نشریه بین المللی مهندسی صنایع و مدیریت تولید زمستان ۱۳۸۹- صفحه ۱۱۵-۱۰۴

شماره ۴، جلد ۲۱، زمستان ۱۳۸۹

نشریه بین المللی مهندسی صنایع و مدیریت تولید

نشریه بین المللی مهندسی صنایع و مدیریت تولید
دانشگاه علم و صنعت ایران

ISSN: 2008-4870

http://IJIEPM.iust.ac.ir/

مدل سازی و حل مساله زمانبندی کار کارگاهی با زمانهای آماده‌سازی وابسته به توالی

مهدی بهروزی و کوروش عشقی*

کلمات کلیدی

زمانبندی کار کارگاهی،
زمانهای آماده‌سازی وابسته به توالی
جدایی ناپذیر،
بهینه‌سازی دسته ذرات،
فاکتورادیک،
آنیلینگ شبیه‌سازی شده،
جستجوی تصادفی حریصانه

چکیده:

در یک دهه اخیر تحقیقات نسبتا کمی بر روی مساله زمانبندی کار کارگاهی با زمان‌های آماده‌سازی وابسته به توالی جدایی‌ناپذیر (ISDSJSP) انجام شده است. در زمینه حل این مساله نیز الگوریتم بهینه‌سازی دسته ذرات (PSO) مورد توجه قرار نگرفته است. در این مقاله مساله ISDSJSP با تابع هدف کمینه‌سازی زمان پایان تمام کارها مدنظر قرار گرفته و یک مدل برنامه‌ریزی خطی عدد صحیح مختلط و یک الگوریتم ترکیبی PSO برای آن ارایه شده است. جواب بدست آمده توسط PSO با استفاده از یک الگوریتم آنیلینگ شبیه‌سازی شده بهبود داده شده است. عملکرد الگوریتم پیشنهادی (HPSO) نسبت به دو الگوریتم دیگر بر روی مسایل نمونه تولید شده در این مقاله آزمایش شده است. نتایج حاصل، دقت و کارایی بیشتر جوابهای حاصل از HPSO نسبت به دو الگوریتم دیگر را نشان می‌دهند.

۱. مقدمه

در طی سالیان متمادی به دلیل پیچیدگی زیاد مساله زمانبندی کار کارگاهی در تحقیقات صورت گرفته معمولا فرضهای ساده کننده زیادی برای مساله در نظر گرفته می‌شده است. یکی از این فرضهای ساده کننده صفر یا ناچیز در نظر گرفتن زمانهای آماده سازی است. تا یک دهه پیش در بسیاری از تحقیقات زمانهای آماده سازی اغلب یا صفر در نظر گرفته می‌شدند (از آنها صرفنظر می‌شد) یا بصورت مستقل از توالی و به عنوان جزیی از زمان پردازش در نظر گرفته می‌شد. اما در حالتی که زمان آماده‌سازی یک کار بر روی یک ماشین به کار قبلی پردازش شده بر روی آن ماشین وابسته باشد و با تغییر آن کار، زمان آماده‌سازی نیز کم یا زیاد شود، زمان آماده‌سازی را نمی‌توان داخل زمان پردازش گنجاند و باید آن را به صورت جدا در نظر گرفت. این حالت در بسیاری از مسایل عملی مانند صنایع شیمیایی، صنایع چاپ، بخش رنگ در صنایع خودروسازی، صنایع دارویی و بسیاری از فرایندهای تولیدی رخ می‌دهد. این نوع زمان آماده‌سازی که وابسته به توالی[۲] نامیده می‌شود، در یک دهه اخیر مورد توجه بسیاری از محققین قرار گرفته است.

در این مقاله گونه‌ای از زمانهای آماده‌سازی مورد توجه قرار گرفته است که اولا مقدار آن برای هر کار به ماشین مورد نظر نیز بستگی دارد (وابسته به توالی است) که این مساله را می‌توان با استفاده از شیوه نگارش سه تایی گراهام [۱] به صورت $Jm|s_{ijk}|C_{\max}$ نشان داد و ثانیا جدایی ناپذیر[۳] است. زمان آماده‌سازی جدایی ناپذیر (که منظور از آن جدا نشدن آماده سازی از کار یا چسبیده بودن آن به کار است) در حالتی رخ می‌دهد که شروع آماده‌سازی یک کار بر روی یک ماشین مستلزم آماده بودن ماشین و کار مورد نظر برای انجام آماده‌سازی است. در این مقاله زمان‌های راه‌اندازی و پاکسازی نیز در غالب زمانهای آماده‌سازی در نظر گرفته می‌شوند.

گری و همکارانش [۲] ثابت کردند که مساله $Jm||C_{\max}$ یک مساله NP-hard است. در نتیجه NP-hard بودن مساله

تاریخ وصول: ۸۸/۱۲/۵

تاریخ تصویب: ۸۹/۳/۲۶

مهدی بهروزی، فارغ‌التحصیل کارشناسی ارشد، دانشکده مهندسی صنایع، دانشگاه صنعتی شریف، mbehroozi@alum.sharif.edu

***نویسنده مسئول مقاله: دکتر کوروش عشقی**، استاد، دانشکده مهندسی صنایع، دانشگاه صنعتی شریف، eshghi@sharif.edu

[2] *Sequence Dependent*
[3] *Inseparable*

$Jm|s_{ijk}|C_{\max}$ به سادگی با فرض صفر بودن تمامی زمانهای آماده‌سازی اثبات می‌شود. در مورد مطالعات صورت گرفته در زمینه زمانهای آماده‌سازی وابسته به توالی در محیط مساله JSP می‌توان به الگوریتم شاخه و کران[1] [۳]، موجه کردن جوابهای غیر موجه [۴]، الگوریتم بهینه سازی اجتماع مورچگان (ACO)[2] [۵]، الگوریتم ژنتیک (GA)[3] [۶]، الگوریتم جستجوی محلی[4] [۷]، قوانین ابتکاری زمانبندی (قوانین اولویت‌بندی[5]) [۸]، الگوریتم ابتکاری انتقال گلوگاه (SB)[6] [۹]، مقایسه روشهای جستجوی محلی با الگوریتم‌های ابتکاری آزمندی[7] و قوانین اولویت‌بندی [۱۰] و الگوریتم ژنتیک در ترکیب با قوانین اولویت‌بندی [۱۱] اشاره کرد. همچنین بهروزی و عشقی [۱۲] یک الگوریتم بهینه‌سازی دسته ذرات (PSO) در ترکیب با یک الگوریتم GRASP و یک الگوریتم آنیلینگ شبیه‌سازی شده برای مساله $Jm||C_{\max}$ ارایه کردند که در آن یک رابطه بهنگام‌سازی جدید برای سرعت ذرات توسعه داده شده بود. آنها همچنین از یک شیوه تبدیل جدید برای ایجاد یک تناظر یک به یک بین جواب گسسته مساله کار کارگاهی و جوابهای پیوسته مورد پذیرش الگوریتم استفاده کردند. مقاله حاضر الگوریتمهای ارایه شده در [۱۲] را با انجام اصلاحات لازم برای مساله $Jm|s_{ijk}|C_{\max}$ بکار گرفته است.

در ادامه مقاله در بخش ۲ تعریف مساله و مدلسازی آن ارایه می‌شود. الگوریتم پیشنهادی در بخش ۳ ارایه شده است. در بخش ۴ الگوریتم پیشنهادی بر روی تعدادی از مسایل نمونه تولید شده در این تحقیق، آزمایش شده و نتایج حاصل از آن گزارش شده است. سرانجام نتیجه‌گیری و پیشنهادهایی برای مطالعات آتی در بخش ۵ ارایه شده است.

۲. تعریف و مدلسازی مساله

۲-۱. تعریف مساله

مساله ISDSJSP مورد بررسی را بطور خلاصه می‌توان بصورت زیر تعریف کرد. در مساله زمانبندی کار کارگاهی با زمانهای آماده‌سازی وابسته به توالی m ماشین و n کار وجود دارد که در آن کارها باید بر روی ماشینها پردازش شوند. پردازش کار j بر روی ماشین i یک فعالیت نامیده می‌شود و به صورت زوج مرتب (i, j) و یا بصورت O_{ij} نشان داده می‌شود. بین هر دو فعالیت یک کار یک رابطه پیش‌نیازی وجود دارد ولی بین هر دو فعالیت از کارهای مختلف هیچگونه رابطه پیش‌نیازی وجود ندارد. به هر کاری برای پردازش بر روی یک ماشین، یک زمان پردازش و یک زمان آماده‌سازی که به کار قبلی پردازش شده بر روی آن ماشین وابسته است اختصاص می‌یابد. هر یک از کارها برای پردازش بر روی ماشینها دارای یک مسیر پردازش مختص خود است که بوسیله توالی ماشینها مشخص می‌شود و در واقع روابط پیش‌نیازی مابین فعالیتهای آن را نشان می‌دهد. در این مقاله، برای نشان دادن مسیر پردازش هر کار ابتدا کلیه ماشینهای موجود را به صورت $M_1, M_2, ..., M_m$ نامگذاری می‌کنیم و سپس مسیر پردازش را به صورت مجموعه $JP_j = \{M_{1j} = i_1, M_{2j} = i_2, ..., M_{mj} = i_m\}$ تعریف می‌کنیم که در آن مقدار i_k در عبارت $M_{kj} = i_k \; ; \; \forall k = 1,2,...,m$ برابر با عددی است که نشان دهنده‌ی اولویت ماشین M_k (فعالیت O_{kj}) در مسیر پردازش کار j است. در واقع عبارت $M_{kj} = i_k$ نشان می‌دهد که ماشین M_k، i_k امین ماشین در مسیر پردازش کار j (فعالیت (k, j) از کار j) خواهد بود. در صورتی که کار j به ماشین M_k نیاز نداشته باشد $i_k = 0$ خواهد بود و در غیر اینصورت عددی بین ۱ تا m_j را اختیار می‌کند که m_j نشان‌دهنده‌ی تعداد ماشینهایی است که کار j به آنها نیاز دارد.

