رویکرد شی گرا برای مساله ی زمان بندی انعطاف پذیر و چند هدفه ی کار کارگاهی

چکیده

کنترل سیستم های تولیدی انعطاف پذیر بسیار پیچیده است و تولید سیستم های کنترل کننده برای این حوزه مساله دشوار است. مساله زمان بندی انعطاف پذیر کار کارگاهی (FJSP) یکی از موارد این حوزه است. این مساله ای است که مشخصه های مساله زمان بندی کار کارگاهی (JSP) را به ارث می برد. FJSP زیر-مساله ی مسیریابی اضافی ای را افزون بر JSP دارد. در زیر-مساله مسیریابی، هر عملیات به یک دستگاه از بین مجموعه ای از دستگاه های فعال تخصیص می یابد. در زیر-مساله برنامه ریزی، توالی عملیات های تخصیص یافته در هنگام بهینه سازی تابع (های) هدف به دست می آید. در این مقاله، رویکرد شی گرا (OO) برای FJSP چند هدفه همراه با الگوریتم بهینه سازی بازپخت شبیه سازی شده ارائه می شود. رویکردهای راه حل در مقالات عموما از طرح کدگذاری دو رشته ای برای بیان این مساله استفاده می کنند. با این حال، متدولوژی تحلیل، طراحی و برنامه نویسی OO به ارائه ی موثر این مساله در طرح کدگذاری کمک می کند، که به ادغام عملی راه حل مساله با سیستم های کنترل تولید که در آن ها اغلب از پارادایم OO استفاده می شود منجر می شود. طراحی FJSP به روش OO از طریق استفاده از دیاگرام کلاس UML حاصل می آید، و این طراحی کدگذاری مساله را به یک ساختار داده ی واحد تقلیل می دهد که در آن شی عملیات FJSP می تواند داده های خود در مورد دستگاه های جایگزین را در ساختار داده ی خودش به صورت سلسله مراتبی حفظ کند. ارتباطات چند-به-چند بین عملیات ها و دستگاه ها از طریق درج یک کلاس جدید بین آنها به دو ارتباط یک-به-چند تبدیل می شوند. کمینه کردن سه تابع هدف زیر در این مقاله در نظر گرفته می شود: حداکثر زمان تکمیل، بار کاری دستگاه دارای بیشترین بار و بار کاری کل همه دستگاه ها. چند مجموعه معیار به منظور نشان دادن اثربخشی رویکرد پیشنهاد شده اجرا می شوند. ثابت می شود که استفاده از رویکرد OO برای FJSP چند هدفه نه تنها به ساخت سیستم های کنترل تولید موثر، بلکه به دستیابی به راه حل های موثر نیز منجر می شود.

کلیدواژه ها: کنترل تولید شی گرا؛ طراحی شی گرا؛ زمان بندی انعطاف پذیر و چند هدفه ی کار کارگاهی؛ الگوریتم بازپخت شبیه سازی شده.

۱. مقدمه

مسائل زمان بندی نقش بسیار مهمی را در سیستم های صنعتی بسیاری ایفا می کنند. این مسائل توجه پژوهشگران بسیاری را جلب می کنند (بایکاسوقلو و همکاران، ۲۰۰۴، بروکر و شلی، ۱۹۹۰، دای و همکاران، ۲۰۱۳، گری و همکاران، ۱۹۷۶، لیو و همکاران، ۲۰۱۴، لو و همکاران، ۲۰۱۴، وانگ و لیو، ۲۰۱۴، شیا و وو، ۲۰۰۵، ژینگونگ و یونگ، ۲۰۱۵ و ژائو و همکاران، ۲۰۱۴). اکثر مسائل زمان بندی مسائل بهینه سازی ترکیبی پیچیده هستند و حل کردن آن ها بسیار دشوار است (شیا و وو، ۲۰۰۵ و زیبالوس، ۲۰۱۰). یکی از مسائل زمان بندی پیچیده JSP است. در کار کارگاهی، هر کار می تواند مسیر پردازش متفاوتی را از درون سیستم داشته باشد، و از این رو مساله زمان بندی بسیار پیچیده می شود (فهمی، بالاکریشنان، و المکاوی، ۲۰۱۱). کار کارگاهی شاخه ای از زمان بندی تولید است. این مساله یک مساله ی NP-سخت معروف است (گری و همکاران، ۱۹۷۶). FJSP مساله ای است که مشخصه های JSP را به ارث می برد. FJSP به دلیل اهمیتش در صنعت توجه پژوهشگران بسیاری را جلب کرده است. FJSP زیر-مساله مسیریابی اضافی را افزون بر JSP دارد. در زیر-مساله مسیریابی، هر عملیات به یک دستگاه از بین مجموعه ای از دستگاه های توانمند تخصیص داده می شود. در زیر-مساله زمان بندی، توالی عملیات های تخصیص یافته در هنگام بهینه سازی تابع (های) هدف به دست می آید. بنابراین، FJSP دو مشکل را ایجاد می دهد؛ تخصیص عملیات و یافتن زمان بندی بهینه ی عملیات (شیا و وو، ۲۰۰۵).

