طبقه بندی الگوریتم های ژنتیکی برای حل مساله JSP و مقایسه کارائی آنها

نویسنده اول

کمال کیانی

دانشگاه آزاد اسلامی واحد زنجان

Kamal_kiani_۸۱@yahoo.com

نویسنده دوم

نسیم همایونی

دانشگاه آزاد اسلامی واحد زنجان

nassim۲۰۰۲۱@yahoo.com

چکیده

الگوریتم های ژنتیکی برای حل مسائل Np-Hard طراحی Neural Network ها , Nonlinear Dynamic System , Strategy Planning و... به کار می روند.از مهمترین موارد استفاده ی GA در حل مسائل NP- Hardاعمال آن بر مساله JSP می باشد.JSP (JOB SHOP SCHEDULING) از سری مسائل زمان بندی می باشد که در صنعت کاربرد بسیاری دارد چرا که JSP به بررسی نحوه ی توزیع کار ها بین ماشین های انجام دهنده ی آن ها می پردازد , به طوری که این کارها در کمترین زمان ممکن انجام شوند. از طرفی بررسی مساله JSP می تواند به عنوان مدلی برای حل سایر مسائل Np-Hard از قبیل TSP استفاده شود.

کلمات کلیدی :*GA , initial population , fitness function , offspring ,representation, فضای حالت*

مقدمه

تاکنون روشهای بسیاری برای حل مساله ی JSP ارائه شده روشهایی مانند [calier ۸۹] branch and bound , [adams ۸۹] shifiting bottleneck , [van larrhoven ۹۲] simulated annealing , ... از آنجا که این متدها نیاز به زمان محاسباتی قابل توجه ویا محاسبات ریاضی پیچیده ای دارند استفاده از الگوریتم های ژنتیکی برای حل مساله ی JSP در طی دو دهه ی گذشته توسط [Davis ۸۵] , [Leipins ۸۷] , [Cleveland ۸۹] , [Nakano ۹۱] , [Yamada ۹۲] , [Davidor ۹۳] به صورت موفقیت آمیزی آغاز گردید و تا کنون نیز ادامه دارد [۲]. در این مقاله به دسته بندی الگوریتم های ژنتیکی اعمال شده برای حل مساله ی JSP پرداخته وسپس به شرح و مقایسه ی بهترین نتایج حاصل از اعمال این الگوریتم ها می پردازیم.

۱. Genetic Algorithm

Genetic Algorithm بر گرفته از نظریه تکاملی Darwin می باشد بر اساس این نظریه مسائل توسط یک روند تکاملی حل می شوند که نتیجه آن رسیدن به بهترین جواب ممکن است محاسبات تکاملی در دهه ۱۹۶۰ توسط Rosenberg معرفی شد . بعد ها نظریه این دانشمند توسط سایر محققان توسعه یافت تا اینکه Jon Holland در سال ۱۹۷۵ Genetic Algorithm را به عالم علم شناساند [۵] .

مراحل استاندارد GA

۱. representation : کد کردن جواب مسئله در یک کروموزم مانند representation های استاندارد Binary , Tree , Value , Permutation

۲. initial population : تولید فضای حالت اولیه شامل N کروموزم مختلف که به صورت تصادفی انتخاب شده اند.

۳. Fitness Function : اعمال تابعی برای محاسبه مطلوبیت کروموزم های موجود در فضای حالت

۴. new generation : تولید نسل جدید شامل ۳ مرحله ی زیر می باشد:

- selection : انتخاب دو کروموزم از فضای حالت به عنوان parent مانندروشهای انتخاب Rank , Elitism , Roulette Wheel , ...
- crossover : عملگری است که بر روی parent های انتخاب شده اعمال شده منجر به ایجاد offspring می شود مانند روش های استاندارد Single point, ۲-Point , Uniform
- mutation : تغییر دادن offspring ایجاد شده وتولید یک کروموزم جدید

۵. test:اعمال fitness function بر روی کروموزم جدید جهت سنجش میزان مطلوبیت آن , در صورتی که کروموزم جدید مطلوب بود اجرای الگوریتم متوقف می شود و در غیر این صورت این کروموزم به فضای حالت افزوده شده اجرای الگوریتم از مرحله ی ۴ ادامه می یابد.

