود دارد. ۵- هر ماشین در هر لحظه فقط یک عملگر را می‌تواند انجام دهد. ۶-پردازش هر عملگردر حال پردازش تا لحظه‌ی اتمام، قطع نمی‌شود. ۷-زمان جابجایی کارها، تاخیر ماشین‌ها، اختلالات موجود در ماشین‌ها و ... قابل اغماض هستند.

در این مسئله، هدف آن است که زمان اتمام تمامی عملگرها با حفظ ترتیب به کمترین مقدار خود برسد. تابع هدف به صورت رابطه‌ی ۱ تعریف می‌شود. برای این مسئله، روابط ۱، ۲، ۳، ۴ برای i=۱,۲,...,n و j=۱,۲,...,h و k=۱,۲,...,m برقرار است.

$$Min.C_{Max} = Min\left\{Max.\sum_{i=1}^{n}\sum_{j=1}^{h}(s_{ijk} + t_{ijk})\right\} \qquad (۱)$$

که در آن s_{ijk} زمان شروع o_{ij} توسط ماشین k است.

$$t_{ijk} > 0 \qquad (۲)$$

که نشان‌دهنده‌ی آن است که زمان انجام عملگرها مقداری مثبت است.

$$s_{ijk} + t_{ijk} \leq s_{ij+1k} \qquad (3)$$

که این رابطه نمایش دهنده‌ی محدودیت انجام عملگرها به ترتیب برای یک کار است. متغیر X_{ijk} که تخصیص هر عملگر به یک ماشین را نشان می‌دهد به صورت رابطه‌ی ۴ تعریف می‌شود.

$$X_{ijk} = \begin{cases} 1 & if\ o_{ij}\ assigned\ to\ machine\ k \\ . & otherwise \end{cases} \qquad (4)$$

رابطه‌ی ۵ نشان می‌دهد که حداقل یک ماشین برای انجام o_{ij} وجود دارد.

$$\forall i,j \quad \sum X_{ijk} \geq 1 \qquad (5)$$

و همچنین رابطه‌ی ۶ نشان می‌دهد که هر ماشین حداکثر انجام به انجام تنها یک عملگر در لحظه است.

$$\forall k \quad \sum\sum X_{ijk} = 1 \qquad (6)$$

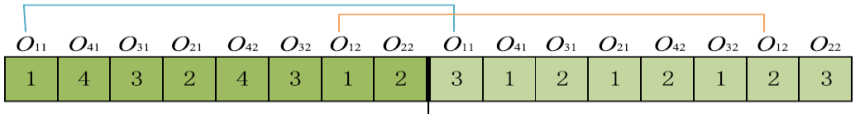
۳- معرفی الگوریتم‌های پایه

۳-۱- الگوریتم ژنتیک

الگوریتم ژنتیک(GA) معروف‌ترین الگوریتم بین الگوریتم‌های بهینه‌سازی برپایه‌ی جمعیت، برای حل مسائل بهینه‌سازی ترکیبی است. در این الگوریتم، راه‌حل‌ها به صورت کروموزوم نمایش داده می‌شود و با استفاده از جمعیت راه‌حل‌ها، تکرار نسل‌های جمعیت و توجه به برازندگی افراد جمعیت، به سمت یک جواب بهینه همگرا می‌شود.

۳-۱-۱- نمایش راه‌حل‌ها

برای نمایش راه‌حل‌ها برای مسائل FJSP، یک رشته‌ی عددی به اندازه‌ی دوبرابر تعداد کل عملگرهای مورد نیاز همه‌ی کارها، در نظر گرفته می‌شود. در نیمه‌ی اول هر رشته، در هر خانه هر عدد، مربوط به کار هر عملگر است و تکرار اعداد کار نیز ترتیب عملگرهای آن کار را نشان می‌دهد. در نیمه‌ی دوم هر رشته نیز، هر جایگاه، عدد ماشینی که وظیفه‌ی انجام عملگر متناظر آن جایگاه در نیمه‌ی اول را نمایش می‌دهد. برای مثال شکل ۱، نمایش یک راه حل از مسئله‌ی واقع در جدول۱ را به تصویر می‌کشد.



شکل ۱: نمایش یک راه‌حل از مسئله‌ی FJSP واقع در جدول ۱، نیمه‌ی اول توالی عملگرها و نیمه‌ی دوم تخصیص ماشین‌ها را نمایش می‌دهد.

نمایش موجود در شکل ۱، نشان می‌دهد ماشین ۱ به ترتیب $o_{۴۱}$، $o_{۲۱}$، $o_{۳۲}$ و ماشین ۲ به ترتیب $o_{۳۱}$، $o_{۴۲}$، $o_{۱۲}$ و ماشین ۳ به ترتیب $o_{۱۱}$ و $o_{۲۲}$ را انجام می‌دهند.

