ارایه الگوریتمی کارا برای حل مساله فلوشاپ ترکیبی با محدودیت انباره‌های میانی

رضا توکلی مقدم و فریبرز جولای
گروه مهندسی صنایع، دانشکده فنی، دانشگاه تهران
tavakoli@ut.ac.ir ; fjolai@ut.ac.ir
هاشم وحدانی
دانشجوی دکتری مهندسی صنایع، دانشگاه صنعتی شریف
hashemvahdani@yahoo.com

واژه‌های کلیدی:
مساله فلوشاپ ترکیبی، مسدود شدن، انباره‌های میانی، شبیه‌سازی گسسته پیشامد، الگوریتم‌های فرا ابتکاری.

چکیده:

مساله فلوشاپ ترکیبی با هدف کمینه کردن زمان کل انجام کارها و با محدودیت ظرفیت انباره‌های میانی از جمله مسایل بهینه‌سازی ترکیبی در خانواده مسایل NP-Hard به شمار می‌رود. در این مقاله، بین هر دو ایستگاه متوالی محلی برای نگهداری کارهای نیمه تمام با ظرفیت محدود در نظر گرفته شده که این محدودیت بحث مسدود شدن ماشین‌ها را پیش می‌کشاند و بر پیچیدگی مساله بطور قابل ملاحظه‌ای می‌افزاید. جواب مساله برداری با طول N (تعداد کارها) می‌باشد که توالی انجام کارها در ایستگاه اول را مشخص می‌کند. به موازات هر جواب برداری یک فرایند تکمیلی بر اساس اصول شبیه‌سازی گسسته-پیشامد و تعریف قوانین تقدم-تاخری خاص با هدف ارایه کاراترین برنامه زمانبندی عملیات توسعه داده شده است. در این مقاله بعد از تشریح مدل به منظور یافتن کاراترین الگوریتم حل برای این مساله از دو الگوریتم فرا ابتکاری با تاکید بر روش ترکیبی ژنتیک- جستجوی ممنوع استفاده شده و مقایسه بین این روشها و کاراترین روش‌های ارایه شده در مقالات عملی ارایه شده است. با بهره‌گیری از خصوصیات مثبت هر یک از الگوریتم‌های ژنتیک و جستجوی ممنوع و ایجاد ترکیب مناسب از عملگرهای مختلف این دو الگوریتم در جستجوی فضای جواب، شاهد افزایش قابل ملاحظه قدرت الگوریتم پیشنهادی در رسیدن به بهترین جواب، در مقایسه با سایر روشها خواهیم بود. از جمله دستاوردهای جنبی این مقاله، بهره‌گیری از روش پیشنهادی برای حل مسایل با ساختارهای جواب مشابه(جایگشتی از N مولفه)، از جمله مساله برنامه ریزی تک ماشینه می‌باشد که به نوبه خود حائز اهمیت است.

۱- مقدمه

مساله زمانبندی فلوشاپ با ماشین‌های موازی (FSPM)[1] با هدف کمینه‌کردن کل زمان انجام کارها[2] که بـه فلوشـاپ ترکیبـی نیـز معـروف است، در سالهای اخیر توجه زیادی را به خود جلب کرده است. این مساله به عنوان زیر مجموعـه‌ای از مـسایل عمـومی‌تـر خطـوط جریـان انعطاف‌پذیر[3] محسوب می‌شود که در مقایسه با این سیستم‌ها از لحاظ ترتیبی که کارها به ایستگاههای مختلف وارد می‌شـوند، تفـاوت‌هـایی وجود دارد. سیستم تولیدی مزبور از چندین ایستگاه کاری متوالی با مجموعه‌ای از ماشین‌های موازی در هر ایستگاه که همه کارهـا بـا یـک توالی یکسان تمامی مراحل را پشت سر می‌گذارند، تشکیل شده است. درهر مرحله، هر کاری مجاز است بر روی ماشینی که در حال حاضـر آماده به کار است تخصیص یابد و مدت زمان انجام کار برای تمامی ماشین‌ها در هر ایستگاه برای یک کار مشخص، یکـسان مـی‌باشـد. از موارد قابل توجه مدل مزبور این است که با در نظر گرفتن یک ماشین در هر ایستگاه، مساله به حالت فلوشـاپ معمـولی تبـدیل مـی‌شـود و بدین ترتیب تمامی قواعد و نتایجی که در مورد سیستم‌های فلوشاپ صدق می‌کند، کاملاً قابل تعمیم به مدل فوق می‌باشند.

مسالهٔ FSPM در ساده‌ترین حالت خود یعنی بدون در نظر گرفتن هر گونه محدودیت‌هـای جنبـی از جملـه محـدودیت ظرفیـت انبارههـای میانی، زمان‌های آماده سازی وابسته به توالی و غیره به شدت NP-Hard می‌باشد. بدین معنا که دست‌یابی به جواب بهینـه مـساله در یـک زمان چند جمله‌ای وابسته به اندازهٔ مساله، میسر نمی‌باشد. گری [۱] نشان داد در صورتیکه تعداد ایستگاه‌های متوالی بیش از سه عدد باشد و در هر ایستگاه تنها یک ماشین موجود باشد، مساله به شدت NP-Hard می‌باشد، چه قطع کار مجـاز باشـد و چـه نباشـد. بـدین ترتیـب بـا اضافه‌شدن ماشین‌های موازی در هر ایستگاه و همچنین در نظرگرفتن محدودیت ظرفیت انبارههای میانی، مطابق فرضیات موجـود در ایـن مقاله، پیچیدگی مساله بهینه‌سازی ترکیبی فوق به شدت افزایش می‌یابد.

تاکنون الگوریتم‌های مختلفی برای حل بهینهٔ مساله فوق ارایه شده که از مهمترین آنها می‌توان به روش شاخه و تحدید [۲] و برنامـه‌ریـزی عدد صحیح مختلط [۳] اشاره کرد. با این وجود نتایج محاسباتی نشان می‌دهد که عملاً بکارگیری این الگوریتم‌ها در صورتیکه تعـداد کارهـا از یک حد معین بیشتر شود عملاً غیر ممکن می‌باشد [۴]. بدین ترتیب الگوریتم‌های فرا ابتکاری[4] جهت دستیابی بـه بهتـرین جـواب‌هـای ممکن (نه لزوماً بهینه) تنها ابزار موجود جهت حل مسأله به شمار می‌روند که البته در سالهای اخیـر اسـتفاده از ایـن الگـوریتم‌هـا بـویژه در مسایل زمانبندی عملیات، بشدت افزایش پیدا کرده است. ساندرا راجاوان و سایرین [۵] روشی را برای حل مسألهٔ خاصی که شامل دو مساله فلوشاپ موازی بوده و تعداد ماشین‌ها در هر فلوشاپ برابر دو می‌باشد، ارایه کردند. در ایـن مـدل، مجموعـهٔ ماشـین‌هـای یـک فلوشـاپ از فلوشاپ دیگر سریع‌ترمی‌باشد و زمانی که یک قطعه به یک فلوشاپ تخصیص داده شد، دیگر نمی‌توانـد بـه فلوشـاپ دیگـر در مرحلـهٔ بعـد منتقل شود.

لی و ویراکتاراکیس [۶] الگوریتمی سازنده برای مساله فلوشاپ ترکیبی دو مرحله‌ای توسعه داده و سپس الگوریتمی سازنده و عمـومی بـرای مساله فلوشاپ ترکیبی با L ایستگاه ارایه دادند. آنها برای مساله فلوشاپ ترکیبی دو مرحله‌ای، از قاعدهٔ جانسون برای تولید یک توالی اولیـه استفاده کردند که با کمک قاعدهٔ FAM[5] و LBM[6] توالی نهایی ایجاد می‌شد. برای مساله فوق با L ایستگاه، آنها الگوریتم خود را برای هـر دو ایستگاه متوالی مورد استفاده قرار دادند و در صورت فرد بودن تعداد ایستگاهها، از یک ایستگاه مجازی با تعداد صفر ماشین در آن استفاده کردند.

نویکی و همکاران [۷] از الگوریتم جستجوی ممنوع(TS) برای حل مساله فلوشاپ ترکیبی L مرحله‌ای با هدف کمینه کردن کل زمان انجام کارها استفاده کردند. در روش ایشان، یک توالی کامل بصورت (I_{j1},C_{j1}) نشان داده می‌شود که بردار I_{j1} نشان دهندهٔ ماشینی است کـه کـار j

1) Flow Shop with Parallel Machines (FSPM)
2) Makespan
3) Flexible Flow Line
4) Meta-heuristic
5) First Available Machine
6) Last Busy Machine

ام در مرحلهٔ l ام به آن تخصص می‌یابد و C_{jl} نشان دهندهٔ زمان اتمام کار jام در مرحلهٔ l ام می‌باشد. آنها از روش جایگذاری[1] برای جستجو در فضای همسایگی هر جواب استفاده کردند. ساختار بکار برده از یک بردار V بصورت (l,a,x,b,y) تشکیل شده است که نشان دهندهٔ جایگزینی کاری که در مکان x ام و بر روی ماشین a ام در مرحلهٔ l ام قرار دارد به جای کاری است که در مکان y ام بر روی ماشین b ام در همان مرحله قرار دارد. به منظور حذف حرکت‌های زائد[2] آنها قوانینی را برای حذف حرکت‌هایی که مشخصاً هیچ‌گونه بهبودی در جواب مساله ایجاد نمی‌کنند، تعریف کردند و بدین ترتیب تا حدی اندازهٔ بردار همسایگی کاهش یافت. اخیراً نگنمان [۸] چندین الگوریتم جستجوی محلی از جمله جستجوی ممنوع و آنیلینگ شبیه‌سازی شده (SA) را برای حل مساله FSPM توسعه داده است. در هر یک از این الگوریتم‌ها او چندین ساختار مختلف برای ایجاد همسایگی یک جواب و جستجو در همسایگی مزبور از جمله روشی که نویکی [۷] ارایه داده بود را بررسی کرد. برای سنجش کارایی الگوریتم‌ها او از پایان‌نامه وندولد [۹] که چندین روش مختلف برای محاسبهٔ حد پایین جواب (LB[3]) ارایه داده است، استفاده کرد. نتایج نشان دهندهٔ آن بود که روش TS با ساختار همسایگی که توسط نویکی ارایه شده، کارائی بالاتری نسبت به سایر الگوریتم‌ها دارد.