۲.فاکتور های مهمی که در نتیجه نهایی GA موثرند

- اولین فاکتور مهم انتخاب یک representation مناسب برای مساله می باشد به طوریکه کمترین خطا را درتغییرات تصادفی داشته باشد و خطاهای حیاتی برای مساله تولید نکند .

- دومین فاکتور مهم تعریف صحیح fitness function است به طوریکه معیاری دقیق برای انتخاب بهترین جواب را ارائه نماید اگر این تابع درست تعریف نشود ممکن است الگوریتم موفق به یافتن جواب ضحیح برای مساله نشود.

- فاکتور مهم بعدی در GA انتخاب اندازه ی initial population , نرخ استفاده از عملگر های mutation و crossover می باشد همچنین نوع وقدرت selection نیز در انتخاب کروموزم های بهتر موثر است اگر initial population کوچک انتخاب شده باشد ممکن است جواب های زیادی از مسئله را در بر نگیرد و در صورت بزرگ انتخاب کردن اندازه ی initial population قدرت عملیات selection کاهش خواهد یافت.

- یکی دیگر از فاکتورهای مهمی که باید مد نظر قرار گیرد جلوگیری از به وجود آمدن همگرایی زود رس است چرا که همگرایی زود رس سبب رسیدن به یک جواب نسبی می شود.(همگرایی زود رس زمانی به وجود می آید که کروموزم انتخابی به عنوان parent مطلوبیت بالاتری نسبت به کروموزم هایی که در عمل crossover با آن ها شرکت میکند داشته باشد.)

۳.دسته بندی GA

GA هـای کلاسـیک از رشته ای باینری برای بیان کردن راه حل مسئله استفاده می کردند این نوع representation بـرای مسـائلی از قبیل TSP و JSP مناسب نبود به ا ین علت که یک راه حل مستقیم و موثر برای نگاشت جواب های ممکن به صورت یک رشته ی باینری وجود نداردلذا representation های جدیدی ارائه شدند که خود به دو دسته ی direct و indirect تقسیم می شوند در روش indirect جواب مسئله در یک ساختار ترتیبی تحت عنوان کروموزم بـه صورتی کد می شود که نشان دهنده ی ترتیب انجام oprestion ها توسط ماشین نمی باشد ولی در روش direct سـاختار ترتیبی (representation) ارئه شده عینا نشان دهنده ی ترتیب انجام operation ها توسط ماشین موجود است[۳].

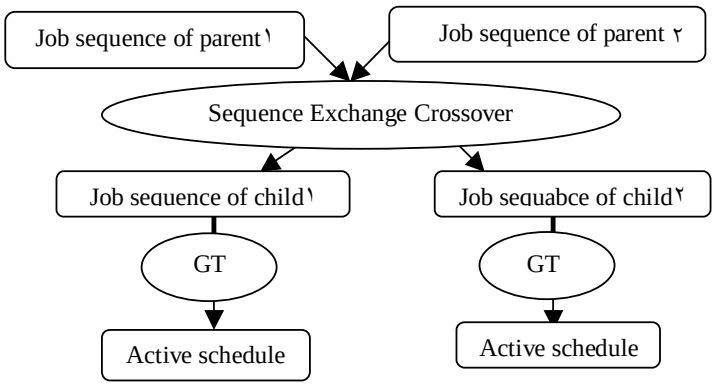
۴. دسته بندی مسائل JSP

مسـائل JSP بـه دو دسته ی عمده تقسیم می شوند: ۱: Classical JSP : در اینگونه مسائل m ماشین یکسان و n کار (job) داریم و باید به دنبال راهی باشیم که بهینه ترین حالت از توزیع کارها بین ماشین ها را نتیجه دهد.