۳-۱-۲-مقداردهی اولیه‌ی جمعیت

مقداردهی اولیه‌ی جمعیت، به دلیل تاثیر آن بر روی زمان همگرایی، مقدار برازندگی جمعیت و کیفیت جواب‌ها، حائز اهمیت است. در این گزارش این عملیات، به صورت تصادفی است ولی اولویت با کارهایی است که بیشترین مقدار زمان را نیاز دارند و ماشین‌هایی که سرعت عمل بیشتری دارند

۳-۱-۳- عملگرهای تغییر الگوریتم ژنتیک

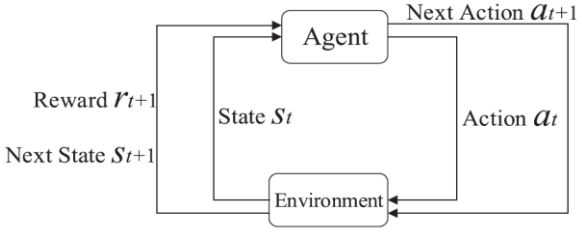
عملیات بازترکیب و جهش، عملگرهای مهم الگوریتم ژنتیک هستند، بازترکیب مسئولیت ادغام ژن‌های والد با هم و به ارث رسیدن ویژگی‌های والدین به فرزند و جهش مسئولیت اکتشاف و حفظ تنوع را بر عهده دارد. بازترکیب با احتمال p_c و جهش با احتمال p_m انجام می‌شود. دراین مسئله، بازترکیب باید یک روش حفظ ترتیب باشد که در ادامه به آن پرداخته می‌شود.

۳-۱-۴- عملیات انتخاب

عملیات انتخاب شامل انتخاب والدین از جمعیت برای عملیات بازترکیب و انتخاب افراد برای جایگزینی در جمعیت جدید است. در این گزارش از یک استراتژی نخبه سالاری برای بهبود کیفیت افراد جمعیت استفاده شده است.

۳-۲- یادگیری تقویتی (RL)

یادگیری تقویتی، یک روش یادگیری است که در آن عامل[۱] مسئله، مدام با محیط تعامل می‌کند و برای هر عملیاتی[۲] که انجام می‌دهد پاداش[۳] یا مجازاتی (پاداش منفی) دریافت می‌کند. این عامل با توجه به این پاداش یا مجازات، تجربه کسب می‌کند و یاد می‌گیرد که در هر حالت[۴] از مسئله چه اقدامی را انجام دهد. در شکل ۲، روند این الگوریتم نمایش داده شده است. هدف الگوریتم، رسیدن به بیشترین پاداش است.



شکل ۲: : روند الگوریتم یادگیری تقویتی

[۱] agent
[۲] action
[۳] reward
[۴] state

SARSA و Q-learning، ۲ نمونه از RLها هستند. در این الگوریتم‌ها مقادیری تحت عنوانQ-valueها وجود دارد که معیاری برای انتخاب عملیات در هر حالت است و متناسب با پاداش‌ها و تجارب به دست می‌آیدی. این مقادیر در جدول Q-value ذخیره می‌شود که به تعداد حالت‌ها، سطر و به تعداد عملیات‌ها ستون دارد. این جدول در ابتدای الگوریتم با مقادیر صفر مقداردهی شده است که به معنای عدم وجود تجربه و اطلاعات در آن حالات و عملیات‌ها است. برای مثال مقدار دهی اولیه‌ی جدول Q-value به صورت جدول ۲ می‌باشد. برای بروزرسانی مقادیر Q_value ها در الگوریتم SARSA و Q-learning به ترتیب رابطه‌ی ۷ و ۸ عمل می‌شود.

$$Q(s_t, a_t) \leftarrow (1-\alpha)Q(s_t, a_t) + \alpha\left(r_{t+1} + \gamma Q(s_{t+1}, a_{t+1})\right) \qquad (۷)$$

$$Q(s_t, a_t) \leftarrow (1-\alpha)Q(s_t, a_t) + \alpha\left(r_{t+1} + \gamma \max_a Q\left(s_{t+1}, a_{t+1}\right)\right) \qquad (۸)$$