در این مدل نمی‌توان فرض کرد که هر کاری باید بر روی همه ماشینها پردازش شود چرا که در صورت عدم نیاز یک کار به یک ماشین حتی با صفر در نظر گرفتن زمان پردازش آن کار بر روی آن ماشین با توجه به تغییر توالی و در نتیجه تغییر زمانهای آماده‌سازی نمی‌توان این فرض را برقرار کرد. بنابراین طول مسیر پردازش برای کارهای مختلف متفاوت خواهد بود. در این مساله هر کار در هر لحظه زمانی تنها می‌تواند بر روی یک ماشین پردازش شود و هر ماشین نیز در هر لحظه زمانی حداکثر می‌تواند یک کار را پردازش کند. همه کارها در زمان صفر در دسترس هستند و هیچ یک از کارها دارای موعد تحویل نمی‌باشند. هدف مساله یافتن یک زمانبندی است که زمان پایان تمام کارها (کار آخر) را کمینه کند.

۲-۲. مؤلفه‌ها و متغیرهای مورد استفاده

در زیر تعریف مؤلفه‌های بکار گرفته شده (s_{ikj} و p_{ij}) و متغیرهای مورد استفاده ($C_{\max}$، ts_{ij}، t_{ij} و x_{ijk}) در مدل ارایه شده است:

s_{ikj} : زمان آماده‌سازی وابسته به توالی مورد نیاز برای پردازش کار j روی ماشین i اگر بلافاصله پس از کار k بر روی این ماشین انجام شود

p_{ij} : زمان پردازش کار j روی ماشین i

$C_{\max}$: زمان پایان تمام کارها

ts_{ij} : زمان شروع آماده‌سازی کار j روی ماشین i

t_{ij} : زمان آغاز پردازش کار j روی ماشین i

x_{ijk}: متغیر تعیین کننده توالی کارها که برابر یک است اگر پردازش کار k بلافاصله پس از کار j روی ماشین i انجام شود و در غیر اینصورت صفر است

[1] *Branch and Bound*
[2] *Ant Colony Optimization*
[3] *Genetic Algorithm*
[4] *Local Search Algorithm*
[5] *Dispatching Rules*
[6] *Shifting Bottleneck Procedure (SBP)*
[7] *Greedy Heuristic Algorithms*

تذکر: متغیرهای x_{i0j} یا x_{ij0} (مؤلفه‌های s_{i0j} یا s_{ij0}) به ترتیب برای حالاتی تعریف می‌شوند که در آنها کار j کار اولی یا کار آخری است که باید روی ماشین i پردازش شود.

۲-۳. مدل ریاضی مساله

برای مساله ISDSJSP مورد بررسی یک مدل برنامه‌ریزی خطی عدد صحیح مختلط توسعه داده شده است که در ادامه ارایه می‌شود.

$$\text{Min } C_{\max}$$

S. to:

$$t_{ij} + p_{ij} + s_{ij0}x_{ij0} \le C_{\max} \quad , \ i \in \mathrm{I} \ , \quad j = \vartheta \qquad (۱)$$

$$t_{ij} + p_{ij} \le ts_{lj} \quad , \ i,l \in \mathrm{I} \ , \ j \in \vartheta \ , \ 0 < M_{ij} < M_{lj} \ , \ M_{ij}, M_{lj} \in JP_j \qquad (۲)$$

$$t_{ij} + p_{ij} \le ts_{ik} + M\left(1 - x_{ijk}\right) \quad , \ i = \mathrm{I} \ , \ j,k \in \vartheta \qquad (۳)$$

$$ts_{ij} + \sum_{\substack{k=0 \\ k \ne j}}^{n} s_{ikj}x_{ikj} \le t_{ij} \quad , \ i \in \mathrm{I} \ , \quad j \in \vartheta \qquad (۴)$$

$$\sum_{\substack{k=0 \\ k \ne j}}^{n} x_{ijk} = 1 \quad , \ i \in \mathrm{I} \ , \ j \in \left\{\{0\} \cup \{\vartheta\}\right\} \ , \ M_{ij} > 0 \ , \ M_{ij} \in JP_j \qquad (۵)$$

$$\sum_{\substack{j=0 \\ j \ne k}}^{n} x_{ijk} = 1 \quad , \ i \in \mathrm{I} \ , \ k \in \left\{\{0\} \cup \{\vartheta\}\right\} \ , \ M_{ik} > 0 \ , \ M_{ik} \in JP_k \qquad (۶)$$

$$x_{ijk} \le M_{ij} \times M_{ik} \quad , \ i \in \mathrm{I} \ , \quad j,k \in \vartheta \ , \ M_{ij} \in JP_j \ , \ M_{ik} \in JP_k \qquad (۷)$$

$$x_{ij0} \le M_{ij} \quad , \ i \in \mathrm{I} \ , \quad j \in \vartheta \ , \ M_{ij} \in JP_j \qquad (۸)$$

$$x_{i0k} \le M_{ik} \quad , \ i \in \mathrm{I} \ , \quad k \in \vartheta \ , \ M_{ik} \in JP_k \qquad (۹)$$

$$t_{ij} \ge 0 \ , \quad ts_{ij} \ge 0 \quad , \ i \in \mathrm{I} \ , \ j \in \vartheta \qquad (۱۰)$$

$$x_{ijk} \in \{0,1\} \quad , \ i \in \mathrm{I} \ , \ j,k \in \vartheta \qquad (۱۱)$$

در مدل ارایه شده منظور از مجموعه I مجموعه تمام ماشینها بصورت $\mathrm{I} = \{1,2,...,m\}$ است و منظور از مجموعه ϑ مجموعه تمام کارها بصورت $\vartheta = \{1,2,...,n\}$ است. مجموعه JP_j نیز همانطور که پیشتر هم گفته شد مسیر پردازش کار j را نشان می‌دهد و شامل جملاتی بصورت M_{ij} است.

مدل ارایه شده یک مدل جدید است که در این مقاله توسعه داده شده و در آن زمانهای آماده‌سازی وابسته به توالی بصورت جدایی‌ناپذیر در نظر گرفته شده است. در این مدل رابطه (۱) به ما اطمینان می‌دهد که زمان پایان هیچ کاری از مقدار بهینه تابع هدف بزرگتر نیست. رابطه (۲) مسیر تکنولوژیکی پردازش کار j بر روی ماشین‌های مختلف و محدودیت زمانی بین پایان پردازش آن بر روی ماشین قبلی و شروع آماده‌سازی آن بر روی ماشین بعدی را نشان می‌دهد. این محدودیت در واقع باعث جدایی ناپذیر بودن آماده‌سازی نیز می‌شود، چرا که تا زمانی که پردازش کار بر روی ماشین قبلی تمام نشده و کار به پای ماشین بعدی نرسیده است، آماده‌سازی آن بر روی ماشین بعدی نمی‌تواند انجام شود. رابطه (۳) محدودیت توالی کارها روی هر ماشین است. رابطه (۴) ارتباط مابین زمان شروع پردازش و زمان شروع آماده‌سازی هر کار بر روی هر ماشین را نشان می‌دهد. به عبارت دیگر پردازش هیچ کاری تا زمانی که آماده‌سازی آن تمام نشده است، نمی‌تواند شروع شود. رابطه (۵) نشان می‌دهد که هر گاه پردازش یک کار بر روی ماشین i تمام شد تنها و تنها یک کار به عنوان کار بعدی برای پردازش روی آن ماشین انتخاب می‌شود، بجز هنگامی که کار انجام شده، کار آخر ماشین باشد (در این حالت تنها و تنها یک کار مجازی به عنوان کار بعدی انتخاب می‌شود). رابطه (۶) نشان می‌دهد که پردازش هر کاری بر روی ماشین i تنها و تنها پس از یک کار می‌تواند انجام شود، بجز هنگامی که کار مورد نظر کار اول ماشین باشد (در این حالت هر کار تنها و تنها پس از یک کار مجازی انجام می‌شود). روابط (۷)، (۸) و (۹) مقدار صفر را بر متغیرهای x_{ijk} به ازای کارهای j یا k ای که ماشین i در مسیر پردازش آنها نیست، تحمیل می‌کنند. روابط (۱۰) و (۱۱) نیز نوع و علامت متغیرهای مورد استفاده در مدل را نشان می‌دهند. اعتبار مدل ارایه شده با روش تحلیلی و بررسی منطقی روابط محدودیت‌ها و همچنین با روش محاسباتی و با حل تعدادی از مسایل نمونه با نرم افزار Lingo 8.0 مورد بررسی قرار گرفت و به تایید رسید.

