بسم الله الرحمن الرحیم

دانشگاه صنعتی شاهرود

دانشکده ریاضی

گروه ریاضی کاربردی

بررسی مسئله زمانبندی کارگاه گردش کاری جایگشتی

دانشجو : سمیرا محمودیان

استاد راهنما:

دکتر احمد نزاکتی رضازاده

استاد مشاور:

دکتر جعفر فتحعلی

پایان نامه جهت اخذ درجه کارشناسی ارشد ریاضی کاربردی

آذرماه ۸۹

شماره :
تاریخ :
ویرایش

بسمه تعالی

دانشگاه صنعتی شاهرود
مدیریت تحصیلات تکمیلی
فرم شماره (۶)

فرم صورتجلسه دفاع پایان نامه تحصیلی دوره کارشناسی ارشد

با تأییدات خداوند متعال و با استعانت از حضرت ولی عصر (عج) جلسه دفاع از پایان نامه کارشناسی ارشد **خانم سمیرا محمودیان** رشته **ریاضی** گرایش **کاربردی** تحت عنوان: **بررسی مسئله زمانبندی کارگاه گردش کاری جایگشتی** که در تاریخ ۸۹/۹/۲۰ با حضور هیأت محترم داوران در دانشگاه صنعتی شاهرود برگزار گردید به شرح زیر است :

قبول (با درجه : عالی امتیاز ۱۸٫۵۹) ☒	دفاع مجدد ☐	مردود ☐	

۱ـ عالی (۲۰ ـ ۱۹) ۲ـ بسیار خوب (۱۸/۹۹ ـ ۱۸)

۳ـ خوب (۱۷/۹۹ ـ ۱۶) ۴ـ قابل قبول (۱۵/۹۹ ـ ۱۴)

۵- نمره کمتر از ۱۴ غیر قابل قبول

عضو هیأت داوران	نام ونام خانوادگی	مرتبهٔ علمی	امضاء
۱ـ استادراهنما	دکتر احمد نزاکتی رضا زاده	استادیار	
۲ـ استاد مشاور	دکتر جعفر فتحعلی	استادیار	
۳ـ نماینده شورای تحصیلات تکمیلی	دکتر محمد آرشی	استادیار	
۴ـ استاد ممتحن	دکتر علیرضا ناظمی	استادیار	
۵ ـ استاد ممتحن	دکتر محمدرضا صافی	استادیار	

تأیید رئیس دانشکده :

تقدیم خالصانه به پدر، مادر و همسر عزیزم

وهمه آنان که اینچنین اند

مهربان و بی ادعا

قدردانی و تشکر

سپاس خاضعانه به پیشگاه کلام آفرین، که جوانه ی واژه ها به مدد نغمه ی خداوندیش به گل می نشیند. سپاس و ستایش پروردگاری را که رحمت بی پایانش روشنی بخش دل و دیده ام گردید و نعمت وجود خانواده ای مهربان را به من ارزانی داشت که در سایه ی این نعمت ها توانستم از گلستان علم و معرفت اساتید و دانش پژوهان توشه ای گرفته و از خرمن اطلاعاتشان خوشه ای برچینم و گامی گر چه بسیار کوچک در جهت خدمت به دانش دوستان بردارم. اینک که با عنایت و یاری خداوند تدوین و نگارش این پایان نامه به پایان رسیده بر خود لازم می دانم تا از تمامی کسانی که مرا در این راه یاری نمودند قدردانی نمایم، خصوصا استاد گرامی آقای دکتر احمد نزاکتی رضازاده که در تمام مراحل انجام این پایان نامه با راهنمائی ها و مساعدت های بی دریغ و بذل و توجهی مشفقانه پاسخگوی تمامی سوالات من بوده اند، و بر خود لازم می دانم از زحمات آقای دکتر جعفر فتحعلی کمال تشکر را نمایم که مرا همواره از راهنمایی های علمی خود بهره مند نموده اند. همچنین از آقایان دکتر علیرضا ناظمی و دکتر محمدرضا صافی که قبول زحمت نموده و داوری این پایان نامه را به عهده گرفته اند تشکر می نمایم. و نیز از سایر اساتید گروه ریاضی دانشگاه صنعتی شاهرود که افتخار شاگردی آنها را داشته و یا به نحوی از محضرشان کسب فیض نموده ام سپاسگزاری می نمایم.

در پایان از خانواده عزیزم که همیشه و در تمام مراحل زندگی راهنما و مشوق من بوده اند سپاسگزارم.

امیدوارم این پایان نامه برای اهل فن و دوستداران دانش مفید واقع شود.

تعهد نامه

اینجانب ......سمیرا محوریان...... دانشجوی دوره کارشناسی ارشد رشته ......ریاضی کاربردی...... دانشکده ......ریاضی......
دانشگاه صنعتی شاهرود نویسنده پایان نامه ......بررسی مسئله زمانبندی کارگاه گردش کاری جایگشتی
تحت راهنمائی......دکتر نیزالهی...... متعهد می شوم

- تحقیقات در این پایان نامه توسط اینجانب انجام شده است و از صحت و اصالت برخوردار است
- در استفاده از نتایج پژوهشهای محققان دیگر به مرجع مورد استفاده استناد شده است .
- مطالب مندرج در پایان نامه تاکنون توسط خود یا فرد دیگری برای دریافت هیچ نوع مدرک یا امتیازی در هیچ جا ارائه نشده است
- کلیه حقوق معنوی این اثر متعلق به دانشگاه صنعتی شاهرود می باشد و مقالات مستخرج با نام « دانشگاه صنعتی شاهرود » و یا « Shahrood University of Technology » به چاپ خواهد رسید .
- حقوق معنوی تمام افرادی که در به دست آمدن نتایج اصلی پایان نامه تأثیرگذار بوده اند در مقالات مستخرج از پایان نامه رعایت می گردد.
- در کلیه مراحل انجام این پایان نامه ، در مواردی که از موجود زنده (یا بافتهای آنها) استفاده شده است ضوابط و اصول اخلاقی رعایت شده است.
- در کلیه مراحل انجام این پایان نامه، در مواردی که به حوزه اطلاعات شخصی افراد دسترسی یافته یا استفاده شده است اصل رازداری ، ضوابط و اصول اخلاق انسانی رعایت شده است .

امضای دانشجو محوریان

تاریخ:
۸۹/۱۰/۱۷

مالکیت نتایج و حق نشر

- کلیه حقوق معنوی این اثر و محصولات آن (مقالات مستخرج ، کتاب ، برنامه های رایانه ای ، نرم افزار ها و تجهیزات ساخته شده است) متعلق به دانشگاه صنعتی شاهرود می باشد . این مطلب باید به نحو مقتضی در تولیدات علمی مربوطه ذکر شود .
- استفاده از اطلاعات و نتایج موجود در پایان نامه بدون ذکر مرجع مجاز نمی باشد.

چکیده

مسائل زمانبندی از جمله مسائل بهینه سازی به شمار می روند که تا کنون روی مسائل زمانبندی، توسعه و گسترش آن کارهای زیادی انجام شده است.

تنوع و پیچیدگی مسائل زمانبندی ما را برآن داشت که در این پایان نامه تنها به بررسی مسئله زمانبندی n کار برروی m ماشین در محیط کارگاه گردش کاری جایگشتی با زمان های پردازش معلوم پرداخته و دو مدل ابتکاری با هدف کمینه کردن مجموع زمان تکمیل کارها ارائه دهیم، همچنین دو مدل ارائه شده را از لحاظ بهینگی جواب نهایی ارزیابی و جواب های آن را با یکی از مدلهای موجود مقایسه می کنیم

ما در این پایان نامه با توجه به ارکان موضوع مورد تحقیق در ابتدا به معرفی مسائل زمانبندی و مفاهیم اولیه آن می پردازیم و پس از آن به معرفی مسئله زمانبندی کارگاه گردش کاری جایگشتی پرداخته به طوری که خواننده پس از مطالعه ای این فصل یک شناخت نسبی از مسئله فوق به دست آورد. در فصل سوم این پایان نامه مروری بر پیشینه موضوع صورت گرفته و از میان الگوریتم های ابتکاری فراوانی که توسط محققین و دانشمندان مطرح شده، چند مورد از بهترین الگوریتم ها تشریح شده است. البته شایان ذکر است که نویسنده این پایان نامه با استفاده از کد های *Matlab* برنامه چند مورد از الگوریتم های مطرح شده در این فصل را نوشته و نتایج مقایسات خود را در این فصل گنجانده است. در فصل چهارم دو الگوریتم ابتکاری برای حل مسئله فوق با معیار کمینه کردن مجموع زمان تکمیل کارها پیشنهادشده است و دو مدل فوق با استفاده از کدهای *Matlab* برنامه نویسی شده و با یکی از بهترین روش های موجود مقایسه شده است.

کلمات کلیدی: زمانبندی، الگوریتمهای ابتکاری، کمینه کردن، بهینه ساز

و درنهایت در فصل پنجم به تشریح کامل الگوریتم ژنتیک پیوندی که توسط لین- یو تسنگ[1] و یا- تی لین[2] در سال ۲۰۰۹ مطرح شده خواهیم پرداخت و کارایی روش فوق را نسبت به سایر روش ها بررسی خواهیم کرد.

کلید واژه : زمانبندی ؛ کارگاه گردش کاری جایگشتی؛ الگوریتم ابتکاری؛ الگوریتم ژنتیک پیوندی.

[1] Lin-Yu Tseng

[2] Ya -Tai Lin

فصل اول

آشنائی با مفاهیم اولیه زمانبندی

۱-۱ مقدمه

رویکرد علمی به مسئله برنامه‌ریزی عملیات، ریشه در انقلاب صنعتی و تلاشهای هنری گانت[1] دارد. در اکثر قریب به اتفاق مسائل برنامه‌ریزی به طور اعم و مسئله تعیین توالی عملیات به طور اخص، تنوع حالات مختلف حل مسئله و ترکیب‌های حاصل از این تنوع، برنامه‌ریزان را با طیف گسترده‌ای از راه‌حلهای مسئله روبرو می‌سازد.

انتخاب و به‌کارگیری بهترین راه‌حل از میان این طیف گسترده، بدون بهره‌گیری از رویکردهای برنامه‌ریزی ریاضی تقریبا غیرممکن است. از این روست که می‌توان اذعان کرد، استفاده از مدل‌های نمادین و بهره‌گیری از رویکردهای برنامه‌ریزی ریاضی در حل این گونه مسائل نقطه عطفی در سیر تحول روش‌های برنامه‌ریزی است.

۱-۲ تعریف زمانبندی[2]

زمانبندی عبارت است از تخصیص منابع برای انجام فعالیت‌ها در طول زمان . در اغلب مسائل زمانبندی یک مجموعه از منابع یا سرویس دهنده‌ها (ماشین) و یک مجموعه از مصرف کننده‌ها (کار) داریم که باید منابع به مصرف کننده به نحوی اختصاص یابند که یکی از معیارهای عملکرد، بهینه شوند.

در تمامی مسایل مورد بررسی زمانبندی تعداد ماشین ها (منابع) و کارها (فعالیت ها) محدود می باشند. تعداد کارها معمولا با n و تعداد ماشین ها با m و مدت زمان پردازش کار J_i برروی ماشین M_j با t_{ij}

---

[1] Henry gant

[2] scheduling

نمایش داده می شود. برای مثال جدول زیر شیوه نمایش n کار و m ماشین با زمان پردازش معلوم را نشان می دهد. در جدول زیر J_i (i=۱ ، ۲.......، n) نشان دهنده کار i ام و M_j (j= ۱ ، ۲.......، m) معرف ماشین jام است.

جدول ۱-۱ شیوه نمایش مفروضات یک مسئله زمانبندی

کار / ماشین	J_1	J_2	J_3	---------	J_n
M_1	t_{11}	t_{21}	t_{31}	---------	t_{n1}
M_2	t_{12}	t_{22}	t_{32}	---------	t_{n2}
⋮	⋮	⋮	⋮	⋮	⋮
M_m	t_{1m}	t_{2m}	t_{3m}		t_{nm}

۱-۲-۱ معیار عملکرد

معیار عملکرد رابطه مستقیم با اهداف مسأله دارد و در حقیقت همان تابع هدف[1] مسأله می‌باشد که در اغلب مسائل به صورت کمینه کردن زمان ظاهر می‌شود. همچنین تابع هدف ممکن است حداقل زمان اتمام کل سیستم [2]، حداقل کردن مجموع زمان تکمیل کارها[3] و.....باشد.

[1] Objective funtion

[2] makespan

[3] Total flowtime

<u>حداقل زمان اتمام کل سیستم:</u> این حالت بیانگر این است که هدف مساله یافتن بهترین ترتیب انجام کارها به منظورانجام تمامی کارها درسریعترین زمان ممکن است (حداقل سازی زمان انجام آخرین کار). کاربرد این تابع هدف در مواقعی است که سفارش ها بصورت گروهی تحویل مشتری می گردند. در چنین حالتی زمان تکمیل آخرین کار، زمان قابل تحویل محموله سفارشی است.

<u>حداقل کردن مجموع زمان تکمیل کارها</u> : معمولا هرچه مدت نگهداری محصولات افزایش یابد هزینه های نگهداری افزایش خواهد یافت. لذا حداقل سازی مجموع زمانهای تکمیل کار ها به کاهش هزینه تولید منجر می شود. در مواردی که هر سفارش به محض تکمیل تحویل مشتری می شود این تابع هدف کاربرد دارد.

۱-۳ دسته‌بندی مسائل زمانبندی

مسائل و مدل های زمانبندی را می توان به روش های متفاوتی طبقه بندی نمود، برای رده‌بندی مدل‌های عمدهٔ زمانبندی لازم است ترکیب منابع و رفتار کارها مشخص شود لذا به طور کلی می‌توان آن ها را به چهار گروه زیر تقسیم‌بندی کرد.

<u>۱-زمانبندی ایستا [1]:</u> مجموعه کارهای در دسترس برای زمانبندی در طول زمان تغییر نمی‌کند.

<u>۲-زمانبندی پویا [2]:</u> یک یا چند کار جدید به مجموعهٔ کارها در طول زمان افزوده می‌شود.

---

[1] Static scheduling

[2] Dynamic scheduling

۳-زمانبندی تک‌ماشینی[1]: مسأله‌ای است که در آن تنها یک منبع یا ماشین وجود دارد، لذا مسأله تک-ماشینی را با شرایط زیر می‌توان مشخص کرد.

- مجموعه n کار مستقل در زمان صفر برای پردازش در کارگاه موجود است.
- زمان آماده‌سازی کارها برای پردازش، مستقل از ترتیب انجام آنهاست و می‌توان آن را به عنوان بخشی از زمان پردازش در نظر گرفت.
- شرح کارها از پیش تعیین شده است.
- یک ماشین به طور پیوسته دسترس‌پذیر است و تا زمانی که کاری در انتظار انجام است ماشین بیکار نگه‌داشته نمی‌شود.
- هنگامی که پردازش کار بروی ماشین شروع شود تا زمان تکمیل عملیات بدون وقفه ادامه پیدا می-کند.

تحت این شرایط رابطه یک‌به‌یک بین توالی n کار و جایگشت شمارهٔ کارها (n ،.......۲ ، ۱) وجود دارد. بنابراین، تعداد کل راه‌حل‌های مجزا برای مسأله تک‌ماشینی برابر $n!$ حالت مختلف است که این تعداد ترتیب‌های مختلف توالی n عضو است.

تعریف: هرگاه بتوان یک زمانبندی عملیات را با جایگشتی از اعداد صحیح کاملا مشخص کرد به آن زمانبندی جایگشتی[2] می‌گوئیم.

1 Single Machine

2 Permutation Sceduling

نکته: در توصیف زمانبندی‌های جایگشتی مفید است که از علامت کروشه برای بیان ترتیب عملیات اســتفاده می‌شود. از این‌رو ۲ = [۵] نشانگر آن است که کار شماره ۲ از لحاظ ترتیب عملیات نوبت پنجم را دارد.

۴-زمانبندی چندماشینی[1]: بر خلاف زمانبندی تک‌ماشینی، تعداد منابع یا ماشین‌ها در این نوع زمانبندی بیش از یک ماشین می‌باشد.

۱-۴ انواع مسائل زمانبندی

۱-۴-۱ زمانبندی کار کارگاهی[2]:

در یک مدل زمانبندی کار کارگاهی با m ماشین

- هر یک از کارها دارای مسیر و فرایند پردازش خاص خود است.

برای مثال مسئله چهار کار $(J_i \quad i=1،2،3،4)$ و چهـار ماشــین $(M_j \quad j=1،2،3،4)$ زیر را در نظر بگیرید که هر کار دارای مسیر پردازش خاص خود است.

$J_1: M_1 - M_2 - M_3 - M_4$

$J_2: M_3 - M_2 - M_1 - M_4$

$J_3: M_3 - M_4 - M_2 - M_1$

$J_4: M_4 - M_2 - M_1 - M_3$

---

[1] Multi machine

[2] Jop shop

۱-۴-۲ زمانبندی کارکارگاهی بدون شرط[۱]

- در این حالت فرض می شود که m ماشین موجود است.
- هر یک از کارها بایستی توسط تمامی ماشین ها پردازش شود.
- ممکن است زمان پردازش یک کار به وسیله تعدادی از ماشین ها صفر باشد.
- محدودیتی در زمینه توالی پردازش هر یک از کارها به روی ماشین ها وجود ندارد.
- مسئول زمان بندی بنا به ماهیت تابع هدف می تواند در مورد توالی انجام هر کاری به روی ماشین ها و توالی انتخاب ماشین ها به روی هر کار تصمیم گیری کند.

یا به عبارتی می توان گفت، ترتیب انجام پردازش توسط ماشین های مختلف روی کارها مهم نباشد.

۱-۴-۳ زمانبندی کارگاه گردش کاری [۲]

در مسئله زمانبندی کارگاه گردش کاری n کار و m ماشین وجود دارد به طوری که

- هر کار به ترتیب به وسیله تمام ماشین ها پردازش می شود.
- در هر لحظه هر کار توسط حداکثر یک ماشین می تواند پردازش شود.
- در هر لحظه هر ماشین حداکثر یک کار را می تواند پردازش کند.

---

[1] Open shop

[2] Flow shop

- فقط یکبار یک کار توسط یک ماشین پردازش می شود و نمی تواند قبل از تکمیل شدن به پایان برسد.

۱- ۴ – ۴ زمانبندی کارگاه گردش کاری جایگشتی [1]

این نوع زمانبندی همانند زمانبندی کارگاه گردش کاری است با این تفاوت که ترتیب کارها توسط ماشین اول تعیین کننده ترتیب کارها توسط تمام ماشین هاست.

[1] Permutation flow shop

فصل دوم

معرفی مسئله زمانبندی کارگاه

گردش کاری جایگشتی

۲-۱ مقدمه

از زمان چاپ اولین مقاله توسط جانسون[1]، درباره توالی عملیات کارگاه گردش کاری در سال ۱۹۵۴ این مسئله مورد توجه بسیاری از پژوهشگران قرار گرفت. در مسئله کارگاه گردش کاری جایگشتی m ماشین و n کار وجود دارد. هر کار نیازمند m عملیات است و برای هر عملیات یک ماشین متفاوت لازم است. n کار با توالی یکسان روی m ماشین انجام می شوند. زمان فرایند کار i بر روی ماشین j به صورت t_{ij} (n ،.......۲ ، ۱ =i) و (m ،.......۲ ، ۱ =j) بیان می شود. هدف، یافتن بهترین ترتیب انجام و تکمیل کارهاست. بهینگی توالی عملیات با در نظر گرفتن یک معیار کارایی مانند کمینه کردن مجموع زمان تکمیل کارها[2] یا زمان اتمام کل سیستم[3] و .... مشخص می شود.

تفاوت عمده کارگاه گردش کاری جایگشتی با کارگاه گردش کاری در حالت عمومی این است که در حالت اول کارها در مراحل مختلف (ماشینها) از یکدیگر سبقت نمی گیرند. به عبارت دیگر ترتیب کارها روی ماشین اول، تعیین کننده ترتیب کارها روی تمام ماشین هاست. بنابراین تعداد کل راه حلهای مجزا برای مساله کارگاه گردش کاری جایگشتی برابر $n!$ حالت مختلف است. در حالت عمومی، کارها ممکن است در مراحل مختلف فرایند از یکدیگر سبقت بگیرند و تعداد کل راه حلهای مجزا برای مسئله فوق برابر $(n!)^m$ حالت مختلف است. در عمل، اکثر مسائل کارگاه گردش کاری از نوع جایگشتی است و سبقت کارها از یکدیگر به ندرت اتفاق می افتد.

---
1 Johnson
2 Totall flowtime
3 makespan

۲-۲ تعریف مساله

همان طور که در فصل اول بیان شد در مسئله زمانبندی کارگاه گردش کاری جایگشتی n کار $\{J_1,J_2, \ldots \ldots J_n\}$ و m ماشین $\{M_1,M_2, \ldots \ldots M_m\}$ وجود دارد، به طوری که

۱- هر کار به ترتیب به وسیله تمام ماشین ها پردازش می شود.

۲- مدت زمان فرایند کار J_i بر روی ماشین M_j به صورت t_{ij} بیان می شود.

۳- در هر لحظه هر کار برروی حداکثر یک ماشین می تواند پردازش شود.

۴- در هر لحظه هر ماشین حداکثر یک کار را می تواند پردازش کند.

۵- فقط یکبار یک کار برروی یک ماشین پردازش می شود و نمی تواند قبل از تکمیل شدن به پایان برسد.

مفروضات مسئله را می توان به صورت جدول زیر به تصویر کشید.

جدول ۲-۱ : مفروضات مسئله کارگاه گردش کاری جایگشتی

کار / ماشین	J_1	J_2	J_3	---------	J_n
M_1	t_{11}	t_{21}	t_{31}	---------	t_{n1}
M_2	t_{12}	t_{22}	t_{32}	---------	t_{n2}
⋮	⋮	⋮	⋮	⋮	⋮
M_m	t_{1m}	t_{2m}	t_{3m}		t_{nm}

حال فرض کنید $\pi=\{\pi_1, \pi_2, \ldots, \pi_n\}$ یک جایگشت از کارها باشد، لذا با توجه به تعریف مسئله زمانبندی کارگاه گردش کاری جایگشتی، پردازش کار $J_{\pi i}$ بر روی ماشین M_j زمانی شروع می شود که کار $J_{\pi i-1}$

برروی ماشین M_j و کار $J_{\pi i}$ برروی ماشین M_{j-1} پردازش شده باشد لذا زمان اتمام کار $J_{\pi i}$ بر روی ماشین M_j را با $c(\pi_i, j)$ نشان می دهیم و تعریف می‌کنیم:

$$c(\pi_i, j) = \max\{c(\pi_{i-1}, j), c(\pi_i, j-1)\} + t_{\pi_i, j} \qquad (۱-۲)$$
$$i = 1,2,......n \qquad j = 1,2,3......m$$

به طوری که $c(\pi_0, j)=0$ و $c(\pi_i, 0)=0$

۲-۲-۱ معیار عملکرد مجموع زمان تکمیل کارها

با توجه به مفروضات بالا زمان تکمیل کار $J_{\pi i}$ برابر است با $c(\pi_i, m)$ که عبارت است از زمان تکمیل کار $J_{\pi i}$ برروی آخرین ماشین M_m . لذا مجموع زمان تکمیل کارها را برای جایگشت π را به صورت زیر می توان تعریف کرد:

$$C_{sum}(\pi) = \sum_{i=1}^{n} c(\pi_i, m)$$

لذا مسئله زمانبندی کارگاه گردش کاری جایگشتی با معیار عملکرد کمینه کردن مجموع زمان تکمیل کارها یک جایگشت π^* را در مجموعه همه جایگشت های Π جستجو می کند به طوری که :

$$\forall \pi \in \Pi \qquad c_{sum}(\pi^*) \le c_{sum}(\pi).$$

۲-۲-۲ معیار عملکرد زمان اتمام کل سیستم

زمان اتمام کل سیستم را برای جایگشت π می توان به صورت زیر تعریف کرد:

$$c_{\max}(\pi) = c(\pi_n, m)$$

که در آن $c(\pi_n, m)$ عبارت است از زمان تکمیل آخرین کار $(J_{\pi n})$ بر روی آخرین ماشین (M_m) .

لذا مسئله زمانبندی کارگاه گردش کاری جایگشتی با معیار عملکرد کمینه کردن زمان اتمام کل سیستم، جایگشت π^* را در مجموعه همه جایگشت های Π جستجو می کند به طوری که :

$$\forall \pi \in \Pi \qquad c_{\max}(\pi^*) \le c_{\max}(\pi).$$

۲-۳ مسئله جانسون

مساله کار در کارگاه گردش کاری جایگشتی با دو ماشین با هدف کمینه کردن زمان اتمام کل سیستم را مسئله جانسون می نامند. نتایجی که در اصل توسط جانسون ارائه شده است اکنون اصول استاندارد نظریه زمانبندی محسوب می شود. در فرمولبندی این مساله کار j با زمان انجام کار t_{j1} برروی ماشین ۱ و با زمان انجام کار t_{j2} بر روی ماشین ۲ بعد از عملیات ماشین ۱ مشخص می شود .

جدول ۲-۲ : مفروضات مساله جانسون

کار / ماشین	J_1	J_2	....	J_n
M_1	t_{11}	t_{21}	....	t_{n1}
M_2	t_{12}	t_{22}	....	t_{n2}

توالی بهینه را می توان طبق قاعده زیر برای مرتب کردن کارها مشخص کرد.