که در این روابط منظور از s_t حالت مسئله در زمان t و منظور از a_t عملیات انجام شده در این زمان است. همچنین r_{t+1} مقدار پاداشی است که وقتی عامل در حالت s_t عملیات a_t را انجام می‌دهد است. γ نیز نرخ تخفیف و α مقدار نرخ آموزش است که مقادیری ثابت هستند. همچنین $\max_a Q(s_{t+1}, a_{t+1})$ نمایش دهنده‌ی بیشترین مقدار *Q_value* در جدول برای s_{t+1} بین عملیات‌های مجاز است در حالی که منظور از $Q(s_{t+1}, a_{t+1})$ مقدار *Q_value* ی برای s_{t+1} که متناظر با استراتژی است که انتخاب می‌شود. در *Q_learning* عملکرد بهتری نسبت به *SARSA* دارد ولی تأثیر آموزش کم‌تری دارد در حالی که در الگوریتم *SARSA* تا تأثیر آموزش بیشتری دارد و زودتر به همگرایی می‌رسد ولی ممکن است به راحتی در بهینه‌ی های محلی گیر کند.

$$Q(s_t, a_t) = \begin{array}{c} \\ s_1 \\ s_2 \\ s_3 \\ \vdots \\ s_n \end{array} \begin{array}{c} \begin{array}{ccccc} a_1 & a_2 & a_3 & \cdots & a_m \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix} \end{array}$$

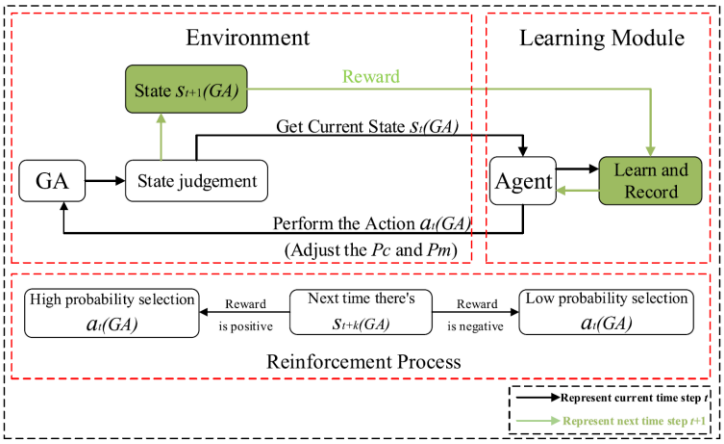
جدول ۲: مقداردهی اولیه‌ی جدول Q-value

۴- الگوریتم پیشنهادی

متناسب با موضوع‌های گفته شده، درمی‌یابیم که باید p_m و p_c نه مقدار خیلی زیاد و نه مقدار خیلی کم بگیرند و متناسب با روند الگوریتم باید مقدار مناسب در نظر گرفته شود. در این روش با استفاده از تجارب قبلی و پیش‌بینی آینده این مؤلفه‌ها تنظیم می‌شود.

۴-۱-ترکیب الگوریتم‌های GA و RL

در شکل ۳ ترکیب الگوریتم GA و RL به تصویر کشیده شده است که در آن محیط متناسب با الگوریتم GA است و فاز آموزش بر پایه‌ی الگوریتم RL است.



شکل ۳: ترکیب الگوریتم‌های GA و RL

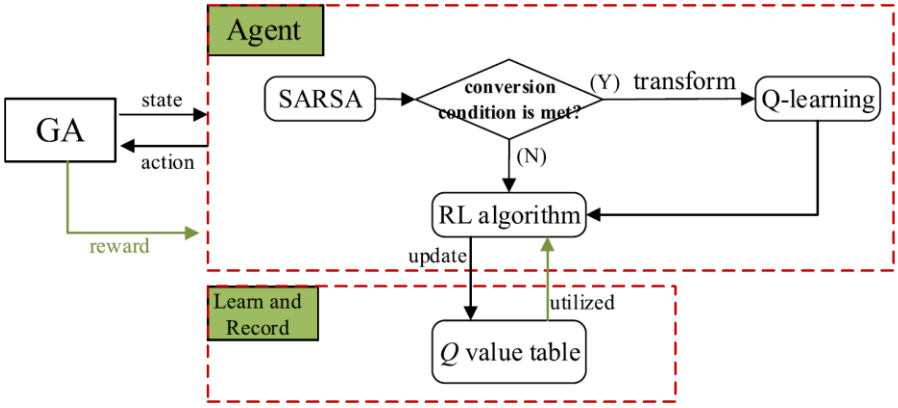
الگوریتم ترکیبی ذکر شده دارای چهار فاز می‌باشد:۱-حالت GA در نسل t ام به دست می‌آید. ۲- متناسب با استراتژی تعیین شده، عملیات که همان بروزرسانی p_m و p_c انجام می‌شود. ۳-بررسی و کارهای مورد نیاز انجام شده و حالت *GA* عوض می‌شود و بازخورد به دست می‌آید. ۴-عملیات و بازخورد به دست ذخیره می‌شود و مقادیر لازم بروزرسانی می‌شوند.