۳. الگوریتم پیشنهادی برای مساله

توانایی‌های الگوریتم PSO و نتایج مطلوبی که از خود بخصوص برای حل مساله $Jm \| C_{\max}$ نشان داده است [۱۲]، مؤلفین را ترغیب کرد تا در این مقاله با انجام تغییراتی عمده بر روی الگوریتم PSO ارایه شده توسط بهروزی و عشقی [۱۲]، از آن برای حل مساله $Jm \mid s_{ijk} \mid C_{\max}$ استفاده کنند. در الگوریتم ترکیبی PSO اصلاح شده همچون الگوریتم PSO ارایه شده در [۱۲] جمعیت اولیه توسط یک الگوریتم GRASP تولید می‌شود. برای الگوریتم GRASP نیز اصلاحات لازم ارایه شده است. همچون [۱۲] به منظور جستجوی بهتر در اطراف هر جواب یافته شده توسط PSO

از یک الگوریتم جستجوی محلی (LS)[1] استفاده شده است و نیز از یک الگوریتم SA برای بهبود نهایی بهترین جواب بدست آمده از الگوریتم GRASP-PSO/LS استفاده شده است. تفاوت الگوریتم جستجوی محلی و الگوریتم SA ارایه شده در این مقاله نسبت به آنچه در [۱۲] آمده است در ساختار همسایگی آن است. بعبارت دیگر گراف انفصالی برای انطباق بر مساله مورد بررسی اصلاح شده است.

۳-۱. تضمین موجه ماندن جوابها

در این مقاله از روش نمایش بر مبنای فهرست اولویت[2] برای جوابهای مساله استفاده شده است. در این روش نمایش هر جواب بصورت یک رشته از کارها نشان داده می‌شود که در آن یک رشته از m زیر رشته هر یک متناظر با یک ماشین (مجموعا m ماشین)، تشکیل می‌شود. این زیر رشته‌ها توالی پردازش فعالیتها بر روی ماشین را نشان نمی‌دهند که اگر اینگونه بود ممکن بود جوابهای تولیدی موجه نباشند (محدودیت مسیر پردازش کارها را رعایت نکنند). این زیر رشته‌ها بصورت فهرستهای اولویت تفسیر می‌شوند و اولویت پردازش یک کار بر کار بعدی خود را تنها در صورتی که پیش از آن کار قابل پردازش باشد، نشان می‌دهد [۱۳].

در الگوریتمهای فرا ابتکاری زمانبندی، جواب قانونی[3] به جوابی گفته می‌شود که در آن هر کاری دقیقا یک بار در توالی کارهای هر ماشین، ذکر شده باشد [۱۳]. با توجه به اینکه با استفاده از روش نمایش بر مبنای فهرست اولویت در هر مرحله تنها فعالیتهایی را اختصاص می‌دهیم که فعالیتهای پیش‌نیاز آن انجام شده باشد، زمانبندی بدست آمده از هر جواب قانونی محدودیت روابط پیش‌نیازی فعالیتها (محدودیت مسیر پردازش کارها) را برآورده می‌کند و در نتیجه موجه است. توضیحات بیشتر راجع به شیوه نمایش مورد استفاده را می‌توانید در [۱۲ و ۱۳] ببینید.

۳-۲. ایجاد تناظر یک به یک و تضمین قانونی بودن جوابها

جواب مساله ISDSJSP بصورت توالی اعداد صحیح و گسسته است در حالی که جواب مورد پذیرش الگوریتم PSO برداری از اعداد پیوسته است. در اینجا برای انطباق شیوه نمایش جواب بر مبنای فهرست اولویت با شیوه مورد پذیرش الگوریتم، از شیوه تبدیل مبنای اعداد توسعه داده شده توسط بهروزی و عشقی [۱۲] استفاده شده است. در این روش که از تبدیل مبنای فاکتوریلی[4] برای ایجاد تناظری یک به یک بین اعداد هر زیر رشته با یک عدد پیوسته مبنای ۱۰ استفاده می‌شود، قانونی بودن جوابها نیز تضمین می‌شود. برای توضیحات بیشتر راجع به این روش می‌توانید به [۱۲] مراجعه کنید.

۳-۳. ایجاد جمعیت اولیه

با توجه به اثرگذاری کیفیت جوابهای اولیه بر کیفیت جواب نهایی الگوریتم PSO، در این مقاله به جای جوابهای تصادفی از یک دسته جوابهای با کیفیت نسبتا خوب و با پراکندگی مناسب که بوسیله یک الگوریتم سازنده جواب[5] تولید می‌شوند، به عنوان جمعیت اولیه استفاده شده است. با این کار ضمن حفظ پراکندگی جوابهای اولیه، در سرعت الگوریتم و کیفیت جواب نهایی آن بهبود حاصل خواهد شد. بدین منظور از الگوریتم جستجوی تصادفی حریصانه (GRASP)[6] ارایه شده توسط بهروزی و عشقی [۱۲] و با انجام اصلاحاتی استفاده شده است. الگوریتم اصلاح شده در زیر ارایه می‌شود:

گام ۱: با استفاده از مجموعه‌هایی بصورت $JP_j=\{M_{1j}=i_1, M_{2j}=i_2, ..., M_{mj}=i_m\}$ که مسیر پردازش کار j را نشان می‌دهند و جزو داده‌های مساله هستند، ماتریسی به نام *JobPath* که سطرهای آن نشان دهنده‌ی کارها و ستونهای آن نشان دهنده‌ی ماشین‌ها است، تشکیل دهید. بعبارت دیگر هر مجموعه *JP* را در یک سطر این ماتریس بگذارید. با استفاده از ماتریس *JobPath* ماتریس *machPlan* که نشان دهنده‌ی کارهایی است که هر ماشین باید انجام دهد را بصورت ذیل بدست آورید:

- قرار دهید؛ $i:=1,\ j:=1,\ k:=1$.
- اگر ji امین عنصر ماتریس *JobPath* مثبت (ناصفر) بود، ik امین عنصر ماتریس *machPlan* را برابر با j قرار دهید و سپس قرار دهید؛ $j:=j+1,\ k:=k+1$ و به بند بعد بروید. در غیر اینصورت قرار دهید؛ $j:=j+1$ و به بند بعد بروید.
- اگر $i=m$ و $j=n+1$ بود متوقف شوید. اگر $i\le m$ و $j\le n$ بود بند قبل را تکرار کنید. اگر $i<m$ و $j=n+1$ بود، قرار دهید؛ $k:=1,\ i:=i+1,\ j:=1$ و بند قبل را تکرار کنید.

تعداد اعداد مثبت موجود در هر سطر i از ماتریس *machPlan* را که نشان دهنده‌ی تعداد کارهایی است که ماشین M_i باید پردازش کند را بدست آورده و این مقدار را با متغیر $machLen_i$ نشان دهید.

گام ۲: مجموع زمانهای آماده‌سازی و پردازش هر کار بر روی هر ماشین را بدست آورید و کارها را بر اساس قانون SSPT[7] بر

[1] *Local Search*
[2] *Preference list-based representation*
[3] *Legal solution*
[4] *Factorial base transformation*
[5] *Constructive Algorithm*
[6] *Greedy Randomized Adaptive Search Procedure (GRASP)*
[7] *Shortest Setup and Processing Time first*

روی ماشینها مرتب کنید و رتبه هر کار بر روی هر ماشین را با $SSPT_{ij}$ نشان دهید (در صورت تساوی رتبه برابر بگیرند). پس از آن مقادیر $LSPT_{ij}$ (برای در نظر گرفتن قانون LSPT[1]) را بصورت $LSPT_{ij} = SSPT_{i(n-j+1)}$ به ازای تمام i و jهایی که $SSPT_{ij}$ برای آنها تعریف شده است، بدست آورید.

گام ۳: مقدار f_{ij} را به ازای $i = 1,...,m$, $j = 1,...,n$ به صورت زیر بدست آورید (هرچه مقدار f_{ij} کمتر باشد اولویت کار j برای پردازش بر روی ماشین i بیشتر است) :

- اگر $JobPath_{ji} = 0$ است مقدار f_{ij} را برابر یک مقدار بسیار بزرگ قرار دهید.
- اگر $JobPath_{ji} \neq 0$ است مقدار f_{ij} را با استفاده از رابطه (۱۲) محاسبه کنید که در آن پارامترهای w_1، w_2 و w_3 مؤلفه‌های وزنی هستند و برای تنظیم اهمیت نسبی چهار عامل استفاده شده در این رابطه بکار می‌روند.

$$f(O_{ij}) = w_1 M_{ij} + w_2 SSPT_{ij} + w_3 LSPT_{ij} \qquad (۱۲)$$

گام ۴: سطر i ام ماتریس $machPlan$ را در فهرستی به نام $A(i)$ بریزید و آن را به ترتیب صعودی مقادیر f_{ij} به ازای تمام کارهای jموجود در آن سطر مرتب کنید. سپس کارهای متناظر با γ درصد اول این مقادیر را در فهرست جدیدی به نام $B(i)$ به عنوان مناسب‌ترین کارهای نامزد برای تخصیص بر روی ماشین i ذخیره کنید. سپس قرار دهید $s := 1$ و $t := 1$ و به گام بعد بروید.

گام ۵: مقادیر فهرست‌های $A(i)$ و $B(i)$ را عینا در دو فهرست جدید به نامهای $A'(i)$ و $B'(i)$ بریزید. حال قرار دهید $i := 1$ و $p := 1$ و به گام بعد بروید.