قضیه ۲-۱ : (قاعده جانسون) در توالی بهینه کار i زودتر از کار j قرار می گیرد اگر

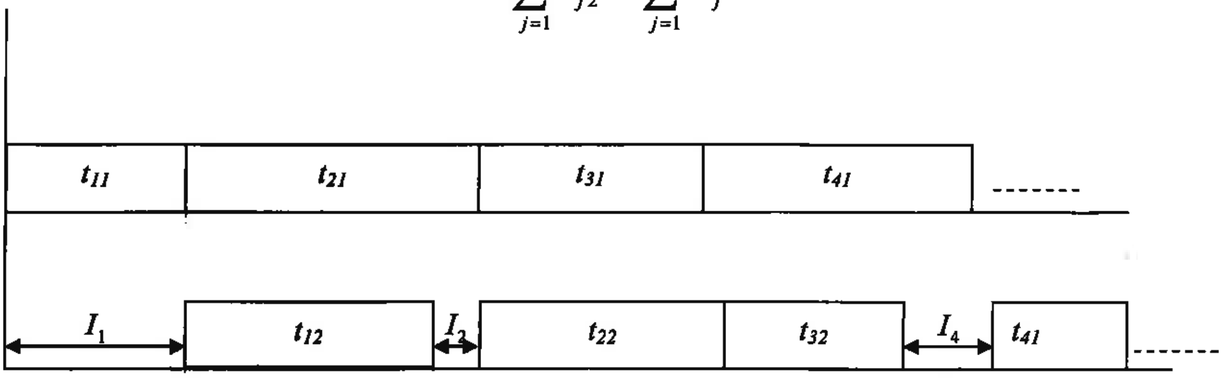
$$Min\ \{t_{i1}\ ,\ t_{j2}\} \le Min\ \{t_{i2}\ ,\ t_{j1}\}$$

عملا توالی بهینه مستقیما با پذیرش این نتیجه بنا می شود. نوبتها در توالی با استفاده از مکانیزم تک گذاری پر می شود که در هر مرحله، کاری را مشخص می کند که باید اولین یا آخرین نوبت موجود در توالی را پر کند.

اثبات :

با توجه به شکل ۲-۱، عملیات روی ماشین ۱ را می توان بدون در نظر گرفتن زمان بیکاری انجام داد. حال اگر I_j نمایشگر زمان بیکاری ماشین ۲ درست پیش از انجام کار jام باشد همانگونه که در شکل نشان داده شده است زمان اتمام کل سیستم را می توان به صورت زیر نوشت:

$$M = \sum_{j=1}^{n} t_{j2} + \sum_{j=1}^{n} I_j .$$



شکل ۲-۱ نمودار گانت در مساله کارگاه گردش کاری با ۲ ماشین

در رابطه فوق حاصل $\sum_{j=1}^{n} t_{j2}$ برابر با مقدار ثابتی است لذا کمینه کردن $\sum_{j=1}^{n} I_j$ هم ارز با کمینه کردن زمان اتمام کل سیستم است. لذا خواهیم داشت:

$$I_1 = t_{11}$$
$$I_2 = \max\{0, t_{11} + t_{21} - t_{12} - I_1\}$$
$$I_3 = \max\{0, t_{11} + t_{21} + t_{31} - t_{12} - t_{22} - I_1 - I_2\}$$

وبه طور کلی

$$I_j = \max\{0, \sum_{i=1}^{j} t_{i1} - \sum_{i=1}^{j-1} t_{i2} - \sum_{i=1}^{j-1} I_i\}.$$

همچنین می بینیم که

$$I_1 + I_2 = \max\{ t_{11}, t_{11} + t_{21} - t_{12} \}$$
$$I_1 + I_2 + I_3 = \max\{ t_{11}, t_{11} + t_{21} - t_{12}, t_{11} + t_{21} + t_{31} - t_{12} - t_{22} \}$$

و به طور کلی

$$\sum_{i=1}^{j} I_i = \max_{1\le k\le j} \{\sum_{i=1}^{k} t_{i1} - \sum_{i=1}^{k-1} t_{i2}\}.$$

برای ساده تر شدن رابطه بالا ، Y_k را به صورت زیر تعریف می کنیم.

$$Y_k = \sum_{i=1}^{k} t_{i1} - \sum_{i=1}^{k-1} t_{i2}$$

به طوری که

$$\sum_{i=1}^{j} I_i = \max_{1\le k\le j} \{Y_k\}.$$

ولذا هدف مطلوب، کمینه کردن $\sum_{i=1}^{n} I_i = \max_k \{Y_k\}$ است.

حال برنامه زمانی S را در نظر بگیرید که ترتیب تعیین شده طبق قضیه ۲-۱ در آن صدق نمی کند. این بدین معنی است که جفت کار مجاور i و j که j به دنبال i می آید وجود دارد به طوری که

$$\min \{t_{i1} , t_{j2}\} > \min \{t_{i2} , t_{j1} \} \qquad (۱)$$

فرض کنید برنامه زمانی S' نشان دهنده همان برنامه زمانی S باشد به جز در این مورد که در برنامه زمانی S' جای کارهای i و j در توالی با هم عوض شده است. همچنین مجموعه B را مجموعه کارهایی در نظر بگیرید که در دو برنامه، قبل از کارهای i و j هستند. در این صورت نامعادله ۱ را می توان به صورت زیر نوشت.

$$\max \{-t_{i1} , - t_{j2}\} < \max \{-t_{i2} , -t_{j1} \} \qquad (۲)$$

حال به دو طرف ۲ مقدار ثابت Q را اضافه می کنیم که در آن

$$Q = \sum_{k\in B} t_{k1} + t_{i1} + t_{j1} - \sum_{k\in B} t_{k2}$$

بنابرین داریم :

$$\max\{Q-t_{i1}, Q-t_{j2}\} < \max\{Q-t_{i2}, Q-t_{j1}\} \qquad (۳)$$

حال مشاهده می شود که :

$$Q - t_{i1} = \sum_{k\in B} t_{k1} + t_{j1} - \sum_{k\in B} t_{k2} = Y_j(S')$$

$$Q - t_{j2} = \sum_{k\in B} t_{k1} + t_{i1} + t_{j1} - \sum_{k\in B} t_{k2} - t_{j2} = Y_i(S')$$

$$Q - t_{j1} = \sum_{k\in B} t_{k1} + t_{i1} - \sum_{k\in B} t_{k2} = Y_i(S)$$

$$Q - t_{i2} = \sum_{k\in B} t_{k1} + t_{i1} + t_{j1} - \sum_{k\in B} t_{k2} - t_{i2} = Y_j(S).$$

از این رو رابطه (۳) به صورت زیر تبدیل می شود:

$$\max\{Y_j(S'), Y_i(S')\} < \max\{Y_i(S), Y_j(S)\}.$$

بنابراین با جابجایی کارهای i و jمقدار $\max_k\{Y_k\}$ نه تنها افزایش نمی یابد بلکه کاهش نیز پیدا می کند. لذا در هر توالی که نامعادله قضیه ۲-۱ در مورد آن کاملا برقرار نباشد ، جابجایی هر جفت مجاور، زمان اتمام کل سیستم را افزایش نمی دهد بلکه در واقع شاید هم آن را کاهش دهد.

۲-۳-۱ الگوریتم جانسون۱

گام یک : مقدار $min_i \{t_{i1}, t_{i2}\}$ را پیدا کنید.

گام دو : (الف) اگر کمترین زمان انجام کار، مربوط به ماشین ۱ باشد کار را در اولویت اول قراردهید وبه گام سه بروید.

(ب) اگر کمترین زمان انجام کار مربوط به ماشین ۲ باشد کار را در اولویت آخر قرار دهید و به گام سه بروید.

گام سه : کار تخصیص داده شده را از فهرست کارها حذف کنید و به قدم ۱ برگردید تا زمانی که تمام نوبت های توالی پر شود. (حالات هم ارزی به طور اختیاری تخصیص داده می شود.)

مثال ۲-۱

برای روشن شدن الگوریتم، مساله ۵ کار را که در جدول ۲-۳ نشان داده شده است در نظر بگیرید محاسبات نشان می دهد که چگونه توالی بهینه را با استفاده از الگوریتم جانسون در پنج مرحله می توان بنا کرد. در هر مرحله، کمترین زمان پردازش کارهای برنامه ریزی نشده را باید مشخص کرد . سپس در قدم دوم نوبت یک کار در ترتیب مشخص می شود و این فرایند آن قدر ادامه می یابد تا توالی {۲ ، ۵ ، ۴ ، ۱ ، ۳}با زمان اتمام کل سیستم برابر۲۴ حاصل می شود.

جدول ۲-۳ داده های مثال ۲-۱

کار / ماشین	J_1	J_2	J_3	J_4	J_5
M_1	۳	۵	۱	۶	۷
M_2	۶	۲	۲	۶	۵

جدول ۲-۴ اجرای گام به گام الگوریتم جانسون

برنامه زمانی جزئی	تخصیص	کمینه t_{ik}	کارهای زمانبندی نشده	
_ _ _ _ ۳	۳=[۱]	t_{31}	۱،۲،۳،۴،۵	مرحله ۱
۲_ _ _ ۳	۲=[۵]	t_{22}	۱،۲،۴،۵	مرحله ۲
۲ _ _ ۱ ۳	۱=[۲]	t_{11}	۱،۴،۵	مرحله ۳
۲ ۵ _ ۱ ۳	۵=[۴]	t_{52}	۴،۵	مرحله ۴
۲ ۵ ۴ ۱ ۳	۴=[۳]	$t_{41}=t_{42}$	۴	مرحله ۵

با توجه به بخش ۲-۲-۲ ، و رابطه ۲-۱ زمان اتمام کل سیستم برای جایگشت {۲ ، ۵ ، ۴ ، ۱ ، ۳} = π برابر است با $c_{\max}(\pi)=c(\pi_5,2)$.

لذا داریم:

$$c(\pi_1,1)=\max\{c(\pi_0,1),c(\pi_1,0)\}+t_{11}=0+1=1$$

$$c(\pi_1,2)=\max\{c(\pi_0,2),c(\pi_1,1)\}+t_{12}=1+2=3$$

$$c(\pi_2,1)=\max\{c(\pi_1,1),c(\pi_2,0)\}+t_{21}=1+3=4$$

$$c(\pi_2,2)=\max\{c(\pi_2,1),c(\pi_1,2)\}+t_{22}=4+6=10$$

$$c(\pi_3,1)=\max\{c(\pi_2,1),c(\pi_3,0)\}+t_{31}=4+6=10$$

$$c(\pi_3,2)=\max\{c(\pi_2,2),c(\pi_3,1)\}+t_{32}=10+6=16$$

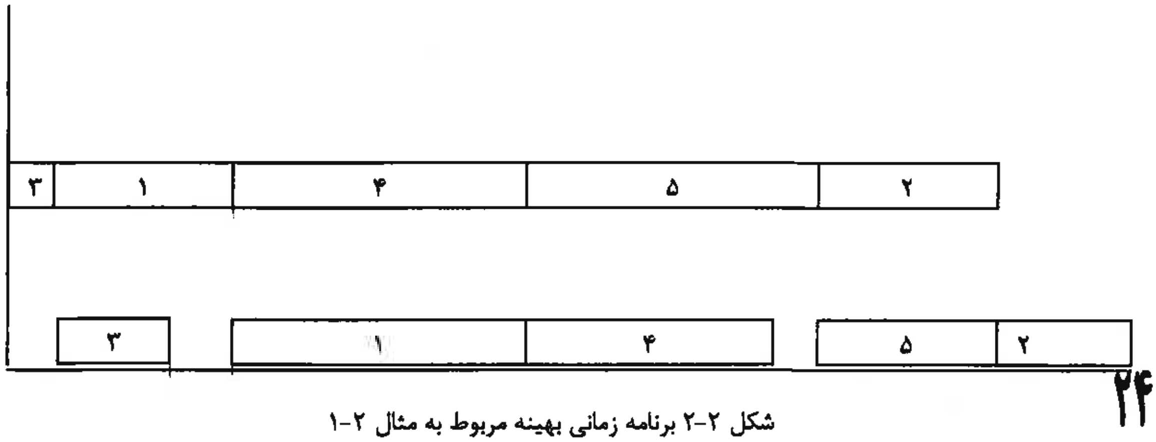
$$c(\pi_4,1)=\max\{c(\pi_3,1),c(\pi_4,0)\}+t_{41}=10+7=17$$

$$c(\pi_4,2)=\max\{c(\pi_3,2),c(\pi_4,1)\}+t_{42}=17+5=22$$

$$c(\pi_5,1)=\max\{c(\pi_4,1),c(\pi_5,0)\}+t_{51}=17+5=22$$

$$c(\pi_5,2)=\max\{c(\pi_4,2),c(\pi_5,1)\}+t_{52}=22+2=24.$$

در نتیجه زمان اتمام کل سیستم برای جایگشت بالا برابر با ۲۴ می باشد.



شکل ۲-۲ برنامه زمانی بهینه مربوط به مثال ۲-۱

الگوریتم جانسون ۲ که در ادامه ارائه می شود شاید در اولین نگاه متفاوت با الگوریتم ۲-۳-۱ به نظر آید،اما این الگوریتم در واقع همان الگوریتم ۲-۳-۱ می باشد که به صورتی دیگر بیان شده است.

۲-۳-۲ الگوریتم جانسون ۲

گام یک : مجموعه u را برابر $\{j \mid t_{j1} < t_{j2}\}$ و مجموعه v را برابر $\{j \mid t_{j1} \geq t_{j2}\}$ قرار دهید.

گام دو: اعضای مجموعه u را بر حسب t_{j1} ها به ترتیب غیر نزولی و اعضای مجموعه v را برحسب t_{j2} ها به ترتیب غیر صعودی مرتب کنید.

گام سه: توالی بهینه، مجموعه مرتب شده u است که مجموعه مرتب شده v به دنبال آن قرار دارد.

مثال ۲-۲ (حل مثال ۲-۱ با استفاده از الگوریتم جانسون ۲)

با حل مثال ۲-۱ با استفاده از الگوریتم بالا در گام یک مجموعه $u=\{۳،۱\}$ و مجموعه $v=\{۲،۴،۵\}$ حاصل می شود. با اعمال گام دو و مرتب کردن مجموعه u به صورت صعودی بر اساس t_{j1} ها داریم $u=\{۳،۱\}$ و با مرتب کردن مجموعه v به صورت نزولی بر حسب t_{j2} ها داریم $v=\{۴،۵،۲\}$. لذا توالی بهینه مجموعه مرتب

شده u است که مجموعه مرتب شده v به دنبال آن قرار دارد لذا جواب بهینه عبارت است از {۲، ۵، ۴، ۱، ۳} .

جانسون همچنین پی برد که در حالت سه ماشین، اگر با استفاده از الگوریتم ۲-۳-۲ برنامه زمانی بهینه یکسانی برای زیر مساله دو ماشینی مشخص شده با مجموعه $\{t_{i1},t_{i2}\}$ و زیر مساله دیگر مشخص شده با $\{t_{i2},t_{i3}\}$ ارائه کند، این توالی برای کل مساله سه ماشین بهینه است.

مثال ۲-۳

مسئله ۵ کار و ۳ ماشین زیر را در نظر بگیرید با حل مسئله با استفاده از روش سه ماشینه جانسون داریم:

جدول ۲-۵ داده های مثال ۲-۳

کار / ماشین	J_1	J_2	J_3	J_4	J_5
M_1	۳	۵	۱	۶	۷
M_2	۶	۲	۲	۶	۵
M_3	۷	۱	۳	۴	۲

مسئله فوق را به دو زیر مسئله زیر تبدیل کرده و روش جانسون دو ماشینه را برای هر کدام از مسائل زیر بکار می بریم.

جدول ۲-۶ زمان های پردازش برروی ماشین ۱و۲

کار / ماشین	J_1	J_2	J_3	J_4	J_5
M_1	۳	۵	۱	۶	۷
M_2	۶	۲	۲	۶	۵

جدول ۲-۷ زمان های پردازش برروی ماشین ۲ و ۳

ماشین \ کار	J_1	J_2	J_3	J_4	J_5
M_2	۶	۲	۲	۶	۵
M_3	۷	۱	۳	۴	۲

پس از اجرای الگوریتم ۲-۳-۲ برای جدول ۲-۶ به جواب {۲ ، ۵ ،۴ ،۱ ،۳}با زمان اتمام کل سیستم M=۲۴ و برای جدول ۲-۷ نیز به همان جواب می رسیم. لذا چون در دو زیر مسئله به یک جواب نائل شده ایم لذا برنامه زمانی فوق جواب بهینه مسئله بالا می باشد.

فصل سوم

حل مسئله زمانبندی کارگاه گردش کاری جایگشتی با استفاده از الگوریتم های ابتکاری

۳-۱ مقدمه و تاریخچه

حل مسئله زمانبندی کارگاه گردش کاری جایگشتی در سالهای اخیر برای محققان و دانشمندان جذابیت بسیاری داشته است. به طوری که در سال ۱۹۷۶ گری[1] وهمکاران حداقل کردن زمان اتمام کل سیستم و کمینه کردن مجموع زمان تکمیل کارها را در مسئله زمانبندی کارگاه گردش کاری جایگشتی جزء مسائل *Np-hard* معرفی کردند. از این رو دانشمندان علاقمند به طراحی روش های ابتکاری[2] و فرا ابتکاری[3] شدند. در این فصل مروری بر الگوریتم های ابتکاری که برای حل مسئله فوق با دو معیار کمینه کردن زمان اتمام کل سیستم و کمینه کردن مجموع زمان تکمیل کارها مطرح شده است، می پردازیم. در این راستا پالمر[4](۱۹۶۵)، کمپل [5] و همکاران(۱۹۷۰)، گوپتا[6] (۱۹۷۲)، داننبرینگ[7] (۱۹۷۷) ، ناواز [8] و همکاران(۱۹۸۳)، راجکوپال[9] (۱۹۸۸) و فرامینان[10] (۲۰۰۳) به ارائه روش های ابتکاری برای حل مسئله با معیار کمینه کردن زمان اتمام کل سیستم پرداختند. ولی در سالهای اخیر حل مسئله با معیار کمینه کردن مجموع زمان تکمیل کارها برای دانشمندان جذابیت بیشتری داشته است. به طور کلی روش های ابتکاری برای حل مسئله فوق با معیار کمینه کردن مجموع زمان تکمیل کارها را می توان به سه دسته تقسیم بندی کرد.

۱- روش های ابتکاری سودمند[11]. ونگ[12] و همکاران (۱۹۹۷)

[1] Garey
[2] heuristic
[3] metaheuristic
[4] palmer
[5] Campell.
[6] Gupta
[7] Dannenbring
[8] Nawaz
[9] Rajgopal
[10] faraninan
[11] Constructive heuristics
[12] wang

۲- روش های ابتکاری بهبود دهنده[1]. هو[2] (۱۹۹۵)

۳- روش های ابتکاری مرکب[3].

در این راستا محققان و دانشمندان، علاقمند به ارائه الگوریتم های ابتکاری برای حل مسئله با معیار کمینه کردن مجموع زمان تکمیل کارها شدند. از مهمترین کارهای انجام شده دراین زمینه می توان به روش هایی که توسط زیگلر و رجندران[4] (۱۹۹۷)، وو و ییم[5] (۱۹۹۸)، لیو و ریورس[6] (۲۰۰۱)، الله وردی و الدویسان[7] (۲۰۰۲)، فرامینان و لیستن[8] (۲۰۰۵و۲۰۰۳) و دیپاک لاها و سرین[9] (۲۰۰۹) پیشنهاد داده اند اشاره کرد.

در این فصل ما ابتدا مروری بر روشهای ابتکاری برای حل مسئله با معیار کمینه کردن زمان اتمام کل سیستم خواهیم کرد و سپس اشاره ای به روشهای ابتکاری با معیار کمینه کردن مجموع زمان تکمیل کارها خواهیم داشت.

۳-۲ روش های ابتکاری با معیار کمینه کردن زمان اتمام کل سیستم

۳-۲-۱ الگوریتم پالمر

این الگوریتم که در سال ۱۹۶۵ توسط پالمر مطرح شده را از لحاظ کیفی می توان به صورت زیر بیان کرد:

1 Improvement heuristics
2 Ho
3 Composite heuristics
4 Rajendran and Ziegler
5 Woo and Yim
6 Liu and Reeves
7 Allahverdi and Aldowaisan
8 Faraminan and Leisten
9 Dipak laha and Sarin

اولویت در توالی عملیات با کارهایی است که زمان پردازششان کمتر است. درحالی که ممکن است راههای بسیاری برای اجرای این حکم وجود داشته باشد، پالمر محاسبه شاخص شیب s_j، برای هر کار را به عنوان معیار پیشنهاد کرده است.

$$s_j = -\sum_{i=1}^{m} [m-(2i-1)]t_{ji} / 2 \qquad (۱-۳)$$

که در رابطه فوق t_{ji} برابر با زمان پردازش کار jام برروی ماشین iام می باشد، و m تعداد کل ماشینها است.

سپس برنامه زمانی با مرتب کردن کارها براساس رابطه زیر تعیین می شود.

$$s_{[1]} \geq s_{[2]} \geq \cdots\cdots \geq s_{[n]}. \qquad (۲-۳)$$

مثال ۳-۱

برای توضیح روش پالمر مسئله زیر با ۵ کار و ۳ ماشین را در نظر بگیرید.که داده های مربوط به آن در جدول زیر آمده است

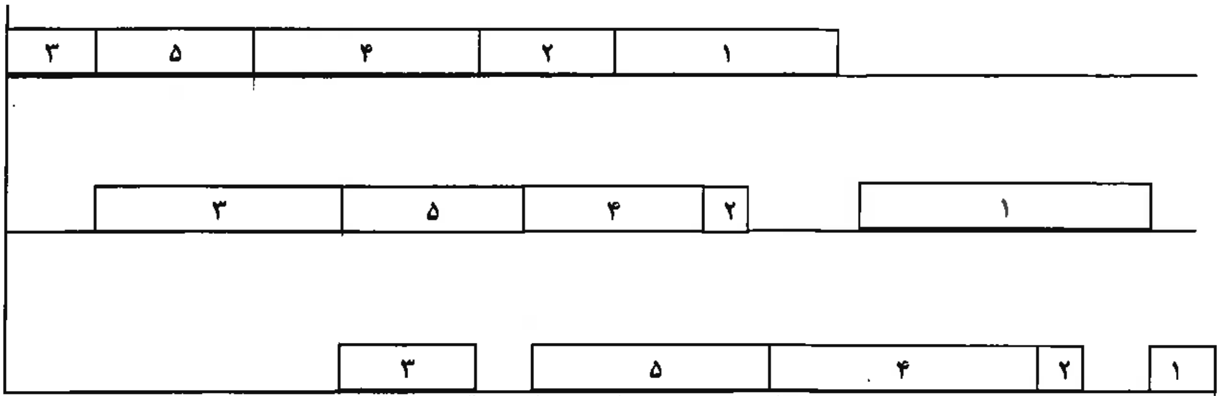
جدول ۳-۱ داده های مثال ۳-۱

کار / ماشین	J_1	J_2	J_3	J_4	J_5
M_1	۶	۴	۳	۹	۵
M_2	۸	۱	۹	۵	۶
M_3	۲	۱	۵	۸	۶

با استفاده از روش پالمر به ازای m=۳ طبق رابطه (۳-۱) شاخص شیب برابر با $s_j=t_{j3}-t_{j1}$ بدست می آید لذا

s_1 = -۴ s_2 = -۳ s_3 = ۲ s_4 = -۱ s_5= ۱.

از توالی به دست آمده طبق رابطه (۳-۲) برنامه زمانی {۱ ، ۲، ۴ ، ۵ ، ۳}به دست می آید که زمان اتمام کل سیستم در دنباله فوق طبق شکل زیر M= ۳۷ است.



شکل (۳-۱)برنامه زمانی مربوط به مثال ۳-۱

۳۷

۳-۲-۲ الگوریتم گوپتا

این الگوریتم که در سال ۱۹۷۱ توسط گوپتا به ازای $m>2$ ارائه شد همانند روش پالمر است با این تفاوت که

$$s_j = \frac{e_j}{\min_{1\le k\le m-1}\{t_{jk}+t_{jk+1}\}}$$

(۳-۳)

$$e_j = \begin{cases} 1 & if \quad t_{j1} < t_{jm} \\ -1 & if \quad t_{j1} \ge t_{jm} \end{cases}$$

و سپس کارها را طبق رابطه (۳-۲) مرتب می کنیم.

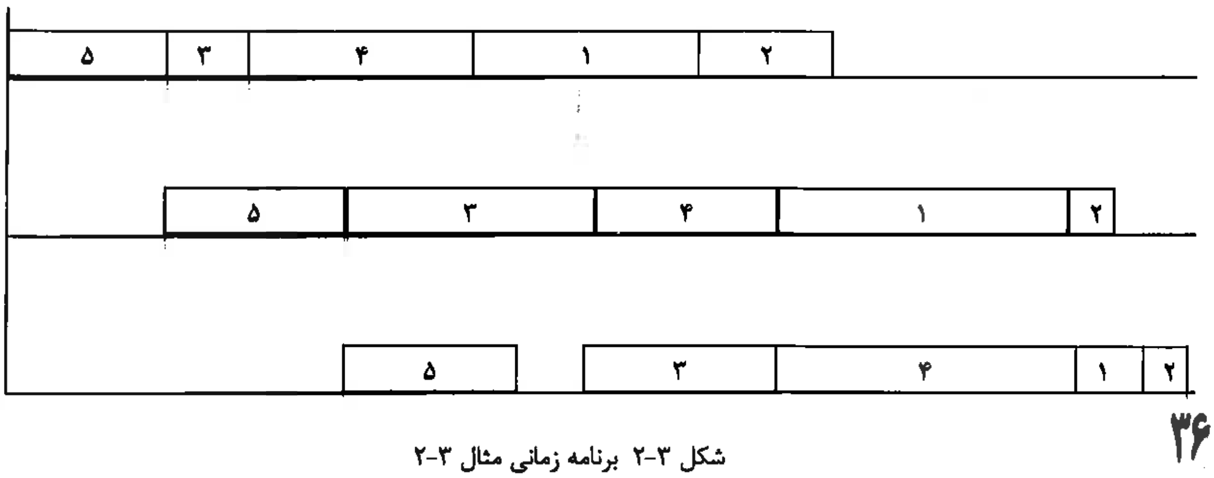
گوپتا این روش را با روش پالمر مقایسه کرد و نتیجه گرفت که این روش در اکثر موارد جایگشت بهتری نسبت به روش پالمر ارائه می دهد

مثال ۳-۲

با حل مثال ۳-۱ با استفاده از الگوریتم گوپتا داریم :

$$s_1 = -\frac{1}{10} \qquad s_2 = -\frac{1}{2} \qquad s_3 = \frac{1}{12} \qquad s_4 = -\frac{1}{13} \qquad s_5 = \frac{1}{11}.$$

براساس رابطه (۳-۲) برنامه زمانی {۲ ، ۱ ، ۴ ، ۳ ، ۵}به دست می آید که زمان اتمام کل سیستم طبق شکل۳-۲، M=۳۶ است.