اگر چه الگوریتم جواب بهینه برای JSP کلاسیکی هنوز توسعه نیافته است، روندی در جامعه ی محققان برای مدل سازی و حل نسخه بسیار پیچیده تر JSP وجود دارد. نیاز به مدل سازی و حل FJSP عمدتا به دلیل این واقعیت که ابزارهای دستگاهی مدرن مقدار قابل توجهی قابلیت های همپوشان دارند پدید آمده است (بایکاسوقلو و همکاران، ۲۰۰۴). تعداد مطالعات برای FJSP بیشتر از تعداد مطالعات برای JSP به نظر می رسد. بروکر و شلی

(۱۹۹۰) در میان اولین کسانی بودند که به FJSP پرداختند. آنها یک الگوریتم چند جمله ای را برای حل FJSP با دو کار توسعه دادند. برندیمارته (۱۹۹۳) از یک الگوریتم سلسله مراتبی برای FJSP بر اساس فرا ابتکاری جستجوی ممنوعه استفاده کردند. داوزره-پرز و پائولی (۱۹۹۷) نسخه ی تعمیم یافته ای از مدل گراف تقاطعی را توسعه دادند که قادر به لحاظ کردن این واقعیت است که عملیات ها باید به دستگاه ها برای FJSP تخصیص یابند. ماسترولیلی و گامبردلا (۲۰۰۲) دو تابع همسایگی را همراه با رویه ی جستجوی ممنوعه برای FJSP معرفی کردند.

همچنین زمانی که بیش از یک تابع هدف وجود دارد FJSP مجددا جستجو می شود. بایکاسوقلو و همکاران (۲۰۰۴) رویکرد مدل سازی و راه حل فرا ابتکاری مبتنی بر زبان شناسی را برای FJSP چند هدفه پیشنهاد کردند. شیا و وو (۲۰۰۵) رویکرد بهینه سازی ترکیبی موثر را برای FJSP چند هدفه پیشنهاد کردند. ژانگ، شائو، لی، و گائو (۲۰۰۹) الگوریتم بهینه سازی ازدحام ذرات ترکیبی موثر را برای FJSP چند هدفه پیشنهاد کردند. وانگ ژو، خو، و لیو (۲۰۱۲) الگوریتم کلونی زنبور مصنوعی مبتنی بر پارتو را برای حل موثر FJSP چند هدفه پیشنهاد کردند. لی، پان، و زی (۲۰۱۲) الگوریتم جهش قورباغه ی آمیخته ی ترکیبی را برای راه حل این مساله پیشنهاد کردند. پور و قاسمی شبانکاره (۲۰۱۳) الگوریتم های ژنتیکی ترکیبی و بازپخت شبیه سازی شده (SA) را برای FJSP چند هدفه پیشنهاد کردند. وانگ، وانگ و لیو (۲۰۱۳) الگوریتم توزیع مبتنی بر پارتوی موثر را برای این مساله پیشنهاد کردند.