۴-۲-قسمت آموزش

در این روش ارائه شده، دو الگوریتم SARSA و Q-learning با هم ترکیب شده تا از مزایای هر دو استفاده شود. در هر حالت از الگوریتم یکی از دو الگوریتم SARSA یا Q-learning استفاده می‌شود. در ابتدای الگوریتم از SARSA استفاده می‌شود و مقادیر Q-value بروزرسانی می‌شود. در ادامه‌ی الگوریتم با توجه به رابطه‌ی ۹ تصمیم‌گیری می‌شود که از کدام الگوریتم برای بخش آموزش استفاده شود. این روند در شکل ۴ نمایش داده شده است.

$$RL = \begin{cases} sarsa & if\ N_{ti} < \frac{N_{ts} \times N_{ta}}{۲} \\ Q-learning & if\ N_{ti} \geq \frac{N_{ts} \times N_{ta}}{۲} \end{cases} \qquad (۹)$$

که در آن N_{ti} تعداد تکرار نسل فعلی، N_{ts} تعداد کل حالت‌ها و N_{ta} تعداد کل عملیات‌ها است.



شکل ۴:روند بخش آموزش الگوریتم

۳-۴- مجموعه‌ی حالت‌ها

در این الگوریتم سه مؤلفه‌ی میانگین برازندگی‌های جمعیت، تنوع جمعیت و بهترین برازندگی اهمیت بسیاری برای تعیین حالت‌ها دارند که به ترتیب در رابطه‌های ۱۰، ۱۱ و ۱۲ آمده است. این روابط به نسبت جمعیت اولیه‌ی الگوریتم نرمال سازی شده اند.

$$f^* = \frac{\Sigma f(x_i^t)}{\Sigma f(x_i^1)} \qquad (۱۰)$$

$$d^* = \frac{\Sigma \left| f(x_i^t) - \frac{\Sigma f(x_i^t)}{N} \right|}{\Sigma \left| f(x_i^1) - \frac{\Sigma f(x_i^1)}{N} \right|} \qquad (۱۱)$$

$$m^* = \frac{Max f(x_i^t)}{Max f(x_i^1)} \qquad (۱۲)$$

معیاری برای انتخاب حالت تحت عنوان s^* معرفی می‌شود که مطابق رابطه‌ی ۱۳ می‌باشد.

$$s^* = w_1 f^* + w_2 d^* + w_3 m^* \qquad (۱۳)$$

که در آن wها ضرایب ۳ مؤلفه‌ی ذکر شده هستند که باید جمع آن‌ها یک شود و در این گزارش به ترتیب مقادیر ۰.۳۵، ۰.۳۵، ۰.۳ در نظر گرفته شده است.(به دلیل آن که بهترین برازندگی، تاثیر کمتری نسبت به دو مؤلفه‌ی دیگر در یادگیری دارد مقدار آن کمتر از بقیه قرار داده شده است.)

در این گزارش ۲۰ حالت برای حالت‌ها در نظر گرفته شده است که متناسب با s^* که در چه بازه‌ی پنج صدمی از ۰ تا ۱ می‌افتد، تعیین می‌شود به گونه‌ای که اگر $s^* \in [۰,۰.۰۵) \rightarrow s = s(۱)$ و $s^* \in [۰.۰۵, ۰.۱۰) \rightarrow s = s(۲)$ و ... .

۴-۴-مجموعه‌ی عملیات‌ها

برای دو مؤلفه‌ی p_c و p_m نیز مشابه حالت‌ها عمل می‌شود با این تفاوت که p_c باید در بازه‌ی [۰.۹ , ۰.۴] قرار بگیرد و این بازه به ۱۰ قسمت تقسیم شود و p_m نیز باید در بازه‌ی [۰.۰۱ , ۰.۲۱] قرار گیرد و به ۱۰ قسمت تقسیم شود. برای مثال $s^* \in [۰,۰.۰۵) \rightarrow \begin{cases} p_c = random\ number\ from\ [۰.۴,۰.۴۵) \\ p_m = random\ number\ from\ [۰.۰۱,۰.۰۳) \end{cases}$

۴-۵- روش پاداش دهی

برای پاداش عملکرد، اگر میانگین برازندگی جمعیت یا بهترین فرد جمعیت بهبود پیدا کرد مقداری مثبت و اگر بدتر شد مقداری منفی تخصیص داده می‌شود. این مقادیر مطابق روابط ۱۴و۱۵ هستند.

$$r_C = \frac{Maxf(x_i^t) - Maxf(x_i^{t-۱})}{Maxf(x_i^{t-۱})} \qquad (۱٤)$$

$$r_m = \frac{\sum f(x_i^t) - \sum f(x_i^{t-۱})}{\sum f(x_i^{t-۱})} \qquad (۱۵)$$

که منظور از r_c میزان پاداش برای مؤلفه‌ی p_c و منظور از r_m میزان پاداش برای مؤلفه‌ی p_m است.