شکل ۳-۲ برنامه زمانی مثال ۳-۲

۳-۲-۳ الگوریتم CDS

کمپل و همکاران در سال ۱۹۷۰ الگوریتمی برای حل مسئله با معیار کمینه کردن زمان اتمام کل سیستم ارائه دادند که این روش بر پایه دو خاصیت بنا نهاده شده است

۱- از قاعده جانسون به صورت ابتکاری اسنفاده می شود

۲- معمولا چند برنامه زمانی تولید می شود که از میان آنها بهترین انتخاب می شود.

الگوریتم CDS متناظر است با استفاده چند مرحله ای از قاعده جانسون در مساله جدید که از مسئله اصلی نتیجه گرفته شده است. در این روش، $m\text{-}1$ توالی با تقسیم کردن m ماشین به دو گروه و استفاده از الگوریتم جانسون به دست می آید و سپس از میان $m\text{-}1$ توالی، دنباله ای با مینمم زمان اتمام کل سیستم انتخاب می شود. در واقع روش فوق را می توان به صورت زیر بیان کرد:

در مرحله اول قرار می دهیم :

$$t'_{j1} = t_{j1}$$
$$t'_{j2} = t_{jm}$$

و سپس با استفاده از قاعده جانسون یک برنامه زمانی به دست می آوریم.

در مرحله دوم قرار می دهیم :

$$t'_{j1} = t_{j1} + t_{j2}$$
$$t'_{j2} = t_{jm} + t_{jm-1}$$

و مجددا با استفاده از قاعده جانسون یک دنباله زمانی جدید به دست می آوریم.

به طور کلی در مرحله i ام

$$t'_{j1} = \sum_{k=1}^{i} t_{jk}$$
$$t'_{j2} = \sum_{k=1}^{i} t_{j,m-k+1}$$

لذا در هر مرحله ترتیب انجام کارها با استفاده از قاعده جانسون به دست می آید، بعد از مرحله $m-1$ ، کمترین زمان اتمام کل سیستم از میان $m-1$ برنامه زمانی مشخص می شود.

کمپل و همکاران الگوریتم خود را به طور جامع آزمودند و عملکرد آن را در مقایسه با روش پالمر در مورد چند مساله بررسی کردند. آنها دریافتند که الگوریتم CDS به طور کلی هم برای مسائل کوچک و هم برای مسائل بزرگ اثر بخش تر است. به علاوه به ازای $n \leq ۲۰$ زمان های مورد نیاز کامپیوتر مرتبه بزرگی یکسانی را در مورد هر دو روش نشان داد. فقط در مورد برخی که اندکی بزرگتر بودند مساله مقایسه ارزش جواب با زمان محاسباتی مصرف شده پیش می آمد.

مثال ۳-۳

با حل مثال ۳-۱ با استفاده از روش CDS در مرحله اول داریم

جدول ۳-۲ نتایج مرحله اول الگوریتم CDS

کار / ماشین	J_1	J_2	J_3	J_4	J_5
t'_{j1}	۶	۴	۳	۹	۵
t'_{j2}	۲	۱	۵	۸	۶

لذا با اجرای قاعده جانسون برنامه زمانی {۳ ، ۵ ، ۴ ، ۱ ، ۲}با زمان اتمام کل سیستم M=۳۵ حاصل می شود. در مرحله دوم داریم:

جدول ۳-۳ نتایج مرحله دوم الگوریتم CDS

ماشین \ کار	J_1	J_2	J_3	J_4	J_5
t'_{j1}	۱۴	۵	۱۲	۱۴	۱۱
t'_{j2}	۱۰	۲	۱۴	۱۳	۱۲

با استفاده از قاعده جانسون برنامه زمانی {۲ ، ۱ ، ۴ ،۳ ،۵} با زمان اتمام کل سیستم M=۳۶ به دست می آید. لذا بهترین جواب، برنامه زمانی {۲ ، ۱ ، ۴ ، ۵ ، ۳} با M=۳۵ است.

۳-۲-۴ الگوریتم *NEH*

ناواز و همکاران در سال۱۹۸۳ الگوریتمی ابتکاری برای حل مسئله زمانبندی کارگاه گردش کاری جایگشتی ارائه دادند که نسبت به سایر روش هایی که تا قبل از آن مطرح شده بود کاراتر و مناسب تر بود. الگوریتم *NEH* را می توان به صورت زیر بیان کرد:

گام یک : برای هر کار J_i (i=۱ ،۲،.....n) متوسط زمان تکمیل کار را محاسبه می کنیم.

$$T_i = \sum_{j=1}^{m} t_{ij} \qquad i = 1,2,\ldots..n \,. \tag{۳-۴}$$

و سپس کارها را به صورت نزولی براساس متوسط زمان تکمیل کار مرتب می کنیم و در مجموعه H قرار می دهیم.

گام دو : قرار می دهیم k=۱

اولین کار از مجموعه H را انتخاب و آن را در اولین موقعیت در جواب جاری S قرار می دهیم .

گام سه : $k=k+1$

k امین کار را از مجموعه H را انتخاب و آن را به k موقعیت ممکن در S اضافه می کنیم واز میان k دنباله ایجاد شده یکی را با کمترین زمان اتمام کل سیستم جزئی انتخاب و آن را به عنوان جواب جاری S قرار می دهیم.

گام چهار: اگر $k=n$ توقف کن در غیر این صورت به گام سه بروید.

مثال ۳-۴

با حل مثال ۳-۱ با استفاده از الگوریتم NEH در گام یک داریم:

$$T_1 = ۱۶ ، \quad T_2 = ۶ ، \quad T_3 = ۱۷ ، \quad T_4 = ۲۴ ، \quad T_5 = ۱۷.$$

لذا مجموعه H برابر است با H={۲-۱-۳-۵-۴} .

به ازای $k=$ ۱ ، جواب جاری S={۴} است. و به ازای $k=$ ۲ دو دنباله {۵-۴} و{۴-۵} به ترتیب با زمان اتمام کل سیستم جزئی ۳۰ و ۲۷ حاصل می شود لذا جواب جاری{۴-۵}=S می باشد. با تکرار دوباره گام سه به ازای $k=$ ۳ سه دنباله {۳-۵-۴}و {۵-۳-۴} و {۵-۴-۳} به ترتیب با زمان اتمام کل سیستم جزئی ۳۲ و۳۳ و ۳۳ حاصل می شود لذا S={۳-۵-۴} به عنوان برنامه زمانی جزئی انتخاب می شود. با مقدار $k=$۴ دنباله های {۱-۳-۵-۴} و {۳-۱-۵-۴} و {۳-۵-۱-۴} و {۳-۵-۴-۱}به ترتیب با زمان اتمام کل سیستم ۴۳ و ۴۰ و۳۹ و ۳۴ حاصل می شود لذا S={۳-۵-۴-۱}. به ازای $k=$ ۵ پنج دنبالهٔ {۲-۳-۵-۴-۱} و {۱-۴-۵-۲-۳} و {۱-۴-۲-۵-۳} و {۱-۲-۴-۵-۳} و {۲-۱-۴-۵-۳} و {۳-۵-۴-۱-۲} به ترتیب با زمان اتمام کل سیستم ۳۸ و ۳۷ و ۳۶ و ۳۷ و ۳۵ حاصل می شود لذا بهترین جواب با استفاده از روش NEH ، S={۳-۵-۴-۱-۲} با زمان اتمام کل سیستم M=۳۵ است.

مثال ۳-۵

با حل مثال ۳-۱ با استفاده از قاعده جانسون سه ماشینه داریم :

جدول ۳-۴ زمان های پردازش برروی ماشین ۱ و ۲

کار / ماشین	J_1	J_2	J_3	J_4	J_5
M_1	۶	۴	۳	۹	۵
M_2	۸	۱	۹	۵	۶

جدول ۳-۵ زمان های پردازش برروی ماشین ۲ و ۳

کار / ماشین	J_1	J_2	J_3	J_4	J_5
M_2	۸	۱	۹	۵	۶
M_3	۲	۱	۵	۸	۶

پس از اجرای قاعده جانسون دو ماشینه برروی جدول ۳-۴ برنامه زمانی {۳ ، ۵ ، ۱ ، ۴ ، ۲} با زمان اتمام کل سیستم $M = ۳۸$ حاصل می شود. همچنین برای جدول ۳-۵ نیز دو برنامه زمانی {۲ ، ۴ ، ۵ ، ۳ ، ۱} و {۴ ، ۵ ، ۳ ، ۱ ، ۲} با زمان اتمام کل سیستم $M = ۴۱$ بدست می آید.

همانطور که مشاهده می شود با استفاده از روش جانسون سه ماشینه علاوه براینکه برای هر دو زیر مسئله برنامه یکسانی به دست نیامده است مشاهده می شود که جواب های فوق هیچکدام یک برنامه زمانی بهینه را نتیجه نمی دهد.

حل مسئله زمانبندی کارگاه گردش کاری جایگشتی با معیار کمینه کردن زمان اتمام کل سیستم در سالهای اخیر جذابیت گذشته را برای محققان نداشته و اکثر دانشمندان در سالهای اخیر هدفشان کمینه کردن مجموع زمان تکمیل کارها بوده است لذا از این پس مروری بر روش های ابتکاری که برای حل مسئله فوق با معیار کمینه کردن مجموع زمان تکمیل کار ها است می پردازیم.

۳-۳ روشهای ابتکاری با معیار کمینه کردن مجموع زمان تکمیل کارها

۳-۳-۱ الگوریتم *RZ*

این الگوریتم ابتکاری که توسط رجندرام و زیگلر ارائه شده است شامل دو مرحله است بدین صورت که درمرحله اول یک دنباله از کارها تولید و در مرحله دوم دنباله فوق بهبود بخشیده می شود.

<u>مرحله اول :</u> در این مرحله ابتدا کارها را به صورت صعودی براساس t_{i1} (زمان پردازش توسط ماشین اول) مرتب می کنیم و دنباله فوق را به عنوان اولین کاندیدا در نظر می گیریم. سپس کارها را به صورت صعودی براساس $۲t_{i1}+t_{i2}$ مرتب کرده و دنباله به دست آمده در این مرحله را به عنوان دومین کاندیدا قرار می دهیم و برای تعیین سومین کاندیدا کارها را بر اساس $۳t_{i1}+۲t_{i2}+t_{i3}$ به صورت صعودی مرتب می کنیم. این روند را تا مرحله m ام ادامه داده و در مرحله m ام کارها را به صورت صعودی براساس $mt_{i1}+(m-1)t_{i2}+ \ldots\ldots\ t_{im}$ مرتب می کنیم و m امین جواب کاندیدا را نیز به دست می آوریم . حال از میان m جواب کاندیدا یکی را با کمترین مجموع زمان تکمیل کارها انتخاب کرده و آن را به عنوان جواب پایه در نظر گرفته و به مرحله دوم می رویم.

<u>مرحله دوم :</u> اولین کار از جواب پایه را انتخاب و آن را به n موقعیت ممکن در جواب پایه اضافه می کنیم و از میان n دنباله ایجاد شده یکی را با کمترین مجموع زمان تکمیل کار انتخاب و در مجموعه جواب کاندیدا قرار می دهیم. سپس دومین کار از جواب پایه را به n موقعیت ممکن اضافه و از میان n دنباله ایجاد شده

یکی را با کمترین مجموع زمان تکمیل کار انتخاب و در مجموعه جواب کاندیدا قرار می دهیم، این روند را تا n امین کار ادامه داده و از میان مجموعهٔ جواب های کاندیدا یکی را با کمترین مجموع زمان تکمیل کار انتخاب کرده و به عنوان بهترین جواب در نظر می گیریم.

مثال ۳-۶

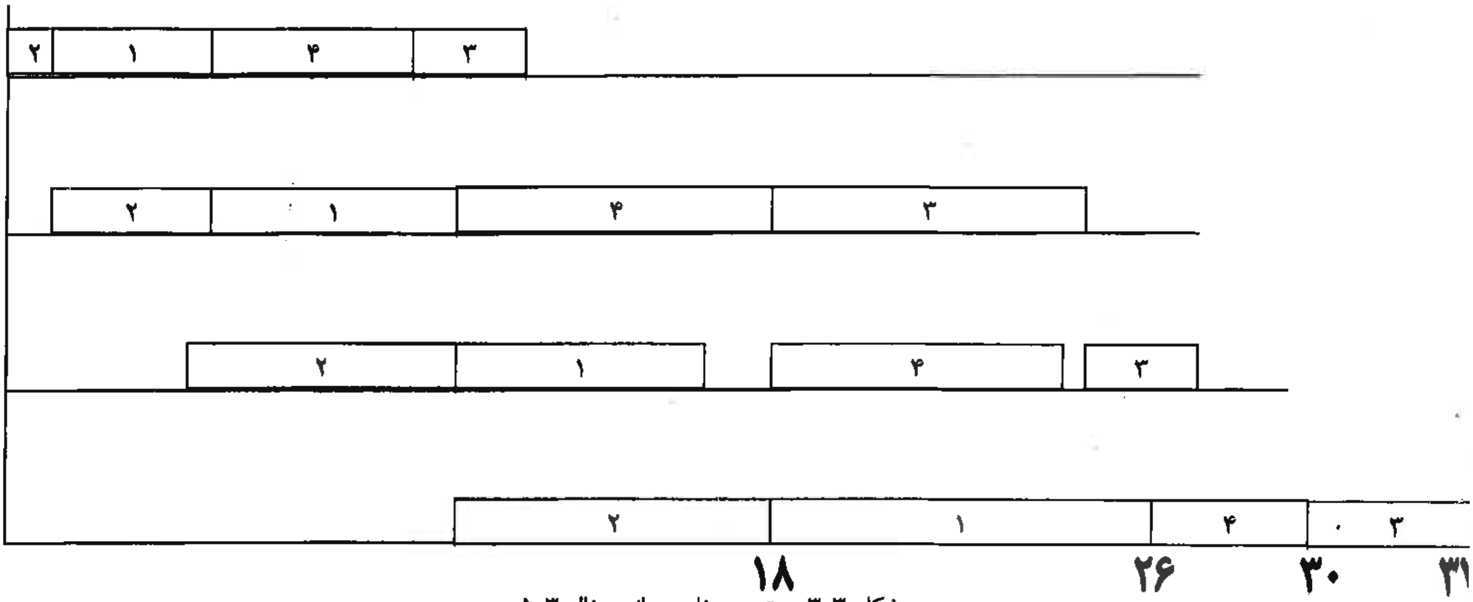
مسئله ۴ کار و ۴ ماشین زیر را در نظر بگیرید. داده های مربوط به مسئله فوق در جدول ۳-۴ آمده است. با حل مسئله با استفاده از روش RZ در مرحله اول چهار دنباله {۳-۱-۴-۲} و {۳-۴-۱-۲}و {۴-۳-۱-۲} و {۴-۱-۳-۲} به ترتیب با مجموع زمان تکمیل کارهای ۱۲۰، ۱۱۱، ۱۱۴، ۱۱۳ حاصل می شود.

جدول۳-۶ داده های مثال ۳-۶

کار / ماشین	J_1	J_2	J_3	J_4
M_1	۳	۱	۴	۲
M_2	۴	۳	۵	۸
M_3	۶	۵	۳	۹
M_4	۸	۹	۷	۴

لذا در مرحله یک دنباله {۳-۴-۱-۲} به عنوان جواب پایه در نظر گرفته می شود و درمرحله دوم ابتدا اولین کار از جواب پایه را به چهار موقعیت ممکن اضافه می کنیم لذا چهار دنباله {۳-۴-۱-۲} و {۴-۳-۲-۱} و {۴-۲-۳-۱} و {۴-۳-۲-۱} به ترتیب با مجموع زمان تکمیل کارهای ۱۱۱، ۱۲۶، ۱۳۲، ۱۲۸ حاصل می شود در نتیجه دنباله {۳-۴-۱-۲} به عنوان یکی از جواب های کاندیدا در نظر گرفته می شود. سپس دومین کار از جواب پایه را به چهار موقعیت ممکن اضافه می کنیم و دنباله های {۳-۴-۲-۱} و {۳-۴-۱-۲} و

{۳-۱-۴-۲} و {۱-۳-۴-۲} با مجموع زمان تکمیل کارهای ۱۲۶، ۱۱۱، ۱۲۰، ۱۱۵ به دست می آید . با ادامه روند فوق برای سومین و چهارمین کار در جواب پایه، بهترین جواب، دنباله {۳-۴-۱-۲} با مجموع زمان تکمیل کارهای ۱۱۱ حاصل می شود.



شکل ۳-۳ بهترین برنامه زمانی مثال ۳-۵

۳-۳-۲ الگوریتم WY

الگوریتم WY در سال ۱۹۹۸ توسط وو و ییم ارائه شد . این الگوریتم را می توان به صورت زیر بیان کرد.

گام یک : فرض می کنیم $k=1$ و U مجموعه همه کارهایی که هنوز تخصیص داده نشده است.

گا م دو : برای هر کار J_i، (i=۱، ۲، .... ، n) متوسط زمان تکمیل کار را طبق رابطه (۳-۴) محاسبه می کنیم، و کاری با مینمم متوسط زمان تکمیل کار انتخاب و آن را در مجموعه جواب جاری A قرار می دهیم . سپس کار انتخاب شده را از مجموعه U برمی داریم .

گام سه $k=k+1$

گام چهار: یک کار از مجموعه U انتخاب و آن را به k موقعیت ممکن در مجموعه جواب جاری اضافه می کنیم و از میان k دنباله ایجاد شده یکی را با کمترین مجموع زمان تکمیل کار جزئی انتخاب و دنباله فوق را به عنوان یک جواب کاندیدا در نظر می گیریم، و کار فوق را از مجموعه U بر می داریم.

گام پنج: اگر مجموعه U خالی نیست برو به گام چهار در غیر این صورت برو به گام شش

گام شش : دنباله ای با مینمم زمان تکمیل کار از میان جواب های کاندید انتخاب می کنیم و آن را به عنوان جواب جاری در مجموعه A قرار می دهیم و مجموعه U را نیز متمم مجموعه A قرار می دهیم.

گام هفت : اگر $k=n$ توقف کن در غیر این صورت به گام سه برو.

۳-۳-۳ الگوریتم الله وردی و الدویسان

در این بخش ما به طور خلاصه هفت روش ابتکاری که توسط الله وردی و الدویسان در سال ۲۰۰۲ ارائه شده است را شرح خواهیم داد و سپس با توجه به جدول ۳-۳-۶ و۳-۳-۷ نتایج مقایسه روش های فوق با روش WY و RZ را بیان می کنیم.

قبل از بیان هفت روش لازم است تعریف زیر را ارائه دهیم.

روش مبادله دو به دو : برای یک دنباله مفروض موقعیت یک جفت از کارها را مبادله می کنیم و تغییرات تابع هدف را محاسبه می کنیم اگربا این مبادله مقدار تابع هدف کاهش پیدا کند کار را در موقعیت جدیدش نگه می داریم در غیر این صورت به حالت قبلی بر می گردیم و این پروسه را برای همه جفت های ممکن از کارها انجام می دهیم.

سه روش ابتکاری اولیه عبارتند از IH_1و IH_2و IH_3 که بر پایه روش NEH بنا شده اند.

روش IH_1: بکاربردن روش مبادله دو به دو برای دنباله حاصل از روش NEH .

قبل از بیان روش $IH2$ ، ایده حلقه را تعریف می کنیم.

تعریف : همان طور که در روش NEH در بخش ۳-۲-۴ دیدید نتیجه این روش به دنباله آغازین وابسته است. انتظار می رود اگر از نتیجه به دست آمده از روش NEH به عنوان دنباله آغازین استفاده شود و سپس تکنیک الحاقی NEH را بکار بریم پیشرفتی در جواب مسئله حاصل شود که به آن به اصطلاح اولین حلقه گوییم . حال اگر جواب به دست آمده از اولین حلقه را به عنوان دنباله آغازین در نظر گرفته و سپس تکنیک الحاقی NEH را بکاربریم به آن به اصطلاح دومین حلقه گوییم.

روش IH_2 : الله وردی و الدویسان با تحقیق بر روی ایده حلقه برای ایجاد تعداد مختلف حلقه ها به این نتیجه رسیدند برای مسئله فوق با ایجاد پنج حلقه در حالت عمومی جواب مسئله تا حدودی بهبود بخشیده می شود. لذا روش IH_2 عبارت است از انتخاب بهترین دنباله از میان ۵ حلقه.

روش IH_3 : بکار بردن روش مبادله دو به دو برای نتیجه حاصل از روش IH_2.

روش IH_4 : بکار بردن روش مبادله دو به دو برای نتیجه حاصل از روش WY .

روش IH_5 : بکار بردن روش مبادله دو به دو برای نتیجه حاصل از روش RZ .

روش IH_6 : همان طور که در بخش ۳-۳-۱ بیان شد، روش RZ دارای دو مرحله است که در مرحله اول یک دنباله پایه به دست می آوریم و سپس در مرحله دوم دنباله پایه بهبود بخشیده می شود. لذا کیفیت جواب نهایی فقط به مرحله دوم وابسته نیست بلکه به دنباله به دست آمده در مرحله اول نیز وابسته است. در روش IH_6 نتیجه به دست آمده از روش WY را به عنوان جواب پایه در مرحله اول در نظر گرفته و سپس به مرحله دوم می رویم.

روش IH_7 : روش مبادله دو به دو را برای نتیجه حاصل از روش IH_6 بکار می بریم.

۴-۳-۳ مقایسه کارایی هفت روش الله وردی و الدویسان با دو روش RZ و WY

در این بخش نگاهی به کارایی روش های ابتکاری مطرح شده خواهیم داشت. الله وردی و الدویسان با استفاده از یک برنامه به زبان فرترن[1] به مقایسه هفت روش خود با دو روش RZ و WY پرداختند. بدین منظور زمان پردازش برروی هر ماشین را به صورت تصادفی بر پایهٔ توزیع یکنواخت گسسته در بازه (۱-۱۰۰)U در نظر گرفتند. البته برای مقایسه کارایی روش های ابتکاری آزمایشات در دو فاز انجام شده است. فاز اول شامل مسائلی با اندازه کوچک که در آن تعداد کارها برابر (۱۲ ، ۱۱ ، ۱۰ ، ۹ ، ۸ ، ۷) و تعداد ماشین ها برابر با (۲۰ ، ۱۵ ، ۱۰ ، ۵) می باشد، برای این مسائل جواب روش ابتکاری با جواب بهینه مقایسه شده است (جواب بهینه با بررسی تمام جایگشت های یک دنباله n تایی حاصل می شود). فاز دوم شامل مسائل با اندازه بزرگ که در آن تعداد کارها برابر با(۴۰۰،۳۰۰،۲۰۰،۱۰۰،۵۰،۵۰۰) و تعداد ماشین ها برابر با (۲۰ ، ۱۵ ، ۱۰ ، ۵) انتخاب شده است، و جواب روش ابتکاری با بهترین جواب مقایسه شده است. در هر دو فاز برای هر ترکیب از n کار و m ماشین ۵۰ مسئله در نظر گرفته شده است. برای مقایسه کارایی روش ابتکاری در مسائلی با اندازه کوچک از سه معیار استفاده شده است.

۱ - $ARPD$[2] (متوسط درصد خطای نسبی) : برابر است با

$$ARPD = \frac{100}{50}\sum_{i=1}^{50}\frac{heuristic_i - optimal_i}{optimal_i}.$$

مقدار $heuristic_i$ برابر است با مقدار تابع هدف (مجموع زمان تکمیل کارها) برای i امین مسئله با استفاده از الگوریتم ابتکاری و $optimal_i$ جواب بهینه مسئله است.

۲ - انحراف معیار[3] : برابر است با

[1] Fortran
[2] Average relative percentage deviation
[3] Standard deviation

$$\delta_x = \sqrt{\frac{\sum_{i=1}^{50}(heuristic_i - \overline{heuristic})^2}{50}}.$$

مقدار $\overline{heuristic}$ برابر با میانگین مقدار تابع هدف های بدست آمده برای ۵۰ مسئله می باشد.

۳ - تعداد دفعاتی که با استفاده از روش ابتکاری به جواب بهینه رسیده ایم.

برای مسائلی با اندازه بزرگ مقدار $ARPD$ برابر است با

$$ARPD = \frac{100}{50}\sum_{i=1}^{50}\frac{heuristic_i - Bestsolution_i}{Bestsolution_i}.$$

مقدار *Best solution* عبارت از بهترین جواب در میان روش های ابتکاری است.