جدیدترین مقاله‌ای که به مساله فوشاپ ترکیبی با محدودیت انباره‌های میانی پرداخته، مربوط به سال ۲۰۰۴ می‌باشد که توسط واردونو و فتحی [۱۰] ارایه شده است. در این مقاله با ارایه روشی سازنده، توالی کامل کارها براساس بردار جواب اولیه که نشان دهندهٔ توالی انجام کارها در اولین ایستگاه می‌باشد، حاصل شده است. سپس با تعریف نوع خاصی از همسایگی براساس جابجایی‌های دوتایی کارها در بردار اولیه، و با استفاده از الگوریتم TS جوابهای نهایی مساله با هدف کمینه کردن کل زمان انجام کارها، ارایه شده است. با توجه به نتایج بسیار مناسب ارایه شده در مقاله ایشان و همچنین وجود تشابه مستقیم در مساله مورد بررسی، معیار اصلی مقایسه کارایی لگوریتم‌های بکار رفته در این تحقیق، همین مقاله انتخاب شده است.

۲- تعریف مساله

فرضیات بکار رفته در این مقاله جهت ارایه مدل به قرار زیر می‌باشد:

۱- تمامی کارها در ابتدای افق برنامه‌ریزی موجود بوده و آمادهٔ انجام عملیات‌درایستگاه اول می‌باشند.

۲- هر ماشین در هر لحظه از زمان فقط قادر به انجام یک کار می‌باشد.

۳- مدت زمان انجام عملیات مربوط به هر کار در هر ایستگاه کاری کاملاً مشخص بوده و برای تمامی ماشین‌ها در آن ایستگاه یکسان می‌باشد.

۴- ماشین‌ها در سرتاسر افق برنامه‌ریزی موجود بوده و قطع کار در هیچ زمانی مجاز نمی‌باشد.

۵-زمانهای آماده سازی برای هر کار با مدت زمان انجام کارها در هر ایستگاه تلفیق شده و کاملاً مستقل از توالی انجام کارها می‌باشند.

۶- ظرفیت انباره‌های میانی بین هر دو ایستگاه متوالی محدود بوده و مشخص شده است. مساله FSPM با این فرض بصورت FSPM/b نمایش داده می‌شود که b بردار ظرفیت انباره‌ها می‌باشد.

پارامترهای بکار رفته در مدل به قرار زیر می‌باشد:

L: تعداد ایستگاههای کاری

m_l: تعداد ماشین‌های موازی در ایستگاه l ام. $(l = 1, \dots L)$

b_l: حداکثر ظرفیت انباره موجود بین ایستگاه l و $l+1$. $(l = 1, \dots, L-1)$

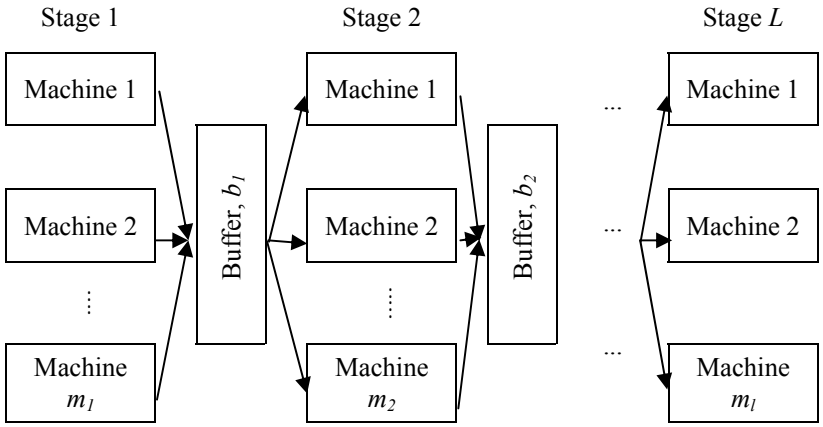
N: تعداد کل کارها

P_{jl}: مدت زمان انجام کار jام در ایستگاه l ام $(j = 1, \dots N), (l = 1, \dots, L)$

C_{max}: مقدار makespan برای هر توالی ارایه شده .

1) Insertion
2) Useless Moves
3) Lower Bound

جهت تبیین هر چه بهتر موضوع، در شکل (۱) نمایش تصویری سیستم تولیدی فوق ارایه شده است:



شکل ۱- شماتیک سیستم تولیدی فلوشاپ ترکیبی با محدودیت انبارهای میانی

مطابق شکل، سیستم فوق از L ایستگاه کاری تشکیل شده است که در هر ایستگاه m_l ماشین مشابه جهت خدمت‌دهی موجود می‌باشند. انباره موجود بین ایستگاه l و l+1 ام را به عنوان انباره l ام می‌شناسیم (l = 1,…, L -1) که ظرفیتی به اندازهٔ b_l دارد. مجموعه‌ای از N کار در اختیار داریم، که اتمام یک کار مستلزم گذراندن تمامی L مرحلهٔ موجود می‌باشد. تمامی کارها یک مسیر ثابت را با شروع از ایستگاه اول تا ایستگاه L ام طی می‌نمایند. به محض خاتمهٔ عملیات یک کار روی یک ماشین معین، در صورت وجود ماشین بیکار در مرحلهٔ بعد کار مزبور به آن ماشین تخصیص می‌یابد. در غیر اینصورت کار باید به انباره مربوطه منتقل شده و تا زمان خالی شدن هر یک از ماشین‌های مرحلهٔ بعد، در آنجا باقی بماند. در صورتیکه از حداکثر ظرفیت انباره‌ها توسط کارهای قبلی استفاده شده باشد، کار مورد نظر بر روی ماشین مربوطه باقی می‌ماند و در طی این مدت هیچ کاری قابل تخصیص به ماشین مزبور نمی‌باشد و به عبارت دیگر کار و ماشین هر دو مسدود[1] می‌شوند تا زمانیکه بتوان کار را به انبارهٔ بعدی منتقل کرد.

۲-۱- نمایش جوابهای مساله

قبل از هر تلاشی برای حل مساله باید ابتدا مناسبترین روش جهت ارایه جواب‌های مساله معرفی شود. بطور کلی دو روش مختلف برای این کار به صورت نمایش ماتریسی و نمایش برداری وجود دارد:

۲-۱-۱- نمایش ماتریسی جوابها

این روش اولین بار توسط نویکی [۷] برای مسئلهٔ FSPM مورد استفاده قرار گرفت. در این روش هر جواب مساله توسط دو ماتریس I و X با ابعاد (N * L) مشخص می‌شود که بصورت زیر تعریف می‌شوند.

I_{jl}: شمارهٔ ماشینی که کار jدر مرحلهٔ l بر روی آن انجام می‌شود. (j= 1,2,…,N) (l= 1,2,…, L)

X_{jl}: توالی انجام کار jام در ایستگاه l ام و برروی ماشین مربوطه را نشان می‌دهد.

بدین ترتیب به کمک این دو ماتریس قادر خواهیم بود تا تمامی فضای ممکن جواب مساله را تحت پوشش قرار دهیم. نکته‌ای که باید اشاره شود این است که با بکارگیری چنین روشی بر اساس نتایج ارایه شده در مقالات علمی از جمله مقاله [۷]، عملاً پیاده سازی الگوریتم‌های جستجوی محلی کارایی لازم را نخواهند داشت، چرا که با در اختیار داشتن یک جواب، تعداد جوابهای همسایه مسأله آنقدر زیاد می‌باشد که فرآیند جستجو را با مشکل

1) Blocked

مواجه می‌نمایند چرا که معمولاً خیلی از این جوابها، موجه نیز نمی‌باشند. با این توضیحات، در این مقاله از این روش برای نمایش جوابها استفاده نشده است.

۲-۱-۲- نمایش برداری جوابها

این روش اولین بار توسط واردونو و فتحی [۱۰] برای مساله FSPM/b مورد استفاده قرارگرفت. ایده اصلی این روش بر این اساس شکل گرفته که در یک توالی مناسب، هر کاری که زودتر برای انجام فرآیند در مراحل میانی موجود باشد، باید زودتر نیز پروسس شود. در صورتیکه در روش قبلی در بسیاری از حالتها، قطعه‌ای برای پروسس آماده است، اما چون در ترتیب تعریف شده درماتریس X_{jl} بعد از قطعهٔ دیگری قرار دارد که هنوز در ایستگاه قبلی درحال پروسس است، لذا بی‌جهت باید در صف انتظار باقی بماند که این خود بر کارایی روش اثر منفی می‌گذارد. در روش برداری هر جواب (S) در حقیقت نشان دهنده توالی انجام کارها در اولین ایستگاه می‌باشد که با برداری به طول N نمایش داده می‌شود. بدین ترتیب کل فضای جواب متشکل از $N!$ جواب می‌باشد که نسبت به روش ماتریسی به مراتب کوچکتر می‌باشد و این مساله در کارایی الگوریتم‌های حل که با توجه به پیچیدگی این مسایل اغلب روشهای جستجوی محلی می‌باشند تاثیر بسیار مثبتی دارد.