۴-۶- استراتژی انتخاب عملیات

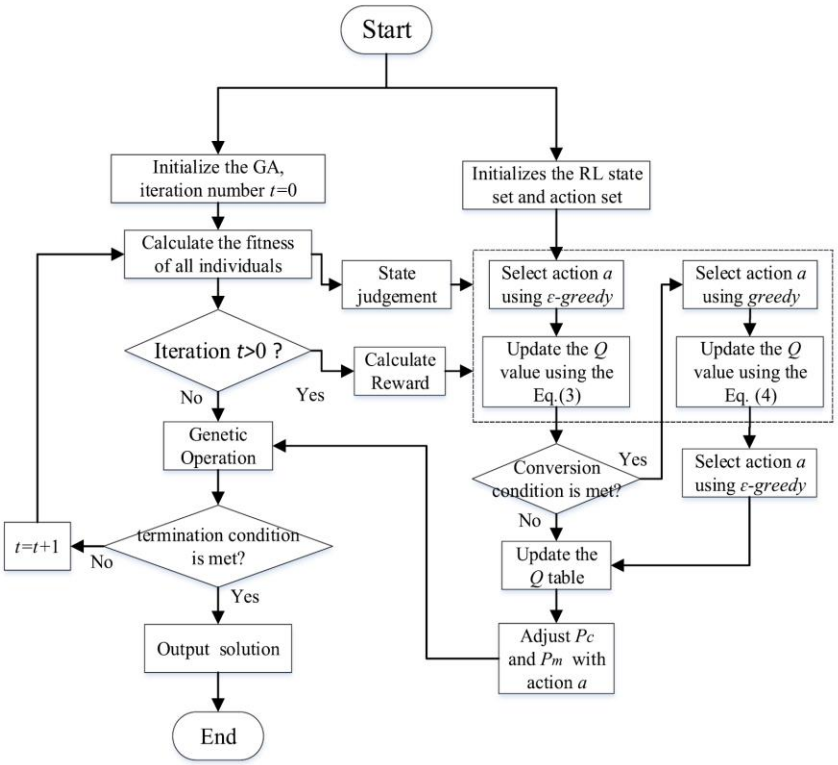
میزان بهره‌برداری و اکتشاف دو مؤلفه‌ی الگوریتم‌های ژنتیک هستن که در تعارض با هم قرار دارند. استراتژی $\varepsilon - greedy$ یک استراتژی انتخاب عملیات است که به هر دو مؤلفه‌ی بهره‌برداری و اکتشاف دقت دارد. این استراتژی در رابطه‌ی ۱۶ نشان داده شده است.

$$\pi(s_t, a_t) = \begin{cases} \max\limits_{a} Q(s_t, a) & \varepsilon \geq r_{.-۱} \\ a(Randomly) & \varepsilon < r_{.-۱} \end{cases} \qquad (۱۶)$$

که در آن منظور از ε نرخ بهره‌برداری و $r_{.-۱}$ یک عدد تصادفی بین صفر و یک است.

۴-۷- روند الگوریتم

روند الگوریتم به صورت کلی در شکل ۵ نمایش داده شده است. و همنین شبه کد آن در الگوریتم ۱ آمده است.



شکل ۵ : فلوچارت روند الگوریتم SLGA

Algorithm 1. SLGA

Initialize the GA: population size (N), maximum iterations (Max_t),
Initialize the RL: Q value table, state set (S), action set (A), policy (ε-*greedy*).
- Set current iteration number t=0. Calculate the fitness of all individuals.
- Choose a random action a_t, set $a \leftarrow a_t$. Calculate the State s_t of GA, set $s \leftarrow s_t$.

While $t \leq Max_t$ **do** /*major loop of SLGA*/
  Calculate the reward r_{t+1} according to Eq. (10) and Eq. (11).
  if conversion condition is not met /*SARSA algorithm*/
    Choose action a_{t+1} with ε-*greedy*.
    Update Q value according to Eq. (3)
  else
    Choose action a_{t+1} with *greedy*. /*Q-learning algorithm*/
    Update Q value according to Eq. (4).
    Update action a_{t+1} with ε-*greedy*.
  end
  - Calculate the state s_{t+1} of GA according to Eq. (6), Eq. (7), Eq. (8) and Eq. (9), update current $s \leftarrow s_{t+1}$.
  - Update current action $a \leftarrow a_{t+1}$.
  - Execute action a. /*action a will take new Pc and Pm*/
  - Execute genetic operation.
  - $t = t + 1$.
  - Calculate the fitness of all individuals.
End
Obtain solution