۲: Flexible JSP : n کـار (job) مسـتقل از هـم داریم که هر job دنباله ای از operation هاست . m ماشین مخـتلف نیـز برای انجام operation ها وجود دارد که هر ماشین operation های نسبت داده شده به خود را یکی پـس از دیگری بدون هیچ وقفه ای انجام می دهد برای انجام هر operation با توجه به ماشین انجام دهنده آن یک زمـان خـاص برای اجرا وجود دارد که در جدول زمانی آورده می شود. FJSP خود به دو دسته ی Total و Partial تقسـیم مـی شـود . در مسائل(T-Fjsp) Total Fjsp هر ماشین توان انجام تمام operation ها را دارد ولی در مسـائل (P-Fjsp) Partial Fjsp هـر operation مـی تـواند توسط زیر مجموعه غیر تهی از ماشین ها ی موجود انجام شود[۱۰].

۵.کاربرد متد GT در حل مساله JSP

از انجاکه ممکن است offspring های تولید شده توسط crossover مطلوب نباشند مجموعه الگوریتم های GT روی آن ها اعمال شده تا آن ها را اصلاح نمایند , با توجه به ایده ی این متد دو نوع عملگر crossover ارائه شده است اولین عملگر sxx نام دارد که توسط Kobayashi در سال ۱۹۹۵ معرفی شد(sxx: در دوکروموزم زیر مجموعه ها یی از job ها قابل تعویض هستند که شامل مجموعه job های یکسان باشند sxx این زیر مجموعه قابل تعویض را بین دو کروموزم عوض می کند تا دو offspring جدید به وجودآید.) الگوریتم پیشنهادی وی GA/GT نام دارد[۲] .



روند الگوریتم GA/GT:۱. initial population ایجاد شده در این الگوریتم شامل Active job sequence هایی می باشد که توسط متد GT به وجود آمده اند . ۲.دو کروموزم تصادفی به عنوان parent انتخاب می شوند و سپس عملگر sxx بر روی آنها اعمال می شود که نتیجه ی آن به وجود آمدن offspring جدید می باشد .۳. در صورت نیاز offspring های جدید توسط متد GT اصلاح می شوند.۴. از مجموعه ی والدین و فرزندان دو کروموزمی که بیشترین fitness را دارند انتخاب شده و به عنوان parent جدید مد نظر قرار می گیرند.

دومین عملگر THX نام دارد که توسط Chang lin در سال ۱۹۹۶ معرفی شد این عملگر نقطه ای را به تصادف در هر یک از دو کروموزم parent انتخاب می کند ولی بر خلاف crossover استاندارد مقادیر قسمت های انتخاب شده را عینا تعویض نمی نماید بلکه مانند الگوریتم GT از فسمت های انتخاب شده برای تغییر مقادیر آنها استفاده می نماید[۳].

۶. T-FJSP

Mesghuni در سال ۱۹۹۸ representation ای تحت عنوان parallel machine برای حل این دسته از مسائل ارائه داد[۴] , همین محقق در سال ۲۰۰۴ برای بهبود GA اعمال شده بر روی T-FJSP یک representation جدید تحت عنوان parallel job معرفی کرد که در ادامه به بررسی هر یک می پردازیم [۹].