۳- چگونگی ارایه برنامه زمانبندی کامل انجام کارها در ایستگاه‌های مختلف براساس بردار جواب اولیه

با در اختیار داشتن بردار جواب (S) با کمک دو پروسهٔ مختلف می‌توانیم توالی کامل عملیات را برای مساله FSPM/b بدست آوریم. در پروسهٔ اول براساس قانون FAM (اولین ماشین در دسترس) کارها را مطابق توالی موجود در بردار به ماشین‌های موجود در اولین ایستگاه تخصیص می‌دهیم و زمان تکمیل هر یک را درلیستی تحت عنوان لیست پیشامدها ثبت می‌کنیم. در مراحل بعدی انتخاب کارها براساس ترتیب صعودی زمانهای تکمیل هر یک در مراحل قبلی واستفادهٔ مجدد از قانون FAM می‌باشد. نکته‌ای که باید به آن توجه داشت وجود انباره‌های با ظرفیت محدود می‌باشد که تغییراتی را در سیستم اعمال می‌کند. از جمله اینکه زمان شروع یک کار در یک ایستگاه لزوماً برابر زمان ختم آن در مرحلهٔ قبلی و یا تکمیل کار قبلی در این ایستگاه نمی‌باشد. چنانچه در زمانی که یک کار در یک ایستگاه تکمیل می‌شود، ظرفیت انباره مربوطه تکمیل شده باشد، کار مورد نظر مجبور است بر روی ماشین فعلی در ایستگاه فوق باقی بماند تا زمانیکه اولین ظرفیت انباره خالی ایجاد شود و به اصطلاح ماشین و قطعه هر دو مسدود می‌شوند. این مسأله بر روی کارهای موجود در انباره ایستگاه قبلی نیز اثر گذاشته و عملاً می‌توان شاهد تأثیر مسدود شدن یک ماشین بر روی تمامی ایستگاههای قبلی بود. این موضوع خود بر پیچیدگی مسأله می‌افزاید چرا که مسدود شدن یک ماشین بر روی دسترس پذیری ماشین‌های موجود درایستگاههای قبلی نیز اثرگذاشته و زمان‌های تکمیل کارها در ایستگاههای قبلی تماماً باید بازنگری شوند. با این توضیحات، برای ایجاد یک توالی کامل براساس یک بردار جواب اولیه از اصول شبیه سازی استفاده کرده و در حقیقت باکمک شبیه سازی گسسته پیشامد و بصورت گام به گام با تعریف برخی قوانین تقدمی با هدف دستیابی به کاراترین زمانبندی ممکن، قادر به ارایه توالی کامل عملیات هستیم .

فرآیند شبیه سازی با m_1 فعالیت که همان تخصیص m_1 کار به m_1 ماشین در ایستگاه اول و براساس اولویت موجود در جواب S است آغاز می‌شود. از آنجایی که زمان‌های پروسس هر کار در ایستگاهها مشخص است، بلافاصله قادریم زمان تکمیل m_1 کار را در مرحلهٔ اول مشخص کنیم. این پیشامدها به ترتیب صعودی زمان وقوع، مرتب شده و لیست اولیه پیشامدها[1] را ایجاد می‌کنند. از این به بعد در هر مرحلهٔ شبیه سازی، اولین پیشامد در لیست فوق با کوچکترین زمان وقوع انتخاب شده و با شناسایی اطلاعات مربوطه از جمله شمارهٔ کار و شمارهٔ ایستگاه و ماشین مربوطه اقدام مقتضی صورت می‌گیرد. در واقع هر پیشامد معادل زمان تکمیل یک کار مشخص در یک ایستگاه مشخص می‌باشد.

با تکمیل کار jدر ایستگاه l یکی از حالات زیر رخ می‌دهد:

1) Event List

حالت اول) اگر ایستگاه l آخرین مرحله برای کار j باشد یعنی $l = L$، کار مورد نظر از سیستم خارج شده و به لیست قطعات تکمیل شده (R) اضافه می‌شود. و وضعیت ماشین مربوط به حالت بیکار تغییر می‌یابد. این پیشامد خود، مجموعه‌ای از پروسه‌های مشخص را فراخوانی می‌کند که در ادامه تشریح می‌گردد.

حالت دوم) اگر ایستگاه l یک ایستگاه میانی باشد و انباره موجود بین ایستگاه l و $l+1$ ظرفیت خالی داشته باشد، کار j به این انباره اضافه می‌شود و وضعیت ماشین مربوطه به حالت بیکار تغییر می‌یابد. زمان ورود هر قطعه به انباره نیز باید ثبت شود چرا که در مراحل بعد برای تخصیص کارها به ایستگاه بعدی مورد نیاز می‌باشد (قوانین تقدمی). مطابق حالت اول این حالت نیز یک سری پروسه‌های مشخص را فراخوانی می‌کند که در ادامه تشریح خواهند شد.

حالت سوم) اگر ایستگاه l یک ایستگاه میانی باشد و ظرفیت انباره موجود بین ایستگاه l و $l+1$ تکمیل باشد، آنگاه وضعیت قطعهٔ j و ماشین مربوطه به حالت مسدود تغییر می‌یابد. در این حالت هیچ گونه پروسهٔ خاصی فراخوانی نمی‌شود و پیشامد بعدی موجود در لیست پیشامدها اجرا می‌شود.

در حالت اول و دوم، کار j که عملیات آن بر روی ماشین مربوطه تمام شده، از روی ماشین برداشته می‌شود و ماشین مربوطه و کار j برای انجام عملیات بعدی (در صورت وجود) آماده می‌باشند. این پیشامد باعث ورود قطعهٔ j به ایستگاه $l+1$ و یا خروج از سیستم و همچنین ورود قطعه‌ای دیگر موجود در انباره قبلی و یا ماشین مسدود شده به مرحلهٔ فعلی می‌شود. به هر ترتیب لیستی از فعالیت‌های مشخص باید اعمال شوند تا تمامی تغییر وضعیت‌های موجه در سیستم اتفاق بیفتد. برای این منظور این فعالیت‌ها را از ایستگاه $l+1$ اعمال کرده و بصورت برگشتی تا ایستگاه اول تغییرات را اعمال می‌کنیم. در اعمال این فعالیت‌ها در سراسر فرایند شبیه سازی، یک ماشین در صورتی در دسترس می‌باشد که مسدود نشده باشد و در حال حاضر هیچ کار دیگری بر روی آن انجام نشود.

فعالیت (۱) بعد از وقوع حالت دوم قابل اعمال می‌باشد یعنی در حالتیکه پیشامد فعلی مربوط به تکمیل کار j در ایستگاه $l < L$ می‌باشد.

فعالیت ۱– مربوط به ایستگاه $l+1$:

اگر یک ماشین در این مرحله در دسترس باشد، کار j از انباره مربوطه به این ماشین منتقل شده و آن انباره خالی می‌شود. زمان شروع کار j در این مرحله همان زمان وقوع پیشامد مربوطه می‌باشد و بلافاصله زمان تکمیل کار j در ایستگاه $l+1$ محاسبه و به لیست پیشامدها اضافه می‌شود و به ایستگاه l بر می‌گردیم.

فعالیت‌های (۲) و (۳) بعد از هر یک از حالت‌های اول و دوم باید اجرا شوند.

فعالیت ۲– مربوط به ایستگاه l:

به وضوح در این مرحله حداقل یک ماشین بیکار در دسترس می‌باشد، چرا که کار j به انباره مرحلهٔ بعدی و یا لیست کارهای تکمیل شده اضافه شده است. در صورتیکه کارهائی در انباره ما قبل ایستگاه l موجود باشند، عملیات را بر روی قطعه‌ای که بیشترین زمان انتظار در صف انباره مربوطه را دارد شروع و زمان تکمیل آن را به لیست پیشامدها اضافه می‌کنیم و انباره مربوطه را آزاد می‌کنیم.

فعالیت ۳– مربوط به ایستگاه‌های $l-1$ تا ایستگاه اول:

از ایستگاه $l-1$ شروع کرده و با حرکت رو به عقب، هر کدام از انباره‌ها که در هر مرحله آزاد می‌شوند، می‌توانند توسط قطعاتی که در ماشین‌های ماقبل خود مسدود شده‌اند (در صورت وجود) بکار گرفته شوند. در هر ایستگاه گام‌های زیر باید صورت گیرد:

گام ۱– در صورتیکه ماشین i در این مرحله در حالت مسدود شده باشد و فضای خالی در انباره بعدی موجود باشد کار مسدود شده را از ماشین به انباره مربوطه منتقل کرده و وضعیت ماشین را به حالت بیکار تغییر می‌دهیم.

گام ۲– در صورتیکه یک ماشین در ایستگاه فعلی بیکار باشد و قطعه‌ای در انباره ماقبل این ایستگاه موجود باشد، عملیات کار مربوطه در این ایستگاه را شروع و بلافاصله زمان تکمیل کار مربوطه را به لیست پیشامدها اضافه می‌کنیم. در صورت وجود بیش از یک قطعه در انباره مربوطه، کاری که زودتر وارد انباره شده از اولویت بالاتری برخوردار است (قانون تقدمی).