جدول ۳-۷ نتایج مقایسه روش های ابتکاری برای مسائلی با اندازه کوچک

n	m	IH1			IH3			WY			IH4			RZ			IH5			IH7		
		Mean error (%)	Std. dev.	Opt (%)	Mean error (%)	Std. dev.	Opt (%)	Mean error (%)	Std. dev.	Opt (%)	Mean error (%)	Std. dev.	Opt (%)	Mean error (%)	Std. dev.	Opt (%)	Mean error (%)	Std. dev.	Opt (%)	Mean error (%)	Std. dev.	Opt (%)
7	5	0.721	1.225	44	0.514	0.824	50	1.123	1.370	24	0.701	1.258	44	0.492	0.969	50	0.408	0.926	58	0.194	0.385	62
	10	0.530	0.864	56	0.336	0.744	70	0.840	1.129	40	0.373	0.692	60	0.402	0.880	64	0.293	0.807	72	0.192	0.602	74
	15	0.506	0.770	46	0.305	0.522	54	0.769	0.859	40	0.448	0.717	54	0.314	0.559	50	0.220	0.507	58	0.142	0.372	72
	20	0.494	0.685	42	0.277	0.445	56	0.761	0.759	18	0.381	0.562	50	0.309	0.523	56	0.234	0.485	72	0.175	0.396	70
8	5	0.987	1.265	30	0.742	1.051	38	1.260	1.598	30	0.396	1.153	40	0.762	1.056	34	0.492	0.755	40	0.500	1.050	60
	10	1.010	1.159	28	0.641	0.890	38	1.035	0.912	20	0.582	0.644	36	0.582	0.910	54	0.449	0.727	60	0.319	0.648	68
	15	0.640	0.774	34	0.527	0.667	44	0.914	0.941	28	0.500	0.647	42	0.363	0.498	46	0.277	0.424	56	0.219	0.438	66
	20	0.546	0.675	32	0.345	0.523	46	0.913	1.064	20	0.452	0.625	40	0.402	0.543	42	0.261	0.390	48	0.276	0.517	60
9	5	1.039	1.207	28	0.891	1.145	32	1.506	1.443	20	0.907	1.189	32	0.994	1.206	28	0.716	1.096	44	0.484	0.820	54
	10	0.944	0.844	16	0.666	0.649	26	1.345	1.192	8	0.796	0.804	18	0.741	0.836	20	0.591	0.735	28	0.359	0.485	46
	15	0.827	0.963	16	0.565	0.663	18	1.142	0.795	6	0.720	0.708	20	0.557	0.511	20	0.417	0.486	34	0.378	0.541	44
	20	0.716	0.808	22	0.406	0.477	28	0.888	1.138	14	0.541	0.550	24	0.403	0.543	34	0.345	0.481	40	0.246	0.384	42
10	5	1.285	1.002	7	1.015	0.978	13	1.759	1.190	3	1.267	0.971	13	0.856	1.068	23	0.664	0.921	37	0.851	0.934	27
	10	1.276	0.803	3	0.849	0.716	10	1.627	1.416	7	0.990	0.711	7	1.000	0.800	17	0.809	0.764	17	0.424	0.574	43
	15	0.872	0.702	13	0.645	0.664	20	1.114	0.974	3	0.676	0.542	17	0.815	0.755	20	0.365	0.661	23	0.394	0.356	30
	20	0.520	0.707	13	0.509	0.493	23	1.166	1.039	3	0.454	0.518	23	0.579	0.616	23	0.341	0.527	37	0.285	0.337	27
11	5	1.469	1.358	10	1.145	1.123	17	1.776	1.276	10	1.132	1.027	17	1.259	1.267	17	1.007	1.194	27	0.678	0.819	33
	10	1.547	1.307	10	1.110	1.036	17	1.988	1.472	0	1.488	1.034	3	1.305	1.064	7	0.924	0.770	13	0.746	0.706	27
	15	1.311	1.132	17	0.654	0.661	27	1.593	1.173	10	1.137	1.117	23	1.176	0.924	7	0.973	0.824	10	0.553	0.528	27
	20	0.866	0.706	17	0.596	0.697	30	1.500	0.818	7	1.094	0.802	13	0.696	0.568	17	0.540	0.532	20	0.647	0.577	20
12	5	1.782	1.197	3	1.250	0.897	3	2.331	1.459	3	1.657	1.132	3	1.722	1.090	0	1.424	1.027	3	1.293	1.034	13
	10	1.949	1.157	0	1.509	1.073	7	2.002	1.220	3	1.809	1.176	3	1.222	0.890	3	1.014	0.761	3	0.903	0.904	17
	15	1.378	0.850	3	1.121	0.894	7	1.482	1.025	7	1.132	0.832	10	1.057	0.821	0	0.829	0.719	3	0.480	0.498	30
	20	1.012	0.756	3	0.504	0.646	7	1.319	0.886	0	0.953	0.715	0	1.031	0.644	0	0.844	0.638	7	0.547	0.552	10

جدول ۳-۸ نتایج مقایسه روش های ابتکاری برای مسائلی با اندازه بزرگ

n	m	IH1			IH3			WY			IH4			RZ			IH5			IH7		
		Mean error (%)	Std. dev.	Opt (%)	Mean error (%)	Std. dev.	Opt (%)	Mean error (%)	Std. dev.	Opt (%)	Mean error (%)	Std. dev.	Opt (%)	Mean error (%)	Std. dev.	Opt (%)	Mean error (%)	Std. dev.	Opt (%)	Mean error (%)	Std. dev.	Opt (%)
50	5	1.249	0.827	6	0.723	0.721	17	1.543	0.762	0	0.824	0.560	5	1.476	0.843	0	0.584	0.590	26	0.278	0.376	46
	10	1.918	1.124	2	0.966	0.691	12	1.872	0.927	0	1.180	0.724	2	1.378	0.824	0	0.673	0.695	27	0.348	0.521	52
	15	1.778	0.917	1	1.040	0.697	8	1.735	0.840	0	1.169	0.716	1	1.182	0.760	0	0.692	0.644	26	0.327	0.494	58
	20	1.634	0.862	2	0.952	0.619	8	1.665	0.797	0	1.041	0.629	2	1.204	0.796	1	0.765	0.697	21	0.272	0.405	57
100	5	0.539	0.503	3	0.453	0.375	12	1.212	0.463	0	0.599	0.352	2	1.581	0.707	0	0.529	0.442	15	0.103	0.221	69
	10	1.326	0.599	1	0.691	0.486	11	1.419	0.542	0	0.844	0.511	4	1.274	0.631	0	0.516	0.492	22	0.192	0.321	63
	15	1.501	0.806	1	0.649	0.569	5	1.407	0.625	0	0.920	0.572	1	1.169	0.636	0	0.498	0.514	35	0.259	0.418	58
	20	1.518	0.742	0	0.598	0.561	9	1.355	0.613	0	0.909	0.534	4	1.117	0.636	0	0.547	0.590	34	0.260	0.357	52
200	5	0.419	0.319	14	0.220	0.222	25	0.853	0.279	0	0.342	0.191	4	1.786	0.697	0	0.564	0.353	5	0.056	0.104	59
	10	0.852	0.436	4	0.414	0.334	17	0.983	0.330	0	0.520	0.296	3	1.320	0.559	0	0.459	0.458	24	0.153	0.251	53
	15	0.933	0.460	1	0.504	0.401	11	0.964	0.430	0	0.570	0.376	1	1.134	0.483	0	0.451	0.393	21	0.133	0.257	67
	20	1.130	0.499	1	0.622	0.428	9	0.951	0.377	0	0.603	0.324	0	1.028	0.421	0	0.388	0.365	21	0.118	0.215	69
300	5	0.310	0.250	14	0.155	0.186	36	0.671	0.191	0	0.235	0.148	1	2.172	0.634	0	0.676	0.345	3	0.060	0.092	55
	10	0.432	0.353	7	0.221	0.235	26	0.773	0.253	0	0.336	0.183	2	1.429	0.455	0	0.544	0.332	5	0.071	0.154	62
	15	0.606	0.358	4	0.302	0.248	16	0.759	0.289	0	0.433	0.258	1	1.153	0.399	0	0.401	0.350	16	0.100	0.173	63
	20	0.787	0.362	1	0.416	0.277	10	0.760	0.290	0	0.482	0.261	2	0.953	0.346	0	0.326	0.321	26	0.105	0.172	62
400	5	0.197	0.197	20	0.087	0.110	40	0.589	0.199	0	0.211	0.129	10	2.393	0.656	0	0.652	0.229	0	0.062	0.098	44
	10	0.295	0.223	18	0.106	0.166	56	0.756	0.220	0	0.360	0.173	0	1.497	0.447	0	0.567	0.318	0	0.116	0.139	40
	15	0.398	0.249	8	0.235	0.264	26	0.655	0.206	0	0.333	0.161	2	1.155	0.398	0	0.401	0.321	8	0.070	0.197	56
	20	0.694	0.391	2	0.380	0.227	4	0.609	0.265	0	0.355	0.268	0	1.089	0.412	0	0.430	0.320	15	0.080	0.198	61
500	5	0.144	0.171	20	0.066	0.097	46	0.579	0.181	0	0.215	0.124	6	2.345	0.711	0	0.667	0.279	0	0.065	0.081	40
	10	0.278	0.240	16	0.089	0.132	56	0.698	0.180	0	0.321	0.172	0	1.751	0.365	0	0.674	0.237	0	0.152	0.169	38
	15	0.349	0.274	10	0.139	0.174	36	0.580	0.232	0	0.322	0.214	0	1.281	0.378	0	0.453	0.300	4	0.061	0.170	54
	20	0.541	0.518	0	0.287	0.252	19	0.504	0.173	0	0.283	0.152	0	1.104	0.359	0	0.406	0.305	12	0.057	0.108	70

۳-۳-۵ الگوریتم فرامینان و لیستن

در این بخش ما ابتدا به بیان یکی از بهترین الگوریتم های ارائه شده برای حل مسئله زمانبندی کارگاه گردش کاری جایگشتی که توسط فرامینان و لیستن مطرح شده می پردازیم، و سپس در بخش ۳-۳-۵ الگوریتمی که توسط دیپاک لاها و سرین روش فرامینان را تا حد زیادی بهبود بخشیده مطرح و نتایج آزمایشات را مقایسه می کنیم.

الگوریتم فرامینان و لیستن دارای پنج گام به شرح ذیل می باشد.

گام یک : قرار می دهیم $k=1$.

- برای هر کار J_i (n،.........،۲ ،$i=1$) متوسط زمان تکمیل هر کار را طبق رابطه (۳-۴) محاسبه می کنیم. سپس کا رها را به صورت صعودی براساس متوسط زمان تکمیل هر کار مرتب می کنیم
- از میان n کار، کاری را با کمترین متوسط زمان تکمیل کار انتخاب و آن را در مجموعه جواب S قرار می دهیم، و $n-1$ کار باقیمانده را در مجموعه R به صورت صعودی قرار می دهیم.

گام دو : $k=k+1$

گام سه : کار $J\in R$ را انتخاب و آن را به k موقعیت ممکن در جواب جاری S اضافه می کنیم.

- از میان k دنباله ایجاد شده یکی را با کمترین مجموع زمان تکمیل کارهای جزئی انتخاب و دنباله فوق را یک جواب کاندیدا در نظر می گیریم و در مجموعه C قرار می دهیم.
- $S:=C$
- کار J را از مجموعه باقیمانده R برمی داریم.

گام چهار : اگر $k>۲$ برای همه i و j هایی که $۱\leq i<k$ و $i<j\leq k$ همهٔ دنباله های ممکن با کارهای متناوب در موقعیت i و j از دنباله جزئی C را به دست می آوریم.

- از میان $\frac{k(k-1)}{2}$ دنباله جزئی اگر یک دنباله $C^{'}$ موجود باشد به طوری که مجموع زمان تکمیل کار جزئی آن کمتر از S باشد قرار می دهیم $S=C$.

گام پنج : اگر مجموعه باقیمانده $R=\Phi$ $(k=n)$ توقف کن در غیر این صورت برو به گام سه.

مثال ۳-۷

مسئله پنج کار و پنج ماشین زیر را در نظر بگیرید

جدول ۳-۹ داده های مثال ۳-۷

ماشین \ کار	J_1	J_2	J_3	J_4	J_5
M_1	۱۶/۸۹۳۹	۷۸/۸۳۹۹	۳۱/۴۹۹۱	۵۲/۷۹۶۲	۱۷/۲۳۳۶
M_2	۵۹/۹۹۴۲	۲۶/۷۷۱۲	۶۵/۰۹۹۸	۶۸/۵۴۳۰	۷۴/۳۱۸۹
M_3	۴۵/۱۵۳۱	۹/۲۱۴۵	۲۳/۴۳۹۷	۹۰/۵۰۷۱	۱۵/۹۳۳۰
M_4	۸۱/۹۳۰۱	۵۳/۷۵۷۶	۹۸/۶۲۱۲	۸/۶۶۱۲	۴۴/۳۸۲۵
M_5	۱۱/۴۵۲۰	۹۵/۲۶۶۰	۱/۴۵۴۲	۷۶/۹۴۱۲	۸۱/۰۹۵۷

با حل مسئله با استفاده از الگوریتم فرامینان در گام یک، پس از محاسبه زمان پردازش هر کار و مرتب سازی آن دنباله {۴-۲-۵-۳-۱} حاصل می شود لذا S={۱} و R={۳-۵-۲-۴} . در قدم بعدی کار ۳ را به دو موقعیت ممکن در دنباله S اضافه می کنیم، لذا دو دنباله {۱-۳} و {۳-۱}، به ترتیب با مجموع زمان تکمیل کارهای ۵۳۲/۱۵ و ۵۱۹/۴۶ حاصل می شود. در نتیجه به ازای k=۲ S ={۳-۱} و R={۵-۲-۴} به دست می آید. حال کار ۵ را به سه موقعیت ممکن در دنباله S اضافه می کنیم. دنباله های {۳-۱-۵} و {۳-۵-۱}

و {۵-۳-۱} به ترتیب با مجموع زمان تکمیل کار جزئی ۹۰۱/۷۵ و ۸۹۳/۳۰ و ۹۴۷/۵۴ حاصل می شود و چون $k>3$ در گام چهار فرامینان با توجه به دنباله {۳-۵-۱}=S با مجموع زمان تکمیل کارهای جزئی ۸۹۳/۳۰ دنباله های {۳-۱-۵} و {۱-۵-۳} و{۵-۳-۱} به ترتیب با مجموع زمان تکمیل کار جزئی ۹۰۱/۷۵ و ۹۳۳/۶۹ و ۹۴۷/۵۴ تولید می شود. لذا به ازای k=۳ ، {۳-۵-۱}=S و {۲-۴}=R می باشد. با ادامه روند فوق برای الگوریتم فرامینان برای k=۴، در ابتدا دنباله های زیر {۳-۵-۱-۲} و {۳-۵-۲-۱} و {۳-۲-۵-۱} و {۲-۳-۵-۱} با مجموع زمان تکمیل کار جزئی ۱۴۲۳/۳ و۱۴۳۸/۱ و۱۳۹۵/۸ و۱۳۸۹/۳تولید می شود. لذا در این گام {۲-۳-۵-۱}=S و {۴}=R . و با اجرای گام چهار فرامینان برای k=۴ دنباله های {۲-۳-۱-۵} و {۲-۱-۵-۳} و {۱-۳-۵-۲} و{۲-۵-۳-۱} و{۵-۳-۲-۱} و {۳-۲-۵-۱} با مجموع زمان تکمیل کارهای ۱۴۲۸/۰ و ۱۴۴۰/۷ و۱۴۳۷/۸ و۱۴۷۰/۹ و ۱۴۰۸/۱ و ۱۳۹۵/۸تولید می شود. لذا در این گام جواب تغییری نمی کند. با ادامه روند فوق برای k=۵ به جواب نهایی {۵-۳-۲-۴-۱}=S با مجموع زمان تکمیل کارهای ۱۸۹۵/۰ می رسیم.

۳-۳-۵ الگوریتم دیپاک لاها و سرین

این الگوریتم که در سال ۲۰۰۹ توسط دیپاک لاها و سرین مطرح شد در واقع با اعمال تغییر در یکی از گام های الگوریتم فرامینان جواب مسئله را تا حدود زیادی بهبود می بخشد. این الگوریتم را می توان به صورت زیر بیان کرد.

گام یک : قرار می دهیم $k=1$.

- برای هر کار J_i (i=۱، ۲،.........n) متوسط زمان تکمیل هر کار را طبق رابطه (۳-۴) محاسبه می کنیم. سپس کا رها را براساس متوسط زمان تکمیل هر کار به صورت صعودی مرتب می کنیم.

- از میان n کار، کاری را با کمترین متوسط زمان تکمیل کار انتخاب و آن را در مجموعه جواب S قرار می دهیم، و $n-1$ کار باقیمانده را در مجموعه R به صورت صعودی قرار می دهیم.

گام دو : $k=k+1$

گام سه : کار $J \in R$ را انتخاب و آن را به k موقعیت ممکن در جواب جاری S اضافه می کنیم.

- از میان k دنباله ایجاد شده یکی را با کمترین مجموع زمان تکمیل کارهای جزئی انتخاب و دنباله فوق را یک جواب کاندیدا در نظر می گیریم و در مجموعه C قرار می دهیم.
- $S:=C$
- کار J را از مجموعه باقیمانده R برمی داریم.

گام چهار : اگر $k>2$ هر کار از مجموعه C، (به جز kامین کار اضافه شده) $k-1$ موقعیت برای جابجایی دارد.

- از میان $(k-1)^2$ دنباله جزئی به دست آمده اگر یک دنباله C' موجود باشد به طوری که مجموع زمان تکمیل کار جزئی آن کمتر از S باشد قرار می دهیم $S=C$

گام پنج : اگر مجموعه باقیمانده $R=\Phi$ $(k=n)$ توقف کن در غیر این صورت برو به گام سه.

همان طور که مشاهده می کنید این الگوریتم در گام یک ، دو و سه همانند الگوریتم فرامینان است و فقط در گام چهار با یکدیگر تفاوت دارند.

مثال ۳-۸

با حل مثال ۳-۶ با استفاده از الگوریتم دیپاک لاها به جواب نهایی {۳-۴-۱-۲-۵} با مجموع زمان تکمیل کارهای ۱۷۹۳/۸ می رسیم. البته جواب فوق جواب بهینه برای مسئله فوق می باشد.

۳-۳-۷ مقایسه کارایی روش فرامینان و دیپاک لاها

با استفاده از کدهای *Matlab* برنامه دو روش فرامینان و دیپاک لاها برروی کامپیوتر شخصی تهیه شده است، که در این بخش به مقایسه روش دیپاک لاها و روش فرامینان می پردازیم. بدین منظور برای مقایسه کارایی، آزمایشات را در دو فاز انجام داده ایم. فاز اول شامل مسائلی با اندازه کوچک که در آن تعداد کارها برابر با ۸ ، ۷ ، ۶ ، ۵ =n و تعداد ماشین ها برابر با ۲۰ ، ۱۵ ، ۱۰ ، ۵ =m است. فاز دوم شامل مسائلی با اندازه بزرگ که در آن تعداد کارها برابر با ۵۰ ، ۴۰ ، ۳۰ ، ۲۰ ، ۱۰ =n و تعداد ماشین ها برابر با ۲۰ ، ۱۵ ، ۱۰ ، ۵ =m می باشد. برای هر ترکیب از n کار و m ماشین، بیست مسئله در نظر گرفته ایم که زمان پردازش هر کار برروی هر ماشین را به صورت تصادفی بر پایه توزیع یکنواخت گسسته U(۱-۹۹) به دست آورده ایم. برای مقایسه کارایی دو روش، دو معیارزیر را بکار برده ایم:

۱- $ARPD$. همان طور که قبلا به آن اشاره شد برابر است با

$$ARPD = \frac{100}{20}\sum_{i=1}^{20}\left(\frac{Heuristic_i - Best_i}{Best_i}\right).$$

در رابطه فوق مقدار $Heuristic_i$ برابر است با مقدار تابع هدف برای i امین مسئله با استفاده از الگوریتم ابتکاری و مقدار $Best_i$ برای مسائل با اندازه کوچک همان جواب بهینه مسئله است و برای مسائل با اندازه بزرگ برابر با بهترین جواب میان دو روش فوق برای i امین مسئله است.

۲- تعداد دفعاتی که با استفاده از روش ابتکاری به جواب بهینه برای مسائل با ا ندازه کوچک و بهترین جواب برای مسائل با اندازه بزرگ می رسیم.

جدول ۳-۹ و ۳-۱۰ نشان از برتری روش دیپاک لاها و سرین نسبت به روش فرامینان به خصوص برای مسائلی با اندازه بزرگ را دارد.

جدول ۳-۱۰ مقایسه روش فرامینان و دیپاک لاها برای مسائلی با اندازه کوچک

n	*m*	*No. of Problem*	*Faraminan and Listen* ARPD	*Faraminan and Listen* No. of Optimal.	*Dipak laha And Sarin* ARPD	*Dipak laha And Sarin* No. of Optimal.
۵	۵	۲۰	۰/۱۲۱۴	۱۷	۰/۰۴۱۹	۱۸
	۱۰	۲۰	۰/۰۵۲۵	۱۷	۰/۰۲۳۳	۱۸
	۱۵	۲۰	۰/۱۳۹۳	۱۷	۰	۲۰
	۲۰	۲۰	۰/۲۱۵۵	۱۲	۰/۱۰۰۲	۱۷
۶	۵	۲۰	۰/۴۰۵۸	۱۱	۰/۱۷۹۶	۱۶
	۱۰	۲۰	۰/۴۳۷۲	۱۱	۰/۰۱۴۳	۱۶
	۱۵	۲۰	۰/۲۹۷۰	۱۳	۰/۱۸۷۸	۱۶
	۲۰	۲۰	۰/۱۱۲۶	۱۵	۰/۰۷۸۲	۱۷
۷	۵	۲۰	۰/۴۲۸۵	۹	۰/۱۴۵۱	۱۵
	۱۰	۲۰	۰/۶۶۹۹	۹	۰/۱۹۲۶	۱۶
	۱۵	۲۰	۰/۲۶۸۴	۱۲	۰/۱۴۵۳	۱۵
	۲۰	۲۰	۰/۱۷۵۳	۱۳	۰/۰۸۳۶	۱۵
۸	۵	۲۰	۱/۲۳۱۹	۸	۰/۳۱۰۶	۱۳
	۱۰	۲۰	۰/۷۰۹۰	۷	۰/۶۲۵۰	۶
	۱۵	۲۰	۰/۴۷۷۸	۹	۰/۲۴۲۲	۱۰
	۲۰	۲۰	۰/۳۶۷۵	۶	۰/۱۰۰۰	۱۴

جدول۳- ۱۱ مقایسه روش فرامینان و دیپاک لاها برای مسائلی با اندازه بزرگ

n	m	No.of problem	Faraminan and Listen ARPD	Faraminan and Listen No.of bestsoulation	Dipak laha And sarin ARPD	Dipak laha And sarin No. of bestsoulation
۱۰	۵	۲۰	۰/۶۱۱۷	۸	۰/۱۱۳۶	۱۷
	۱۰	۲۰	۰/۵۰۲۵	۵	۰/۰۰۱۳	۱۹
	۱۵	۲۰	۰/۶۳۲۵	۷	۰/۰۶۲۶	۱۹
	۲۰	۲۰	۰/۲۱۷۳	۸	۰/۰۹۳۰	۱۸
۲۰	۵	۲۰	۰/۷۲۵۱	۴	۰/۱۶۱۳	۱۶
	۱۰	۲۰	۰/۸۵۹۶	۳	۰/۰۸۸۱	۱۷
	۱۵	۲۰	۰/۸۱۶۶	۲	۰/۰۱۹۹	۱۸
	۲۰	۲۰	۰/۶۸۵۸	۳	۰/۰۵۰۸	۱۷
۳۰	۵	۲۰	۰/۴۴۹۷	۷	۰/۲۳۷۶	۱۳
	۱۰	۲۰	۰/۷۵۶۱	۳	۰/۰۴۰۷	۱۷
	۱۵	۲۰	۱/۰۰۱۲	۴	۰/۰۴۱۶	۱۶
	۲۰	۲۰	۰/۹۵۶۵	۲	۰/۰۵۴۲	۱۸
۴۰	۵	۲۰	۰/۹۹۸۸	۴	۰/۱۲۸۱	۱۶
	۱۰	۲۰	۱/۱۹۳۱	۳	۰/۱۵۵۰	۱۷
	۱۵	۲۰	۰/۹۶۹۵	۲	۰/۱۰۳۸	۱۸
	۲۰	۲۰	۱/۰۷۲۱	۲	۰/۰۶۳۷	۱۸
۵۰	۵	۲۰	۰/۳۰۸۱	۸	۰/۳۴۹۴	۱۲
	۱۰	۲۰	۱/۵۸۱۶	۳	۰/۰۸۷۴	۱۷
	۱۵	۲۰	۱/۰۸۹۲	۱	۰/۰۱۸۴	۱۹
	۲۰	۲۰	۰/۷۰۲۱	۴	۰/۰۶۳۹	۱۶

فصل چهارم

ارائه دو الگوریتم ابتکاری برای حل مسئله زمانبندی کارگاه گردش کاری جایگشتی

۴-۱ مقدمه

همانطور که قبلا اشاره شد حل مسئله زمانبندی کارگاه گردش کاری جایگشتی در سال های اخیر برای دانشمندان و محققان جذابیت بسیاری داشته و الگوریتم های ابتکاری و فرا ابتکاری بسیاری در سالهای اخیر برای حل مسئله فوق با معیار های متفاوت ارائه شده است.

در این فصل نیز دو الگوریتم ابتکاری برای مسئله فوق با معیار کمینه کردن مجموع زمان تکمیل کارها ارائه شده است که در ادامه به تشریح الگوریتم ها پرداخته و سپس به ارزیابی دو مدل ارائه شده با روش فرامینان و لیستن که در بخش ۳-۳-۵ بدان اشاره شده است می پردازیم.

شایان ذکر است که برای دو روش پیشنهادی و روش فرامینان و لیستن با استفاده از کدهای ***Matlab*** برنامه ای نوشته شده که در پیوست آورده شده است.