گام ۶: فعالیت p ام از ماشین i را برای افزودن به جواب جزیی S_s، در صورتی که $B'(i)$ هنوز خالی نشده است به صورت تصادفی از فهرست $B'(i)$ انتخاب کنید، در غیر اینصورت این فعالیت را به ترتیب کمترین مقادیر f از فهرست $A'(i) - B'(i)$ انتخاب کنید. فعالیت (کار) مورد نظر را از فهرست $B'(i)$ $\left(A'(i) - B'(i)\right)$ حذف کنید. حال شرایط ذیل را بررسی کنید:

- اگر $i = m$ و $p < machLen_i$ بود قرار دهید؛ $i := 1$ و $p := p + 1$ و گام ۶ را تکرار کنید.
- در صورتی که $i < m$ و $p \leq machLen_i$ است قرار دهید؛ $i := i + 1$ و گام ۶ را تکرار کنید.
- اگر $i = m$ و $p = machLen_i$ بود، به گام بعد بروید.

گام ۷: جواب بدست آمده را به عنوان S_s ذخیره نمایید و آن را به یک زمانبندی فعال[2] تبدیل کرده و مقدار تابع هدف آن $C_{\max}(S_s)$ را با مقدار تابع هدف بهترین جواب بدست آمده تا این مرحله $C_{\max}(S^*)$، مقایسه کنید. در صورتی که $C_{\max}(S_s) < C_{\max}(S^*)$ است آن را جایگزین S^* فعلی کنید. سپس در صورتی که $s < N_s$ و $t \leq nIter$ باشد با قرار دادن $s := s + 1$ به گام ۵ برگردید. که در آن N_s تعداد جوابهای مورد نیاز در هر تکرار است و $nIter$ نیز تعداد تکرارهای مورد نیاز برای اجرای الگوریتم است. در غیر اینصورت اگر $s = N_s$ و $t < nIter$ باشد با قرار دادن $s := 1$ و $t := t + 1$ به گام ۵ برگردید و اگر $s = N_s$ و $t = nIter$ بود، متوقف شوید.

هر جواب الگوریتم GRASP مورد استفاده بصورت فهرستی از کارها برای تک تک ماشینها بدست می‌آید. فهرست‌های $A(i)$ در گام ۴ الگوریتم بر اساس ماتریس $machPlan$ برای هر ماشین i بگونه‌ای تولید می‌شوند که فقط و فقط کارهایی در آن فهرست شوند که آن ماشین باید آنها را پردازش کند. از طرف دیگر در گام ۶ با اختصاص هر فعالیت (کار) به یک ماشین، آن فعالیت (کار) از فهرست $B'(i)$ یا فهرست $A'(i) - B'(i)$ حذف می‌شود. در نتیجه هیچ‌گاه در یک رشته مربوط به یک ماشین، شماره هیچ کاری تکرار نمی‌شود. بنابراین جوابهای بدست آمده همواره قانونی خواهند بود. همچنین با توجه به اینکه هر یک از این فهرست‌ها بصورت فهرست اولویت کارها بر روی ماشین مربوطه تفسیر می‌شوند جوابهای بدست آمده همواره موجه نیز هستند.

۳-۴. گراف انفصالی و ساختار همسایگی

در این مقاله یک گراف انفصالی جدید که قابلیت در نظر گرفتن زمانهای آماده‌سازی وابسته به توالی را دارد، توسعه داده شده است. نمونه‌ای از این گراف انفصالی در شکل ۱ نشان داده شده است که در آن فرض شده است که ۳ کار 1، j و n به ماشین یکسان i نیاز دارند که اولین ماشین در مسیر پردازش کارهای 1 و n است و ماشینهای l و r آخرین ماشینهایی باشند که به ترتیب کارهای 1 و n در مسیر پردازش خود به آنها نیاز دارند. می‌دانیم که در گراف انفصالی سنتی دو نوع گره و دو نوع کمان وجود دارد. دو نوع گره شامل دو گره ابتدایی و انتهایی و سایر گره‌های میانی است. گره ابتدایی (S) برای شروع کارها و گره انتهایی (F) به منزله پایان کارها است. گره‌های میانی به طور کلی به صورت (i,j) نشان داده می‌شوند که نشان دهنده فعالیتی از کار j است که بر روی ماشین i انجام می‌شود. گره‌های متناظر با برخی از فعالیتها برای پرهیز از شلوغی در شکل نشان داده نشده‌اند. کمانهای موجود در گراف انفصالی سنتی عبارت‌اند از کمانهای ربطی (ارتباطی)[3] که بصورت

[1] *Longest Setup and Processing Time first*

[2] *Active Schedule*
[3] *Conjunctive arcs*

خطوط پر (⟶) نشان داده می‌شوند و کمانهای فصلی (انفصالی)[1] که بصورت خط‌چین و به شکل (⟵⟶) نشان داده می‌شوند. کمانهای ربطی بین فعالیتهای مختلف یک کار رسم می‌شوند و مسیر تکنولوژیکی پردازش کار را نشان می‌دهند. بر روی هر کمان ربطی زمان پردازش فعالیتی که این کمان از گره مربوط به آن خارج می‌شود، نوشته می‌شود. کمانهای فصلی در دو جهت بین فعالیتهای کارهای مختلف رسم می‌شوند و در واقع فعالیتهایی از این کارها را که به یک ماشین یکسان برای پردازش نیاز دارند، به صورت یک زیر گراف کامل (خوشه[2]) به هم مرتبط می‌کنند. روی هر کمان فصلی زمان پردازش فعالیتی که این کمان از گره مربوط به آن خارج می‌شود باضافه زمان آماده سازی مربوط به فعالیتی که این کمان به گره متناظر آن وارد می‌شود، نوشته می‌شود.

در گراف انفصالی جدید دو نوع گره جدید تعریف شده است. گره اول گره $\langle i \rangle$ است که نشان دهنده‌ی آماده‌سازی اولین کار بر روی ماشین i است و بر روی کمان خروجی از آن که نوعی کمان جدید است که در این گراف تعریف کرده‌ایم (⟶⋯⟶)، مقدار s_{i0j} نوشته می‌شود که در آن کار j نشان دهنده‌ی کاری است که به عنوان اولین کار بر روی ماشین i انجام می‌شود و این مقدار مربوط به عملیات راه‌اندازی بر روی ماشین i است. از بین تمام گره‌های موجود به شکل $\langle i \rangle$ برای هر ماشین i تنها یک گره به همراه کمان خروجی از آن انتخاب می‌شود و سایر کمانها و گره‌های متناظر آنها حذف خواهند شد. گره مذکور بصورت یک فعالیت مجازی با زمان پردازش s_{i0j} قبل از اولین فعالیت ماشین i در می‌آید و کمان مربوط به آن نیز به یک کمان ربطی تبدیل می‌شود. گره دیگر تعریف شده، گره $\langle i \rangle$ است که نشان دهنده‌ی عملیات پاکسازی آخرین کار (فعالیت) بر روی ماشین i است و بر روی کمان ورودی به آن (⟶⋯⟶) مقدار p_{ij} مربوط به فعالیتی که آن کمان از آن خارج شده است نوشته می‌شود و بر روی کمان خروجی از آن (⟶⋯⟶) مقدار s_{ij0} نوشته می‌شود که در آن کار j نشان دهنده‌ی کاری است که به عنوان آخرین کار بر روی ماشین i انجام می‌شود و این مقدار مربوط به عملیات پاکسازی بر روی ماشین i است. از بین تمام گره‌های $\langle i \rangle$ به ازای هر ماشین i تنها یک گره به همراه کمان‌های ورودی و خروجی آن انتخاب می‌شود و سایر کمانها و گره‌های متناظر آنها حذف خواهند شد. گره مذکور بصورت یک فعالیت مجازی با زمان پردازش s_{ij0} پس از آخرین فعالیت ماشین i در می‌آید و کمانهای مربوط به آن نیز به کمانهای ربطی تبدیل می‌شود.

ایجاد هر جواب موجه در مساله به منزله انتخاب یک جهت در هر جفت کمان انفصالی موجود مابین هر دو فعالیتی که به یک ماشین برای پردازش نیاز دارند، است، بگونه‌ای که دوری نیز در زیر گراف مذکور وجود نداشته باشد. زمان پایان تمام کارها ($C_{\max}$) از محاسبه طولانی‌ترین مسیر (مسیر بحرانی) در گراف ارتباطی (جواب موجه) حاصل از انتخاب یک جهت برای هر کمان انفصالی و انتخاب یکی از گره‌های $\langle i \rangle$ و همچنین یکی از گره‌های $\langle i \rangle$ و کمانهای مربوطه برای هر ماشین i بدست می‌آید. تعریف همسایگی در این مقاله بر اساس قضیه بالاش است. بالاش در [۱۴] قضیه‌ای را به اثبات رساند که اگر جهت یک کمان انفصالی واقع بر مسیر بحرانی یک جواب موجه را برعکس کنیم، جواب حاصل همچنان موجه خواهد ماند. این قضیه اساس تعریف همسایگی در بسیاری از الگوریتمهای جستجوی محلی است که از گراف انفصالی برای پیاده‌سازی الگوریتم کمک می‌گیرند.