در سیستم های تولیدی واقعی، تحلیل مساله زمان بندی و توسعه الگوریتم ها و رویه ها تنها بخشی از کار است. این رویه باید در سیستمی جاسازی شود که تصمیم گیرنده را قادر به استفاده ی واقعی از آن می کند. این سیستم باید در سیستم اطلاعات سازمان ادغام شود، که می تواند کاری دشوار باشد (پیندو، ۲۰۱۲) ادغام مدل طراحی با اطلاعات فرآیند و زمان بندی به صورت آنی به منظور افزایش کیفیت محصول، کاهش هزینه، و کوتاه کردن چرخه تولید محصول لازم است (سورماز، آروموگام، هاری هارا، پاتل، و نیروکوندا، ۲۰۱۰). طراحان سیستم به جهت گیری الگوریتم ها و رویه های زمان بندی توسعه یافته نسبت به سیستم اطلاعات تولید واقعی شان نیاز دارند، زیرا سیستم زمان بندی پیوندهای بسیاری با سیستم های مختلف دیگر در سازمان دارد. گاهی اوقات تطبیق الگوریتم

زمان بندی با سیستم اطلاعات تولید، نسبت به توسعه ی الگوریتم یا رویه، بسیار بیشتر طول می کشد. ادغام سیستم های زمان بندی با سیستم اطلاعات جاری برای برنامه نویسان کامپیوتر کار بی اهمیتی نیست.

طراحی سیستم های نرم افزاری فعالیت بسیار پیچیده ای است، زیرا به شدت تحت تاثیر کیفیت مورد نیاز در محصولات نهایی قرار دارد (وازکوئز، آندرس دیاز پیس، و کامپو، ۲۰۱۴). تحلیل، طراحی و برنامه نویسی OO تا الان به مدت ۳۰ سال در دست توسعه بوده است، با این حال در دهه ی گذشته بود که متدولوژی های رشد و توسعه ی سریعی را با استفاده از پیشرفت های توانایی سرعت و پردازش در توسعه ی نرم افزارهای بسیار پیچیده دریافت کرد (مورنو رینا و همکاران، ۲۰۱۲). برنامه نویسی OO روشی است که به شدت ساخت یافته است. این روش شامل ماژولار بودن و نظم منطقی ساختارهای نرم افزاری است (شیزین و منگوانگ، ۲۰۰۰). فناوری های OO پرکاربردترین فناوری برنامه نویسی کامپیوتری در فناوری های اطلاعاتی کنترلی تولیدی امروزی هستند (فرناندز د کانته و همکاران، ۲۰۱۳، کوان، ۲۰۱۱، لیائو و هو، ۲۰۱۱ و لین و همکاران، ۲۰۰۳). بنابراین محققان در حوزه ی زمان بندی دستگاه باید تناسب راه حل شان با فناوری های اطلاعاتی مدرن را در نظر بگیرند.

استفاده از فناوری اطلاعات (IT) یک ممکن کننده ی قابل توجه فرصت هاست (نگای، پنگ، الکساندر، و مون، ۲۰۱۴). سیستم های مدیریت پایگاه داده ی (DBMS) امروزی که در سیستم های کنترل تولید استفاده می شوند عموما بر اساس اصول برنامه نویسی OO ساخته می شوند. آنها از زبان پرس و جوی ساخت یافته (SQL) در هنگام ضبط و بازیابی داده ها استفاده می کنند. Oracle، MySql، Sybase، و MS SQL Server نمونه هایی از DBMS ها هستند که معمولا در سیستم های کنترل تولید استفاده می شوند. DBMS ها در ارتباط با الگوریتم های زمان بندی استفاده می شوند. بنابراین رابطه ی بسیار نزدیکی در میان برنامه نویسی OO، زمان بندی دستگاه و DBMS وجود دارد.

داده های مرتبط با کار و دستگاه را می توان بر اساس متدولوژی برنامه نویسی OO پردازش و در یک DBMS ثبت کرد. این داده ها را می توان در اشیاء کپسوله کرد و می توانند برای اهداف زمان بندی و ثبت / بازیابی داده ها استفاده شوند. شکل ۱ برخی ویژگی ها و رفتارهای اساسی کار معمولی و کلاس دستگاه را در یک سیستم اطلاعات تولید نشان می دهد.

دستگاه	کار
نام	نام
نوع	نوع
قابلیت	مقدار
ساخت	اولویت
کپی	آماده
حذف	موعد
نمایش	ساخت
	کپی
	حذف
	نمایش

شکل ۱. تعریف کلاس کار و دستگاه (اقتباس از پیندو، ۲۰۱۲).