۴-۲ الگوریتم های پیشنهادی

روش پیشنهاد شده توسط فرامینان و لیستن بر اساس زمانبندی جزئی است که برای اولین بار توسط ناواز و همکاران در سال ۱۹۸۳ به نام روش *NEH* مطرح شده بود (بخش ۳-۲-۴) . یعنی از یک حدس اولیه برای زمانبندی اولین کار استفاده و سپس در هر تکرار، کاری جدید به دنباله اضافه می شود تا در مرحله n ام، به یک دنباله n تایی می رسیم. دو روش پیشنهادی ما نیز از این ایده تبعیت می کنند.

نکته قابل توجه این است که روش های پیشنهادی با روش *WY* که توسط وو و ییم (بخش ۳-۳-۲) ارائه شده است شباهت دارد، با این تفاوت که روش *WY* در هر تکرار یک دنباله جزئی حاصل می شود ولی با اعمال یک محدودیت در روش پیشنهادی اول که با نام *MN1* نام گذاری شده حداقل یک دنباله جزئی و در روش پیشنهادی دوم با نام MN2، در هر تکرار سه دنباله جزئی حاصل می شود.

۴-۲-۱ الگوریتم $MN1$

گام یک: قرارمی دهیم k=۱ و برای هر کار J_i $(i=1 ، 2 ،...... n)$ ، متوسط زمان تکمیل هرکار را محاسبه می کنیم

$$T_i = \sum_{j=1}^{m} t_{ij} \qquad i = 1,2........n.$$

و از میان n کار، کاری را با مینمم متوسط زمان تکمیل هرکار انتخاب و در مجموعه A قرار می دهیم.

گام دو : $k=k+1$

گام سه : برای هر کدام از دنباله های مجموعه A ، متمم آن (A') را تشکیل می دهیم.

گام چهار : یک کار از مجموعه A' انتخاب و آن را به k موقعیت ممکن در دنباله متناظر با آن درمجموعه A اضافه می کنیم و دنباله های ایجاد شده را در مجموعه R قرار می دهیم و کار فوق را از مجموعه A' بر میداریم.

گام پنج : اگر مجموعه A' خالی نیست برو به گام چهار در غیر این صورت A را خالی کن برو به گام شش.

گام شش : برای هر کدام از دنباله های مجموعه R مقدار مجموع زمان تکمیل کارها را محاسبه می کنیم، و مقدار مجموع زمان تکمیل کارها، برای دنباله i ام در مجموعه R را برابر با TFT_i و کمترین مقدار مجموع زمان تکمیل کارها در مجموعه R را $min\ (TFT_i)$ در نظر می گیریم.

گام هفت : برای دنباله های موجود در مجموعه R دنباله هایی که در شرط $\frac{TFT_i}{\min(TFT_i)} \le 1+\frac{\varepsilon}{2^{k-2}}$ صدق می کند را در مجموعه A قرار دهید.

گام هشت: اگر $k < n$ برو به گام سه در غیر این صورت از دنباله های مجموعه A یکی را با کمترین مجموع زمان تکمیل کارها انتخاب کن.

مقدار ε را طی ازمایشات انجام شده برای n های کوچک (۸ ، ۷ ، ۶ ،۵ = n) برابر با ۰/۱ در نظر می گیریم، البته هر چه مقدار ε، بیشتر باشد جواب مسئله بهتر ولی زمان انجام محاسبات طولانی تر می شود.

مثال ۴-۱

مساله چهار کار و چهار ماشین زیر را در نظر بگیرید.

جدول ۴-۱ داده های مثال ۴-۱

کار / ماشین	J_1	J_2	J_3	J_4
M_1	۴۱/۸۹۲۲	۵/۸۶۶۱	۸۹/۴۶۶۲	۹۳/۵۸۹۱
M_2	۴۹/۱۰۴۷	۴۸/۹۴۶۸	۳۴/۰۹۶۵	۸۹/۲۰۵۳
M_3	۳۷/۱۸۶۲	۱۱/۸۹۷۹	۷۷/۴۶۴۷	۳۹/۱۹۴۴
M_4	۹۳/۵۸۹۱	۸۹/۲۰۵۳	۳۹/۱۹۴۴	۱۳/۹۳۳۴

با اجرای گام به گام الگوریتم پیشنهادی در گام یک داریم A={۲}.

لذا برای ۲=k، {۴، ۳، ۱}=A' و مجموعه R برابر است با

R	TFT	R	TFT
{۱، ۲}	۳۴۶/۳۲۱۰	{۲، ۳}	۳۲۴/۶۴۰۲
{۲، ۱}	۲۷۳/۰۸۳۷	{۴، ۲}	۵۲۰/۱۴۴۷
{۳، ۲}	۴۶۴/۹۸۸۶	{۲، ۴}	۳۴۹/۰۸۲۷

که تنها دنباله {۱، ۲ } با مجموع زمان تکمیل کارهای ۲۷۳/۰۸۳۷ در شرط گام شش صدق می کند. لذا برای k=۳ و {۴، ۳ }=A' داریم

R	TFT	R	TFT
{۱، ۲، ۳}	۷۴۸/۴۷۴۶	{۱، ۲، ۴}	۸۶۲/۸۶۲۴
{۱، ۳، ۲}	۵۹۳/۴۰۵۷	{۱، ۴، ۲}	۶۴۸/۷۱۹۷
{۳، ۱، ۲}	۵۳۲/۳۲۱۹	{۴، ۱، ۲}	۵۵۶/۷۶۴۲

که دو دنباله {۳، ۱، ۲} و {۴، ۱، ۲} در شرط گام شش صدق می کند لذا {{۴، ۱، ۲}، {۳، ۱، ۲}}=A و {{۳}، {۴}}=A'. برای k=۴ نیز داریم:

R	TFT	R	TFT
{۳، ۱، ۲، ۴}	۱۲۶۸/۸	{۴، ۱، ۲، ۳}	۱۱۲۱/۶
{۳، ۱، ۴، ۲}	۱۰۱۱/۶	{۴، ۱، ۳، ۲}	۹۶۶/۶
{۳، ۴، ۱، ۲}	۵۹۱۴/۴	{۴، ۳، ۱، ۲}	۹۰۵/۵
{۴، ۳، ۱، ۲}	۹۰۵/۵	{۳، ۴، ۱، ۲}	۹۱۴/۴

که از میان دنباله های بالا، دنباله {۴، ۳، ۱، ۲} با مجموع زمان تکمیل کارهای ۹۰۵/۵ به عنوان بهترین جواب انتخاب می شود.

البته با بررسی تمام جایگشت های چهارتایی، جواب فوق جواب بهینه مسئله است.

۴-۲-۲ الگوریتم *MN2*

این روش در گام های اول تا ششم همانند الگوریتم *MN1* می باشد و تنها تفاوت آن در اجرای گام هفت است. لذا الگوریتم را می توان به صورت زیر بیان کرد.

گام یک: قرارمی دهیم k=۱ و برای هر کار J_i (i=۱ ، ۲،......n) متوسط زمان تکمیل هر کار را محاسبه می کنیم

$$T_i = \sum_{j=1}^{m} t_{ij} \qquad\qquad i = 1,2........n.$$

و از میان n کار، کاری را با مینمم متوسط زمان تکمیل هرکار انتخاب و در مجموعه A قرار می دهیم.

گام دو : k=k+۱

گام سه : برای هر کدام از دنباله های مجموعه A ، متمم آن را (A') تشکیل میدهیم.

گام چهار : یک کار از A' انتخاب و آن را به k موقعیت ممکن در دنباله متناظر با آن در A اضافه می کنیم و دنباله های ایجاد شده را در مجموعه R قرار می دهیم و کار فوق را از مجموعه A' بر میداریم.

گام پنج : اگر A' خالی نیست برو به گام چهار در غیر این صورت A را خالی کن برو به گام شش.

گام شش : برای هر کدام از دنباله های مجموعه R مقدار مجموع زمان تکمیل کارها را محاسبه می کنیم.

گام هفت : از میان دنباله های مجموعه R سه دنباله با کمترین مجموع زمان تکمیل کارها انتخاب و در مجموعه A قرار می دهیم.

گام هشت : اگر $k < n$ برو به گام سه در غیر این صورت از دنباله های مجموعه A یکی را با کمترین مجموع زمان تکمیل کارها انتخاب کن.

مثال ۴-۲

با حل مثال ۴-۱ با استفاده از دومین الگوریتم پیشنهادی ، روند حل مسئله برای k=۱ همانند مثال قبل می باشد و برای k=۲ دنباله های {{۱، ۲}، {۲، ۳}،{۲، ۱}}=A در شرط گام شش صدق می کند لذا {{۴، ۳}،{۱، ۴}،{۳، ۴}} = A'.

برای k=۳ داریم :

R	*TFT*	*R*	*TFT*
{۳، ۲، ۱}	۷۴۸/۴۷۴۷	{۲، ۱ ، ۴}	۵۵۶/۷۶۴۲
{۲، ۳، ۱}	۵۹۳/۴۰۵۶	{۱، ۲ ، ۳}	۶۰۸/۲۷۸۴
{۲، ۱، ۳}	۵۳۲/۳۲۱۹	{۴، ۲، ۳}	۸۷۳/۸۹۹۶
{۴، ۲، ۱}	۸۶۲/۸۶۲۵	{۲، ۴، ۳}	۶۶۴/۸۵۴۶
{۲، ۴، ۱}	۶۴۸/۷۱۹۶	{۲، ۳، ۴}	۶۵۵/۸۹۴۷
{۳، ۱، ۲}	۷۷۷/۸۶۱۹	{۱، ۴ ، ۲}	۷۵۶/۷۹۷۹
{۱، ۳، ۲}	۷۰۱/۶۴۱۸	{۱، ۲، ۴}	۶۳۰/۰۰۱۵
{۴، ۱، ۲}	۸۶۴/۰۴۷۶		

دنباله های {{۱، ۳، ۲}، {۴، ۱، ۲}، {۳، ۱، ۲}}=A سه دنباله ای هستند که در شرط گام شش صدق می کند. لذا {{۴}، {۳}، {۴}}=A'

و در مرحله نهایی برای k=۴ داریم :

R	*TFT*	*R*	*TFT*
{۴، ۲، ۱، ۳}	۱۲۶۸/۸	{۲، ۳، ۱، ۴}	۹۶۶/۶
{۲، ۴، ۱ ،۳}	۱۰۱۱/۶	{۴، ۲ ، ۳، ۱}	۱۲۷۹/۱
{۲، ۱، ۴، ۳}	۹۱۴/۴	{۲، ۴، ۳، ۱}	۱۰۳۲
{۲، ۱، ۳، ۴}	۹۰۵/۵	{۲، ۳، ۴، ۱}	۱۰۴۵
{۳، ۲، ۱، ۴}	۱۱۲۱/۶		

لذا جواب نهایی روش فوق دنباله {۴، ۳، ۱، ۲} با مجموع زمان تکمیل کارهای ۹۰۵/۵ می باشد.

۴-۳ مقایسه کارایی روش های ارائه شده

با استفاده از کدهای *Matlab* برنامه دو روش پیشنهادی و روش فرامینان و لیستن برروی کامپیوتر شخصی تهیه شده است که در پیوست C کدهای آن را آورده ایم . در جدول ۴-۲ و ۴-۳ نتایج مقایسه اولین روش پیشنهادی و دومین روش پیشنهادی با روش فرامینان و لیستن آورده شده است. برای هر ترکیب از n کار و m ماشین، بیست مسئله در نظر گرفته ایم که زمان پردازش هر کار برروی هر ماشین را به صورت تصادفی بر پایه توزیع یکنواخت گسسته U(۱-۹۹) به دست می آوریم.

برای مقایسه کارایی دو روش، دو معیارزیر را بکار برده ایم.:

۱- $ARPD$ (متوسط درصد خطای نسبی). همان طور که قبلا به آن اشاره شد برابر است با

$$ARPD = \frac{100}{20}\sum_{i=1}^{20}\left(\frac{Heuristic_i - Best_i}{Best_i}\right).$$

در رابطه فوق مقدار $Heuristic_i$ برابر است با مقدار تابع هدف برای i امین مسئله با استفاده از الگوریتم ابتکاری و مقدار $Best_i$ برای مسائل با اندازه کوچک همان جواب بهینه مسئله است و برای مسائل با اندازه بزرگ برابر با بهترین جواب در میان دو روش (روش پیشنهادی و روش فرامینان) برای i امین مسئله است.

۲- تعداد دفعاتی که با استفاده از روش ابتکاری به جواب بهینه برای مسائلی با اندازه کوچک و بهترین جواب برای مسائلی با اندازه بزرگ می رسیم.

۴-۳-۱ مقایسه اولین روش پیشنهادی با روش فرامینان و لیستن

همانطور که در جدول ۴-۲ مشاهده می کنید، برای مسائل با اندازه کوچک که در آن تعداد کارها برابر با ۸ ، ۷ ، ۶ ، ۵ =n و تعداد ماشین ها برابر با ۲۰ ، ۱۵ ، ۱۰ ، ۵ =m است ، برتری اولین روش پیشنهادی مشهود است و تقریبا در اغلب مسائل با اندزه کوچک به جواب بهینه می رسیم. ولی یکی از مشکلات اساسی این روش برای مسائلی با سایز بزرگ می باشد. چون این روش وابسته به مقدار ε است لذا با در نظر گرفتن ۰/۱ =ε برای مسائلی با اندازه کوچک به نتیجه قابل قبولی می رسیم ولی برای مسائلی با اندازه بزرگ اگر مقدار ۰/۱ =ε باشد زمان انجام محاسبات نسبت به روش فرامینان بیشتر می شود و اگر مقدار ۰/۱< ε در نظر گرفته شود از دقت جواب کاسته می شود.

لذا روش فوق برای مسائل با اندازه کوچک نسبت به روش فرامینان برتری کامل دارد و برای مسائلی با اندازه بزرگ، شاید استفاده از روش فرامینان بخصوص در شرایطی که معیار زمان نیز مطرح می باشد، بهتر است.

جدول ۴-۲: نتایج مقایسه اولین روش پیشنهادی با روش فرامینان برای مسائلی با اندازه کوچک

n	*m*	*No.of problem*	*Faraminan and Listen* ARPD	*Faraminan and Listen* No.of optimal	*Proposed Method* ARPD	*Proposed Method* No. of Optimal
۵	۵	۲۰	۰/۲۵۶۵	۱۶	۰	۲۰
	۱۰	۲۰	۰/۱۴۲۲	۱۷	۰	۲۰
	۱۵	۲۰	۰/۱۰۸۳	۱۵	۰	۲۰
	۲۰	۲۰	۰/۰۱۴۵	۱۸	۰	۲۰
۶	۵	۲۰	۰/۲۸۸۵	۱۵	۰	۲۰
	۱۰	۲۰	۰/۳۸۱۱	۱۰	۰/۰۲۶۱	۱۹
	۱۵	۲۰	۰/۳۳۵۸	۱۴	۰	۲۰
	۲۰	۲۰	۰/۳۲۱۰	۱۳	۰	۲۰
۷	۵	۲۰	۰/۶۴۲۲	۸	۰/۰۲۷۸	۱۹
	۱۰	۲۰	۰/۳۶۳۳	۱۱	۰/۰۰۹۵	۱۹
	۱۵	۲۰	۰/۲۶۲۰	۱۰	۰	۲۰
	۲۰	۲۰	۰/۴۹۳۲	۱۰	۰	۲۰
۸	۵	۲۰	۱/۲۷۲۰	۷	۰/۱۵۵۴	۱۶
	۱۰	۲۰	۱/۰۳۲۰	۵	۰/۰۳۴۸	۱۹
	۱۵	۲۰	۰/۵۵۲۵	۶	۰/۰۲۹۵	۱۸
	۲۰	۲۰	۰/۳۶۸۹	۹	۰	۲۰

۴-۳-۲ مقایسه دومین روش پیشنهادی با روش فرامینان و لیستن

در جدول ۴-۳ نتایج مقایسه دومین روش پیشنهادی با روش فرامینان نشان داده شده است. همانطور که مشاهده می کنید برای مسائل با اندازه کوچک روش فوق نسبت به روش فرامینان برتری دارد ولی با بزرگ شدن اندازه مسئله (n=۸) از دقت جواب کاسته می شود لذا برای مسائلی با اندازه بزرگ (۱۰< n) چون جواب روش فوق، همسطح و در بعضی موارد ضعیفتر از روش فرامینان بود، از آوردن نتایج آزمایشات خودداری کردیم.

البته به نظر می رسد روش فوق هر چند نسبت به روش فرامینان برتری کامل ندارد ولی نسبت به سایر روش هایی که در فصل سوم بدان اشاره کرده ایم برتری کامل دارد و تنها هدف ما از مقایسه این دو روش با روش فرامینان، جدیدتر بودن روش فرامینان نسبت به سایر روش هایی بود که در فصل سوم مطرح شده است.

جدول ۴-۳: نتایج مقایسه دومین روش پیشنهادی با روش فرامینان برای مسائلی با اندازه کوچک

n	*m*	*No.of problem*	*Faraminan and Listen*		*proposed method*	
			ARPD	*No.of optimal*	*ARPD*	*No. of Optimal*
۵	۵	۲۰	۰/۲۴۸۶	۱۴	۰	۲۰
	۱۰	۲۰	۰/۱۱۸۰	۱۶	۰/۰۳۱۴	۱۸
	۱۵	۲۰	۰/۰۶۸۱	۱۶	۰/۰۱۳۹	۱۸
	۲۰	۲۰	۰/۰۳۲۷	۱۷	۰/۰۲۳۱	۱۹
۶	۵	۲۰	۰/۴۲۸۴	۱۲	۰/۱۶۶۵	۱۸
	۱۰	۲۰	۰/۴۲۰۵	۱۲	۰/۱۱۷۳	۱۵
	۱۵	۲۰	۰/۰۴۳۵	۱۷	۰/۰۳۸۴	۱۸
	۲۰	۲۰	۰/۱۳۱۴	۱۲	۰/۰۹۸۲	۱۷
۷	۵	۲۰	۰/۵۰۵۲	۱۰	۰/۳۹۵۷	۱۴
	۱۰	۲۰	۰/۶۳۹۳	۱۰	۰/۱۹۵۷	۱۷
	۱۵	۲۰	۰/۵۴۷۴	۹	۰/۲۵۸۷	۱۲
	۲۰	۲۰	۰/۲۹۱۲	۹	۰/۰۹۵۸	۱۴
۸	۵	۲۰	۰/۹۸۸۴	۹	۰/۴۸۰۴	۱۱
	۱۰	۲۰	۰/۵۳۴۰	۱۰	۰/۲۸۶۴	۱۱
	۱۵	۲۰	۰/۵۹۹۶	۵	۰/۳۱۰۹	۱۲
	۲۰	۲۰	۰/۳۲۰۹	۷	۰/۱۷۱۱	۱۱

فصل پنجم

حل مسئله زمانبندی کارگاه گردش کاری جایگشتی با استفاده از الگوریتم های فراابتکاری

۵- ۱ مقدمه و تاریخچه

در فصل سوم مروری به حل مسئله زمانبندی کارگاه گردش کاری جایگشتی با استفاده از روش‌های ابتکاری داشتیم و در این فصل هدف ما، حل مسئله فوق با معیار کمینه کردن زمان اتمام کل سیستم و کمینه کردن مجموع زمان تکمیل کارها با استفاده از الگوریتم‌های فراابتکاری[1] می‌باشد.

اشکال اساسی روش‌های ابتکاری این است که این روش‌ها در حالت عمومی خیلی سریع به جواب می‌رسند ولی از لحاظ کیفی، جواب روش‌های فوق نامرغوب است لذا در سال‌های اخیر روش‌های فراابتکاری مختلفی برای حل مسئله زمانبندی کارگاه گردش کاری جایگشتی توسط محققان پیشنهاد شده که از آن جمله می‌توان به :

۱-بازپخت شبیه سازی شده[2]، (*SA*) : اگبو[3] و اسمیت[4] (۱۹۹۰)، ایشاباشی[5] و همکاران (۱۹۹۵)

۲- جستجوی تابو[6] (*TS*) : تیلارد [7] (۱۹۹۰)

---

[1] metaheuristics

[2] Include simulated annealing

[3] ogub

[4] simith

[5] Ishubushi

[6] Tabu search

[7] Taillard

۳- الگوریتم ژنتیک[1] (GA) : ریورز[2] (۱۹۹۵)

۴- الگوریتم‌های فراابتکاری پیوندی[3] : ریورز[4] و یامادا[5] (۱۹۹۸) ، ونگ[6] و ژنگ[7] (۲۰۰۳)

با معیار کمینه کردن زمان اتمام کل سیستم اشاره کرد.

کلیه روش‌های TS, SA, GA پارامتری هستند و باید برای هر مسئله پارامترها مشخص شوند. بزرگترین اشکال این روش‌ها، طولانی بودن زمان حل مسائل نسبت به روش‌های ابتکاری است.

البته برای حل مسئله با معیار کمینه کردن مجموع زمان تکمیل کارها نیز روش‌های فراابتکاری بسیاری ارائه شده که می‌توان به الگوریتم محلی ژنتیک یامادا و ریورز (۱۹۹۸) اشاره کرد. البته الگوریتم مذکور از لحاظ کیفی مرغوب ولی زمان محاسبه آن بسیار طولانی می‌باشد. در سال ۲۰۰۹، لین- یو تسنگ[8] و یا- تی لین[9] الگوریتم جستجوی محلی ژنتیک پیوندی را برای حل مسئله با هر دو معیار ارائه دادند که الگوریتم پیشنهادی، الگوریتم ژنتیک و یک طرح جستجو محلی[10] جدید را با هم ترکیب می‌کند.

---

[1] Gentic algorithm

[2] Reeves

[3] Hybrid metaheuristic

[4] Reeves

[5] Yamada

[6] Wang

[7] Zhang

[8] Lin-Yu Tseng

[9] Ya -Tai Lin

[10] Local search

در این فصل به تشریح الگوریتم فوق می‌پردازیم. در انتها نتایج آزمایشگاهی مقایسه روش فوق را با چند مورد از بهترین روش هایی که در سالهای اخیر مطرح شده است ارائه می دهیم. نتایج آزمایشگاهی بیانگر کارایی بیشتر الگوریتم محلی ژنتیک پیوندی می‌باشد.

۵-۲ ساختار کلی الگوریتم ژنتیک

الگوریتم ژنتیک یک نوع از روش‌های جستجوی تصادفی هستند که به ساز و کار انتخاب و ژنتیک وابسته است و بر خلاف روش‌های جستجوی متداول، با یک مجموعه اولیه از جوابهای تصادفی که آن را جمعیت[1] اولیه می‌نامیم شروع می‌کند. هر عضو جمعیت را یک کروموزوم[2] می‌نامیم که بیانگر یک جواب مسئله است.

رشته یا کروموزوم: نمایانگر یک نقطه در فضای جستجو است که ممکن است به صورت کد و یا به شکل اصلی متغیرهای موردنظر باشد. هر کروموزوم باید نشان‌دهنده یک جواب کامل برای مسئله باشد. گنجاندن اطلاعات اضافی در رشته باعث کاهش سرعت عمل الگوریتم می‌شود.

جمعیت: الگوریتم ژنتیک برخلاف سایر روش‌ها در هر مرحله، از چند نقطه برای انجام عملیات جستجو استفاده می‌کند. کروموزم های مورد استفاده در هر مرحله یک جمعیت تشکیل می‌دهند. این جمعیت می‌تواند با جمعیت مرحله قبل همپوشانی داشته باشد. به این معنی که در هر مرحله، بعضی از اعضای جمعیت قبلی بدون تغییر وارد جمعیت جدید می‌شوند. معمولا جمعیت اولیه به صورت تصادفی از میان جواب‌های مسئله تشکیل می‌شود.

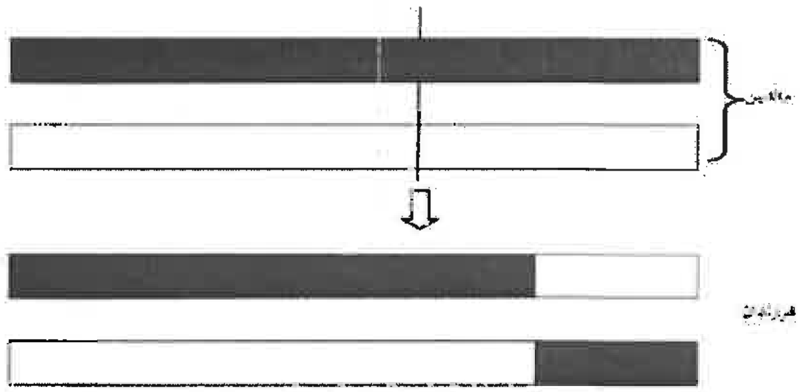
---

[1] Population

[2] Coromosome

هر تکرار الگوریتم را یک نسل[1] می‌نامیم، در هر نسل صلاحیت کروموزم‌ها براساس معیار عملکرد برای تولید نسل بعدی، مورد ارزیابی قرار می‌گیرد.کروموزم‌های بعدی که آنها را فرزندان می‌نامیم، طی مراحل زیر ایجاد می‌شوند:

- ادغام دو کروموزم از نسل جاری با استفاده از عملگر ادغام[2]. این عملگر برای هر جفت از کروموزم ها به صورت تصادفی حداقل یک محل قطع در طول کروموزم انتخاب می‌کند و سپس قسمتی از این دو کروموزم را با هم مبادله می‌کند که حاصل این عمل دو کروموزم جدید با طول مساوی طول کروموزم های اولیه است. دو کروموزم اول را والدین و کروموزم های جدید را فرزندان می‌نامند. سپس رشته‌های جدید جانشین والدین خود در جمعیت می‌شوند. عملکرد عملگر ادغام را می‌توان مطابق شکل ۵-۱ نشان داد.