گام‌های فوق تا رسیدن به ایستگاه اول ادامه می‌یابد. در ایستگاه اول در صورت بیکار شدن یک ماشین، عملیات بر روی اولین کاری که در بردار S وجود دارد و هنوز وارد سیستم نشده، آغاز می‌شود (مگر اینکه هیچ کاری باقی نمانده باشد) و زمان تکمیل آن به لیست پیشامدها اضافه می‌شود. حال با اعمال کلیهٔ این تغییرات، به سراغ اولین رویداد موجود در لیست پیشامدها با کمترین زمان وقوع می‌رویم و این فرآیند ادامه می‌یابد تا زمانیکه هیچ پیشامدی در

لیست پیشامدها باقی نمانده باشد. بمنظور دستیابی به توالی کامل انجام همه کارها در تمامی ایستگاهها، هر زمان که کاری بر روی یک ماشین برنامه‌ریزی می‌شود، اطلاعات مربوط به شماره ماشین مربوطه و ترتیبی که کار بر روی آن انجام شده، ذخیره می‌شود. به این ترتیب با خاتمه پروسه فوق، می‌توان ماتریس‌های جواب I و X مرتبط با بردار S را از اطالاعات ذخیره شده استخراج نمود. در طول فرآیند شبیه‌سازی فوق با برخی حالتهای‌خاص مواجه خواهیم شد که در ادامه تمامی این حالات تشریح می‌گردد.

حالت خاص اول:

درصورتیکه چندین پیشامد در یک لحظه از زمان با هم رخ دهند، آنها را بر اساس ترتیب ایستگاههایی که پیشامدهای فوق در آنها رخ داده‌اند، مرتب می‌کنیم. ابتدا تغییرات مربوط به پیشامد ایستگاههای جلوتر را اعمال می‌کنیم و سپس به سراغ ایستگاههای قبلی می‌رویم. بعنوان مثال اگر در لحظه t بطور همزمان دو پیشامد، یکی مربوط به تکمیل کار در ایستگاه چهار و دیگری در ایستگاه دو رخ دهد، ابتدا پیشامد مرتبط با ایستگاه چهار را بررسی کرده و تغییرات اعمال می‌شود و سپس به سراغ پیشامد حادث شده در ایستگاه دو می‌رویم. دلیل این موضوع روشن است، چرا که نتایج فعالیتهایی که با تکمیل یک کار در ایستگاه چهار رخ می‌دهد، بر روی وضعیت سیستم در ایستگاههای قبلی اثر می‌گذارد (آزادسازی ماشین‌ها و انباره‌های قبلی)، درصورتیکه عکس این مساله صادق نیست.

حالت خاص دوم:

حال درصورتیکه چند پیشامد بطور همزمان در یک ایستگاه بوقوع بپیوندد و نتوان همه کارها را به انباره مربوطه منتقل کرد و ملزم به انتخاب یک کار جهت انتقال باشیم، کاری را انتخاب می‌کنیم که در ترتیب اولیه داده شده (بردار S) جلوتر قرار دارد یا بعبارت دیگر، پیشامد ورود آن به سیستم زودتر شکل گرفته است. درصورتیکه شرایط مشابهی برای انتخاب یک کار بین چند کار که بطور همزمان وارد انباره شده‌اند بوقوع بپیوندد، آن کاری را برای انتقال به ایستگاه بعدی انتخاب می‌کنیم که زمان تکمیل کمتری در ایستگاه فوق داشته باشد.

حالت خاص سوم:

فرآیند شبیه‌سازی فوق‌الذکر با فرض وجود انباره‌هایی با ظرفیت حداقل یک بین هر دو ایستگاه متوالی ارایه شد. در حالتیکه ظرفیت انباره بین ایستگاههای l و l+1 برابر صفر درنظر گرفته شود، حالتهای دوم و سوم بصورت زیر تغییر می‌یابند. این حالت برای ارزیابی کارایی الگوریتم‌های حل مساله مورد استفاده قرار خواهد گرفت.

– حالت دوم در حالتی که ظرفیت انباره‌ها صفر باشد:

با پایان یافتن عملیات کار jدر ایستگاه d اگر ماشین بیکاری در l+1 وجود داشته باشد، قطعه jمستقیما به آن ماشین انتقال می‌یابد و زمان تکمیل آن نیز به لیست پیشامدها اضافه می‌گردد و وضعیت ماشین فعلی به حالت بیکار، تغییر می‌یابد. بعد از آن، فعالیتهای (۲) و(۳) بصورتیکه در قسمت قبل تشریح شد، اعمال می‌گردند.

– حالت سوم در حالتی که ظرفیت انباره‌ها صفر باشد:

درصورتیکه هیچ ماشین بیکاری در مرحله l+1 موجود نباشد، قطعه j و ماشین مربوطه بحالت مسدود درآمده و هیچگونه پیشامدی به لیست پیشامدها اضافه نمی‌شود و بسراغ پیشامد بعدی در لیست فوق می‌رویم. فعالیت (۱) در این حالت هرگز اجرا نمی‌شود. فعالیتهای (۲) و (۳) نیز نیاز به برخی اصلاحات دارند، از جمله اینکه هنگام حرکت بازگشتی از ایستگاه l به ایستگاه اول، چنانچه در ایستگاه k یک ماشین بیکار یافت شود و ظرفیت انباره ماقبل آن صفر باشد، باید بررسی کنیم که آیا هیچ کاری در مرحله k-1 مسدود شده است یا نه. اگر نتیجه بررسی مثبت بود، کار مربوطه را به ایستگاه k منتقل کرده و زمان تکمیل آن را به لیست پیشامدها اضافه می‌کنیم و وضعیت ماشین مسدود شده به حالت بیکار تغییر می‌یابد. درصورتیکه بیش از یک کار در ایستگاه k-1 ام به حالت مسدود موجود باشد، کاری برای انتقال به ایستگاه k انتخاب می‌شود که مدت زمان بیشتری را در حالت مسدود سپری کرده باشد.

۴- نحوه ایجاد مثالهای نمونه جهت بررسی کارایی الگوریتمهای پیشنهادی برای حل مساله فلوشاپ ترکیبی با محدودیت انباره‌های میانی

پیچیدگی بسیار بالای محاسبه توالی بهینه مرتبط با هر بردار جواب و همچنین NP-Hard بودن مساله FSPM/b، عملاً دستیابی به جوابهای بهینه برای مثالهای نمونه، بمنظور بررسی کارایی الگوریتمهای ابتکاری بکار رفته در حل این مساله را در زمان محدود ناممکن می‌سازد. لذا برای حل این مشکل از دو نوع داده‌های ورودی از پیش تعیین شده استفاده می‌شود. در مجموعه داده‌های نوع اول، زمانهای انجام عملیات کارها در ایستگاههای مختلف بصورتی تصادفی و بر اساس قوانینی معین، چنان تعیین می‌شوند که زمان بهینه انجام کل کارها(C_{max}) در سیستم برای انجام مقایسات، براحتی قابل محاسبه می‌باشد.

در مجموعه داده‌های نوع دوم دیگر قادر به محاسبه C_{max} بهینه نیستیم و در این حالت معیار مقایسه را بر اساس میزان اختلاف جوابهای حاصل از الگوریتمهای جستجو با یک مقدار حد پایین برای تابع C_{max} قرار می‌دهیم.

۴-۱- مجموعه داده‌های نوع اول

برای ایجاد چنین مسایل نمونه‌ای از الگوریتم پیشنهادی واردونو و فتحی [۱۰] استفاده شده است. در این حالت خاص تعداد ماشینها در ایستگاههای مختلف، یکسان فرض شده و برابر m درنظر گرفته می‌شود. فرایند ایجاد مساله‌ای با C_{max} بهینه مشخص، در مرحله اول با تخصیص مقادیر با رویکردی معین به ماتریس‌های I و X آغاز شده و سپس زمانهای عملیات چنان تعیین می‌شوند که تضمین کننده بهینه بودن توالی مرتبط با جواب (X, I) باشد. بطور کلی در مورد این نوع خاص از مسایل نمونه، قوانین زیر حکمفرما می‌باشد:

۱. در جواب بهینه مساله، مجموعه یکسانی از کارها به هریک از ماشینها در ایستگاه‌های مختلف تخصیص می‌یابد. بعبارت دیگر، تمامی ستونهای ماتریس I یکسان می‌باشد.

۲. در جواب بهینه مساله، مجموعه کارهایی که یک خانواده را تشکیل می‌دهند و بر روی یک ماشین یکسان انجام می‌گیرند، با یک ترتیب ثابت بر روی ماشینها در ایستگاه‌های مختلف اجرا می‌شوند. بعبارت دیگر، تمامی ستونهای ماتریس X نیز یکسان می‌باشد.

۳. در هر ایستگاه زمانهای عملیات مربوط به کارهایی که اولویت اول را بر روی هر ماشین دارند، با هم برابر بوده و کمتر از زمانهای سایر کارها بر روی هر ماشین می‌باشند. به همین دلیل زمان بیکاری ماشین‌ها برای مرحله l ($l > 1$) قبل از شروع اولین کار در حداقل مقدار خود قرار دارد.