۳-۵. الگوریتم بهینه‌سازی دسته ذرات ترکیبی (HPSO) برای مساله JSSP

الگوریتم بهینه سازی دسته ذرات ترکیبی که به اختصار HPSO خوانده خواهد شد در گامهای زیر خلاصه می‌شود:

گام ۱: مقادیر داده‌ها و پارامترها را بگیرید. ماتریس‌های *JobPath* و *machPlan* را همانگونه که در الگوریتم GRASP توضیح داده شد، تشکیل دهید و مقدار متغیر $machLen_i$ را نیز محاسبه کنید. همچنین فهرستی از اعداد $0,1,2,...,(n-1)$ تهیه کنید و آن را فهرست F بنامید. دو فهرست دیگر از زیرمجموعه‌ای از اعداد $1,2,...,n$ برای هر ماشین i تهیه کنید که در آن هر عدد نشان دهنده‌ی یک کار است و این فهرستها را P_i و P_i' بنامید. این زیر مجموعه برابر با اعداد موجود در سطر i ام ماتریس *machPlan* است.

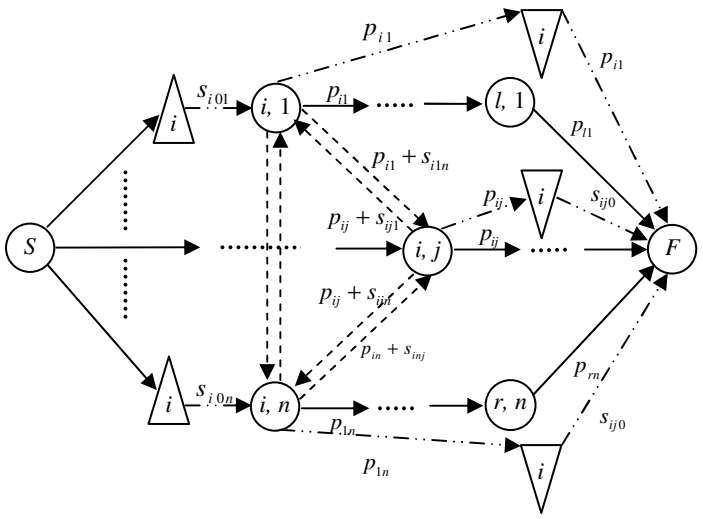
گام ۲: با استفاده از الگوریتم GRASP ارایه شده یک مجموعه جواب اولیه موجه با اندازه *popsize* تولید کنید.

گام ۳: فرض کنید که توالی کارها بر روی ماشین i بصورت توالی اعداد $h_{i(machLen_i)}...h_{i(machLen_i-1)}...h_1$ باشد. این توالی را به عنوان یک جایگشت در نظر گرفته و با استفاده از تبدیل جایگشت‌ها به مبنای فاکتوریل (f) و سپس تبدیل از مبنای f به مبنای ۱۰ و با بکارگیری فهرست‌های F و P_i'، توالی کارهای هر ماشین در هر جواب اولیه را به یک عدد در مبنای ۱۰ تبدیل کنید و سپس با استفاده از اعداد صحیح نامنفی بدست آمده یک بردار m بعدی ایجاد کنید. این بردار، موقعیت ذره مورد استفاده در الگوریتم را نشان می‌دهد. سپس بردارهای m بعدی بدست آمده برای هر جواب r را برابر اولین بردار موقعیت ذره r و بصورت $X_r^1=(x_{r1}^1,...,x_{rm}^1),\ \forall r=1,...,popsize$ قرار دهید و به گام ۴ بروید.

گام ۴: یک مجموعه بردار m بعدی به عنوان اولین بردار سرعت هر ذره r با اندازه *popsize* و بصورت

[1] *Disjunctive arcs*
[2] *Clique*

$V_r^1=\left(v_{r1}^1,...,v_{rm}^1\right)$ ، $\forall r=1,...,popsize$ از اعداد صحیح موجود در بازه $\left[0-x_{ri}^1,(machLen_i\text{ !}-1-x_{ri}^1)\right]$ بصورت تصادفی تولید کنید، سپس به گام ۶ بروید.



شکل ۱. یک گراف انفصالی اصلاح شده برای مساله کار کارگاهی با زمانهای آماده سازی وابسته به توالی در حالت کلی

گام ۵: i امین بعد از بردار سرعت حرکت هر ذره r را در k امین حرکت آن (k امین تکرار الگوریتم که k شمارنده تکرار الگوریتم است) با استفاده از فرمول زیر تغییر دهید:

$$v_{ri}^k \leftarrow \chi\left(wv_{ri}^{k-1}+c_1r_1(p_{ri}^k-x_{ri}^k)+c_2r_2(p_{gi}^k-x_{ri}^k)\right) \quad (۱۳)$$
$$,\forall i=1,2,...,m \ ,\forall r=1,...,popsize$$

که در آن w ضریب وزن اینرسی بوده، بصورت پویا مقادیر نزولی به خود می‌گیرد و در هر مرحله بصورت زیر مشخص می‌شود:

$$w=w_{\max}-\frac{w_{\max}-w_{\min}}{N_{pso}}\times k \quad (۱۴)$$

$w_{\max}$ و $w_{\min}$ ثابت بوده و توسط کاربر تعیین می‌شوند، N_{pso} تعداد کل تکرارها است و χ ضریب محدود سازی سرعت (انقباض)[1] است که به صورت $\chi=\frac{2}{C-2+\sqrt{C^2-4C}}, C=c_1+c_2>4$ تعریف می‌شود. در صورتی که v_{ri}^k در بازه $[0-x_{ri}^k,(machLen_i\text{ !}-1-x_{ri}^k)]$ بود، به گام ۶ بروید و در غیر اینصورت اگر مقدار v_{ri}^k کوچکتر از حد پایین بازه بود قرار دهید:

$$v_{ri}^k=rand\times(0-x_{ri}^k) \quad (۱۵)$$

و اگر بزرگتر از حد بالای بازه بود قرار دهید:

$$v_{ri}^k=rand\times(machLen_i\text{ !}-1-x_{ri}^k) \quad (۱۶)$$

که در دو رابطه اخیر rand یک عدد تصادفی در بازه [0,1] است. سپس به گام ۶ بروید.

گام ۶: با تغییر i امین بعد از بردار موقعیت هر ذره r با استفاده از فرمول زیر، آن ذره را به موقعیت جدید خود انتقال دهید:

$$x_{ri}^{k+1}=\left\lfloor x_{ri}^k+v_{ri}^k\right\rfloor , \quad \forall i=1,2,...,m \quad (۱۷)$$
$$,\forall r=1,...,popsize$$

با موقعیت جدید x_r^{k+1} برای هر ذره r به گام بعد بروید.

گام ۷: ابتدا کلیه مقادیر فهرست P_i را عینا در فهرست P_i' بریزید. سپس i امین بعد ($i=1,2,...,m$) از موقعیت فعلی هر یک از ذرات جمعیت که بصورت $X_r^{k+1}=\left(x_{r1}^{k+1},...,x_{rm}^{k+1}\right),\forall r=1,...,popsize$ نشان داده می‌شود را در نظر گرفته و آن را که عددی در مبنای ۱۰ است، به یک عدد در مبنای فاکتوریل بصورت $d_{i(machLen_i)}d_{i(machLen_i-1)}...d_{i1}$ تبدیل کنید. حال عدد بدست آمده را با توجه به فهرست‌های F و P_i' به یک جایگشت به صورت $h_{i(machLen_i)}h_{i(machLen_i-1)}...h_1$ تبدیل کنید (که نشان دهنده‌ی اولویت کارها بر روی ماشین i است). سپس با این جایگشت‌ها برای هر ماشین i جواب متناظر با هر ذره r را بصورت فهرست اولویت کارها تشکیل دهید و آنها را به زمانبندیهای فعال تبدیل کرده و مقدار تابع هدف ($C_{\max}$) آنها را محاسبه کنید. سپس هر زمانبندی فعال بدست آمده را در دو حافظه S_r و $S_{r_{temp}}$ ذخیره کنید و به گام ۸ بروید.

[1] *Velocity constriction coefficient*

گام ۸: (گام جستجوی محلی: پرقدرت سازی و متنوع سازی): در صورتی که $C_{\max}(S_{r_{temp}}) \le C_{\max}(S_r)$ باشد (مساوی به منظور متنوع سازی)، قرار دهید $S_r \leftarrow S_{r_{temp}}$. در صورتی که N_{ls} تکرار از اجرای گام ۸ گذشته است، با جواب S_r و مقدار تابع هدف $C_{\max}(S_r)$ به گـام ۹ بروید. در غیر اینصورت مسیر بحرانی زمانبندی فعال $S_{r_{temp}}$ را بیابید. کمانهای انفصالی (⟵) موجود بر مسیر بحرانی را مشخص کنید و از بین آنها یکی را به تصادف انتخاب کنید و با استفاده از تعویض زوجی دو کار همسایه در توالی کارهای زیر رشته (ماشین) مربوطه، جهت آن کمان انفصالی را معکوس کنید. مقدار تابع هدف جواب جدید بدست آمده را محاسبه کرده و آن را به زمانبندی فعال جدید $S_{r_{temp}}$ تبدیل کنید و گام ۸ را تکرار نمایید.

گام ۹: موقعیت هر ذره r را با توجه به زمانبندی فعال جدید بدست آمده و با استفاده از تبدیل مبنای فاکتوریلی بهنگام کنید.

گام ۱۰: اگر مقدار فعلی تابع هدف متناظر با موقعیت فعلی هر ذره r کمتر یا مساوی بهترین مقدار قبلی آن ذره ($pbest[r]$) بود، مقدار فعلی تابع هدف را جایگزین $pbest[r]$ نموده و موقعیت فعلی ذره را جایگزین بهترین موقعیت قبلی آن یعنی $P_r = (p_{r1},...,p_{rm})$ کنید. همچنین اگر مقدار فعلی تابع هدف متناظر با موقعیت فعلی هر ذره r کمتر یا مساوی بهترین مقدار قبلی بدست آمده توسط کلیه ذرات ($pbest[gbest]$) بود، قرار دهید: $gbest = r$ و موقعیت فعلی ذره را جایگزین بهترین موقعیت قبلی بدست آمده توسط کلیه ذرات یعنی $P_g = (p_{g1},...,p_{gm})$ کنید.