شکل ۵-۱ عملکرد عملگر ادغام

- تغییر دادن یک کروموزم با استفاده از عملگر جهش[3]. در این مرحله، عملگر جهش بر روی یک مجموعه عمل می‌کند. به این صورت که از بین تمام کاراکترهای موجود در جمعیت، یک کاراکتر انتخاب و مقدار آن

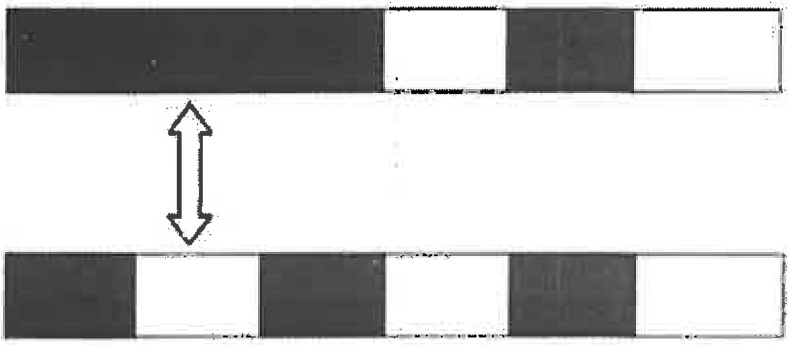
---

[1] Generation

[2] Crossover operator

[3] Mutation operator

تغییر داده می‌شود. انتخاب و تغییر مقدار کاراکتر، هر دو به صورت تصادفی انجام می‌شوند. عملکرد عملگر جهش را می‌توان مطابق شکل ۵-۲ نشان داد



شکل ۵-۲ عملکرد عملگر جهش

انتخاب نسل جدید نیز طی مراحل زیر انجام می‌گیرد:

- ارزیابی و تعیین صلاحیت کلیه کروموزم‌ها، شامل والدین و فرزندان براساس معیار عملکرد.

- انتخاب تعداد ثابتی از بهترین کروموزم‌ها، برابر با تعداد ثابت اعضای جمعیت، ودور ریختن بقیه کروموزم‌ها.

بعد از چند نسل(تکرار)، الگوریتم به سمت بهترین کروموزم‌ها همگرا می‌شود.

۵-۳ بررسی پارامترهای الگوریتم ژنتیک برای مسئله

۱. احتمال ادغام[1] : (P_c) عبارتست از عددی بین صفر تا یک که با این احتمال، عملگر ادغام بر روی پاسخ‌ها عمل می‌کند. اگر P_c برابر صفر انتخاب شود، ادغام هیچگاه اتفاق نمی‌افتد و اگر یک باشد همیشه ادغام انجام می‌شود .

[1]Crossover rate

۲. احتمال جهش[1] : (P_m) این پارامتر شبیه پارامتر P_c می‌باشد و احتمال عمل جهش را بر روی پاسخ‌ها تعیین می‌کند. تذکر آنکه اگر P_m بزرگ انتخاب شود الگوریتم ژنتیک به یک الگوریتم جستجوی احتمالی تبدیل می‌شود .

۳. اندازه جمعیت[2] : (P_s) این پارامتر تعیین‌کننده تعداد کروموزوم‌ها در یک نسل (جمعیت) است. اگر این پارامتر کوچک باشد الگوریتم به جستجوی فضای کوچکی از کل فضای جواب می‌پردازد و جواب خوبی به دست نمی‌آید. همچنین با بزرگ بودن این پارامتر، الگوریتم ژنتیک به کندی پیش می‌رود. براین اساس این پارامتر با توجه به نوع مسئله و امکانات موجود تعیین می‌شود .

۴. شرایط پایانی: (*max-stuck*) این شرط تعیین‌کننده زمان پایان یافتن الگوریتم است. برای شرایط پایانی الگوریتم می‌توان تعداد نسل‌ها، تعداد پاسخ‌های بهینه تکراری، حد صلاحیت افراد جمعیت و... را درنظر آورد. همیشه با زیاد کردن تکرارهای الگوریتم پاسخ‌های بهتر تولید نمی‌شود بلکه پس از رسیدن به حدی از تکرار، الگوریتم متوقف می‌شود و پاسخ‌های بهتر به دست نمی‌آید .

۴-۵ الگوریتم ژنتیک پیوندی

در این بخش به توصیف الگوریتم ژنتیک پیوندی برای حل مسئله با معیار کمینه کردن مجموع زمان تکمیل کارها و زمان اتمام کل سیستم خواهیم پرداخت. در واقع الگوریتم پیشنهادی، الگوریتم ژنتیک را و یک طرح جستجوی محلی جدید را با هم ترکیب می‌کند. این طرح جستجوی محلی متشکل از دو روش جستجوی محلی با عناوین، جستجوی الحاقی (*IS*) و جستجوی الحاقی با برش و مرمت (*ISCR*) است.

[1]Mutation rate

[2] Population size

در این روش، الگوریتم ژنتیک برای انجام یک جستجوی سراسری[1] استفاده می‌شود و از دو روش جستجوی محلی[2] برای انجام جستجوی محلی استفاده می‌شود که البته این دو روش جستجوی محلی نقش‌های مختلفی را در فرایند جستجو بازی می‌کنند. جستجوی الحاقی[3] مسئول جستجو در همسایگی‌های کوچک است در حالی که جستجوی با برش و مرمت[4] مسئول جستجو در همسایگی‌های بزرگ می‌باشد.

در این بخش ابتدا به توصیف الگوریتم ژنتیک می‌پردازیم و در بخش‌های بعدی جزئیات عملگر ادغام و دو روش جستجوی محلی شرح داده خواهد شد.

۵-۴-۱ الگوریتم ژنتیک

گام یک: مجموعه مقادیر اندازه جمعیت (P_s)، نرخ ادغام (P_c)، نرخ جهش (P_m)، شرط پایان (*Max-stuck*) و در آستانه جستجوی محلی (*Local-stuck*) را تعیین می‌کنیم و قرار می‌دهیم $S_l=0$

گام دو: یک جمعیت اولیه به طور تصادفی تولید می‌کنیم .

گام سه: مقدار مجموع زمان تکمیل کارها (زمان اتمام کل سیستم) را برای هر یک از کروموزم‌های جمعیت اولیه به دست می‌آوریم و کروموزمی با بهترین مجموع زمان تکمیل کار (زمان اتمام کل سیستم) را در B^* قرار می‌دهیم و مقدار مجموع زمان تکمیل کارها (زمان اتمام کل سیستم) را در C^* قرار می‌دهیم.

1 Global search

2 Local search

3 Insertion search

4 Cut – rapair

« ادغام، جستجوی محلی، انتخاب » (چگونگی عمل ادغام و جستجوی محلی به ترتیب در بخش ۵-۴-۲ و ۵-۴-۳ شرح داده شده است.)

گام چهار: گام ۵ تا ۷ را P_s . P_c بار تکرار می‌کنیم.

گام پنج: به طور تصادفی دو کروموزم P_1 و P_2 را انتخاب و عملگر ادغام را بر روی کروموزم P_1 و P_2 اعمال و کروموزم فرزند را تولید می‌کنیم.

گام شش: اگر S_1<*Local-stuck* عملگر الحاقی را بکار ببرید در غیر این صورت عملگر الحاقی، با برش و مرمت را بکار برید.

گام هفت : P_1 و P_2 را با دو تا از بهترین‌ها از میان P_1 و P_2 و فرزند جابجا می کنیم.

« جهش » (شیوه عملگر جهش در بخش ۵-۴-۴ شرح داده شده است.)

گام هشت : به طور تصادفی P_s . P_m کروموزم انتخاب کنید و عملگر جهش را بر روی این کروموزم‌ها بکار برید.

گام نه : کروموزم B با بهترین مجموع زمان تکمیل کارها و زمان اتمام کل سیستم را پیدا و مقدار آن را در C قرار می‌دهیم اگر $C<C^*$ آنگاه قرار می‌دهیم $C^* \leftarrow C$ و $B^* \leftarrow B$.

گام ده : اگر S_1>*Max-stuck* آنگاه توقف می‌کنیم در غیر اینصورت به گام چهار می‌رویم.

۵-۴-۲ عملگر ادغام

قبل از معرفی عملگر ادغام، خلاصه‌ای از آرایه‌های متعامد[1] که بر مبنای آن عمل ادغام صورت می‌گیرد را معرفی می‌کنیم.

۵-۴-۲-۱ آرایه متعامد

فرض کنید در یک آزمایش K فاکتور و برای هر فاکتور Q سطح وجود دارد، لذا برای پیدا کردن بهترین حالت Q^K آزمایش باید انجام شود. خیلی اوقات این امر غیر ممکن است یا عامل هزینه برای تست Q^K ترکیب بسیار بالا است. در یک آزمایش که K فاکتور وجود دارد و برای هر فاکتور Q سطح، یک آرایه متعامد $OA(n, K, Q, t)$ را تعریف می‌کنیم که یک آرایه با n سطر و K ستون می‌باشد و در آن n تعداد آزمایشات، K تعداد فاکتورها ، Q تعداد سطوح برای هر فاکتور و t مقاومت می باشد. که در سه شرط زیر صدق می‌کند.

۱- برای هر فاکتور در هر ستون، هر سطح به تعداد دفعاتی یکسان در آرایه متعامد ظاهر شده است.

۲- برای هر t فاکتور در هر t ستون، هر ترکیب از Q سطح به تعداد دفعاتی یکسان ظاهر شده است. (به طوریکه هر ترکیب از Q سطح، که در ستون های هر t سطر متمایز تشکیل شده اند با یکدیگر متمایز باشند)

۳- ترکیبات به طور یکنواخت در فضای همه ترکیبات توزیع شده است. (برای درک بهتر، به مثال ۵-۱ مراجعه شود)

---

[1] Orthogonal array

چون آرایه متعامد در سه شرط فوق صدق می‌کند به طراح آزمایش کمک خواهد کرد، مجموعه نمونه را به جای انجام Q^K آزمایش، برای پیدا کردن بهترین مقدار سطح برای هر فاکتور جایگزین کند.

اغلب ، آرایه‌های متعامد را با $L_n(Q^K)$ نمایش می‌دهند که معمولا مقدار t برابر با ۲ می‌باشد.

برای یک آزمایش، آرایه‌های متعامد مختلفی وجود دارد، بعد از انتخاب آرایه متعامد، معیار زیر برای تعیین بهترین ترکیبات از سطح فاکتورها بکار برده می‌شود.

فرض کنید E_i مقدار ارزیابی (معکوس تابع هدف)، برای i امین آزمایش باشد. تاثیر فاکتور j بر سطح K را با F_{jk} نمایش می‌دهیم و تعریف می‌کنیم :

$$F_{jk} = \sum_{i=1}^{n} E_i A_{ijk}\,, \qquad k = 0,1,2....,Q-1, \quad j = 1,2....,K.$$

که A_{ijk} برابر با یک است اگر سطح j امین فاکتور در i امین آزمایش در آرایه متعامد K باشد در غیر این صورت برابر با صفر می‌باشد. بعد از محاسبه تمام F_{jk} ها، سطح فاکتور j L انتخاب می‌شود اگر

$$F_{jL}= max_{1\leq k\leq Q}\, F_{jk}$$

اکنون با توجه به تعریف ارائه شده در مورد آرایه‌های متعامد به معرفی عملگر ادغام می‌پردازیم.

عملگر ادغام با هدف تولید کروموزم‌های بهتر انجام می‌شود این عملگر یک عملگر ترکیبی است که شامل سه عمل است. اول یک جفت کروموزم را به صورت تصادفی انتخاب می‌کند، دوم یک محلی را برای عمل ادغام به طور تصادفی در طول رشته انتخاب می‌کنیم و سرانجام در سومین مرحله مقدار دو رشته را با توجه به محل ادغام جابه‌جا می‌کنیم.

روش‌های مختلف برای عمل ادغام وجود دارد که معمولا ادغام تک نقطه‌ای و ادغام دو نقطه‌ای در عملگر ادغام استفاده می‌شود. ولی در الگوریتم ژنتیک پیوندی از ادغام چندگانه استفاده می‌شود، در واقع کروموزم پدر و کروموزم مادر به چند زیر دنباله به طور تصادفی تقسیم می‌شود. در این روش برای بیست کار از ۶ نقطه برش و برای پنجاه و صد کار از ۱۴ نقطه برش استفاده می‌شود.

نتایج آزمایشات نشان می‌دهد که افزایش تعداد نقاط برش در عملگر ادغام با افزایش اندازه مسئله، جواب مسئله را بسیار بهبود می‌بخشد.

عملگر ادغام OA به صورت زیر توصیف می‌شود:

گام یک: فرض کنید N تعداد تکه‌هایی است که کاربر می‌خواهد کروموزم‌های والدین P_1 و P_2 را برای تولید فرزند برش دهد. لذا آرایه متعامد $L_{N+1}(2^N)$ را تولید می‌کنیم.

گام دو: به طور تصادفی کروموزم والدین P_1 و P_2 را انتخاب و به طور تصادفی $N-1$ نقطه برش را برای برش P_1 و P_2 به N زیر دنباله انتخاب می‌کنیم.

گام سه: i امین سطر از $L_{N+1}(2^N)$ را بررسی و یک فرزند نمونه C_i $(i=1، ۲،......N+1)$ تولید می‌کنیم. jامین زیر دنباله از C_i، jامین زیر دنباله از P_1 است اگر jامین فاکتور در سطر i ام، OA برابر با صفر در غیر این-صورت jامین زیر دنباله از C_i ، jامین زیر دنباله از P_2 می‌باشد، و C_i را هر زمان که لازم باشد مرمت می-کنیم.

گام چهار: مقدار ارزیابی E_i را برای تمام C_i ها، $(i=1، ۲،......N+1)$ محاسبه می‌کنیم.

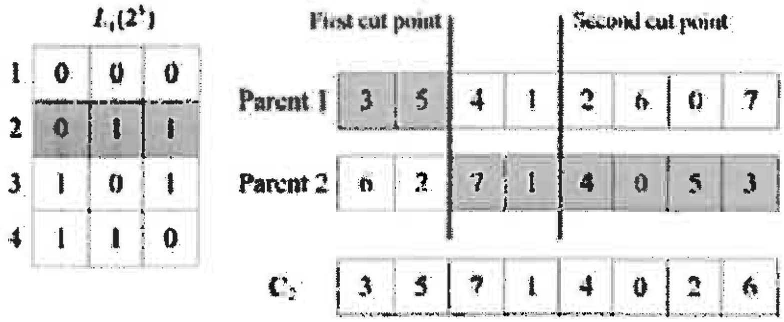
گام پنج: مقدار F_{jk} را برای فاکتور j ام از سطح k ام محاسبه می‌کنیم. $(j=1، ۲،....، N$ و $k=0، 1)$

گام شش: بهترین سطح را برای هر فاکتور پیدا می‌کنیم. بهترین سطح، فاکتور j برابر k است اگر $F_{jk}=max\{F_{j0}\ ,\ F_{j1}\}$ باشد. و سپس بهترین سطح همه فاکتورها را برای تولید فرزند C_{N+2} بکار می‌بریم. و C_{N+2} را در صورت نیاز مرمت کرده و مقدار ارزیابی E_{N+2} را برای آن محاسبه می‌کنیم.

گام هفت: از میان $C_1, C_2, \ldots, C_{N+2}$ یکی را با بهترین مقدار ارزیابی به عنوان کروموزم فرزند انتخاب می کنیم.

مثال ۵-۱

برای روشن‌تر شدن مطالب دو کروموزم والدین، P_1 و P_2 که در شکل ۵-۱ نشان داده شده است را در نظر بگیرید.



شکل ۵-۳ : تولید فرزند با استفاده از عملگر ادغام

همانطور که درشکل مشاهده می شود تعداد تکه هایی که کروموزم های والدین P_1 و P_2 برای تولید فرزند برش داده شده برابر با ۳ است لذا کروموزم های والدین P_1 و P_2 آرایه متعامد، $L_4(2^3)$ را ایجاد می‌کنند، که سطوح آن ۰ ، ۱ می باشد حال برای نمونه برای تولید فرزند C_2، ما باید دومین سطر از $L_4(2^3)$ را کنکاش کنیم، چون ظرفیت این سطر (۰ ۱ ۱) می‌باشد ابتدا اولین زیر دنباله از C_2 را اولین زیر دنباله از P_1 قرار می‌دهیم (چون مقدار اولین عنصر سطر دوم برابر با ۰ است) که مقدارش برابر با ۳ و ۵ است و دومین زیر دنباله

از C_2 را دومین زیر دنباله از P_2 (چون مقدار دومین عنصر سطر دوم برابر با ۱ است) با مقدار ۷ و ۱ قرار می‌دهیم. تا این مرحله احتیاجی به مرمت نیست، زیرا اعداد تکراری ظاهر نشده‌اند و سپس سومین زیر دنباله از C_2 را سومین زیر دنباله از P_2 که اعداد ۴، ۰، ۵، ۳ می‌باشد قرار می‌دهیم. که این مرحله نیاز به مرمت دارد زیرا اعداد ۳و ۵ قبلا در C_2 ظاهر شده‌اند لذا دو مکان آخر C_2 پاک شده‌اند و کارهایی که هنوز ظاهر نشده‌اند در C_2 قرار می‌گیرد.

۵-۴-۳ دو روش جستجوی محلی

پروسه جستجوی محلی، بهترین جواب را در همسایگی یک جواب جستجو می‌کند، لذا فرض کنید π یک جواب در فضای جواب‌ها باشد، همسایگی π را با $N(\pi)$ نمایش می‌دهیم و تعریف می‌کنیم مجموعه همه جواب‌هایی که می‌تواند با بکار بردن بعضی عملگرها به جواب π برسند.

دو عملگر که برای حل مسئله زمانبندی کارگاه گردش کاری جایگشتی طراحی شده است عبارتند از:

۱- عملگر الحاقی[1] $Ins(x , y)$ این عملگر کاری را که در موقعیت x است خارج و آن را به موقعیت y وارد می‌کند.

۲- عملگر مبادله[2] $Exch(x , y)$ این عملگر کاری را که در موقعیت x است با کاری که در موقعیت y است مبادله می‌کند.

---

[1] Insertion operator

[2] Exchange operator

البته با بررسی‌های انجام شده استفاده از عملگر الحاقی در انجام جستجوی محلی برتر از عملگر مبادله است. لذا از عملگر الحاقی برای انجام جستجوی محلی استفاده می‌کنیم. برای یک کروموزم شامل n کار، $n.(n-1)$ عملگر الحاقی می‌توان بکار برد (هر کار می‌تواند در $(n-1)$ موقعیت دیگر به جز موقعیت فعلی‌اش قرار گیرد) ولی بررسی تمام $n.(n-1)$ عملگر الحاقی کاری طاقت‌فرسا خواهد بود لذا برای کنترل بازه موقعیت‌ها، از یک پارامتر α استفاده می‌شود.

همانطور که قبلا بیان شد در الگوریتم ژنتیک پیوندی، از دو نوع روش جستجوی محلی استفاده می‌شد ۱- روش جستجوی الحاقی و ۲- روش جستجوی الحاقی با برش و مرمت که در ادامه به توصیف دو روش فوق خواهیم پرداخت.

در این روش‌ها جایگشت π نشان دهنده یک کروموزم است و $\pi(i)$ نشان دهنده کاری است که در موقعیت i ام از π می‌باشد و C مقدار مجموع زمان تکمیل کارها (زمان اتمام کل سیستم) برای جایگشت π می‌باشد. لذا الگوریتم را می‌توان به صورت زیر بیان کرد.

۵-۴-۳-۱ روش جستجوی الحاقی(π, C, α)

گام یک: لیست جستجو را با SL نمایش داده و قرار می‌دهیم$SL \leftarrow \{1,2,....,n\}$

گام دو: اگر SL خالی نیست، به طور تصادفی p را از مجموعه SL انتخاب می‌کنیم و به گام سه می‌رویم در غیر این‌صورت توقف می‌کنیم و C و π را به عنوان جواب در نظر می‌گیریم.

گام سه: عملگر الحاقی $Ins(p,K)$ را بر روی π برای

$$K=p-1, p-2, ...,max(p-\alpha, 1) \text{ و } K= p+1, p+2, ...min(p+\alpha, n).$$

اجرا می‌کنیم و بعد از هر بار اجرای عملگر الحاقی، مقدار مجموع زمان تکمیل کارها (زمان اتمام کل سیستم) را محاسبه می‌کنیم اگر مقدار بدست آمده بهتر از جواب C باشد آن را جایگزین C می‌کنیم و کروموزم حاصل را نیز جایگزین π کرده و به گام یک رفته در غیر این‌صورت به گام دو می‌رویم.

همانطور که در بالا اشاره شد روش جستجوی الحاقی فقط مسئول جستجو در همسایگی های کوچک می باشد لذا برای همسایگی های بزرگتر روش جستجوی الحاقی با برش و مزمت را در بخش بعدی معرفی خواهیم کرد.

قبل از معرفی روش جستجوی الحاقی با برش و مرمت ، لازم است به تشریح پروسه برش و مرمت بپردازیم. در پروسه برش و مرمت به دنبال دو جفت کار متوالی در دنباله π هستیم به طوری که بیشترین زمان بیکاری را دارند.

زمان بیکاری یک جفت کار مجاور در π به صورت زیر تعریف می شود:

$$IT(\pi_j,\pi_{j+1})=\sum_{x=2}^{m}\max\left\{c(\pi_{j+1},x-1)-c(\pi_j,x),0\right\} \qquad i=1,2.....n.$$

۵-۴-۳-۲ روش برش و مرمت (π,C)

<u>گام یک</u> : لیست برش را $Cl\leftarrow\Phi$ و لیست حرکت را $Ml\leftarrow\Phi$ قرار می دهیم.

<u>گام دو</u>: مقدار $IT(\pi_j , \pi_{j+1})$ برای i =۱، ۲،.........n-۱ محاسبه می کنیم و دو جفت از کارهای مجاور با بیشترین زمان بیکاری را انتخاب می کنیم و آنها را به عنوان دو نقطه برش در Cl قرا می دهیم.

گام سه : اگر Cl خالی نیست یک نقطه برش $(cp,cp+1)$ به طور تصادفی از Cl انتخاب و به گام چهار بروید. در غیر این صورت مقدار C و π را به عنوان جواب در نظر بگیرید.

گام چهار: برای $t\in\{1,2,....cp-1\}$ عملگر الحاقی $Ins(t,cp)$ را اجرا می کنیم و همچنین برای $t\in\{cp+2,cp+3,......n\}$ نیز عملگر الحاقی $Ins(t,cp+1)$ را اجرا کرده و از میان عملگرهای الحاقی بالا هشت عملگر که بیشترین میزان کاهش زمان بیکاری را در بردارد پیدا می کنیم، و هشت عملگر فوق را در Ml قرار می دهیم.

گام پنج : یک عدد تصادفی از بازه (۱ ، ۰] انتخاب می کنیم . اگر عدد فوق بزرگتر از ۰/۵ باشد بهترین عملگر را از Ml انتخاب می کنیم در غیر اینصورت به طور تصادفی یکی از عملگرهای Ml را انتخاب می کنیم. و سپس این عملگر را در π اعمال و C را نیز به روز رسانی می کنیم . سپس $Ml\leftarrow\Phi$ و به گام سه می رویم .

۵-۴-۳-۳ روش جستجوی الحاقی با برش و مرمت $(\pi , C , \alpha , Max\ loop)$

گام یک : قرار می دهیم $\pi^*\leftarrow\pi$ و $C^*\leftarrow C$ و $iter\leftarrow 0$.

گام دو : روش جستجوی الحاقی (π , C, α) را بکار برده و مقدار π و C را به دست آورید . اگر $C<C^*$ بود آنگاه قرار می دهیم $\pi^*\leftarrow\pi$ و $C^*\leftarrow C$

گام سه : روش برش و مرمت (π , C) را اجرا کرده و مقدار π و C را به دست آورید. $\pi^*\leftarrow\pi$ و $C^*\leftarrow C$

گام چهار : $iter\leftarrow iter+1$. اگر $iter\geq Max\text{-}loop$ مقدار π^* و C^* را به عنوان جواب در نظر بگیرید در غیر این صورت به گام دو بروید.

۵-۴-۴ عملگر جهش

عملگر مبادله برای عمل جهش بکار برده می شود. بدین صورت که $P_m . P_s$ کروموزم به صورت تصادفی از جمعیت انتخاب می کنیم، و سپس برای هر کروموزم انتخابی ،عملگر $Exch(x,y)$ را t بار اجرا می کنیم که x و y , موقعیت انتخابی به صورت تصادفی و t یک عدد صحیح بین ۱ تا T که T یک پارامتر از پیش تعریف شده می باشد.

۵-۵ نتایج مقایسات

لین – تسنگ و یا – تی لین برنامه الگوریتم ژنتیک پیوندی را با زبان C++ برروی کامپیوتر شخصی نوشته اند که در زیر نتایج آزمایشات آنان آورده شده است.

آنان آزمایشات خود را در سه مرحله انجام دادند. در مرحله اول به مقایسه کارایی دو روش جستجوی محلی پرداختند، و در آزمایشات مرحله دوم به مقایسه کارایی چهار الگوریتم ، ژنتیک سنتی ، ژنتیک با جستجوی الحاقی و ژنتیک با جستجوی الحاقی برش _ مرمت و ژنتیک پیوندی پیشنهادی پرداخته اند . از آنجا که نتایج مقایسات مسئله زمانبندی کارگاه گردش کاری با معیارکمینه کردن زمان اتمام کل سیستم بسیار شبیه معیار کمینه کردن مجموع زمان تکمیل کارها می باشد، در دو گروه اول نتایج آزمایشات براساس معیار مجموع زمان تکمیل کارها می باشد. در نهایت در مرحله سوم به مقایسه کارایی روش پیشنهادشده با چند مورد از الگوریتم هایی که در چند سال اخیر ارائه شده است، پرداخته اند.