داده های ایستای مربوط به یک کار را می توان در یک شی کار نگاه داشت. به عنوان مثال فهرست عملیات، وزن مرتبط، زمان جابجایی مواد، هزینه حمل و نقل، و موعد، برخی داده های ایستای مربوط به شی یک کار خاص هستند. برخی از رفتارهای اساسی برای یک کار می توانند ساخت، کپی، حذف و نمایش باشند. داده های ایستای مربوط به دستگاه نیز می توانند در شی دستگاه کپسوله شوند؛ فهرست قابلیت ها، سرعت، زمان نگهداری زمان بندی شده و غیره. از سوی دیگر برخی داده های ایستای مربوط به هر دوی کارها و دستگاه ها وجود دارند. به عنوان مثال، زمان راه اندازی وابسته به توالی می تواند به عنوان داده های ایستای مربوط به هر دوی دستگاه ها و

کارها در نظر گرفته شود. این انواع داده ها به عنوان ایستا طبقه بندی می شوند، زیرا به زمان بندی تولید شده بستگی ندارند.

در FJSP، عملیات های کارها دستگاه های جایگزینی را برای پردازش شدن دارند. داده های مربوط به دستگاه جایگزین به داده های هر دوی عملیات های دستگاه و کار وابسته هستند. بنابراین FJSP شامل یک ارتباط اضافی دستگاه-کار است. تحقیقات ارائه شده در مقالات عموما این ارتباط را به دو زیر مساله تقسیم بندی می کنند. زیر-مساله اول تخصیص عملیات ها به یک دستگاه از بین چند دستگاه در دسترس است. زیر-مساله دوم توالی بندی تمام عملیات ها بر روی هر دستگاه است (شائو، لیو، لیو، و ژانگ، ۲۰۱۳).

اگرچه متدولوژی برنامه نویسی OO و مساله FJSP برای محققان جدید نیستند، اما طراحی FJSP توسط پارادایم OO برای هر دوی کارورزان علم و توسعه دهندگان الگوریتم جدید به نظر می رسد. به نظر می رسد که شکافی بین توسعه دهندگان نرم افزار سیستم های کنترل تولید و توسعه دهندگان الگوریتم نظریه زمان بندی وجود دارد، که قبلا گفتیم پر کردن این شکاف کار دشواری است (پیندو، ۲۰۱۲). این تحقیق قرار است که این شکاف را پر کند و پلی بین توسعه دهندگان نرم افزار و توسعه دهندگان الگوریتم باشد.

در این مقاله، ارتباط بین دستگاه و عملیات ها با تعریف یک کلاس جدید بازنمایی می شود. با انجام این کار، نمایش راه حل دو فازی FJSP چند هدفه می تواند به راه حلی تبدیل شود که به ساختار راه حل هوشمند و موثر منجر می شود. ارائه ی موجودیت های سیستم و ارتباطات شان بر روی اشیاء نرم افزار قرار است که پیچیدگی سیستم را بدون از دست دادن کیفیت راه حل کاهش دهد. الگوریتم SA برای رویکرد OO پیشنهاد شده استفاده می شود.

ساختار ادامه این مقاله به شرح زیر است: بخش ۲ تعریف مساله برای FJSP چند هدفه را ارائه می دهد. بخش ۳ طراحی OO برای FJSP را ارائه می دهد. بخش ۴ اعمال الگوریتم SA به رویکرد پیشنهاد شده را ارائه می دهد. بخش ۵ چند مساله تست و راه حل های شان را در رویکرد پیشنهاد شده ارائه می دهد و بخش ۶ این مقاله را خلاصه می کند.

۲. تعریف مساله

FJSP یکی از شایع ترین مسائل زمان بندی تولید در شرکت های تولیدی است. FJSP تعمیمِ JSP کلاسیکی است که در آن عملیات ها مجازند که بر روی هر کدام از دستگاه های مجموعه ای از دستگاه های در دسترس پردازش شوند. بنابراین، FJSP دشوارتر از JSP کلاسیکی است، زیرا یک سطح تصمیم بیشتر را علاوه بر سطح توالی وارد می کند، یعنی مسیرهای کار. تعیین مسیرهای کاری یعنی تعیین اینکه برای هر عملیات، چه دستگاهی، از بین دستگاه های در دسترس، باید آن را پردازش کند (پزلا، مورگانتی، و کیاسچتی، ۲۰۰۸).