۵-۱)Parallel Machine Representation : در این representation کروموزم بیانگر مجموعه ماشین هایی است که به صورت موازی با هم کار می کنند که هر ماشین خود مجموعه ای از operation های نسبت داده شده به آن است در این روش crossover به این صورت انجام می شود که : دو parent {p۱,p۲} و یک ماشین {M۳} به صورت تصادفی انتخاب می کنیم operation های مربوط به M۳ از p۱ را درM۳ از offspring۱ قرار می دهیم (به همین ترتیب برای p۲ و offespring۲) سپس operation های غیر تکراری سایر ماشین های p۲ را در offespring۱ کپی می کنیم(به همین ترتیب سایر ماشین های p۱ باoffespring۲)

M۱	(۱،۱،۰)	(۱،۲،۱)	
M۲	(۲،۲،۲)	(۱،۳،۱)	(۳،۱،۱۵)
M۳	(۲،۱،۱)	(۳،۲،۱۰)	(۲،۳،۱۵)

برای parallel machine یک عملگر Mutation به نام assigned mutation انتخاب کرده. این عملگر به این ترتیب عمل میکند که یک operation ویک کروموزم به صورت تصادفی انتخاب میکند و operation انتخاب شده رادر صورت امکان به یک ماشین دیگر در همان ستون نسبت می دهد. Fitness function (جهت محاسبه زمان کل اجرای همه jobها توسط ماشین ها) به این ترتیب محاسبه میشود که زمان شروع آخرین operation هر ماشین و زمان لازم جهت اجرای آن با هم جمع میشوند ماکزیمم زمان بدست آمده, نشان دهنده زمان کل لازم برای اتمام همه operation ها میباشد.

نتایج محاسباتی :در حالتی که ده job و ده ماشین متفاوت برای انجام کارها داریم Parallel machine در سه حالت مختلف از نظر تشکیل initial population آزمایش شده است.

حالت اول : توسط متدConstraint Logic Programming ۵۰ کروموزم متفاوت برای تشکیل initial population به وجود میایند. بهترین زمان کل بدست آمده در این روش پس از اعمال GA بعد از ۶۰ نسل برابر ۲۲ واحد زمانی بوده است. حالت دوم : initial population توسط متد temporal decomposition به وجود می آید. به این ترتیب که یک کروموزم توسط این متد تولید میشود و سپس با استفاده از عملگرmutation فضای

حالـت به ۵۰ کروموزم گسترش می یابد. بهترین زمان بدست آمده در این حالت بعد از ۶۰ نسل ۱۳ واحد زمانی بوده اسـت. حالت سوم : از سه متد برای ایجاد initial population استفاده شده است. متدSPT ده کروموزم , متد LPT ده کروموزوم و Temporal decomposition نیز ۲۰ کروموزوم. در این حالت بهترین زمان بدست آمده بعد از ۶۰ نسل, ۹ واحد زمانی میباشد. مشاهده میشود هنگامی که چندین متد متفاوت برای ایجاد initial population به کار میروند نتیجه نهایی بهتری حاصل میشود.

۲-۵) parallel job representation : ایـن representation همانند parallel machine از نوع direct است بـا ایـن تفـاوت کـه مشـکلاتی از قبـیل تولـید جواب های غیر مجاز بعد از اعمال crossover را بر طرف نموده است. همچنـین در این روش میتوانیم به صورت کاملا تصادفی کروموزوم های initial population را به وجود آوریم. از طرفـی اعمـال عملگـر های GA روی این representation بسیار ساده تر است. در parallel job کروموزوم حاوی جـواب مسـاله توسط یک ماتریس نمایش داده میشود. در این ماتریس هر ردیف بیان گر مجموعه operation های یک job است و هر عنصر در این ردیف نشان دهنده ماشین انجام دهنده آن operation و زمان شروع آن می باشد.