۵-۵-۱ مقایسه دو روش جستجوی محلی

در این قسمت ۱۵ مسئله در نظر گرفته شده است که شامل ۵ مسئله ۵×۵۰ و پنج مسئله ۱۰×۵۰ و پنج مسئله ۲۰×۵۰ می باشد . چون کارایی روش جستجوی محلی بسیار وابسته به حدس اولیه است، آنان به طور تصادفی ۱۰ جواب برای هر مسئله به عنوان جواب اولیه برای دو روش جستجوی محلی در نظر گرفته اند. نتایج مقایسه کارایی دو روش جستجوی محلی در جدول ۵-۱ نشان داده شده است که البته ازدو معیار برای مقایسه کارایی استفاده کرده اند.

۱- *Avg time*: بیانگر میانگین زمان اجرا (بر حسب ثانیه) می باشد.

۲- *Avg PRD* : بیانگر میانگین درصد خطای نسبی می باشد که به صورت زیر تعریف می شود:

$$AvgPRD = {C_{Avg} - C_{Best}} / {C_{Best}} \times 100\%.$$

C_{Best} برابر با مقدار مجموع زمان تکمیل کارها در میان بهترین جواب پیدا شده تا آن مرحله است.

جدول ۵-۱ مقایسه دو روش جستجوی محلی

Methods	Parameters		Result	
	Max_Loop	x	AvgTime	AvgPRD
Insertion Search		10 (n/5)	0.0244	7.0682
		25 (n/2)	0.0497	3.3590
		50 (n)	0.0596	2.8919
Insertion Search with Cut-and-Repair	2	10 (n/5)	0.0416	6.2704
	2	25 (n/2)	0.0881	3.0528
	5	10 (n/5)	0.0862	4.6525
	5	25 (n/2)	0.1800	2.1780
	10	25 (n/2)	0.3351	1.9180
	50	25 (n/2)	1.5317	1.2705

با نگاهی اجمالی به جدول ۵-۱ می توان چنین برداشت کرد که هر چند در روش جستجوی الحاقی، زمان پردازش، نسبت به روش جستجوی الحاقی با برش- مرمت کمتر است اما جواب از لحاظ کیفی مناسب نیست.

یکی از پارامترهای مهم در روش جستجوی محلی مقدار α می باشد. همانطور که در جدول بالا مشاهده می کنید وقتی مقدار α از (*n*/۵) ۱۰ به (*n*/۲) ۲۵ تغییر می کند کیفیت جواب به دست آمده با استفاده از روش جستجوی الحاقی به طور قابل توجهی بهبود بخشیده می شود. ولی وقتی α از (*n*/۲) ۲۵ به (*n*) ۵۰ تغییر می کند کیفیت جواب به دست آمده فقط اندکی بهتر می شود.

۵-۵-۲ مقایسه الگوریتم ژنتیک سنتی با الگوریتم ژنتیک پیوندی

در این بخش نیز ۱۵ مسئله همانند بخش قبلی در نظر گرفته شده است. چهار الگوریتم به نام های ژنتیک سنتی، ژنتیک با جستجوی الحاقی، ژنتیک با جستجوی الحاقی برش- مرمت و الگوریتم ژنتیک پیوندی برروی این ۱۵ مسئله، با ده بار تکرار برای حل مسئله زمانبندی کارگاه گردش کاری جایگشتی با معیار کمینه کردن مجموع زمان تکمیل کارها مقایسه شده اند.

در جدول ۵-۲ میانگین زمان پردازش (*Avg time*) و میانگین درصد خطای نسبی (*Avg PRD*) نشان داده شده است. نرخ ادغام (P_c) برابر با ۵۰٪ و نرخ جهش (P_m) برابر با ۵٪ و مقدار ***Local-stuck*** برابر با (*Max-stuck/2*) در نظر گرفته شده است. مقدار پارامتر T که در عملگر جهش از آن استفاده می شود برابر با ۵ در نظر گرفته شده و مقادیر سایر پارامترها در جدول ۵-۲ نشان داده شده است.

P_s نشان دهنده سایز جمعیت، *Max-stuck* نشان دهنده شرط پایان و α_1 بیانگر بازه جستجو الحاقی و α_2 معرف بازه جستجو الحاقی با برش- مرمت می باشد.

جدول ۵-۲ : مقایسه الگوریتم ژنتیک با الگوریتم ژنتیک پیوندی

Methods	Parameters					Result	
	P_s	Max_Stuck	Max_Loop	α_1	α_2	AvgTime	AvgPND
Traditional GA	50	10				0.1420	6.0335
	100	10				0.3395	4.3444
	500	10				2.2902	2.0434
	2500	10				14.1663	1.8513
GA + Insertion search	50	10		10 (n/5)		11.7818	0.8681
	50	10		25 (n/2)		23.0639	0.6702
	50	10		50 (n)		29.6705	0.6691
GA + Insertion Search with Cut-and-Repair	50	10	2		10 (n/5)	26.9018	0.6447
	50	10	2		25 (n/2)	51.5301	0.4912
	100	10	2		10 (n/5)	59.2918	0.4942
	100	10	2		25 (n/2)	109.5557	0.3990
Hybrid GA	50	10	2	10 (n/5)	25 (n/2)	22.2896	0.6485
	50	10	2	25 (n/2)	25 (n/2)	29.2571	0.5630
	100	10	2	10 (n/5)	25 (n/2)	46.3747	0.4725
	100	10	2	25 (n/2)	25 (n/2)	65.4471	0.4246
	100	20	2	25 (n/2)	25 (n/2)	95.1199	0.3445
	100	30	2	25 (n/2)	25 (n/2)	118.5326	0.3477
	150	10	2	25 (n/2)	25 (n/2)	99.0642	0.3524
	150	20	2	25 (n/2)	25 (n/2)	150.3324	0.2970
	150	30	2	25 (n/2)	25 (n/2)	186.4431	0.2807
	200	10	2	25 (n/2)	25 (n/2)	131.4619	0.3057
	200	20	2	25 (n/2)	25 (n/2)	192.3615	0.2486
	200	30	2	25 (n/2)	25 (n/2)	254.0652	0.2436

همانطور که در جدول ۵-۲ مشهود است الگوریتم ژنتیک با جستجوی الحاقی و مقدار$\alpha_۱ = ۱۰$ و $P_s = ۵۰$ نسبت به روش الگوریتم ژنتیک سنتی با $P_s = ۲۵۰۰$ هم از لحاظ زمان پردازش و هم از لحاظ کیفیت جواب بهتر می باشد. همین طور الگوریتم ژنتیک با جستجوی الحاقی با برش و مرمت و مقدار $\alpha_۲ = ۱۰$ برتر از الگوریتم ژنتیک با جستجوی الحاقی با مقدار$\alpha_۱ = ۵۰$ می باشد. همچنین الگوریتم ژنتیک پیوندی با مقادیر $P_s = ۵۰$، $Max\text{-}stuck = ۱۰$ و $\alpha_۱ = ۱۰$ و $\alpha_۲ = ۲۵$ هم از لحاظ زمان پردازش و هم از لحاظ کیفیت جواب از الگوریتم ژنتیک با جستجوی الحاقی و مقدار $\alpha_۱ = ۲۵$ برتر است. همچنین اگر الگوریتم ژنتیک پیوندی مقادیر $P_s = ۱۰۰$ و $Max\text{-}stuck = ۱۰$ ، $\alpha_۱ = ۱۰$ و $\alpha_۲ = ۲۵$ نتایج بهتری هم از لحاظ زمان و هم از لحاظ کیفیت

جواب نسبت به حالتی که در الگوریتم ژنتیک با جستجوی الحاقی با برش و مرمت با مقادیر P_s =۵۰ و α_2 =۲۵ حاصل می شود.

لذا با توجه به جدول بالا می توان نتیجه گرفت الگوریتم ژنتیک سنتی روشی کارا برای حل مسئله فوق نمی باشد، حتی زمانی که سایز جمعیت خیلی بزرگ باشد (P_s= ۲۵۰۰). علت آن را می توان این گونه توجیه کرد که روش فوق یک روش جستجوی سراسری است و برای انجام جستجوی محلی مناسب نیست. لذا برای حل این مشکل، با استفاده از جستجوی محلی می توان جواب مسئله را تا حدود زیادی بهبود بخشید. البته الگوریتم ژنتیک با روش الحاقی با برش _مرمت (*ISCR*) جواب بهتری نسبت به الگوریتم ژنتیک با جستجوی الحاقی (*IS*) دارد زیرا روش *ISCR* هم همسایگی کوچک و هم همسایگی بزرگ را پوشش می دهد ولی روش (*IS*) فقط مسئول همسایگی های کوچک است. لذا دو روش *IS* و *ISCR* در الگوریتم ژنتیک پیوندی نقش موثری را ایفا می کنند. در الگوریتم ژنتیک پیوندی، روش *IS* مکررا تا زمانی که پیشرفت داشته باشیم، استفاده می شود و تنها وقتی که در روش *IS* به یک نقطه بهینه محلی برسیم از روش *ISCR* برای رهایی از این حالت استفاده می شود .

چون روش *ISCR* زمانبرتر از روش *IS* می باشد لذا الگوریتم ژنتیک پیوندی از لحاظ زمانی نیز کارامدتر از روش الگوریتم ژنتیک با جستجوی الحاقی با برش و مرمت است.

۵-۵-۳ مقایسه الگوریتم ژنتیک پیوندی با سایر روش ها

در این بخش نتایج مقایسه کارایی الگوریتم ژنتیک پیوندی برای حل مسئله زمانبندی کارگاه گردش کاری جایگشتی با معیار کمینه کردن مجموع زمان تکمیل کارها با روش های پیشنهاد شده توسط لیو و ریورز(۲۰۰۱)، رجندران و زیگلر (۲۰۰۴) و تسگتیرن[1] وهمکاران (۲۰۰۷) در جدول ۵-۳ آورده شده است. در این جدول LR نشان دهنده نتیجه بهترین جواب روش ارائه شده توسط لیو و ریورز می باشد.

M-MMAS و *PACO* معرف نتیجه دو روش فرا ابتکاری با استفاده از الگوریتم مورچگان[2] است که توسط رجندران و زیگلر ارائه شده است، *PCO* بیانگر نتیجه روش فراابتکاری ارائه شده توسط تسگترین و همکاران می باشد. پارامترهای مورد استفاده برای حل مسئله با استفاده از الگوریتم ژنتیک پیوندی که در جدول ۵-۳ بدان نیاز است در جدول ۵-۴ آورده شده است. در جدول ۵-۳ ، *Min* برابر با کمترین مقدار مجموع زمان تکمیل کارهای به دست آمده با اجرای ده بار روش مورد نظر است و *Avg* بیانگر متوسط مجموع زمان تکمیل کارهای به دست آمده با اجرای ده بار الگوریتم ژنتیک پیوندی برای یک مسئله می باشد. همانطور که در جدول ۵-۳ مشاهده می شود در میان ۹۰ مسئله در نظر گرفته شده، در ۶۶ مسئله اکیدا بهترین جواب حاصل شده و در ۲۴ مسئله باقیمانده در ۲۳ مورد همراه با سایر روش ها به بهترین جواب رسیده اند. (در جدول ۵-۳ بهترین جواب، پررنگ تر نشان داده شده است.)

[1] Tasgetiren

[2] Ant colony algorithm

جدول ۵-۳ : بررسی مقایسه کارایی الگوریتم ژنتیک پیوندی با چهار روش دیگربا معیار کمینه کردن مجموع زمان تکمیل کارها

n × m	LR	M-MMAS	PACO	PSO	Hybrid GA	
	Min	Min	Min	Min	Min	Avg
20 × 5	14226	14056	14056	14033	14033	14037.8
	15446	15151	15214	15151	15151	15180.4
	13676	13416	13403	13301	13301	13328.7
	15750	15486	15505	15447	15447	15475.3
	13633	13529	13529	13529	13529	13529.0
	13265	13139	13123	13123	13123	13123.0
	13774	13559	13674	13548	13548	13586.3
	13968	13968	14042	13948	13948	13964.7
	14456	14317	14383	14295	14295	14319.6
	13036	12968	13021	12943	12943	12969.4
20 × 10	21207	20980	20958	20911	20911	20925.1
	22927	22440	22591	22440	22440	22459.8
	20072	19833	19968	19833	19833	19845.7
	18857	18724	18769	18710	18710	18751.4
	18939	18644	18749	18641	18641	18661.9
	19608	19245	19245	19249	19245	19294.6
	18723	18376	18377	18363	18363	18364.3
	20504	20241	20377	20241	20241	20255.9
	20561	20330	20330	20330	20330	20330.0
	21506	21320	21323	21320	21320	21329.5
20 × 20	34119	33623	33623	34975	33623	33719.0
	31918	31604	31597	32659	31587	31590.7
	34552	33920	34130	34594	33920	33925.0
	32159	31698	31753	32716	31661	31691.1
	34990	34593	34642	35455	34557	34587.3
	32734	32637	32594	33530	32564	32570.1
	33449	33038	32922	33733	32922	32989.9
	32611	32444	32533	33008	32412	32429.4
	34084	33625	33623	34446	33600	33611.9
	32537	32317	32317	33281	32262	32271.6
50 × 5	65663	65768	65546	65058	64853	64924.6
	68664	68828	68485	68298	68173	68263.2
	64378	64166	64149	63577	63367	63524.0
	69795	69113	69359	68571	68281	68522.1
	70841	70331	70154	69698	69551	69670.8
	68084	67563	67664	67138	67013	67120.0
	67186	67014	66600	66338	66294	66405.5
	65582	64863	65123	64638	64560	64635.8
	63968	63735	63483	63227	63029	63190.1
	70273	70256	69831	69195	69037	69187.0

$n \times m$	LR	M-MMAS	PACO	PSO	Hybrid CA	
	Min	Min	Min	Min	Min	Avg
50 × 10	88770	89599	88942	88031	87598	87783.5
	85600	83612	84549	83624	83001	83312.0
	82456	81655	81338	80609	80224	80453.4
	89356	87924	88014	87053	86787	87050.8
	88482	88826	87801	87263	86646	86910.4
	89602	88394	88269	87255	86826	87003.0
	91422	90686	89984	89259	88996	89371.1
	89549	88595	88281	87192	86860	87345.6
	88230	86975	86995	86102	85841	86104.3
	90787	89470	89238	88631	88293	88578.9
50 × 20	129095	127348	126962	126622	126073	126448.3
	122094	121208	121098	122173	119360	119737.0
	121379	118051	117524	118719	116856	117194.8
	124083	123061	122807	123028	121028	121404.4
	122158	119920	119221	121202	118736	118943.0
	124061	122369	122262	123217	121066	121516.3
	126363	123609	125351	125586	123580	123879.3
	126317	124543	124374	125714	122770	123175.7
	125318	124059	123646	124932	121872	122364.3
	127823	126582	125767	126311	124354	124849.6
100 × 5	256789	257025	257886	254762	254619	255198.4
	245609	246612	246326	245315	243817	244825.0
	241013	240537	241271	239777	238075	239697.2
	231365	230480	230376	228872	228291	228787.0
	244016	243013	243457	242245	241255	241742.3
	235793	236225	236409	234082	233583	234260.1
	243741	243935	243854	242122	241458	241930.4
	235171	234813	234579	232755	232283	232813.0
	251291	252384	253325	249959	249269	250235.7
	247491	246261	246750	244275	243879	244647.4
100 × 10	306375	305004	305376	303142	300834	302291.8
	280928	279094	278921	277109	277209	278049.9
	296927	297177	294239	292465	290198	291703.2
	309607	306994	306739	304676	303669	305501.3
	291731	290493	289676	288242	287136	288222.6
	276751	276449	275932	272790	273172	273951.3
	288199	286545	284846	282440	281306	283139.0
	296130	297454	297400	293572	293628	294827.2
	312175	309664	307043	305605	304276	305295.3
	298901	296869	297182	295173	293465	295056.2
100 × 20	383865	373756	372630	374351	370603	371741.8
	383976	383614	381124	379792	375982	377991.6
	383779	380112	379135	378174	373554	375336.2
	384854	380201	380765	380899	376236	378454.3
	383802	377268	379064	376187	373524	374314.3
	387962	381510	380464	379248	374705	377063.5
	384839	381963	382015	380912	376998	378533.3
	397264	393617	393075	392315	388058	389421.6
	387831	385478	380359	382212	378474	380306.2
	394861	387948	388060	386013	383283	384028.7

جدول ۵-۴ : پارامترهای موردنیاز الگوریتم ژنتیک پیوندی با معیار کمینه کردن مجموع زمان تکمیل کارها

	P_s	P_c (%)	P_m (%)	Max_ Stuck	Local_ Stuck	OA	α_1	α_2	Max_ Loop
$n = 20$	20	50	5	10	5	$L_8(2^7)$	10	10	2
$n = 50$	100	50	5	10	5	$L_{16}(2^{15})$	10	25	2
$n = 100$	100	50	5	10	8	$L_{16}(2^{15})$	10	50	2

نتایج مقایسه زمان اجرای الگوریتم ژنتیک پیوندی با معیار کمینه کردن مجموع زمان تکمیل کارها با روش *PSO* در جدول ۵-۵ آورده شده است. همانطور که مشاهده می شود زمان اجرای الگوریتم ژنتیک پیوندی بسیار سریعتر از روش *PSO* می باشد.

جدول ۵-۵ : مقایسه زمان اجرای الگوریتم ژنتیک پیوندی با روش PSO

$n \times m$	PSO	Hybrid GA
20 × 5	3.18	0.11
20 × 10	7.21	0.22
20 × 20	11.93	0.39
50 × 5	41.71	16.55
50 × 10	74.49	40.33
50 × 20	143.32	82.23
100 × 5	222.28	94.17
100 × 10	407.88	251.62
100 × 20	824.41	585.86

در انتها لین – تسنگ و یا – تی لین به مقایسه الگوریتم ژنتیک پیوندی برای حل مسئله زمانبندی کارگاه گردش کاری جایگشتی با معیار کمینه کردن زمان اتمام کل سیستم پرداخته اند. لذا بدین منظور روش خود را با روش های *MMAS* و *PACO* رجندران و زیگلر(۲۰۰۴) و روش *PSO* تسگترین و همکاران (۲۰۰۷)

مقایسه کرده اند. پارامترهای موردنیاز برای الگوریتم ژنتیک پیوندی در جدول ۵-۶ آورده شده است و نتایج مقایسات نیز در جدول ۵-۷ نشان داده شده است.

در جدول ۵-۷ ، ۱۱۰ مسئله درنظر گرفته شده است (برای هر کدام از مجموعه های ۵ ×۲۰ ، ۱۰ ×۲۰، ۲۰×۲۰، ۵ ×۵۰ ، ۵۰×۱۰ ، ۵۰×۲۰ ، ۱۰۰×۵ ، ۱۰۰×۱۰ ، ۱۰۰×۲۰ ، ۲۰۰×۱۰ ، ۲۰۰×۲۰ ، ۱۰ مسئله).

جدول ۵-۶ : پارامترهای موردنیاز الگوریتم ژنتیک پیوندی با معیار کمینه کردن زمان اتمام کل سیستم

	P_s	P_c (%)	P_m (%)	Max_Stuck	Local_Stuck	OA	α_1	α_2	Max_Loop
$n=20$	100	50	5	10	5	$L_8(2^7)$	10	10	10
$n=50$	100	50	5	10	5	$L_8(2^7)$	10	10	10
$n=100$	100	50	5	10	5	$L_8(2^7)$	10	10	10
$n=200$	100	50	5	10	5	$L_8(2^7)$	10	10	10

جدول ۵-۷ : بررسی مقایسه کارایی الگوریتم ژنتیک پیوندی با سه روش دیگربا معیار کمینه کردن زمان اتمام کل سیستم

	MMAS	PACO	PSO			Hybrid GA		
	Avg	Avg	Avg	Min	Max	Avg	Min	Max
20 × 5	0.762	0.704	0.03	0.00	0.41	0.022	0.000	0.097
20 × 10	0.890	0.843	0.02	0.00	0.39	0.026	0.000	0.119
20 × 20	0.721	0.720	0.05	0.00	0.31	0.045	0.000	0.133
50 × 5	0.144	0.090	0.00	0.00	0.00	-0.003	-0.015	0.031
50 × 10	1.118	0.746	0.57	0.00	1.54	-0.269	-0.463	-0.042
50 × 20	2.013	1.855	1.36	0.40	2.22	0.280	-0.208	0.823
100 × 5	0.064	0.072	0.00	0.00	0.13	0.028	-0.023	0.183
100 × 10	0.451	0.040	0.18	0.00	0.79	0.081	-0.175	0.452
100 × 20	1.030	0.985	1.45	0.57	2.89	0.171	-0.325	0.677
200 × 10			0.18	0.00	0.64	-0.072	-0.181	0.091
200 × 20			1.35	0.69	1.95	0.157	-0.432	0.786

$$Avg = {(M_{average} - M_{best})}\Big/{M_{best}} *100.$$

به طوری که M_{best} برابر با میانگین بهترین زمان اتمام کل سیستم در ۱۰ مسئله برای هر مجموعه می باشد، و *Min* و *Max* به ترتیب نشان دهنده مینیمم خطای نسبی و ماکزیمم خطای نسبی می باشد.

همانطور که در جدول ۵-۷ مشهود است الگوریتم ژنتیک پیوندی کاراتر از سه روش دیگر است.

۵-۶ نتیجه گیری

در این فصل الگوریتم ژنتیک پیوندی برای حل مسئله کارگاه گردش کاری جایگشتی با دو معیار کمینه کردن مجموع زمان تکمیل کارها و کمینه کردن زمان اتمام کل سیستم به عنوان یک الگوریتم فراابتکاری مطرح شد. در این روش، الگوریتم ژنتیک به عنوان یک روش جستجوی سراسری استفاده می شود لذا برای انجام جستجوی محلی از ترکیب دو روش جستجوی محلی به نام های جستجوی الحاقی *(IS)* که مسئول جستجو در همسایگی های کوچک می باشد و روش جستجوی الحاقی با برش و مرمت *(ISCR)* که همسایگی های بزرگتر را نیز پوشش می دهد، استفاده می شود.

در واقع پیوند الگوریتم ژنتیک با دو روش جستجوی محلی که ویژگیها و امکانات متفاوتی دارند، یک الگوریتم کارا و موثر را می سازد.

همانطور که نتایج آزمایشات نشان می دهد با به کار بردن الگوریتم ژنتیک پیوندی برای حل مسئله با معیار کمینه کردن مجموع زمان تکمیل کارها از ۹۰ مسئله در ۶۶ مسئله اکیدا به بهترین جواب و در ۲۴ مسئله باقیمانده در ۲۳ مورد همراه با سایر روش ها به بهترین جواب رسیده اند. همین طور زمانی که از روش فوق

برای حل مسئله با معیار کمینه کردن زمان اتمام کل سیسنم استفاده شده است نتایج ، نشان از برتری روش فوق نسبت به روش های *MMAS*، *PACO* و *PSO* را دارد.

لذا می توان نتیجه گرفت که پیوند یک برنامه جستجوی سراسری موثر و یک برنامه جستجوی محلی کارامد، یک روش فرا ابتکاری خوب را نتیجه می دهد و همچنین ترکیب روش جستجوی محلی که به میزان متفاوت، همسایگی ها را جستجو می کند تا حد زیادی قابلیت جستجو را در برنامه جستجو محلی بهبود می بخشد.

پیوست A

مراجع

Allahverdi, A., Aldowaisan, T., 2002. New heuristics to minimize total completion time in m-machine flowshops. International Journal of Production Economics 77, 71–83.

Campbell, H.G., Dudek, R.A., Smith, M.L., 1970. A heuristic algorithm form the n job, m machine sequence problem. Management Science 16 (10), B630–B637.

Dannenbring, D.G., 1977. An evaluation of flow shop sequencing heuristics. Management Science 23 (11), 1174–1182.

Dipak Laha, Subhash C. Sarin., 2009. Aheuristic to minimize total flowtime in permutation flowshop . Omega 37 , 734–739

Framinan, J.M., Leisten, R., Ruiz-Usano, R., 2005. Comparison of heuristics for minimisation in permutation flowshops. Computers and Operations Research 32, 1237–1254.

Framinan JM, Leisten R. 2003. An efficient constructive heuristic for flowtime minimization in permutation flow shops. Omega;31:311–7.

Garey, M.R., Johnson, D.S., Sethi, R., 1976. The complexity of flowshop and jobshop scheduling. Mathematics of Operations Research 1, 117–129.

Grabowski, J., Wodecki, M., 2004. A very fast tabu search algorithm for the permutation flow shop problem with makespan criterion. Computers and Operational Research 31, 1891–1909.

Gupta JND., 1972. Heuristic algorithms for multistage flowshop scheduling problem. AIIE Transactions;4:11–8.

Ho, J.C., 1995. Flowshop sequencing with mean flow time objective. European Journal of Operational Research 81, 571–578.

Ishubuchi, M., Masaki, S., Tanaka, H., 1995. Modified simulated annealing for the flow shop sequencing problems. European Journal of Operational Research 81, 388–398.

Johnson, S.M., 1954. Optimal two-and-three stage production schedules with setup times included. Naval Research Logistics Quarterly 1, 61–68.

Lin- Yu Tseng., Ya-Tai Lin.,2009. A hybrid genetic local search algorithm for the permutation flowshop scheduling problem. European Journal of Operational Research 198 , 84–92

Liu, J., Reeves, C.R., 2001. Constructive and composite heuristic solutions to the P/ / ∑Ci scheduling problem. European Journal of Operational Research 132, 439–452.

Liu, D.S., Tan, K.C., Huang, S.Y., Goh, C.K., Ho, W.K., 2008. On solving multiobjective bin packing problem using evolutionary particle swarm optimization. European Journal of Operational Research 190 (2), 357–382.

Montgomery, D.C., 1991. Design and Analysis of Experiments, third ed. Wiley, NewYork.

Nawaz, M., Enscore, Jr. E., Ham, I., 1983. A heuristic algorithm for the m-Machine, n-Job flow-Shop Sequencing Problem. Omega 11, 91–95.