۴. هر کاری که در مرحله l تکمیل می‌شود، بلافاصله قابل انتقال به مرحله l+1 بوده و زمان تکمیل آن را می‌توان بلافاصله به لیست پیشامدها اضافه کرد. بدین ترتیب با شروع فعالیت یک ماشین تا زمان تکمیل تمام کارهای محوله، هیچگونه بیکاری برای ماشین وجود ندارد. نکته دیگر اینکه در این برنامه زمانبندی در حالت بهینه، از هیچ انباره‌ای استفاده نمی‌شود. بدین ترتیب برنامه زمانبندی ارایه شده قابل استفاده در هر دو نوع مساله FSPM و FSPM/b می‌باشد

قضیه: برنامه زمانبندی مرتبط با جواب (I, X) حاصل از الگوریتم فوق، یک برنامه زمانبندی بهینه برای مساله مورد نظر می‌باشد (اثبات این قضیه در مقاله واردونو و فتحی [۱۰] ارایه شده است).

۴-۲- مجموعه داده‌های نوع دوم

الگوریتم بکار گرفته شده برای محاسبه حد پایین (LB) برای مساله فلوشاپ ترکیبی برگرفته از پایان نامه وندولد [۹] می‌باشد. در ابتدا جهت تسهیل در محاسبه LB تعاریف زیر ارایه می‌گردند:

$$P_{j,\,a\to b} = P_{j,\,a} + P_{j,\,a+1} + \ldots + P_{j,\,b}$$

$P'_{x,\,a\to b}$: **xامین کوچکترین مقدار در بین کلیه $P_{j,a\to b}$ ها**

$P'_{x,a}$: **کمترین زمان انجام کار در ایستگاه a**

m_k : **تعداد ماشین‌های موجود در ایستگاه k**

اصول محاسبه LB بر مبنای محاسبه متوسط حداقل زمانهای بیکاری ماشین‌ها قبل از شروع کارها در هر ایستگاه و همچنین متوسط حداقل بیکاری ماشین بعد از اتمام آخرین کار تا زمان تکمیل کل کارها، ارایه شده است. برای محاسبه LB نیازمند محاسبه سه مؤلفه مجزا α_l، T_l و β_l برای هر یک از ایستگاهها هستیم.

نحوه محاسبه α_l :

این مؤلفه نشانگر متوسط زمان بیکاری ماشین‌های ایستگاه l ($l>1$) قبل از شروع فعالیت می‌باشد. برای محاسبه α_l قبل از هر چیز باید به محاسبه γ_l پرداخت که نشانگر حداقل مجموع تاخیر ماشین‌ها در ایستگاه l می‌باشد. این تاخیر بدان دلیل است که کارها برای ورود به ایستگاه l حتما بایستی تمامی ایستگاههای قبلی را پشت سر گذاشته باشند.

γ_l بر اساس الگوریتم زیر محاسبه می‌شوند:

گام ۱: $\gamma_l = 0$ و $K=l$

گام ۲: S را شماره بزرگترین مرحله قبل از مرحله K ($S<K$) قرار می‌دهیم، بطوریکه تعداد ماشین‌های موجود در آن از تعداد ماشین‌های موجود در ایستگاه K کمتر باشد ($m_S < m_K$). اگر چنین ایستگاهی موجود نباشد، الگوریتم خاتمه می‌یابد، در غیراینصورت مقدار $\gamma_l + \sum_{x=1}^{d_{KS}} P'_{x,s}$ را در γ_l، قرار می‌دهیم.

در این رابطه d_{KS} تفاوت تعداد ماشین‌های موجود در ایستگاه K و S می‌باشد. $d_{KS} = m_K - m_S$

گام ۳: K را برابر با S قرار داده و به گام ۲ می‌رویم.

α_l بعنوان متوسط بیکاری ماشین‌های موجود در مرحله l قبل از شروع عملیات یک کار می‌باشد و بصورت زیر محاسبه می‌شود:

$$\alpha_l = \begin{cases} \frac{1}{m_l}\left(\sum_{x=1}^{m_l} P'_{x,1\to(l-1)} + \gamma_l\right) & l>1 \\ 0 & l=1 \end{cases}$$

نحوه محاسبه β_l :

بصورت مشابه، هر ماشینی در ایستگاه l ($l<L$) بعد از اتمام کلیه کارهای محوله تا زمان خاتمه کارها در ایستگاه L، زمان بیکاری دارد. بدین ترتیب مشابه γ_l در این حالت نیز، φ_l را بعنوان حداقل مجموع بیکاری ماشین‌ها در مرحله l بعد از اتمام کارها، تعریف می‌کنیم. φ_l بر اساس الگوریتم زیر محاسبه می‌شود:

گام ۱: $\varphi_l = 0$ و $K=l$

گام ۲: S را برابر شماره کوچکترین ایستگاه بعد از ایستگاه K ($S>K$) قرار می‌دهیم، بطوریکه تعداد ماشین‌های موجود در S از تعداد ماشین‌های موجود در ایستگاه K کمتر باشد. ($m_S < m_K$)

اگر چنین ایستگاهی موجود نباشد، الگوریتم خاتمه می‌یابد، در غیراینصورت مقدار $\varphi_l + \sum_{x=1}^{d_{KS}} P'_{x,s}$ را در φ_l، قرار می‌دهیم.

تعریف d_{KS} در این قسمت نیز مشابه آن چیزی است که برای محاسبه γ_l ارایه شد.

گام ۳: K را برابر با S قرار داده و به گام ۲ می‌رویم.

β_l بعنوان متوسط بیکاری ماشین‌های موجود در مرحله l بعد از اتمام کارهای محوله بصورت زیر محاسبه می‌شود:

$$\beta_l = \begin{cases} \frac{1}{m_l}\left(\sum_{x=1}^{m_l} P'_{x,1\to(l-1)} + \varphi_l\right) & l>1 \\ 0 & l=L \end{cases}$$

نحوه محاسبه T_l:

مؤلفه سوم که باید محاسبه شود، T_l می‌باشد که بعنوان متوسط بار کاری محوله به هر ماشین در ایستگاه l می‌باشد.

$$T_l = \frac{1}{m_l}\sum_{j=1}^{N} P_{jl}$$

بدین ترتیب حد پایین زمان کل انجام کارها برای سیستم تولیدی فلوشاپ ترکیبی(LB) بصورت زیر بدست می‌آید:

$$LB = \max \{ \alpha_l + \beta_l + T_l \} \qquad l = 1, 2, \ldots, L$$

۵- الگوریتم های فرا ابتکاری بکاررفته در حل مساله

در این قسمت بدون اینکه به معرفی جزئیات هر یک از الگوریتم‌های ژنتیک (GA)[1] و الگوریتم ژنتیک ترکیبی (HGA)[2] مورد استفاده در حل مساله پرداخته شود، مکانیزم بکار گرفته شده در حل مساله مورد مطالعه تشریح می‌گردد.

۵-۱- الگوریتم ژنتیک (GA)

به منظور بکارگیری الگوریتم ژنتیک برای حل هر مساله‌ای ، قبل از هر چیز باید نوع کدینگ مساله جهت ارایه جوابها انتخاب شود. بنا بر آنچه که دربخش (۲) در مورد انوع ارایه جوابها برای مساله مورد بحث، ذکر گردید به نظر می رسد بهترین نوع کدینگ، کدینگ جهشی باشد. در این نوع کدینگ هر کروموزوم از یک رشته اعداد که همان نمایش برداری انجام کارها می‌باشد، تشکیل شده است. با مشخص شدن نوع کروموزوم‌ها باید نوع عملگرهای تقاطعی و جهشی نیز، به عنوان اصلی‌ترین عملگرهای مورد استفاده در این الگوریتم مشخص شود. عملگر تقاطعی مورد استفاده عملگر تقاطعی ترتیبی[3] (OX) می‌باشد که به طریق زیر عمل می‌کند:

- قسمتی از یکی از کرومزومهای والد را بصورت تصادفی انتخاب می کنیم
- این زیر رشته را دقیقاً و در همان موقعیت موجود به فرزندان منتقل می کنیم
- تمامی نمادهای موجود در کروموزوم فرزند را که در والد دیگر (دوم) نیز وجود دارد، از کروموزوم والد دوم حذف می‌کنیم
- نمادهای باقیمانده از کروموزوم والد دوم را در کروموزوم فرزند از چپ به راست در موقعیتهای خالی قرار دهید.

برای اعمال عملگر جهشی نیز به طریق زیر عمل می‌شود:

- بصورت تصادفی دو موقعیت را در کروموزوم فعلی انتخاب کرده و محتویات این دو موقعیت را با هم عوض می‌کنیم.

حال با مشخص شدن نوع عملگرهای اصلی، نحوه اجرای الگوریتم ژنتیک بصورت گام به گام تشریح می‌شود:

گام ۱- جمعیت اولیه کروموزوم‌ها را به تعداد معین (pop-size) و بصورت تصادفی تولید کنید.

گام ۲- مقدار تابع هدف (makespan) را برای هر کروموزوم محاسبه کرده و بهترین جواب(S^*) را ذخیره کنید.

گام ۳- بر اساس "مکانیزم انتخاب"، تعداد pop-size کروموزوم را از بین کروموزوم های موجود انتخاب کنید.

گام ۴- بصورت تصادفی مجموعه‌ای از کروموزوم ها را برای اعمال عملگر تقاطعی انتخاب کنید.

گام ۵- بصورت تصادفی مجموعه‌ای از کروموزوم ها را برای اعمال عملگر جهشی انتخاب کنید.

گام ۶- مقدار تابع هدف را برای تمامی فرزندان جدید محاسبه کرده و بهترین مقدار را با S^* مقایسه کنید. در صورتیکه این جواب از S^* بهتر باشد آن را در S^* قرار دهید.

گام ۷- شرط توقف را بررسی کنید. اگر شرط برقرار نیست به گام ۳ برگردید.