FJSP را می توان به شرح زیر تعریف کرد. مجموعه ای از n کار و مجموعه ای از m دستگاه وجود دارد. M نشان دهنده ی مجموعه ی تمام دستگاه هاست. هر کار i از دنباله ای از ni عملیات تشکیل شده است. هر عملیات $O_{i,j}$ $(i = 1, 2, \dots, n; j = 1, 2, \dots, n_i)$ کار i باید در یک دستگاه M_k از بین مجموعه ای از دستگاه های سازگار $M_{i,j}$ (برای $M_k \in M_{i,j}$، $M_{ij} \subseteq M$) پردازش شود. اهداف در FJSP چند هدفه به عنوان کمینه سازی معیارهای زیر داده می شوند:

- حداکثر زمان تکمیل دستگاه ها (بازه تولید)
- حداکثر بار کاری دستگاه.
- بار کاری کل دستگاه ها.

مفروضات زیر برای FJSP چند هدفه ایجاد می شوند:

- هر عملیات نمی تواند قطع شود.
- هر دستگاه می تواند حداکثر یک عملیات را در هر زمان انجام دهد.
- دستگاه ها زمان راه اندازی ندارند.
- زمان جابجایی بین عملیات ها وجود ندارد.

نمادهای استفاده شده در تعریف FJSP چند هدفه به شرح زیرند (ژانگ و همکاران، ۲۰۰۹):

n: تعداد کل کارها؛

m: تعداد کل دستگاه ها؛

ni: تعداد کل عملیات های کار i؛

$O_{i,j}$: j امین عملیات کار i؛

$M_{i,j}$؛ مجموعه دستگاه های در دسترس برای عملیات $O_{i,j}$؛

P_{ijk}؛ زمان پردازش $O_{i,j}$ بر روی دستگاه k؛

t_{ijk}؛ زمان شروع عملیات $O_{i,j}$ بر روی دستگاه k؛

$C_{i,j}$؛ زمان تکمیل عملیات $O_{i,j}$؛

h ،i:

اندیس کارها، $i,h = 1, 2, \dots , n$؛

k؛ اندیس دستگاه ها، $k = 1, 2, \dots , m$؛

j,g؛ اندیس توالی عملیات، $j, g = 1, 2, \dots , n_i$؛

C_k؛ زمان تکمیل M_k،

W_k؛ بار کاری M_k؛

X_{ijk} = ۱، اگر دستگاه k برای عملیات Oij انتخاب شود

۰، در غیر این صورت

مدل ریاضی را می توان به صورت زیر ارائه داد (ژانگ و همکاران، ۲۰۰۹):

$$min\ f_1 = \max_{1\le k\le m}(C_k) \tag{1}$$

$$min\ f_2 = \max_{1\le k\le m}(W_k) \tag{2}$$

$$min\ f_3 = \sum_{k=1}^{m} W_k \tag{3}$$

s.t.

$$C_{ij} - C_{i(j-1)} \ge P_{ijk}x_{ijk},\ j = 2,\ldots,n_i;\ \ \forall i, j \tag{4}$$

$$[(C_{hg} - C_{ij} - t_{hgk})x_{hgk}x_{ijk} \ge 0] \vee [(C_{ij} - C_{hg} - t_{ijk})x_{hgj}x_{ijk} \ge 0], \forall (i,j),(h,g),k \tag{5}$$

$$\sum\nolimits_{k\in M_{ij}} x_{ijk} = 1,\ \forall i, j \tag{6}$$

نامعادله (۴) قید تقدم عملیات ها را تضمین می کند. نامعادله (۵) تضمین می کند که هر دستگاه تنها می تواند یک عملیات را در هر زمان پردازش کند. معادله (۶) بیان می کند که یک دستگاه را می توان از مجموعه ی دستگاه های در دسترس برای هر عملیات انتخاب کرد (ژانگ و همکاران، ۲۰۰۹).