گام ۱۱: شروط توقف زیر را بررسی کنید.

- **شرط توقف ۱:** تعداد تکرارهای الگوریتم به اندازه از پیش تعیین شده‌ی N_{pso} برسد ($k > N_{pso}$ شود).
- **شرط توقف ۲:** میزان بهبود تابع هدف در N'_{pso} تکرار متوالی از مقدار از پیش تعیین شده‌ی ε کمتر باشد.

در صورتی که هیچ یک از شروط توقف برقرار نشده است، با قرار دادن $k := k + 1$ به گام ۵ برگردید. در غیر اینصورت زمانبندی فعال متناظر با $P_g = (p_{g1},...,p_{gm})$ را به عنوان بهترین جواب الگوریتم GRASP-PSO/LS و با نام S' شناخته، $C_{\max}(S')$ را محاسبه و آن را به گام ۱۲ منتقل کنید.

گام ۱۲ (گام پرقدرت سازی) : با استفاده از الگوریتم جستجوی محلی ذکر شده در گام ۸ سعی کنید جواب S' را به جواب بهتری تبدیل کنید. این کار را به تعداد N_{pow} تکرار کنید. سپس قرار دهید $S'' \leftarrow S'$ ، $S^* \leftarrow S'$ و $T \leftarrow T_0$. پس از آن نیز به گام ۱۳ بروید.

گام ۱۳ (الگوریتم آنیلینگ شبیه سازی شده) : با استفاده از الگوریتم جستجوی محلی که در گام ۸ توضیح داده شد جواب S'' را به جواب جدید S''_{new} تبدیل کنید و مقدار $\Delta = C_{\max}(S''_{new}) - C_{\max}(S'')$ را محاسبه کنید. قرار دهید $S'' \leftarrow S''_{new}$. اگر $\Delta \le 0$ بود قرار دهید $S' \leftarrow S''$ و $C_{\max}(S') \leftarrow C_{\max}(S'')$. در این حالت اگر $C_{\max}(S') < C_{\max}(S^*)$ بود، قرار دهید $C_{\max}(S^*) \leftarrow C_{\max}(S')$ و $S^* \leftarrow S'$. اگر $\Delta > 0$ بود، یک عدد تصادفی در بازه [0,1] تولید کنید و آن را r بنامید، آنگاه اگر $r \le e^{(-\frac{\Delta}{T})}$ بود، قرار دهید $S' \leftarrow S''$ و $C_{\max}(S') \leftarrow C_{\max}(S'')$ و در غیر اینصورت قرار دهید $S'' \leftarrow S'$ و $C_{\max}(S'') \leftarrow C_{\max}(S')$. در صورتی که L تکرار از اجرای این گام گذشته است متوقف شوید و با قرار دادن $T \leftarrow \alpha T$ به گام ۱۴ بروید که در آن $0 < \alpha < 1$ و ضریب خنک سازی[1] است. در غیر اینصورت گام ۱۳ را تکرار کنید.

گام ۱۴ (ادامه SA): مادامی که دمای فعلی T بزرگتر از یا مساوی با دمای پایانی T_f است به گام ۱۳ برگردید، در غیر اینصورت متوقف شوید و S^* را با مقدار تابع هدف $C_{\max}(S^*)$ به عنوان بهترین جواب الگوریتم HPSO بگیرید.

در این الگوریتم از رابطه بهنگام‌سازی سرعت و شیوه تبدیل جوابهای JSSP به جوابهای مورد پذیرش PSO ارایه شده در [۱۲] استفاده شده است. الگوریتم برای حالتی توسعه داده شده است که فرض پردازش هر کاری بوسیله همه ماشینها صادق نباشد. همچنین برای جستجوی بهتر در اطراف جوابهای بدست آمده توسط PSO یک الگوریتم LS در داخل آن تعبیه شده است. یک الگوریتم SA نیز برای بهبود جواب نهایی بدست آمده توسط الگوریتم GRASP-PSO/LS بکار گرفته شده است. فلوچارت این الگوریتم با فلوچارت ارایه شده توسط مؤلفین در [۱۲] تفاوتی ندارد اما شبه کد ارایه شده در [۱۲] نیازمند تغییراتی بویژه در گامهای ۱، ۴ و ۵ است.

۴. نتایج محاسباتی

از آنجا که مسایل نمونه مشهور و قابل انطباق بر فروض مساله یافت نشد، به منظور آزمایش عملکرد الگوریتم ارایه شده آن را بر روی تعدادی از مسایلی که با استفاده از مسایل نمونه مشهور موجود در ادبیات مساله کار کارگاهی سنتی تولید شده‌اند اجرا کرده و نتایج را با دو الگوریتـم دیگر مقایسه نموده‌ایم. مسایل نمونه اسـتفاده شده از ادبیات مساله کار کارگاهی سنتی عبـارت‌اند از: FT06، FT10 و FT20 [۱۵] که گاهـی با عنوان MT نیز اسـتفاده می‌شوند و

[1] *Cooling factor*

LA01 تا LA40 [۱۶]. مجموعا ۴۳ مساله انتخاب شده است که تمامی آنها در تارنمای کتابخانه الکترونیکی تحقیق در عملیات ارایه شده توسط بیزلی [۱۷] به آدرس اینترنتی URL: http://people.brunel.ac.uk/~mastjjb/jeb/info.html موجود است. عملکرد الگوریتم HPSO (GRASP-PSO/LS-SA) در مقایسه با دو الگوریتم GRASP-PSO/LS و GRASP-SA بر روی این مسایل سنجیده شده است. هر سه الگوریتم با استفاده از زبان C و در محیط Visual C++ کد شده و برای حل مسایل نمونه بر روی یک دستگاه رایانه 2.4 GHz Celeron (CPU), 512 MB (RAM) اجرا شده است.

۴-۱. مسایل نمونه و تنظیم پارامترها

بمنظور درنظرگرفتن زمانهای آماده‌سازی وابسته به توالی در مسایل نمونه FT 06&10&20 و LA 1-40 ، بصورت تصادفی زمانهای راه‌اندازی و پاکسازی (s_{ij0} و s_{i0j}) در بازه $[0,\frac{p_{\min}+p_{\max}}{2}]$ و سایر زمانهای آماده‌سازی (s_{ijk} , $j,k\neq 0$) در بازه $[0,\frac{p_{\min}+p_{\max}}{3}]$ تولید شده‌اند که در آنها $p_{\max}$ و $p_{\min}$ به ترتیب بیشترین و کمترین زمانهای پردازش کارها هستند. به منظور تنظیم پارامترهای الگوریتم، تعدادی از مسایل از هر یک از این ۲ دسته مساله به عنوان نمونه انتخاب شد و با استفاده از تجزیه تحلیل حساسیت، پارامترها بگونه‌ای تنظیم شدند که بهترین جواب را بدست دهند. پارامترهایی که برای این مسایل بهترین جواب را ارایه می‌دادند بر روی دسته مسایل نظیر آنها به کار گرفته شدند. این مسایل عبارت‌اند از: FT 10 و LA 02&15&21&29&35. بهترین نتایج بدست آمده برای تنظیم پارامترهای الگوریتم‌های GRASP-PSO/LS، GRASP-SA و HPSO عبارتند از: $\gamma=\%65$ ، $w_1=3.3$ ، $w_2=0.3$ ، $w_3=0.7$ ، $w_{\max}=1$ ، $w_{\min}=0.5$ ، $c_1=2.1$ و $c_2=2.1$. مقادیر سایر پارامترها در جدول ۱ آمده است. از آنجا که در الگوریتم HPSO از فرایند SA به منظور جستجوی دقیقتر در پیرامون بهترین جواب یافته شده توسط فرایند GRASP-PSO/LS استفاده می‌شود، دمای اولیه T_0 نسبتا کوچک انتخاب شده است. تنظیم پارامترها برای مسایلی انجام شده است که بصورت خاص در این مقاله تولید شده‌اند، اما از آنجا که مسایل بررسی شده طیف وسیعی از مسایل آسان تا بسیار سخت را در بر می‌گیرد، محققان برای مسایل جدید می‌توانند مقادیر پارامترهای ذکر شده فوق را بکار برند و در مورد پارامترهای ذکر شده در جدول ۱ با توجه به سطح سختی مساله از بازه موجود برای هر پارامتر در این طیف استفاده کنند. بدیهی است برای مسایل جدید ممکن است نیازمند تنظیم دقیقتری برای پارامترها باشیم.