مکانیزم انتخاب:

روش مورد استفاده در این قسمت روش رتبه بندی می‌باشد که از مقاله میکالویکس [۱۱] استخراج شده است. این روش کروموزوم‌ها را بر اساس رتبه‌ای که در جمعیت فعلی دارند انتخاب می‌کند بطوریکه کروموزوم‌های با تابع هدف (C_{max}) کمتر، شانس بیشتری برای انتخاب داشته باشند. برای این منظور بعد از محاسبه مقدار تابع هدف مطابق آنچه در فصل قبل ذکر شد، کروموزوم‌ها را به ترتیب از بهترین تا بدترین مرتب می‌کنیم و به هر کروموزوم، احتمال انتخابی برابر با Pr مطابق روش زیر تخصیص می‌دهیم:

$$\Pr(\text{rank}) = q - (\text{rank}-1) * r \qquad \text{rank}=1,\ldots,\text{pop-size}$$

1) Genetic Algorithm
2) Hybrid Genetic Algorithm
3) Order Crossover

q و r اعداد ثابتی هستند که توسط شخص تصمیم گیرنده تعیین می‌شوند. بدین ترتیب بالاترین احتمال انتخاب به رتبه یک و کمترین احتمال به رتبه آخر اختصاص می‌یابد. از آنجاییکه مجموع این احتمالات باید یک شود می توان به رابطه زیر رسید:

$$q = \frac{r(pop-size-1)}{2} + \frac{1}{pop-size}$$

با قرار دادن r=0 عملاً هیچ گونه تفاوتی بین کروموزوم‌ها قائل نمی شویم. میکالویکس [۱۱] پیشنهاد کرده که بهترین مقدار برای q عددی بین $\frac{1}{pop-size}$ و $\frac{2}{pop-size}$ می‌باشد. سپس احتمال تجمعی هر رتبه را بصورت زیر محاسبه می کنیم:

$$Q(rank) = \sum_{j=1}^{pop-size} \Pr(j)$$

حال فرایند انتخاب pop'(i) بصورت زیر صورت می گیرد:

– به تعداد جمعیت موجود عدد تصادفی R'(i) که توزیع یکنواخت بین صفر و یک دارند، تولید می‌کنیم. برای i=1 تا i=pop-size گام های زیر را تکرار می‌کنیم:

a) $x = \arg\min \{ Q(j): Q(j) > R'(i) \}$
b) $pop'(i) = pop(x)$

با این روش کروموزوم های با Pr بالا ممکن است چندین بار انتخاب شوند که خود باعث تاثیر هر چه بیشتر این کروموزوم‌ها در ایجاد نسل بعدی خواهد شد. حال به ازای هر یک از کروموزوم‌های انتخاب شده یک عدد تصادفی بین صفرو یک تولید می کنیم و چنانچه این عدد کمتر از P_c (احتمال اعمال عملگر تقاطعی) بود، کروموزوم مربوطه را برای تولید فرزندان جدید با عملگرتقاطعی که در قسمت قبل تشریح شد انتخاب می کنیم. حال بصورت متوالی و دو به دو کروموزوم ها را انتخاب کرده و عملیات تقاطعی و تولید فرزندان جدید صورت می‌گیرد. لازم به توضیح است در صورت فرد بودن تعداد کروموزوم‌هایی که برای این مرحله انتخاب شده‌اند، کروموزوم آخر را حذف می‌کنیم. مشابه این قسمت برای اعمال عملگر جهشی نیز اقدام می‌کنیم. این بار بعد از تخصیص عددی تصادفی به هر کروموزوم، این عدد را با P_m مقایسه می کنیم. در صورتیکه عدد فوق از Pm کوچکتر باشد، عملیات جهش بر روی کروموزوم صورت می‌گیرد.

شرط توقف: در دو صورت الگوریتم پایان می‌یابد:

۱– تعداد تولید نسلها به عددی از پیش تعیین شده (Iter-max)برسد.

۲– به تعداد معینی تکرار (Count)، هیچ گونه بهبودی در بهترین مقدار ذخیره شده داده نشود.

مقادیر پارامترهای بکاررفته در حل مساله FSPM/b به کمک الگوریتم ژنتیک بشرح زیر می‌باشد:

Pop-size=100 Iter- max = 1000 $P_m = 0.5$ $P_c = 0.7$ Count = 100 $q = \frac{1.7}{pop-size}$

۵-۲- الگوریتم ژنتیک ترکیبی (GA-TS)

در این روش با ترکیب الگوریتم ژنتیک با روش جستجوی ممنوع سعی در ارتقاء کارایی هر یک از الگوریتم‌های فوق داریم. روش جستجوی ممنوع (TS) یکی از کاراترین الگوریتم های جستجوی محلی به حساب می آیدکه فقط از یک جواب در هر مرحله برای جستجو استفاده می‌کند، لذا احتمال اینکه خیلی از نواحی مناسب را در جستجو از دست بدهد نسبتاً زیاد است. از طرف دیگر استفاده نامناسب و بی‌دلیل از جمعیتی بزرگ و بصورت موازی و بدون توجه به معیارهای بهینه سازی در الگوریتم ژنتیک، ممکن است عملاً تاثیر مثبتی بر روی مساله نداشته و تحول قابل ملاحظه‌ای در هر تکرار رخ ندهد. بدین ترتیب با ترکیب روشهای فوق عملاً می توان کاستی‌های هر یک را کاهش داده و نواقص را به حداقل رساند.

قبل از بیان نحوه پیاده‌سازی الگوریتم به بیان برخی مشخصه‌های مربوط به الگوریتم جستجوی ممنوع می‌پردازیم.

- فهرست ممنوع:

ثبت ویژگیهای حرکت در روش جستجوی ممنوع محدودیت‌هایی را بنام محدودیت‌های ممنوع که مانع انتخاب حرکتهای تکراری تا تعداد معینی تکرار جهت جلوگیری از بازگشت به جوابهای قبلی و افتادن در دام بهینه‌های محلی، بوجود می‌آورد. از مجموعهٔ این محدودیت‌های ممنوع، فهرست ممنوع تشکیل می‌شود. برای این منظور از حافظه کوتاه مدت برای ثبت خواص حرکتهایی که در انتقال از یک جواب به جواب همسایگی آن صورت گرفته، استفاده می‌شود. قبل از اینکه حرکتی صورت گیرد باید مطمئن شد که چنین حرکتی د رحال حاضر در فهرست ممنوع موجود نیست.

- طول فهرست ممنوع:

بیان کنندهٔ حداکثر تعداد تکرارهای ممنوع در فرایند جستجوی کلی می‌باشد که می‌تواند عددی ثابت و یا تابعی از پارامترهای مساله باشد. مطالعات تجربی نشان می‌دهد که اندازه خیلی کوچک برای طول این فهرست منجر به قرار گرفتن زودهنگام در حلقه تکرار و اندازه بزرگ برای فهرست ممنوع ما را از بررسی بسیاری از جوابها باز می‌دارد. تا قبل از پر شدن این فهرست هر حرکت ممنوع به لیست اضافه می‌شود و پس از پر شدن لیست، با هر حرکت ممنوع جدید قدیمی‌ترین عضو لیست حذف و عضو جدید به لیست اضافه می‌شود.

- سطح رضایتمندی[1]

عدم انتخاب برخی حرکتهای موجود در لیست ممنوع ممکن است مانع پذیرش بسیاری از جوابهای خوب باشد. به همین دلیل معیاری با عنوان سطح آرمانی تعریف می‌شود که اجازه انجام بعضی از حرکتهای ممنوع را می‌دهد. مثلاً اگر تمامی حرکتهای موجود ممنوع باشند و هیچ حرکتی امکان پذیر نباشد، قدیمی‌ترین ممنوعیت ایجاد شده انتخاب و رفع ممنوعیت می‌شود. ویا چنانچه حرکت انتخابی منجر به مقداری برای تابع هدف شود که از بهترین جواب فعلی نیز بهتر باشد، آن حرکت را انتخاب می‌کنیم.

با این تS^* وضیحات، گامهای اصلی الگوریتم به شرح زیر می باشد:

گام ۱- ایجاد جمعیت اولیه تصادفی به اندازه Pop-size و محاسبه مقدار تابع هدف برای هر یک ازکروموزومها و قرار دادن بهترین مقدار در S^* (به عنوان بهینه کلی) و Iter-max=0.

گام ۲- انجام گامهای زیر به تعداد Iter-in بار:

- تخصیص S_i^* به هر کروموزوم به عنوان معیاری برای پذیرش معیار رضایت‌مندی (در ابتدا S^* را در تمامی S_i^* ها قرار می‌دهیم)(i= 1,.., Pop-size)
- تعداد N همسایگی برای هر کروموزوم i با عملگر تعویض جفتی۲ و بصورت تصادفی ایجاد کنید.
- بهترین جواب را در این همسایگی انتخاب و حرکتهای منتخب را به لیست ممنوع کروموزوم مربوطه اضافه کنید.

در صورتیکه این مقدار از S_i^* بهتر باشد آنرا جایگزین S_i^* کنید.

تکرار بعدی را با این جواب منتخب ادامه دهید.

گام ۳- کروموزوم‌های نهایی حاصل از گام ۲ را برای اعمال عملگر تقاطعی انتخاب و تابع هدف را برای فرزندان جدید محاسبه کنید.

گام ۴- تعداد Pop-size کروموزوم با بهترین مقدار تابع هدف را انتخاب و S^* جدید را محاسبه کرده و Iter-max=Iter-max+1. در صورتیکه شرایط خاتمه برآورده نشده به گام ۲ بروید.