الگوریتم ۱: شبه کد الگوریتم SLGA

۵- پیاده‌سازی الگوریتم و نتایج به دست آمده

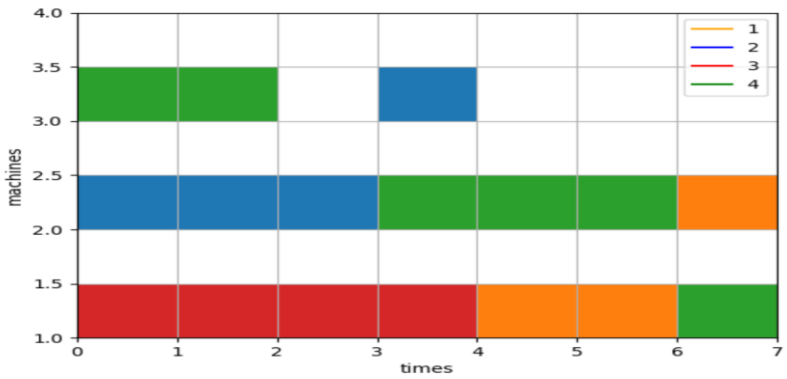
۵-۱- توضیح الگوریتم پیاده سازی شده

با توجه به مطالب گفته شده برای حل مسئله‌ی FJSP و الگوریتم GA، الگوریتم ژنتیکی پیاده سازی شد که در آن از نمایش موجود ذکر شده در بخش ۳-۱-۱ استفاده شده است. روش انتخاب، به صورت انتخاب بر اساس برازندگی (*FPS*)[۱]، تابع برازندگی منفی مقدار واحد زمان برای اتمام کارها در نظر گرفته شده است. هر فرد انتخاب شده با احتمال متناسب با آن و با استفاده از جهش درهم ریزی[۲] تغییر پیدا کرده و نیمه‌ی اول هر کروموزوم ساخته می‌شود (قسمت مربوط به توالی عملگرها). و متناسب با مقدار احتمال در نظر گرفته شده برای این که آیا از ماشین با کمترین زمان مصرفی برای عملگر داده شده استفاده شود یا خیر، یک ماشین مجاز برای آن عملگر انتخاب می‌شود. شرط خروج الگوریتم رسیدن به مقدار بیشینه‌ی تعداد نسل‌ها است. همچنین برای این که همیشه بهترین در طول کل نسل‌ها فراموش نشود آن را ذخیره نگه می‌داریم(مستقل از جمعیت). ورودی این الگوریتم یک جدول از زمان مصرفی هر عملگر برای هر ماشین و خروجی آن بهترین راه‌حل به دست آمده از الگوریتم است.

۵-۲-آزمایش‌ها

۵-۲-۱-آزمایش اول

برای داده‌های جدول ۱، الگوریتم اجرا شد و به جواب بهینه دست یافت که در شکل ۶ آمده است. این راه حل سعی کرده است که برای هر عملگر، ماشین سریع تر انتخاب کند به گونه ای که کمترین میزان بدون کار بودن ماشین‌ها را داشته باشیم. این مسئله برای اندازه‌ی جمعیت و حداکثر تعداد نسل ۱۰۰ اجرا شده است.