۴-۲. نتایج بدست آمده

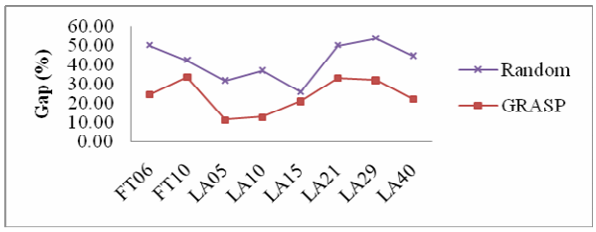
پیشتر گفته شد که از یک تکرار یک الگوریتم GRASP که بر اساس یک ساختار آزمندانه تصادفی عمل می‌کند، در الگوریتم HPSO استفاده شده است. استفاده از این الگوریتم باعث می‌شود که جوابهای اولیه الگوریتم HPSO علاوه بر کیفیت خوب، از پراکندگی کافی نیز برخوردار باشند. از طرف دیگر سرعت الگوریتم نیز بسیار زیاد است و تفاوت زمانی چندانی با تولید تصادفی جوابها ندارد. نتایج حاصل از مقایسه کیفیت جوابهای تولیدی از الگوریتم GRASP با کیفیت جوابهای تولیدی بصورت تصادفی برای تعدادی از مسایل نمونه ساده و سخت، در جدول ۲ نشان داده شده است. هر کدام از مسایل ۵ بار حل شده است و فاصله بین میانگین آنها با بهترین جواب شناخته شده برای حالت بدون زمان آماده سازی بر حسب درصد محاسبه شده است. میانگین فاصله‌ها برای الگوریتم GRASP ۲۴/۰۷٪ است و میانگین فاصله‌ها برای جوابهای تصادفی ۴۲/۱۶٪ است. نسبت این دو میانگین برابر ۰/۵۷ است. به عبارت دیگر کیفیت جواب‌های تولیدی GRASP تقریبا دو برابر بهتر از جوابهای تصادفی است. مقایسه نتایج حاصل از GRASP و جوابهای تصادفی در شکل ۲ نشان داده شده است.

جدول ۱. بهترین مقادیر بدست آمده برای پارامترهای مورد استفاده در الگوریتم‌های GRASP-PSO/LS، GRASP-SA و HPSO

Parameters	FT 6 & 10 & 20			LA 1-10			LA 11-20			LA 21-30 & 36-40			LA 31-35		
	GRASP-PSO/LS	GRASP-SA	HPSO	GRASP-PSO/LS	GRASP-SA	HPSO	GRASP-PSO/LS	GRASP-SA	HPSO	GRASP-PSO/LS	GRASP-SA	HPSO	GRASP-PSO/LS	GRASP-SA	HPSO
$popsize$	30	3000	25	25	2000	20	30	3000	25	45	4000	35	35	1000	30
N_{ls}	100	-	80	100	-	50	100	-	50	150	-	120	100	-	70
N_{pso}	342	-	255	323	-	236	350	-	268	386	-	325	352	-	262
N'_{pso}	140	-	100	100	-	75	110	-	80	180	-	120	120	-	90
N_{pow}	1100	2000	320	620	2000	150	950	2000	200	1500	3000	400	1050	2500	300
T_0	-	300	50	-	200	20	-	350	30	-	400	80	-	500	20
T_f	-	0.1	0.1	-	0.1	0.1	-	0.1	0.1	-	0.05	0.05	-	0.1	0.1
α	-	0.985	0.98	-	0.98	0.98	-	0.987	0.98	-	0.99	0.99	-	0.988	0.985
L	-	2000	500	-	1000	120	-	2000	335	-	3000	850	-	2500	180

جدول ۲. مقایسه نتایج حاصل از الگوریتم GRASP و جوابهای تصادفی

Problem	n	m	BKS	GRASP		Random	
				Average	Gap(%)	Average	Gap(%)
FT06	6	6	55	68.6	24.73	82.6	50.18
FT10	10	10	930	1244	33.76	1326.2	42.60
LA05	10	5	593	662.2	11.67	782.8	32.01
LA10	15	5	958	1083.8	13.13	1315.2	37.29
LA15	20	5	1207	1464.6	21.34	1523.6	26.23
LA21	15	10	1046	1395	33.37	1571.6	50.25
LA29	20	10	1157	1529.4	32.19	1781.4	53.97
LA40	15	15	1222	1495.4	22.37	1769	44.76



شکل ۲. مقایسه نتایج حاصل از الگوریتم GRASP و جوابهای تصادفی

هر یک از مسایل نمونه بدست آمده با هر یک از ۳ الگوریتم ذکر شده ۵ بار حل شده است و بهترین مقدار بدست آمده از تابع هدف در هر مساله به همراه متوسط زمان صرف شده الگوریتم برای رسیدن به بهترین جوابها در جدول ۳ نوشته شده است. در هر مساله کمترین مقدار بدست آمده توسط این ۳ الگوریتم پر رنگ شده است.

در این جدول همچنین جواب بهینه مساله کلاسیک بدون زمانهای آماده‌سازی به همراه جواب یافته شده توسط نرم افزار Lingo 8.0 برای مسایل تولید شده با زمانهای آماده‌سازی در مدت زمان یک ساعت ارایه شده است. برای هر الگوریتمی تعداد مسایلی که آن الگوریتم توانسته است جوابهایی بهتر از دو الگوریتم دیگر بیابد و میانگین زمان اجرای الگوریتم برای همه ۴۳ مساله بکار گرفته شده به همراه میزان متوسط انحرافات برای هر الگوریتم نسبت به جواب یافته شده توسط Lingo در سه سطر در انتهای جدول آورده شده است.

همانطور که مشاهده می‌شود، الگوریتم HPSO برای این مساله بسیار بهتر از دو الگوریتم دیگر عمل می‌کند و کمترین میزان انحراف نسبت به حد بالای یافته شده توسط Lingo را دارد و این جوابهای خوب را در زمانی بسیار کوتاه نسبت به زمان محاسباتی Lingo بدست می‌آورد. HPSO توانسته است برای ۲۹ مساله از ۴۳ مساله جوابهای بهتری نسبت به دو الگوریتم دیگر بیابد که از بین آنها برای مساله FT06 دو الگوریتم HPSO و GRASP-SA جواب یکسانی را یافته‌اند. در مواردی که HPSO موفق نشده است کمترین مقدار را در مقایسه با دو الگوریتم دیگر بیابد، اختلاف چندانی با بهترین مقدار یافته شده ندارد.

کمترین میانگین زمان اجرای الگوریتم مربوط به الگوریتم GRASP-PSO/LS است و بیشترین آن مربوط به الگوریتم GRASP-SA است. این مطلب نشان می‌دهد که هر چه پیچیدگی مساله و ابعاد فضای جواب بیشتر و بزرگتر شوند الگوریتمهای تکاملی از آنجا که فضای جوابها را بصورت موازی مورد جستجو قرار می‌دهند، از نظر زمانی نسبت به الگوریتمهای جستجوی محلی که در هر مرحله از یک جواب به جواب دیگر می‌روند، کارایی بیشتری دارند. هر چند که در برخی موارد ممکن است کیفیت جوابهای حاصل از الگوریتمهای تکاملی کمی پایین‌تر باشد. این مطلب با مقایسه عملکرد این دو الگوریتم در جدول فوق روشنتر می‌شود. لازم به ذکر است که بخش زیادی از زمان صرف شده در الگوریتم GRASP-PSO/LS مربوط به فرایند جستجوی محلی بکار رفته در آن است و فرایند بهینه سازی دسته ذرات بسیار سریع است. با وجود اینکه الگوریتم GRASP-SA از نظر زمانی کارایی پایینتری دارد ولی قابلیت بالای آن در جستجوی دقیق فضای جوابها را نباید از نظر دور داشت که آن را می‌توان در کیفیت خوب جوابهای آن دید که در اغلب موارد فاصله چندانی با جوابهای HPSO ندارد. الگوریتم HPSO در واقع با ترکیب این دو الگوریتم توانسته است مزایای هر دو الگوریتم را در یکجا جمع کند.

جدول ۳.خلاصه نتایج حاصل از سه الگوریتم برای مسایل تولید شده

Prob.	Size			Classical Prob. Opt.	Lingo UB (in 3600 Sec.)	GRASP-PSO/LS		GRASP-SA		HPSO	
	n	×	*m*			Best Cmax	Time	Best Cmax	Time	Best Cmax	Time
FT06	6	×	6	55	66	67	23	**66**	41	**66**	34
FT10	10	×	10	930	1227	1254	46	1271	134	**1252**	57
FT20	20	×	5	1165	1414	1432	87	1479	287	**1428**	98
LA01	10	×	5	666	820	827	29	854	23	**820**	39
LA02	10	×	5	655	817	**803**	32	848	23	817	32
LA03	10	×	5	597	731	**720**	22	769	23	731	33
LA04	10	×	5	590	805	813	19	807	23	**805**	30
LA05	10	×	5	593	699	734	34	707	23	**699**	45
LA06	15	×	5	926	1054	1089	33	1060	51	**1054**	56
LA07	15	×	5	890	1083	1096	41	1134	50	**1083**	64
LA08	15	×	5	863	942	1048	26	1019	51	**1002**	49
LA09	15	×	5	951	1052	1110	91	1065	50	**1063**	114
LA10	15	×	5	958	1055	1087	24	1085	50	**1077**	47
LA11	20	×	5	1222	1363	1398	61	1398	371	**1391**	84
LA12	20	×	5	1039	1196	1239	59	1234	367	**1228**	82
LA13	20	×	5	1150	1310	1401	56	1359	311	**1357**	79
LA14	20	×	5	1292	1343	1429	81	1395	369	**1385**	104
LA15	20	×	5	1207	1462	**1486**	79	1502	231	1492	102
LA16	10	×	10	945	1173	1235	54	**1204**	111	1209	77
LA17	10	×	10	784	991	**1018**	43	1026	110	1043	66
LA18	10	×	10	848	1060	1087	42	1073	109	**1071**	65
LA19	10	×	10	842	1049	1122	49	1136	108	**1096**	72
LA20	10	×	10	902	1116	1201	76	1186	108	**1173**	99
LA21	15	×	10	1046	1372	1474	144	1458	250	**1452**	178
LA22	15	×	10	927	1285	1365	149	1345	544	**1303**	183
LA23	15	×	10	1032	1250	1304	201	**1288**	540	1294	235
LA24	15	×	10	935	1252	1297	231	**1265**	540	1271	265
LA25	15	×	10	977	1246	1351	251	1347	543	**1325**	285
LA26	20	×	10	1218	1499	1634	489	1686	998	**1612**	523
LA27	20	×	10	1235	1627	1689	315	**1668**	1005	1677	349
LA28	20	×	10	1216	1472	1625	300	1622	998	**1583**	334
LA29	20	×	10	1152	1568	**1702**	501	1734	1007	1704	535
LA30	20	×	10	1355	1624	1709	551	**1700**	1002	1701	585
LA31	30	×	10	1784	2139	2199	843	2163	1727	**2161**	877
LA32	30	×	10	1850	2333	2387	609	**2355**	1730	2357	643
LA33	30	×	10	1719	2119	2201	1009	2128	1743	**2119**	1043
LA34	30	×	10	1721	2230	2256	897	2231	1746	**2230**	966
LA35	30	×	10	1888	2343	2385	1103	2369	1724	**2343**	1173
LA36	15	×	15	1268	1640	1792	890	1787	813	**1781**	959
LA37	15	×	15	1397	1741	**1890**	548	1920	821	1903	618
LA38	15	×	15	1196	1564	1701	677	**1677**	809	1684	747
LA39	15	×	15	1233	1563	1723	443	**1675**	805	1679	513
LA40	15	×	15	1222	1494	1659	687	1642	818	**1627**	757
No. of better Cmax						6		9		29	
Average of running time						278		539		309	
Average of relative deviation						4.8		4.4		3.2	