مراجع

Nowicki, E., Smutnicki, C., 1996. A fast tabu search algorithm for the permutation flow-shop problem. European Journal of Operational Research 91, 160–175.

Ogub, F.A., Simith, D.K., 1990. Simulated annealing for the permutation flowshop problem. Omega 19, 64–67.

Palmer, D., 1965. Sequencing jobs through a multi-stage process in the minimum total time – a quick method of obtaining a near optimum. Operations Research Quarterly 16, 101–107.

Rajagopalan, D. and Karimi, I.A., 1988. Scheduling in serial mixed storage multiproduct processes with transfer and set-up times. In Foundations of Computer Aided Process Operations, edited by G.V. Reklaitis, and H.D. Spriggs, pp. 679–686, (Elsevier: New York).

Rajendran, C., Ziegler, H., 1997. An efficient heuristic for scheduling in a flowshop to minimize total weighted flowtime of jobs. European Journal of Operational Research 103, 129–138.

Rajendran, C., Ziegler, H., 2004. Ant-colony algorithms for permutation flowshop scheduling to minimize makespan/total flowtime of jobs. European Journal of Operational Research 155 (2), 426–438.

Rajendran, C., Ziegler, H., 2005. Two ant-colony algorithms for minimizing total flowtime in permutation flowshops. Computer and Industrial Engineering 48, 789–797.

Reeves, C.R., 1995. A genetic algorithm for flowshop sequencing. Computers and Operational Research 22, 5–13.

Reeves, C.R., Yamada, T., 1998. Genetic algorithm, path relinking and the flowshop sequencing problem. Evolutionary Computation 6, 45–60.

Taillard, E., 1990. Some efficient heuristic methods for the flow shop sequencing problem. European Journal of Operations Research 47, 65–74.

Taillard, E., 1993. Benchmarks for basic scheduling problems. European Journal of Operational Research 64, 278–285.

Tan, K.C., Cheong, C.Y., Goh, C.K., 2007. Solving multiobjective vehicle routing problem with stochastic demand via evolutionary computation. European Journal of Operational Research 177, 813–839.

Tasgetiren, M.F., Liang, Y.-C., Sevkli, M., Gencyilmaz, G., 2007. A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem. European Journal of Operational Research 177, 1930–1947.

Tsai, J.T., Liu, T.K., Chou, J.H., 2004. Hybrid Taguchi-genetic algorithm for global numerical optimization. IEEE Transactions on Evolutionary Computation 8 (4), 365–377.

Wang, L., Zheng, D.Z., 2003. A effective hybrid heuristic for flow shop scheduling. International Journal of Advanced Manufacturing Technology 21, 38–44.

Wang, C., Chu, C., Proth, J.-M., 1997. Heuristic approaches for $n/m/F/\sum Ci$ scheduling problems. European Journal of Operational Research 96, 636–644.

Woo, D.S., Yim, H.S., 1998. A heuristic algorithm for mean flowtime objective in flowshop scheduling. Computers and Operational Research 25, 175–182.

Yagmahan, B., Yenisey, M.M., 2008. Ant colony optimization for multi-objective flow shop scheduling problem. Computers and Industrial Engineering 54 (3), 411–420.

مراجع

Yamada, T., Reeves, C.R., 1998. Solving the Csum permutation flowshop scheduling problem by genetic local search. In: Proceedings of 1998 IEEE International Conference on Evolutionary Computation, pp. 230–234

توکلی مقدم، ر.، جولای، ف.، وحدانی، ه.،« ارایه الگوریتمی کارا برای حل مساله فلوشاپ ترکیبی با محدودیت انباره های میانی» .، چهارمین کنفرانس بین المللی مهندسی صنایع - ۱۳۸۴

حسینی، م.، جولای،ف.، «زمانبندی کارها در محیط کارگاه گردش کاری با معیار حداقل سازی مجموع دیر کرد و زود کرد کارها» ، مدیر ساز، سال ششم، شماره ۱،۲، ص ۷۴-۵۷

علیرضا، مهدی.، « مقدمه ای بر الگوریتم ژنتیک و کاربردهای آن »، چاپ اول ، نشر زانیس.

کنت آر، بیکر.،« توالی عملیات و زمانبندی » ، قاسمی طاری، ف..، فاطمی قمی ، م.، ۱۳۷۶.

مروج، محمدتقی.، « کلید MATLAB » ، چاپ اول ، نشر کلید آموزش و توسعه آموزش.

پیوست B

واژه نامه فارسی به انگلیسی

آرایه متعامد........................ Orthogonal array

آزمایش .... Experiment

ابتکاری .... Heuristic

الگوریتم ژنتیک .... Genetic algorithm

انتخاب .... Selection

اندازه جمعیت .... Population size

برش – مرمت .... Cut – repair

بهبود دهنده .... Improvement

پیوندی .... Hybrid

تابع هدف .... Objective function

تک ماشینه .... Single machine

ترتیب صعودی .... Ascending order

جایگشت .... Permutation

جستجو الحاقی .... Insertion search

جستجو محلی .... Local search

واژه نامه

Optimal solution	جواب بهینه
multi machine	چند ماشینه
Empty	خالی
Partial sequence	دنباله جزئی
Scheduling	زمانبندی
Makespan	زمان اتمام کل سیستم
Static scheduling	زمانبندی ایستا
Dynamic scheduling	زمانبندی پویا
Idle time	زمان بیکاری
Processing time	زمان پردازش
Completion time	زمان تکمیل
Global	سراسری
Constructive	سودمند
Crossover operator	عملگر ادغام
Insertion operator	عملگر الحاق
Mutation operator	عملگر جهش
Exchange operator	عملگر مبادله
Metaheuristice	فرا ابتکاری
Performance	کارایی

واژه نامه

Jopshop .... کارکارگاهی

Open shop .... کار کارگاهی بدون شرط

Flow shop .... کارگاه گردش کاری

Coromosome .... کروموزوم

Quality .... کیفیت

Pair – Wise exchange .... مبادله دو به دو

Total flow time .... مجموع زمان تکمیل کارها

Combine .... مرکب

Criterion .... معیار

compair .... مقایسه

Position .... موقعیت

Generation .... نسل

Crossover rate .... نرخ ادغام

Mutation rate .... نرخ جهش

Result .... نتیجه

Neighborhood .... همسایگی

پیوست C

کد Matlab الگوریتم فرامینان و لیستن

```
function [totallflowtime]=faranian(m,n,A)
k=1;
for j=1:1:m
   sumrows(j)=sum(A(j,:));
end
[sums,IX]=sort(sumrows);
%[y,index]=min(sumrows);
index=IX(1,1);
IX=IX(1,2:length(IX));
s=index;
%R=(1:m);
%if index==1
  % R=R(1,2:length(R));
%else
   %R=cat(2,R(1,1:index-1),R(1,index+1:length(R)));
%end
while k~=m
k=k+1;
minjob=IX(1,k-1);
temp=s;
for p=1:1:length(temp)+1

     if p==1
        set=cat(2,minjob,temp);
     else t=cat(2,temp(1,1:p-1),minjob,temp(1,p:length(temp)));
   set=cat(1,set,t);
     end
end
[x,y]=size(set);
for i=1:1:x
   array1=set(i,:);
   for j=1:1:length(array1)
      if j==1
         mat=A(array1(j),:);
      else s=A(array1(j),:);
     mat=cat(1,mat,s);
      end
   end
   [l1,l2]=size(mat);
if i==1
   tft=cost1(l1,l2,mat);
   %makespan=cost2(l1,l2,mat,n);
else tft2=cost1(l1,l2,mat);
```

```
  %makespan2=cost2(l1,l2,mat,n);
tft=cat(2,tft,tft2);
%makespan=cat(2,makespan,makespan2);
end
end
[y,IX1]=sort(tft);
s=set(IX1(1,1),:);
%index=IX(1,k-1);
%if index==1
   %R=R(1,2:length(R));
%else R=cat(2,R(1,1:index-1),R(1,index+1:length(R)));
%end
if k>2
D=zeros(1,length(s));
for i=1:1:k-1
   for j=i+1:1:k
      temp=s;
      l=temp(i);
      temp(i)=temp(j);
      temp(j)=l;
      D=cat(1,temp,D);
   end
end
[x1,y1]=size(D);
for i=1:1:x1-1
   array4=D(i,:);
   for j=1:1:length(array4)
      if j==1
         mat1=A(array4(j),:);
      else s1=A(array4(j),:);
     mat1=cat(1,mat1,s1);
      end
   end
   [l1,l2]=size(mat1);
if i==1
    tft=cost1(l1,l2,mat1);
    %makespan=cost2(l1,l2,mat1,n);
else tft2=cost1(l1,l2,mat1);
  % makespan2=cost2(l1,l2,mat1,n);
tft=cat(2,tft,tft2);
%makespan=cat(2,makespan,makespan2);
end
end
[y2,IX2]=sort(tft);
c=D(IX2(1,1),:);
if y2(1)<y(1)
```

```
  s=c;
end
%index=IX(1,1);
%if index==1
  %R=R(1,2:length(R));
%else R=cat(2,R(1,1:index-1),R(1,index+1:length(R)));
%end
end
end
result=s;
if y2(1)<y(1)
  totallflowtime=y2(1);
else
  totallflowtime=y(1);

end
end
```

کد Matlab الگوریتم دیپاک لاها و سرین

```
function [totallflowtime]=Dipak laha(m,n,A)
k=1;
for j=1:1:m
   sumrows(j)=sum(A(j,:));
end
[sums,IX]=sort(sumrows,2);
%[y,index]=min(sumrows);
index=IX(1,1);
IX=IX(1,2:length(IX));
s=index;
%R=(1:m);
%if index==1
  % R=R(1,2:length(R));
%else
   %R=cat(2,R(1,1:index-1),R(1,index+1:length(R)));
%end
while k~=m
k=k+1;
minjob=IX(1,k-1);
temp=s;
for p=1:1:length(temp)+1

     if p==1
        set=cat(2,minjob,temp);
     else t=cat(2,temp(1,1:p-1),minjob,temp(1,p:length(temp)));
   set=cat(1,set,t);
     end
end
[x,y]=size(set);
for i=1:1:x
   array1=set(i,:);
   for j=1:1:length(array1)
      if j==1
         mat=A(array1(j),:);
      else s=A(array1(j),:);
     mat=cat(1,mat,s);
      end
   end
   [l1,l2]=size(mat);
if i==1
   tft=cost1(l1,l2,mat);
    %makespan=cost2(l1,l2,mat,n);
else tft2=cost1(l1,l2,mat);
```

```
  %makespan2=cost2(l1,l2,mat,n);
tft=cat(2,tft,tft2);
%makespan=cat(2,makespan,makespan2);
end
end
[y,IX1]=sort(tft);
s=set(IX1(1,1),:);
%index=IX(1,k-1);
%if index==1
   %R=R(1,2:length(R));
%else R=cat(2,R(1,1:index-1),R(1,index+1:length(R)));
%end
if k>2
   s1=setdiff(s,minjob);

      for j=1:length(s1)
        if j==1
           f=s;
           temp=f;
           job=s1(j);
           for i=1:length(s)
              if f(i)==job;
        temp(i)=[];
              end
           end

      for p=1:length(temp)+1
if p==1
        set=cat(2,job,temp);
     else t=cat(2,temp(1,1:p-1),job,temp(1,p:length(temp)));
   set=cat(1,set,t);
end
      end
   else  f=s;
           temp=f;
           job=s1(j);
           for i=1:length(s)
              if f(i)==job;
        temp(i)=[];
              end
           end
      for p=1:length(temp)+1
if p==1
        set1=cat(2,job,temp);
     else t=cat(2,temp(1,1:p-1),job,temp(1,p:length(temp)));
   set1=cat(1,set1,t);
```

```
end
        end
        set=cat(1,set,set1);
          end

      end

end
[x1,y1]=size(set);
for i=1:1:x1
   array4=set(i,:);
   for j=1:1:length(array4)
      if j==1
         mat1=A(array4(j),:);
      else s1=A(array4(j),:);
     mat1=cat(1,mat1,s1);
      end
   end
   [l1,l2]=size(mat1);
if i==1
    tft=cost1(l1,l2,mat1);
    %makespan=cost2(l1,l2,mat1,n);
else tft2=cost1(l1,l2,mat1);
  % makespan2=cost2(l1,l2,mat1,n);
tft=cat(2,tft,tft2);
%makespan=cat(2,makespan,makespan2);
end
end
[y2,IX2]=sort(tft);
c=set(IX2(1,1),:);
if y2(1)<y(1)
   s=c;
end

end

result=s;
if y2(1)<y(1)
  totallflowtime=y2(1);
else
   totallflowtime=y(1);

end
end
```

کد *Matlab* اولین الگوریتم پیشنهادی

```
function [totallflowtime]=mahmodian3(m,n,A)
k=1;
for j=1:1:m
   sumrows(j)=sum(A(j,:));
end
[sums,IX]=sort(sumrows);
%[y,index]=min(sumrows);
index=IX(1,1);
IX=IX(1,2:length(IX));
s=index;
while k~=m
k=k+1;
minjob=IX(1,k-1);
temp=s;
for p=1:1:length(temp)+1

      if p==1
         set=cat(2,minjob,temp);
      else t=cat(2,temp(1,1:p-1),minjob,temp(1,p:length(temp)));
   set=cat(1,set,t);
      end
end
[x,y]=size(set);
for i=1:1:x
   array1=set(i,:);
   for j=1:1:length(array1)
      if j==1
         mat=A(array1(j),:);
      else s=A(array1(j),:);
     mat=cat(1,mat,s);
      end
   end
   [l1,l2]=size(mat);
if i==1
    tft=cost1(l1,l2,mat);
else tft2=cost1(l1,l2,mat);
tft=cat(2,tft,tft2);
end
end
[y,IX1]=sort(tft);
s=set(IX1(1,1),:);

if k>2
```

```
D=zeros(1,length(s));
for i=1:1:k-1
   for j=i+1:1:k
      temp=s;
      l=temp(i);
      temp(i)=temp(j);
      temp(j)=l;
      D=cat(1,temp,D);
   end
end
[x1,y1]=size(D);
for i=1:1:x1-1
   array4=D(i,:);
   for j=1:1:length(array4)
      if j==1
         mat1=A(array4(j),:);
      else s1=A(array4(j),:);
    mat1=cat(1,mat1,s1);
      end
   end
   [l1,l2]=size(mat1);
if i==1
    tft=cost1(l1,l2,mat1);
    %makespan=cost2(l1,l2,mat1,n);
else tft2=cost1(l1,l2,mat1);
  % makespan2=cost2(l1,l2,mat1,n);
tft=cat(2,tft,tft2);
%makespan=cat(2,makespan,makespan2);
end
end
[y2,IX2]=sort(tft);
c=D(IX2(1,1),:);
if y2(1)<y(1)
   s=c;
end

end
end
result=s;
if y2(1)<y(1)
  totallflowtime=y2(1);
else
   totallflowtime=y(1);

end
end
```

کد *Matlab* دومین الگوریتم پیشنهادی

```
function [totallflowtime]=mahmodian4(m,n,A)
k=1;
for j=1:1:m
   sumrows(j)=sum(A(j,:));
end
[sums,IX]=sort(sumrows,2);
%[y,index]=min(sumrows);
index=IX(1,1);
IX=IX(1,2:length(IX));
s=index;
R=(1:m);
if index==1
   R=R(1,2:length(R));
else
   R=cat(2,R(1,1:index-1),R(1,index+1:length(R)));
end
while k~=m
k=k+1;
[x2,y2]=size(R);
for i=1:x2
   Ri=R(i,:);
   if i==1
for j=1:1:length(Ri)
   l=Ri(j);
   temp=s(i,:);
      if j==1
         for p=1:length(temp)+1
     if p==1
        set1=cat(2,l,temp);
    else g=cat(2,temp(1,1:p-1),l,temp(1,p:length(temp)));
   set1=cat(1,set1,g);
     end
         end
         set=set1;
      else
          for p=1:length(temp)+1
     if p==1
        set1=cat(2,l,temp);
    else g=cat(2,temp(1,1:p-1),l,temp(1,p:length(temp)));
   set1=cat(1,set1,g);
     end
          end
 set=cat(1,set,set1);
```

```
        end
end
    else
   for j=1:1:length(Ri)
    l=Ri(j);
    temp=s(i,:);
          if j==1
              for p=1:length(temp)+1
        if p==1
            set1=cat(2,l,temp);
      else g=cat(2,temp(1,1:p-1),l,temp(1,p:length(temp)));
    set1=cat(1,set1,g);
        end
              end
              set3=set1;
          else
               for p=1:length(temp)+1
        if p==1
            set1=cat(2,l,temp);
      else g=cat(2,temp(1,1:p-1),l,temp(1,p:length(temp)));
    set1=cat(1,set1,g);
        end
               end
   set3=cat(1,set3,set1);
         end
   end
set=cat(1,set,set3);
      end
end

[x,y]=size(set);
for i=1:1:x
   for j=i+1:x
      if set(i,:)==set(j,:);
         set(j,:)=0;
      end
   end
end
for i=1:x
   if set(i,:)~=0;
   array1=set(i,:);
   for j=1:1:length(array1)
      if j==1
         mat=A(array1(j),:);
      else s=A(array1(j),:);
      mat=cat(1,mat,s);
```

```
      end
    end

    [l1,l2]=size(mat);
 if i==1
    tft=cost1(l1,l2,mat);
     %makespan=cost2(l1,l2,mat,n);
 else tft2=cost1(l1,l2,mat);
   % makespan2=cost2(l1,l2,mat,n);
 tft=cat(2,tft,tft2);
 %makespan=cat(2,makespan,makespan2);
 %sumtm=tft+makespan;
 end
    else
       tft12=max(tft);
    tft=cat(2,tft,tft12);
    end
 end
 [y,IX]=sort(tft);
 for i=1:3
    if i==1
       s=set(IX(1,1),:);
    else d=set(IX(1,i),:);
       s=cat(1,s,d);
    end
 end
 s1=[1:m];
 for i=1:3
   if i==1
      b1=s(i,:);
    R1=setdiff(s1,b1);
   else
      bi=s(i,:);
      Ri=setdiff(s1,bi);
      R1=cat(1,R1,Ri);
   end
   R=R1;
 end
 end

 s=set(IX(1,1),:);

 totallflowtime=y(1);
 end
```

کد *Matlab* بررسی تمام جایگشت های یک دنباله n تایی برای به دست آوردن جواب بهینه

```
function [totallflowtime]=jaygasht(m,n,A)
k=1;
setj=[1];
s=(2:m);
while k~=m
   k=k+1;
   [x,y]=size(setj);
    for i=1:x
      setji=setj(i,:);
      temp=s(1,k-1);

      if i==1
for p=1:1:length(setji)+1
     if p==1
        set=cat(2,temp,setji);
     else t=cat(2,setji(1,1:p-1),temp,setji(1,p:length(setji)));
   set=cat(1,set,t);
     end
end
      else
         for p=1:1:length(setji)+1
     if p==1
        set1=cat(2,temp,setji);
     else t1=cat(2,setji(1,1:p-1),temp,setji(1,p:length(setji)));
   set1=cat(1,set1,t1);
     end
         end
         set=cat(1,set,set1);
      end
    end
   setj=set;
end
set=setj;
[x1,y1]=size(set);
for i=1:1:x1
   array1=set(i,:);
   for j=1:1:length(array1)
      if j==1
         mat=A(array1(j),:);
      else s=A(array1(j),:);
      mat=cat(1,mat,s);
      end
   end
   [l1,l2]=size(mat);
```

```
if i==1
   tft=cost1(l1,l2,mat);
    %makespan=cost2(l1,l2,mat,n);
else tft2=cost1(l1,l2,mat);
  % makespan2=cost2(l1,l2,mat,n);
tft=cat(2,tft,tft2);
%makespan=cat(2,makespan,makespan2);
%sumtm=tft+makespan;
end
end
[y2,IX]=sort(tft);
totallflowtime=y2(1);
result=set(IX(1,1),:);
end
```

کد *Matlab* محاسبه مجموع زمان تکمیل کارها برای یک دنباله

```
function [tft]=cost1(i,j,A)
tft=0;
  c=zeros(i+1,j+1);
      for k=2:1:i+1
       for   l=2:1:j+1

      c(k,l)=max(c(k-1,l),c(k,l-1))+A(k-1,l-1);
       end
       tft=tft+c(k,l);
      end

end
```

کد *Matlab* محاسبه زمان اتمام کل سیستم برای یک دنباله

```
function [makespan]=cost2(i,j,A)
  c=zeros(i+1,j+1);
      for k=2:1:i+1
       for   l=2:1:j+1
      c(k,l)=max(c(k-1,l),c(k,l-1))+A(k-1,l-1);
       end
      end
      makespan=c(i+1,j+1);
end
```

کد *Matlab* مقایسه دو الگوریتم با در نظر گرفتن بیست مثال برای n های کوچک

```
 function [ARPD1,ARPD2,nooptimal1,nooptimal2]=main(m,n,q)
j=0;
k=0;
for i=1:20
   if i==1
      Ai=random('unif',1,99,m,n);
      x1=jaygasht(m,n,Ai);
      e1=cputime;
      x2=faranian(m,n,Ai);
      t2=cputime-e1;
      e2=cputime;
      x3=mahmodian3(m,n,Ai,q);
      t3=cputime-e2;
      if x1==x2
         j=j+1;
      end
      if x1==x3
         k=k+1;
      end
       ARPD1=(x3-x1)/x1;
       ARPD2=(x2-x1)/x1;
      s=cat(2,x1,x2,x3);
      t=cat(2,t2,t3);

   else
      Ai=random('unif',1,99,m,n);
      x1=jaygasht(m,n,Ai);
      e1=cputime;
      x2=faranian(m,n,Ai);
      t2=cputime-e1;
      e2=cputime;
      x3=mahmodian3(m,n,Ai,q);
      t3=cputime-e2;
      if x1==x2
         j=j+1;
      end
      if x1==x3
         k=k+1;
      end
      ARPDi1=(x3-x1)/x1;
      ARPDi2=(x2-x1)/x1;
      s1=cat(2,x1,x2,x3);
      t12=cat(2,t2,t3);
```

```
    s=cat(1,s,s1);
    t=cat(1,t,t12);
    ARPD1=cat(2,ARPD1,ARPDi1);
    ARPD2=cat(2,ARPD2,ARPDi2);
    end
end
set=s;
time=t
Average1=mean(ARPD1,2);
Average2=mean(ARPD2,2);
ARPD1=100*Average1
ARPD2=100*Average2
nooptimal1=k
nooptimal2=j
  end
```

کد *Matlab* مقایسه دو الگوریتم با در نظر گرفتن بیست مثال برای n های بزرگ

```
function [ARPD1,ARPD2,nooptimal1,nooptimal2]=main2(m,n,q)
j=0;
k=0;
for i=1:20
   if i==1
      Ai=random('unif',1,99,m,n);
      e1=cputime;
      x2=faranian(m,n,Ai);
      t2=cputime-e1;
      e2=cputime;
      x3=mahmodian3(m,n,Ai,q);
      t3=cputime-e2;
      if x3>x2
         j=j+1;
         x1=x2;
      elseif x3<x2;
         k=k+1;
         x1=x3;
      else
         x2=x3;
         k=k+1;
         j=j+1;
         x1=x3;
      end
       ARPD1=(x3-x1)/x1;
       ARPD2=(x2-x1)/x1;
      s=cat(2,x2,x3);
      t=cat(2,t2,t3);

   else
      Ai=random('unif',1,99,m,n);
      e1=cputime;
      x2=faranian(m,n,Ai);
      t2=cputime-e1;
      e2=cputime;
      x3=mahmodian3(m,n,Ai,q);
      t3=cputime-e2;
      if x3>x2
         j=j+1;
         x1=x2;
      elseif x3<x2;
         k=k+1;
         x1=x3;
```

```
        else
          x2=x3;
          k=k+1;
          j=j+1;
          x1=x3;
        end
        ARPDi1=(x3-x1)/x1;
        ARPDi2=(x2-x1)/x1;
        s1=cat(2,x2,x3);
       t12=cat(2,t2,t3);
   s=cat(1,s,s1);
   t=cat(1,t,t12);
   ARPD1=cat(2,ARPD1,ARPDi1);
   ARPD2=cat(2,ARPD2,ARPDi2);
   end
end
set=s;
time=t
Average1=mean(ARPD1,2);
Average2=mean(ARPD2,2);
ARPD1=100*Average1
ARPD2=100*Average2
nooptimal1=k
nooptimal2=j
  end
```

Abstract

Scheduling problem are considered optimization problems and many works have been done on these problems and their development and progress up to now.

Diversity and complexity of scheduling problems made us to merely explore the scheduling problems of n job on m machine in the environment of permutation flow shop with given process time in this thesis. we present two heuristic models with the aim of minimizing total flow time and assess these two models considering final answer optimization and compare their answers with one of the existing models.

In this thesis, firstly scheduling problems and its primary concepts are introduced according to subject matter essentials and then scheduling problems of permutation flow shop are introduced so that the reader can attain a relative familiarity with the problem after reading this chapter. At the third chapter, literature of this subject is reviewed and among the many of heuristic algorithms proposed by researchers, a number of the best algorithms are described. Of course it should be mentioned that the researcher in this thesis has written some of the algorithms proposed in this chapter by using Matlab codes of the program and the results of comparisons are mentioned in this chapter. At the fourth chapter, two innovative algorithms are proposed with the criterion of minimizing total time of doing jobs and aforementioned two models are programmed by using Matlab codes and compared by one of the best existing methods.

finally in the fifth chapter, hybrid genetic algorithms proposed by Lin-Yu Tseng and Ya-Tai Lin in 2009 is completely described and the efficiency of this method with respect to the other is explored.

Keywords: scheduling; permutation flow shop; heuristic algorithm; hybrid genetic algorithm