شکل ۶:زمانبندی کارها برای مسئله با اطلاعات مسئله‌ی داده شده در جدول ۱

[۱] Fitness proportionate selection

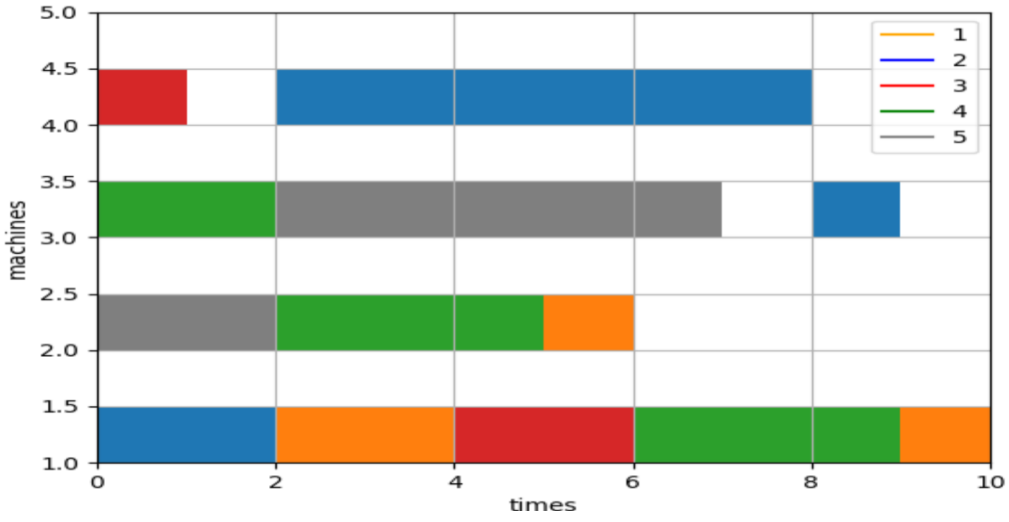
[۲] scramble

۵-۲-۲-آزمایش اول

در مرحله‌ی بعد مسئله ای پیچیده تر انتخاب شد که در آن تعداد ماشین‌ها و تعداد کارها افزایش پیدا کرد و تعداد عملگرهای برای هر کار نیز برابر نبودند. اطلاعات مسئله به صورت جدول ۳ است و نتیجه‌ی به دست آمده نیز بسیار خوب بوده است و در شکل ۷ به تصویر کشیده شده است. این مسئله نیز برای اندازه‌ی جمعیت و حداکثر تعداد نسل ۱۰۰ اجرا شده است.

	ماشین ۱	ماشین ۲	ماشین ۳	ماشین ۴
O_{11}	۲	-	۷	۵
O_{12}	-	۴	۱	۲
O_{13}	۲	۱	۴	-
O_{21}		۱۰	-	-
O_{22}	۲	۵	-	۴
O_{23}	۵	۵	۵	۶
O_{31}	۲	۳	-	۱
O_{32}	۲	۶	۵	۱۰
O_{41}	۲	-	۲	۳
O_{42}	-	۳	۵	۳
O_{43}	۱	-	۲	۴
O_{44}	۲	-	۳	۴
O_{51}	۷	۲	-	۲
O_{52}	۲	-	۳	۴

جدول ۳: جدول مربوط به داده‌های آزمایش دوم



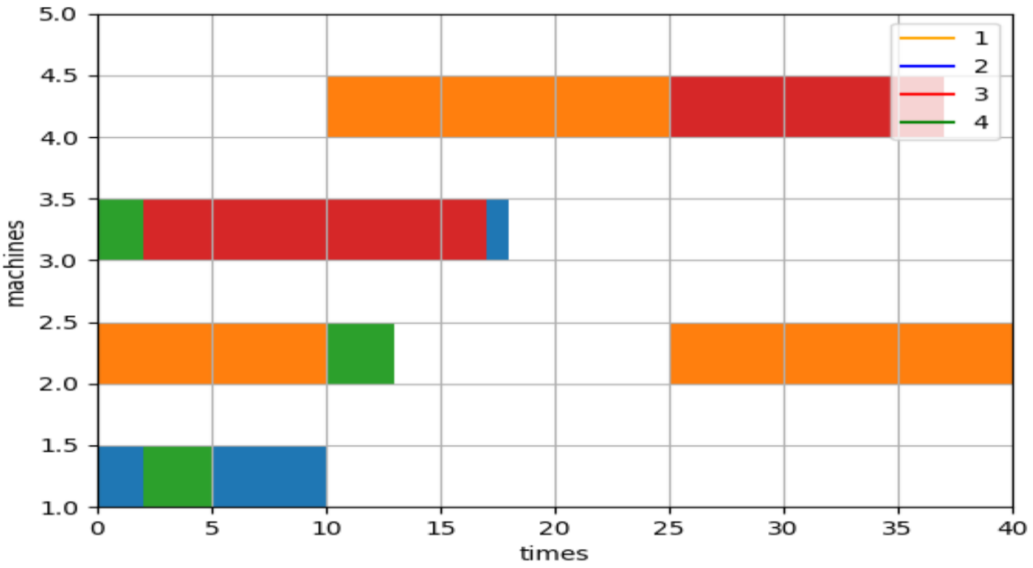
شکل ۷:نتیجه‌ی بدست آمده برای آزمایش دوم

۵-۲-۳-آزمایش سوم

در آزمایش سوم، برای زمان‌های مصرفی هر عملگر برای هر ماشین عددهایی در بازه‌های بزرگ‌تر انتخاب شد تا نتیجه‌ی الگوریتم برای مسائلی که مقدار زمان‌ها در آن بسیار متفاوتند ارزیابی شود.داده‌های مربوط به این آزمایش در جدول ۴ آمده است.در این آزمایش نیز الگوریتم به راه حل بهینه رسیده است و نتیجه‌ی کار به صورت شکل ۸ به دست آمده است. همچنین این مسئله نیز برای اندازه‌ی جمعیت و حداکثر تعداد نسل ۱۰۰ اجرا شده است.