Job۱	(M۱،۰)	(M۱،۱)	(M۴،۴)
Job۲	(M۴،۰)	(M۵،۳)	(M۲،۶)
Job۳	(M۵،۰)	(M۳،۲)	

عملگر Crossover در parallel job تحت عنوان Row Crossover همانند عملگر Crossover در parallel machine عمل میکند. علاوه بر آن نوع دیگری از Crossover به نام Column Crossover نیز بر روی آن قابل اجرا است. عملگر mutation نیز در این نوع از representation به این صورت عمل میکند که یک کروموزوم و یک operation را انـتخاب نموده و برای آن operation ماشین دیگری که زمان اجرای کمتری دارد نسبت میدهد. لازم بـه ذکـر است که انجام عمل mutation در parallel job بدون هیچ صدمه ای به ساختار کروموزوم (تولید جواب غیـر مجـاز) انجـام میشود. Fitness function در این روش به این صورت محاسبه می شود که زمان شروع آخرین operation هـر job را بـا زمـان لازم برای اجرای آن جمع میکنیم و سپس ماکزیمم مقدار بدست آمده را به عنوان fitness آن کروموزوم در نظر میگیریم.

نتایج محاسباتی: در حالتی که ده job و ده ماشین متفاوت برای انجام کارها داریم با در نظر گرفتن ساختار parallel job و اعمـال GA بـر روی آن بعـد از ۱۰ اجرای جداگانه بهترین fitness برابر با ۷ واحد زمانی بوده است. لازم به ذکـر است که این نتیجه پس از ۹۷۰ نسل جدید حاصل شده است. این در صورتی است که بهترین نتیجه بدست آمده از parallel machine بعد از۱۰ اجرای جداگانه و ۱۸۵۳ نسل برابر با ۷ واحد زمانی بوده است.

P-FJSP .۷

تـا سـال ۲۰۰۴ بهتـرین نتـیجه گـزارش شـده از حـل P-FGSP مربوط به مقاله I. kacem بوده است.Kacem اظهارداشـت کـه P-FGSP مسـاله را بـا افـزایش زمان محاسباتی وگسترش فضای حالت , پیچیده تر خواهد نمود. Representation ارائـه شـده توسـط وی assignment table نـام دارد. هـر عـدد ۱ در این جدول نشان دهنده ماشـین مـربوط بـه operation مورد نظر میباشد. عمل Crossover در این نوع representation به این صورت انجـام میشـود کـه ردیـف هـای یـک job در دو parent ثابت نگاه داشته شده و ردیف های jobهای دیگر این دو parent بـا هم تعویض میشوند در این حالت با توجه به ساختار representation ارائه شده , کروموزوم نیازی به باز سازی مجدد ندارد.Mutation نیز با تغییر ماشین مربوط به operation انتخاب شده انجام میشود[۸]

	M۱	M۲	M۳
O۱۱	۱	۰	۰
O۱۲	۰	۱	۰
O۱۳	۰	۱	--
O۲۱	۱	--	۰
O۲۲	۰	۰	۱

۸.استفاده از ساختار های T-Fjsp و P-Fjsp برای کد کردن کروموزم

در سال۲۰۰۴ متد GEANCE توسط N.B.HO معرفی شد که از محاسبات تکاملی برای حل هر دو نوع مساله ی F-JSP استفاده می کند و برای درست کردن جمعیت اولیه ی کروموزم ها از متد CDR بهره می برد[۱۰] . در ادامه به بررسی این متد می پردازیم.

الگوریتم CDR (Composite Dispatching Rule)

از CDR برای آماده سازی اولیه ی(initial population) الگوریتم GEANCE استفاده می شود این الگوریتم به ازای هر job صفی (ساختار این صف می تواند FIFO , LPT , SPT باشد) از operation های مربوط به آن را تشکیل داده سپس یک ترتیب تصادفی از job انتخاب کرده به مجموعه ی ماشین ها نسبت می دهد ماشین ها اولین operation در صف مربوطه را انتخاب و شروع به کار می کنند سپس با اتمام آن ,operation بعدی از صف بیرون کشیده می شود . آنگاه مدت زمان لازم برای اجرای آن در هر ماشین توسط فرمول زیر محاسبه می گردد تا تعیین شود کدام ماشین بهترین حالت را برای انجام آن کار دارد.