در این روش بر خلاف روش جستجوی ممنوع بیش از یک جواب در هر مرحله به تکرار بعدی منتقل می‌شود. هر کروموزوم (به غیر از فرزندان) فهرست ممنوع وابسته به خود را نیز به نسل بعد منتقل می‌کند. فرزندان جدید در صورت انتخاب، با فهرست ممنوع خالی وارد چرخه می‌شوند. عملگر تقاطعی دقیقاً مشابه آن چیزی است که در قسمت قبل توضیح داده شد. در گام ۲ به ازای هر تکرار، دو مولفه به لیست ممنوع اضافه می‌شود که شامل شمارهٔ کارها و موقعیت‌هایی است که این کارها قبل از انتقال در آنها قرار داشته‌اند. بدین ترتیب این فهرست اجازه بازگشت این کارها به موقعیت‌های فوق را تا تعداد تکرار معین نمی‌دهد. شرایط توقف الگوریتم دقیقاً مشابه شرایطی است که در مورد الگوریتم ژنتیک ذکر گردید. مقادیر پارامترهای بکاررفته در حل مساله FSPM/b به کمک الگوریتم ژنتییک ترکیبی به شرح زیر می‌باشد:

Pop-size=100 Iter-max = 500 Iter-in= 50 $P_c = 0.8$ Count = 50

1) Aspiration Criteria
2) Pairwise Exchange

۶- ارایه نتایج محاسباتی

معیار مقایسه کارایی الگوریتم‌های مختلف برای هر دو نوع داده‌های ورودی بصورت زیر محاسبه می‌شود:

$$\delta = \frac{s - s^*}{s^*}$$

S^*: مقدار بهینه یا حد پایین جواب که برای هر نوع مساله ورودی قابل محاسبه می‌باشد.
S: جواب حاصل از اجرای الگوریتم

مقادیر کم برای معیار فوق حاکی از قدرت بالای الگوریتم در جستجوی فضای جواب می‌باشد. در ادامه، نتایج ارایه شده برای دو نوع از داده‌های تعریف شده در بخش ۴ بصورت مجزا ارایه می‌شود.

۶-۱- مجموعه داده‌های نوع اول

در این بخش مسایلی با مشخصات کلی زیر بررسی شده‌اند:

N= 20,30,50 (تعداد کارها) L=2,3,4 (تعداد ایستگاهها) m_i= 2,4,6 (تعداد ماشین‌های موجود در هر ایستگاه)
b_i= 0,1,3

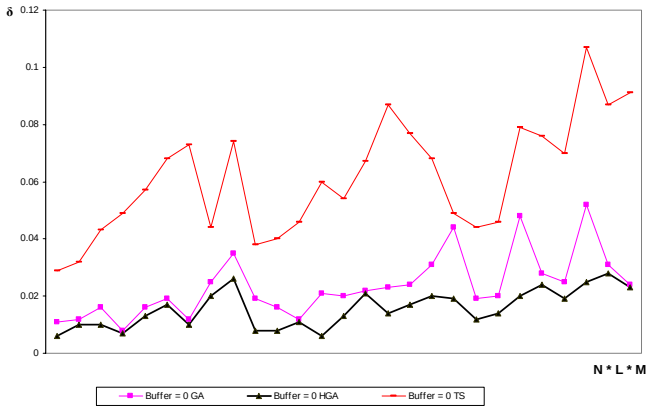
برای هر بار تولید اطلاعات ورودی با مقدار بهینه مشخص برای تابع هدف(C_{max})، به کمک هر دو الگوریتم پیشنهادی یعنی الگوریتم ژنتیک و الگوریتم ژنتیک ترکیبی، مساله را حل کرده و مقدار خروجی برای تابع هدف ذخیره می‌شود. برای هر سیستم مدنظر با اطلاعات کلی بالا، پنج بار این کار انجام شده و میانگین مقادیر خروجی وارد جداول شده است. در جدول ۱ نتایج حاصل از الگوریتم‌های فوق همراه با نتایج ارایه شده در مقاله واردونو و فتحی [۱۰] که حاصل روش جستجوی ممنوع می‌باشد، ارایه شده است.

جدول ۱- مقایسه کارایی الگوریتم‌ها برای داده‌های نوع اول

N*L*M	Genetic Algorithm			Hybrid Genetic Algorithm			Tabu Search	
	b_i=0	b_i=1	b_i=3	b_i=0	b_i=1	b_i=3	b_i=0	b_i=1
	ratio			ratio			ratio	
20*2*2	0.011	0.0031	0.003	0.006	0.0008	0.002	0.029	0.0153
20*2*4	0.012	0.0112	0.009	0.01	0.009	0.007	0.032	0.0348
20*2*6	0.016	0.013	0.016	0.01	0.0033	0.005	0.043	0.0403
20*3*2	0.008	0.0166	0.015	0.007	0.005	0.005	0.049	0.0252
20*3*4	0.016	0.0136	0.015	0.013	0.0097	0.003	0.057	0.0469
20*3*6	0.019	0.0208	0.027	0.017	0.0104	0.016	0.068	0.0658
20*4*2	0.012	0.0109	0.004	0.01	0.0058	0.001	0.073	0.0285
20*4*4	0.025	0.025	0.025	0.02	0.0143	0.011	0.044	0.0455
20*4*6	0.035	0.0306	0.035	0.026	0.0282	0.022	0.074	0.0773
30*2*2	0.019	0.0097	0.005	0.008	0.0071	0.004	0.038	0.0184
30*2*4	0.016	0.0109	0.009	0.008	0.0098	0.004	0.04	0.0323
30*2*6	0.012	0.0138	0.012	0.011	0.0099	0.01	0.046	0.0412
30*3*2	0.021	0.0209	0.006	0.006	0.0053	0.004	0.06	0.0316
30*3*4	0.02	0.0175	0.020	0.013	0.0058	0.007	0.054	0.0445
30*3*6	0.022	0.0118	0.024	0.021	0.0101	0.01	0.067	0.0548
30*4*2	0.023	0.0094	0.009	0.014	0.0041	0.006	0.087	0.0379
30*4*4	0.024	0.0139	0.015	0.017	0.0089	0.01	0.077	0.0467
30*4*6	0.031	0.0195	0.018	0.02	0.0135	0.008	0.068	0.069
50*2*2	0.044	0.007	0.01	0.019	0.0048	0.007	0.049	0.0175
50*2*4	0.019	0.0127	0.012	0.012	0.0067	0.008	0.044	0.0299
50*2*6	0.02	0.0159	0.013	0.014	0.0106	0.005	0.046	0.0337

N*L*M	Genetic Algorithm			Hybrid Genetic Algorithm			Tabu Search	
	b_i=0	b_i=1	b_i=3	b_i=0	b_i=1	b_i=3	b_i=0	b_i=1
	ratio			ratio			ratio	
50*3*2	0.048	0.0113	0.008	0.02	0.0052	0.005	0.079	0.0303
50*3*4	0.028	0.0187	0.017	0.024	0.01	0.014	0.076	0.0523
50*3*6	0.025	0.028	0.022	0.019	0.011	0.015	0.07	0.0574
50*4*2	0.052	0.0206	0.012	0.025	0.011	0.006	0.107	0.0381
50*4*4	0.031	0.0246	0.024	0.028	0.0123	0.019	0.087	0.0666
50*4*6	0.024	0.0274	0.024	0.023	0.0196	0.007	0.091	0.0746
Average	0.024	0.0162	0.015	0.016	0.0093	0.008	0.061	0.0428

در مورد هر سه الگوریتم فوق عملاً تغییراتی در کارایی هر الگوریتم با تغییر ظرفیت بافرها مشاهده می‌شود. بر اساس این جدول، در تمامی سه الگوریتم فوق افزایش ظرفیت، باعث کاهش در مقدار اختلاف مشاهده شده می‌شود. در تحلیل این مطلب باید اشاره شود که ظرفیت صفر برای بافر عملاً تاثیر بسیار زیادی بر روی موجه بودن بردارهای جواب دارد. علت اصلی آن مسدود شدن ماشین‌ها در بسیاری از حالات به ازای جوابهای اولیه متفاوت می‌باشد. با اعمال این محدودیت الگوریتم در فضای محدودتری از لحاظ کیفیت جوابها قرار گرفته و نتایج حاصله افت نسبی کارایی را نشان می‌دهد. جهت نمایش هر چه بهتر اختلافات موجود در میزان کارایی، در شکل ۲ نمایش نموداری مقادیر فوق برای انباره با ظرفیت صفر نشان داده شده است.