	ماشین ۱	ماشین ۲	ماشین ۳	ماشین ۴
$o_{۱۱}$	۲۰	-	۷۰	۱۵
$o_{۱۲}$	۱۷	۱۰	۲۲	-
$o_{۱۳}$	۱۹	۱۵	-	-
$o_{۲۱}$	۲	۵	-	۴
$o_{۲۲}$	۴	۲	۱	۲
$o_{۲۳}$	۵	۵	۵	۶
$o_{۳۱}$	۲۱	۳۱	-	۱۲
$o_{۳۲}$	۲۲	۱۶	۱۵	۲۰
$o_{۴۱}$	۴	-	۲	۳
$o_{۴۲}$	-	۳	۵	۳
$o_{۴۳}$	۱	-	۲	۴
$o_{۴۴}$	۲	-	۳	۴

جدول ۴: جدول مربوط به داده‌های آزمایش سوم



شکل ۸: نتیجه‌ی بدست آمده برای آزمایش سوم

۶-نتیجه گیری

مسئله‌ی FJSP یک مسئله‌ی بسیار پیچیده و دارای قیود زیادی می‌باشد. یکی از روش‌های حل این مسائل استفاده از الگوریتم ژنتیک است که به دلیل اکتشاف زیاد و همچنین توجه به بهبود دادن راه حل‌ها ، انتخاب خوبی برای حل این گونه مسائل می‌باشد. از الگوریتم ژنتیک ذکر شده در گزارش برای حل این مسائل استفاده شده است. نتایج بدست آمده نشان می‌دهد که این الگوریتم جواب‌های خوبی بدست آورده است. استفاده از نمایش جایگشتی برای این گونه مسائل باعث شده است که فضای جستجو تا حد زیادی کوچک شود و زمان رسیدن به جواب‌های مطلوب بسیار کوتاه شود. در این مسائل حفظ مکان ژن‌های والدین می‌تواند کمک زیادی کند پس متناسب با این قضیه از جهش درهم ریزی که بخشی از آن فقط جابجا می‌شود استفاده می‌شود. این الگوریتم سعی می‌کند کارهای طولانی تر و ماشین‌های با سرعت عمل بیشتر برای هر عملگر را در اولویت قرار دهد تا بازدهی بالاتر رود.

الگوریتم SLGA به کمک یادگیری تقویتی دو مؤلفه‌ی p_c و p_m را تنظیم می‌کند که باعث می‌شود هم به اکتشاف و هم به بهره وری در هر نسل متناسب با شرایط، توجه ویژه‌ای شود. برای آموزش از دو الگویتم SARSA و Q-learning استفاده شده است که این دو الگوریتم تا حد زیادی شبیه به هم کار می‌کنند ولی یکی دارای بهره‌وری بیشتر ولی امکان گیر افتادن در نقاط بهینه‌ی محلی را دارد و دیگری بهره‌وری کمتر ولی قابلیت امتحان و اکتشاف بیشتری دارد. با ترکیب هر دوی آن‌ها و استفاده‌ی مناسب در زمان مناسب از آن‌ها باعث می‌شود از مزایای دو الگوریتم استفاده شود. این الگوریتم‌ها با استفاده از تجارب گذشته و پیش‌بینی آینده سعی در انتخاب بهترین مقدار برای p_c و p_m دارند. الگوریتم ترکیبی حاصل شده مزایای الگوریتم ژنتیک و یادگیری تقویتی را داراست. از جمعیت در الگوریتم ژنتیک برای اکتشاف و بهره‌وری و پیش رفتن به سمت نقطه‌ی بهینه استفاده می‌شود. از قدرت یادگیری، تجربه کردن، به یادآوردن گذشته و پیش بینی آینده در یادگیری تقویتی برای بهبود الگوریتم ژنتیک و تنظیم مؤلفه‌های آن استفاده شده است که این کار باعث می‌شود الگوریتم هوشمندانه تر به سمت هدف برود و درموارد مورد نیاز رویکرد اکتشافی داشته باشد. پاداش در این الگوریتم ارائه شده مقدار بهبود میانگین برازندگی جمعیت و بهبود بهترین فرد آن است. به دلیل آن که جمعیت به سمت برازندگی بیشتر رود، در صورت بهبود میانگین جمعیت، پاداش به انتخاب مناسب داده می‌شود و اگر به سمت بدتر شدن پیش رفت مجازاتی تعیین می‌شود. الگوریتم یادگیری تقویتی سعی در آن دارد که به بیشترین پاداش برسد پس به همین دلیل جمعیت نیز به برازندگی بالاتر در طول نسل‌ها می‌رسد

۷-منابع

[۱] A self-learning genetic algorithm based on reinforcement learning for flexible job-shop scheduling problem, Computers & Industrial Engineering (Volume ۱٤۹ , November ۲۰۲۰, ۱۰٦۷۷۸), ghua Chen, Bo Yang, Shi Li, Shilong Wang ,