Sum of (processing times of waiting list of machine m)+processing time of O_{MK} on machine M + remaining processing time on machine M

لازم به ذکر است ترتیب تصادفی انتخاب کار ها در نتیجه نهایی موثر است.

Representation : شامل دو قسمت می شود :۱ - operation order part : نشان دهنده ی ترتیب قرار گرفتن operation ها می باشد . ۲- machine selection part : یک آرایه از مقادیر باینری که ماشین انتخاب شده برای انجام operation ها را مشخص می کند.

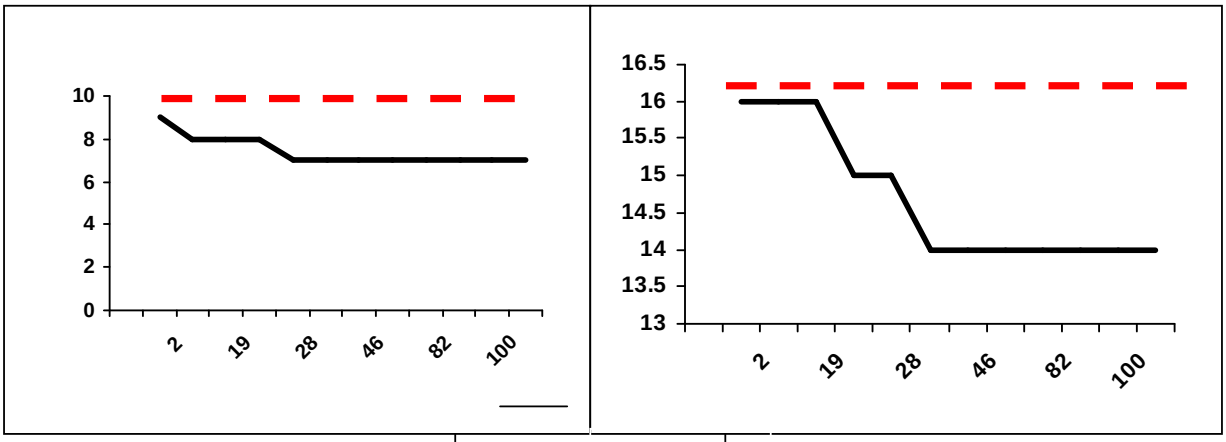
$O_{۱۱}$		$O_{۱۲}$			$O_{۱۳}$		$O_{۲۱}$		$O_{۲۲}$		
$M_۱$	$M_۲$	$M_۱$	$M_۲$	$M_۳$	$M_۲$	$M_۳$	$M_۱$	$M_۲$	$M_۱$	$M_۲$	$M_۳$
۰	۱	۰	۰	۱	۱	۰	۰	۱	۰	۱	۰

Crossover : با توجه به ساختار کروموزم در این الگوریتم ,crossover باید به هر یک از این دو قسمت اعمال شود. برای operation order از ۲-point crossover استفاده کرده و machine selection crossover نیز به این صورت انجام می گیرد که دو عدد random (r۱,r۲) که ۱-l=>r۲ , ۲=<r۱ است انتخاب شده مقادیر بین این دو قسمت در p۱ با مقادیر انتخاب شده به همین روش در p۲ عوض می شوند.

Mutation : نیـز بـه دو قسـمت کـروموزم اعمـال مـی شـود :۱- operation order : دو عدد r۱ و r۲ به صورت تصـادفی اتـنخاب مـی شـوند مقادیـر موجـود در ایـن بـازه تغییـر پـیدا می کند .۲- machine selection : عملگر mutation تحـت تاثیـر OB-Space انجـام می گیرد.(OB-Space شامل مجموعه ای از ماشین های موجود برای انجـام operation و معـیاری برای ایجادیک کروموزم کاراتر در انجام عمل mutation است .) بعد از K نسل مقادیر OB-Space , Update می شوند.