۵. نتیجه گیری و پیشنهادهایی برای مطالعات آتی

در این مقاله مساله زمانبندی کار کارگاهی با زمانهای آماده سازی وابسته به توالی ($Jm \mid s_{ijk} \mid C_{\max}$) مد نظر قرار گرفت که در آن زمانهای آماده سازی وابسته به توالی بصورت جدایی ناپذیر در نظر گرفته شد. مدل ریاضی مساله بصورت یک مدل برنامه ریزی عدد صحیح خطی توسعه داده شد. شیوه نمایش گراف انفصالی برای منطبق شدن بر این مساله اصلاح و تبدیل به یک گراف انفصالی جدید شد که قابلیت نمایش زمانهای آماده‌سازی وابسته به توالی (که شامل زمانهای راه اندازی و پاکسازی نیز است) را دارد.

سپس یک الگوریتم ترکیبی PSO برای حل مساله مورد بررسی توسعه داده شد. از آنجا که الگوریتم PSO برای مسایل با ماهیت پیوسته طراحی شده است، برای تبدیل جوابهای مساله زمانبندی که ماهیت گسسته دارند به جوابهای مورد پذیرش الگوریتم PSO با ماهیت پیوسته و بالعکس، از روش توسعه داده شده توسط مؤلفین در [۱۲] که با استفاده از تبدیل مبنای اعداد و بکارگیری مبنای فاکتوریلی اعداد (سیستم فاکتورادیک) انجام شده بود، استفاده شد. پیچیدگی زمانی این تبدیل برای این مساله نیز چند جمله‌ای ($O(n)$) بوده و کارایی و انطباق‌پذیری بالایی برای آن دارد.

همچنین بدلیل اهمیت کیفیت جوابهای اولیه در الگوریتمهای تکاملی مانند PSO، یک الگوریتم GRASP نیز توسعه داده شد و از یک تکرار آن برای تولید جوابهای (جمعیت) اولیه PSO استفاده شد. برای غلبه بر تمایل الگوریتم تکاملی PSO در به دام افتادن در بهینه‌های محلی نیز از تغییراتی که در رابطه بهنگام‌سازی سرعت ذرات مبتنی بر استفاده همزمان از ضرایب اینرسی (w) و انقباض (χ) در [۱۲] اعمال شد، در حل این مساله هم استفاده شد و نتایج بهتری بدست آمد. همچنین برای جستجوی دقیق‌تر در اطراف بهینه‌های محلی از یک الگوریتم جستجوی محلی (LS) مبتنی بر تعویض جهت کمانهای فصلی موجود بر مسیر بحرانی در هر تکرار الگوریتم و از یک الگوریتم SA در انتهای کار الگوریتم PSO استفاده شد و یک الگوریتم ترکیبی (HPSO) بدست آمد.

نتایج محاسباتی کارایی الگوریتم GRASP ارایه شده را نسبت به جوابهای تصادفی نشان داد. همچنین کارایی الگوریتم پیشنهادی (HPSO) نسبت به دو الگوریتم دیگر و همچنین حد بالای بدست آمده توسط Lingo با توجه به نتایج محاسباتی بر روی مسایل تولید شده به اثبات رسید.

ترکیب سایر الگوریتمهای فرا ابتکاری با الگوریتم PSO به جای الگوریتم SA، همچنین بکارگیری الگوریتم ارایه شده در حل یک مساله واقعی می‌تواند موضوع تحقیقات آتی برای نویسندگان و یا مطالعه کنندگان این مقاله باشد.

مراجع

[۱] بهروزی، مهدی.، عشقی، کوروش.، "بکارگیری الگوریتم ترکیبی بهینه سازی دسته ذرات برای حل مساله سنتی زمانبندی کار کارگاهی"، نشریه بین المللی مهندسی صنایع و مدیریت تولید، جلد ۲۰، شماره ۲، ۱۳۸۸، صفحه ۷۵-۵۷.

[2] Graham, R.L., Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G., *Optimization and Approximation in Deterministic Sequencing and Scheduling: A survey*, Annals of Discrete Mathematics 5, 1979, pp. 287–326.

[3] Garey, M.R., Johnson, D.S., Sethi, R., *The Complexity of Flowshop and Jobshop Scheduling*, Mathematics of Operations Research 1, 1976, pp. 117-129.

[4] Artigues, C., Belmokhtar, S., Feillet, D., *A New Exact Solution Algorithm for the Job Shop Problem with Sequence Dependent Setup Times*, In: Regin, J.C., Rueher, M. (Eds.), 1st International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, Lecture Note in Computer Science, Vol. 3011. Springer, 2004, pp. 37–49.

[5] Zoghby, J., Barnes, J.W., Hasenbein, J.J., *Modeling the Reentrant Job Shop Scheduling Problem with Setups for Metaheuristic Searches*, European Journal of Operational Research 167, 2005, pp. 336–348.

[6] Rossi, A., Dini, G., *Flexible Job-Shop Scheduling with Routing Flexibility and Separable Setup Times using ant Colony Optimisation Method*, Robotics and Computer-Integrated Manufacturing 23, 2007, pp. 503–516.

[7] Mönch, L., Schabacker, R., Pabst, D., Fowler, J.W., *Genetic Algorithm-Based Subproblem Solution Procedures for a Modified Shifting Bottleneck Heuristic for Complex Job Shops*, European Journal of Operational Research 177, 2007, pp. 2100–2118.

[8] Choi, I.C., Choi, D.S., *A Local Search Algorithm for Job Shop Scheduling Problems with Alternative Operations and Sequence-Dependent Setups*, Computers and Industrial Engineering 42, 2002, pp. 43–58.

[9] Vinod, V., Sridharan, R., *Scheduling a Dynamic Job Shop Production System with Sequence-Dependent Setups: An Experimental Study*, Robotics and Computer-Integrated Manufacturing 24, 2008, pp. 435–449.

[10] Sun, X., Noble, J.S., *An approach to job shop scheduling with sequence-dependent setups*, Journal of Manufacturing Systems 18, 1999, pp. 416–430.

[11] Sun, J.U., Yee, S.R., *Job Shop Scheduling with Sequence Dependent Setup Times to Minimize Makespan*, International Journal of Industrial Engineering: Theory Applications and Practice 10, 2003, pp. 455–461.

[12] Cheung, W., Zhou, H., *Using genetic algorithms and heuristics for job shop scheduling with sequence-dependent setup times*, Annals of Operations Research 107, 2001, pp. 65–81.

[13] Cheng, R., Gen, M., Tsujimura, Y., *A Tutorial Survey of Job-Shop Scheduling Problems Using Genetic Algorithms, Part I: Representation*, International Journal of Computers and Industrial Engineering, 30, 1996, pp. 983-997.

[14] Balas, E., *Machine sequencing via disjunctive graphs: An implicit enumeration approach*, Operations Research 17, 1969, pp. 941–957.

[15] Fisher, H., Thompson, G.L., *Probabilistic Learning Combinations of Local Job-Shop Scheduling Rules*, Muth, J.F., Thompson, G.L., (eds.), Industrial Scheduling, Prentice Hall, Englewood Cliffs, New Jersey, 1963.

[16] Lawrance, S., *Resource Constrained Project Scheduling: an Experimental Investigation of Heuristic Scheduling Techniques*, Dissertation, Graduate School of Industrial Administration, Carnegie Mellon University, Pittsburgh, Pennsylvania, 1984.

[17] Beasley, J.E., *OR-Library: Distributing Test Problems by Electronic Mail*, Journal of the Operational Research Society, Vol. 41, No. 11, 1990, pp. 1069–1072.