۳. نمایش راه حل FJSP

در FJSP، عملیات های کارها دستگاه های جایگزینی را برای پردازش شدن دارند. داده های مربوط به دستگاه جایگزین به داده های هر دوی عملیات های دستگاه و کار وابسته هستند. بنابراین FJSP شامل یک ارتباط اضافی دستگاه-کار است. تحقیقات ارائه شده در مقالات عموما این ارتباط را به دو زیر مساله تقسیم بندی می کنند. زیر-مساله اول تخصیص عملیات ها به یک دستگاه از بین چند دستگاه در دسترس است. زیر-مساله دوم توالی بندی

تمام عملیات ها بر روی هر دستگاه است (شائو و همکاران، ۲۰۱۳). این دو زیر-مساله عموما با نمایش مساله با دو کدگذاری مورد رسیدگی قرار می گیرند. نمایش راه حل پیشین در بخش بعد، قبل از نمایش طراحی FJSP با رویکرد OO، ارائه می دهد. بنابراین تفاوت بین متدولوژی های طراحی می تواند به راحتی تشخیص داده شود.

۳,۱. نمایش راه حل FJSP پیشین

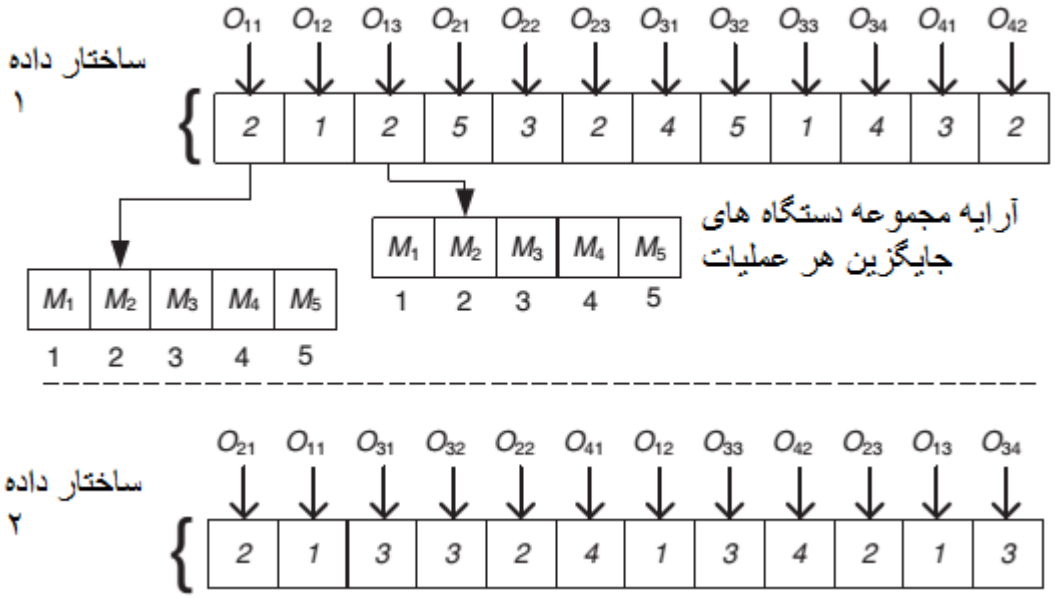
طراحی شی گرای FJSP در تحقیقات اولیه ی این حوزه مساله وجود ندارد. تحقیقات عموما بر موفقیت محاسباتی الگوریتم های خود، بدون در نظر گرفتن راندمان طراحی شان، تمرکز می کنند. در تحقیقات اولیه، این مساله در جایی نمایش داده می شد که ترمیم راه حل مورد نیاز بود. به عنوان مثال، مسقونی، حمادی، و بورن (۱۹۹۷) از نمایش دستگاه موازی و کار موازی استفاده کردند. آنها از نمایش ماتریسی استفاده کردند که در آن هر سطر از مجموعه ای از عملیات های مرتب تشکیل می شود. عیب این نمایش ضرورت مکانیسم ترمیم بر روی نمایش راه حل بود. از آنجا که این مساله دو زیر-مساله دارد، دو ساختار داده ی مستقل در تحقیقات بعدی به منظور اجتناب از کاربرد مکانیسم ترمیمی ترجیح داده می شوند. به عنوان مثال، ژانگ و همکاران (۲۰۰۹) و شائو و همکاران (۲۰۱۳) از دو ساختار داده مستقل به منظور نمایش FJSP استفاده کردند. اولین ساختار داده برای نمایش دستگاه ها و دومین ساختار داده برای نمایش عملیات ها در این مساله استفاده می شود. اولین ساختار داده آرایه ای از مقادیر صحیح است. مقادیر صحیح با اندیس آرایه ی مجموعه دستگاه های جایگزین $M_{i,j}$ هر عملیات $O_{i,j}$ برابر هستند، و این مقادیر صحیح بین ۱ و n_i هستند که i اندیس کارهاست. طول آرایه برابر است با مجموع تمام عملیات های همه کارها. دومین ساختار داده طول برابری با اولین ساختار داده دارد. دومین ساختار داده از دنباله ای از اعداد کار تشکیل می شود که در آن شماره کار i، ni بار رخ می دهد (دنباله ی عملیات ها بر روی هر دستگاه) (ژانگ و همکاران، ۲۰۰۹). برخی محققان بردار تخصیص دستگاه و بردار جایگشت عملیات را برای این دو ساختار داده ترجیح دادند (شائو و همکاران، ۲۰۱۳).