نـتایج محاسـباتی: بـا اعمـال الگوریـتم GENECE برای حل مسئله ای با ۱۰ job و ۱۰ ماشین مختلف و با در نظر گـرفتن شـرایط مسـئله بـه صـورت T-Fjsp بهترین نتیجه ی به دست آمده ۷ واحد زمانی بعد از ۲۰ نسل جدید بود همچنین با اعمال این الگوریتم برای حل مسئله ای با ۱۰ job و ۷ ماشین مختلف و با در نظر گرفتن شرایط مسئله به صـورت P – Fjsp بهتـرین نتـیجه ی بـه دسـت آمده ۱۲ واحد زمانی بوده است و این در حالی است که Kacem با همین شرایط مسئله در۱۵ واحد زمانی به نتیجه ی مطلوب رسید.

Result of ۱۰ jobs by ۱۰ machine P-FJSP



Result of ۱۰ jobs by ۱۰ machine T-FJSP

نتیجه گیری

با توجه به اهمیت مساله Np-Hard (مخصوصا jsp) و تنوع متدهای ارائه شده برای حل آنها و به دست آوردن نتایج مخـتلف نـیاز بـه یـک طـبقه بـندی دسـته مسـائل jsp و مقایسه ی کارایی متد ها برای سازماندهی این قبیل مسائل احساس می شد تا دید بهتر و جامعتری نسبت به این سری از مسائل Np-Hard برای کارهای آینده حاصل شود.

ارائـه ی مـتدهای جدیـد و representation هـای انعطـاف پذیرتر برای حل مسائل P - Fjsp و T-Fjsp هنوز به عنوان انگیره ای جهت ادامه ی کار محققان می باشد

سپاسگزاری

در پایـان از اسـتاد عزیـزمان جناب آقای مهندس امیری که مشوق و راهنمای ما در گردآوری و تهیه مطالب این مقاله بودند تقدیر و تشکر می نماییم.

Reference

[١]Takeshi Yamada and Ryohei Nakano, "A Genetic Algorithm with Multi-Step Crossover for Job-Shop Scheduling Problems ", First IEE/IEEE International Conference on Genetic ALgorithms in Engineering Systems (GALESIA '٩٥), ١٢-١٤ September-١٩٩٥

[٢]Kobayashi, Isao Ono , M.Yamamura,. : "An Efficient Genetic Algorithm for Job Shop Scheduling Problems " , Proc. of ICGA

[٣]Shyh-Chang Lin Erik D. Goodman William F. Punch, "Investigating Parallel Genetic Algorithms on Job Shop Scheduling Problems" , Genetic Algorithms Research and Applications Group Michigan State University-١٩٩٦

[٤]K. Mesghouni, S. Hammadi, P. Borne, "On modeling GA for flexible job-shop scheduling problem," ١٩٩٨

[٥]Marec obito, "Genetic Algorithms" – ١٩٩٨

[٦]H.Chen, J. Ihlow, and C. Lehmann, "A genetic algorithm for flexible JSP" Proc IEEE. International Conference on Robotics and Automation ١٩٩٩.

[٧]I.Kacem, S. Hammadi, and P. Borne, "Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems," IEEE Transactions on Systems, Man and Cybernetics, vol. ٣٢(١), ٢٠٠٢.

[٨] I. Kacem., S. Hammadi, P. Borne, "Pareto-optimality approach for flexible job-shop scheduling problems: hybridization of evolutionary algorithms and fuzzy logic," Mathematics and Computers in Simulation, vol. ٦٠- ٢٠٠٢.

[٩]K. Mesghouni, S. Hammadi, P. Borne, "Evolution programs for job-shop scheduling," Proc. IEEE International Conference on Computational Cybernetics and Simulation-٢٠٠٤

[١٠]N. B. Ho and J. C. Tay " GENACE: An Efficient Cultural Algorithm for Solving the Flexible Job-Shop Problem" , Intelligent Systems Laboratory - Nanyang Technological University, ٢٠٠٤