شکل ۲– نمودار مقایسه معیارهای کارایی الگوریتم‌ها برای داده‌های نوع اول

۶-۲- مجموعه داده‌های نوع دوم

در این بخش مسایلی با مشخصات کلی زیر بررسی شده‌اند:

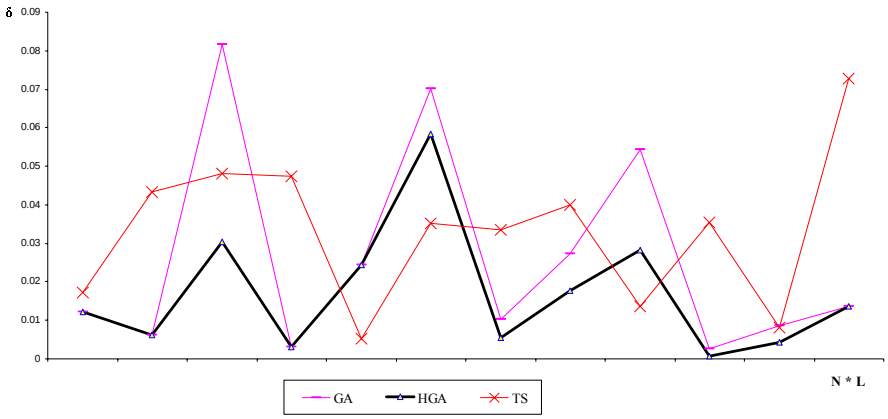
N= 20,30,50 **(تعداد کارها)** L=2,4 **(تعداد ایستگاهها)** b_i= 0,1,N

در این قسمت برای تمامی مسایل نمونه، تعداد ماشین‌ها در ایستگاههای مختلف بصورت تصادفی تولید می‌شود. تعداد ماشین‌ها در هر ایستگاه عددی تصادفی بین ۱ تا ۵ می باشد. انباره با ظرفیت N عملاً نشان دهنده حذف محدودیت فوق می‌باشد. این انتخاب به منظور بررسی هر چه بیشتر تاثیر محدودیت انباره‌ها بر میزان کارایی الگوریتم‌ها صورت گرفته است. برای هر بار تولید اطلاعات ورودی با مقدار حد پایین مشخص برای تابع هدف(C_{max})، به کمک هرالگوریتم پیشنهادی یعنی الگوریتم ژنتیک و الگوریتم ژنتیک ترکیبی، مساله را حل کرده و مقدار خروجی برای تابع هدف ذخیره می‌شود. برای هر سیستم با اطلاعات کلی بالا، پنج بار این کار را انجام داده و میانگین معیارهای کارایی با یکدیگر مقایسه می‌شود.

جهت مقایسه هر چه بهتر، در شکل ۳ نمودار مقایسه کارایی‌های الگوریتم‌های مختلف نمایش داده شده است.

جدول۲- مقایسه کارایی الگوریتم‌ها برای داده‌های نوع دوم

N*L	Genetic Algorithm			Hybrid Genetic Algorithm			Tabu Search		
	b_i=0	b_i=1	b_i=N	b_i=0	b_i=1	b_i=N	b_i=0	b_i=1	b_i=N
	ratio			ratio			ratio		
20*2	0.0121	0.0061	0.0081	0.0121	0.0081	0.0020	0.0173	0.0066	0.0043
20*4	0.0062	0.0079	0.0070	0.0062	0.0026	0.0035	0.0433	0.0217	0.0170
20*6	0.0817	0.0112	0.0208	0.0304	0.0064	0.0160	0.0480	0.0360	0.0360
20*8	0.0030	0.0076	0.0191	0.0030	0.0030	0.0191	0.0475	0.0303	0.0237
30*2	0.0245	0.0368	0.0123	0.0245	0.0184	0.0061	0.0053	0.0020	0.0012
30*4	0.0702	0.0024	0.0166	0.0583	0.0012	0.0131	0.0353	0.0111	0.0034
30*6	0.0103	0.0055	0.0182	0.0055	0.0030	0.0085	0.0335	0.0127	0.0163
30*8	0.0272	0.0450	0.0072	0.0178	0.0194	0.0044	0.0399	0.0134	0.0121
50*2	0.0543	0.0067	0.0067	0.0283	0.0030	0.0052	0.0136	0.0042	0.0031
50*4	0.0027	0.0225	0.0038	0.0008	0.0088	0.0019	0.0355	0.0147	0.0097
50*6	0.0086	0.0071	0.0059	0.0043	0.0012	0.0039	0.0082	0.0018	0.0067
50*8	0.0137	0.0018	0.0120	0.0137	0.0004	0.0078	0.0728	0.0286	0.0215
Average	0.0262	0.0134	0.0115	0.0162	0.0063	0.0076	0.0334	0.0153	0.0129



شکل۳- نمودار مقایسه معیارهای کارایی برای داده‌های نوع دوم

بطور کلی از مجموع نتایج فوق می‌توان به جمع‌بندی زیر در خصوص انتخاب کاراترین الگوریتم برای حل مساله FSPM/b رسید:

- در هر دو نوع مجموعه داده‌ها، الگوریتم ژنتیک ترکیبی کارایی بالاتری را نسبت به سایر الگوریتم‌ها از خود نشان می‌دهد. البته این مساله با صرف زمان بیشتر برای محاسبات همراه می‌باشد. اما با توجه به اینکه میزان افزایش زمان محاسبات با افزایش تعداد کارها و ابعاد مساله کاملاً روند خطی دارد، لذا الگوریتم پیشنهادی فوق به عنوان بهترین گزینه برای استفاده در حل مساله فلوشاپ ترکیبی با محدودیت بافر، پیشنهاد می‌شود.
- ظرفیت بافرهای میانی بر روی کارایی الگوریتم‌ها در هر دو نوع مسایل نوع اول و دوم اثر قابل ملاحظه‌ای دارد .
- در خانواده مسایل نوع اول، عموماً با افزایش تعداد کارها شاهد کاهش میزان کارایی الگوریتم‌ها هستیم ولی در مورد مسایل نوع دوم روند قابل شناسایی، مشاهده نمی‌شود. تحلیل‌های آماری انجام شده با کمک نرم‌افزار SPSS نیز این مطالب را تایید می‌کند.

مقایسه میانگین زمان لازم برای توقف در هر الگوریتم بصورت زیر می‌باشد:

$$T\,(\mathrm{TS}) < T\,(\mathrm{GA}) < T\,(\mathrm{HGA})$$

۷- پیشنهاد برای تحقیقات آتی

در این مقاله نوع خاصی از جوابها تحت عنوان جوابهای برداری، برای نمایش فضای جستجوی مساله بکارگرفته شده است. به دنبال آن نحوه تشکیل برنامه زمانبندی کامل عملیات بر اساس قوانین FAM و برخی قوانین ابتکاری در تعیین تقدم و تاخر های ایجاد شده، معرفی شده است. به عنوان تحقیقات آتی می‌توان بر روی افزایش کارایی نوع دیگری از جوابها تحت عنوان جوابهای ماتریسی متمرکز شده و با معرفی ساختارهای شکل دهی زمانبندی کارا بر اساس بکارگیری قوانین مختلف بکاررفته در متون علمی سعی در افزایش کارایی این روش داشت. در این مساله برای هر ایستگاه انباره‌های میانی بصورت مشترک مورد استفاده قرار می‌گیرد که معمولاً در شرایط واقعی خیلی کاربرد ندارد. بررسی تاثیر مجزا کردن صف مربوط به انباره هر ماشین در ایستگاههای مختلف از جمله مواردی است که می‌توان در تحقیقات آتی وارد مساله کرد.همچنین به منظور هر چه بیشتر نزدیک کردن فضای تولیدی به دنیای واقعی می‌توان مساله دسترس پذیری ماشین‌ها را نیز با اعمال شرایط تصادفی در مدل وارد کرد. ارایه روشهای ابتکاری برای محاسبه حدود پایین زمان کل انجام کارها برای مساله فوق، بطوریکه تا حد ممکن اختلاف موجود با جواب بهینه را کمینه کند، از دیگر موارد قابل بررسی در این مساله می‌باشد. همچنین می‌توان با انتخاب پارامترهایی نظیر زمانهای عملیات بصورت اعداد فازی تا حد زیادی محدودیت‌های دنیای واقعی در مدل‌بندی مساله وارد کرد.

۸- مراجع و مواخذ

[1] Garey, M.R., "The Complexity of flow shop and job shop scheduling", Mathematics of Operations Research,Vol.1,No.2,pp. 117-129, 1976.
[2] Shaukat, A.B., and Hunsucker, J.L., "Branch and bound algorithm for the flow shop with multi processors", European Journal of Operational Research, Vol. 51, pp. 88 – 89, 1991.
[3] Sawik, T., "Mixed Integer Programming for Scheduling Flexible flow line with Limited InterMediate Buffers", Mathematical and Computer Modeling, Vol. 31, pp. 39-52, 2000.
[4] Ding, F.Y. and Kittichartphayak, D., "Heuristic for Scheduling Flexible Flow lines", Computers and Industrial Engineering, Vol. 26, pp. 27-34, 1994.
[5] Sundararaghavan, P.S.; Kunnathur, A.S. and Viswanathan, I.., "Minimizing make span in parallel flow shops", Journal of the Operational Research Society, Vol. 48, pp. 834 – 842, 1997.
[6] Lee, C.–Y. and Vairaktarakis, G.L., "Minimizing make span in hybrid flow shops", Operation Research Letters, Vol. 16, pp. 149 – 158, 1994.
[7] Nowicki, E. and Czeslaw, S., "The flow shop with parallel machines: A tabu search approach", European Journal of Operational Research, Vol. 106, pp. 226 – 253, 1998.
[8] Negenman, E.G., "Local search algorithms for the multiprocessor flow shop scheduling problem". European Journal of Operational Research, Vol. 128, pp. 147 – 158, 2001.
[9] Vandevelde, A., "Minimizing the make span in a multiprocessor flow shop", Master's thesis, Eindhoven University of Technology, 1994.
[10] Wardono, B. and Fathi, Y., "A tabu search algorithm for the multi-stage parallel machine problem with limited buffer capacities", European Journal of Operational Research, Vol. 155, pp. 380-401, 2004.
[11] MIchalewicz, Z., "GENETIC Algorithms + Data Structures = Evolution programs", 3rd edition, Springer–Verlag, New York, 1992.