جدول ۱ چهار کار را نشان می دهد که می توانند در پنج دستگاه پردازش شوند. نمایش راه حل با استفاده از دو ساختار داده مستقل، که در مقالات در دسترس است، در شکل ۲ داده شده است. طول این دو ساختار داده ۱۲ است. ساختار داده ۱ اطلاعات انتخاب دستگاه را برای عملیات های کارها نگه می دارد. به عنوان مثال، دستگاه ۲ به منظور پردازش O_{11} و O_{13} انتخاب می شود. این تصمیم در ساختار داده ۱ نگه داشته می شود. ساختار داده ۲ نیز طول همان ساختار داده ۱ را داراست. این ساختار داده از دنباله ای از شماره های کار تشکیل شده است که در آن شماره کار ni i بار رخ می دهد. این کار می تواند از ایجاد زمان بندی غیر ممکن در هنگام تعویض هر عملیات با اندیس کار متناظر جلوگیری کند. ساختار داده ۲ که به صورت ۳-۱-۲-۴-۱-۳-۴-۲-۳-۳-۲-۱-۲ داده شده است می تواند به عنوان فهرستی از عملیات های O_{21}- O_{11}- O_{31}- O_{32}- O_{22}- O_{41}- O_{12}- O_{33}- O_{42}- O_{23}- O_{13}- O_{34} تفسیر شود. هنگامی که یک راه حل کدگشایی می شود، ابتدا ساختار داده ۲ به دنباله ای از عملیات ها تبدیل می شود. سپس هر عملیات مطابق با ساختار داده ۱ به یک دستگاه پردازش کننده تخصیص می یابد (ژانگ و همکاران، ۲۰۰۹).

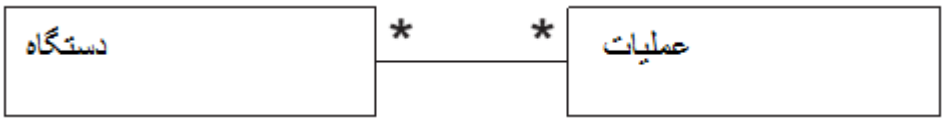
جدول ۱. مساله ۴ × ۵.

		M_1	M_2	M_3	M_4	M_5
کار ۱	$O_{1,1}$	۲	۵	۴	۱	۲
	$O_{1,2}$	۵	۴	۵	۷	۵
	$O_{1,3}$	۴	۵	۵	۴	۵
کار ۲	$O_{2,1}$	۲	۵	۴	۷	۸
	$O_{2,2}$	۵	۶	۹	۸	۵
	$O_{2,3}$	۴	۵	۴	۵۴	۵
کار ۳	$O_{3,1}$	۹	۸	۶	۷	۹
	$O_{3,2}$	۶	۱	۲	۵	۴

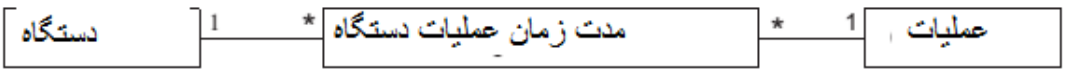
	M_1	M_2	M_3	M_4	M_5	
$O_{3,3}$	۲	۵	۴	۲	۴	
$O_{3,4}$	۴	۵	۲	۱	۵	
$O_{4,1}$	۱	۵	۲	۴	۱۲	۴ کار
$O_{4,2}$	۵	۱	۲	۱	۲	



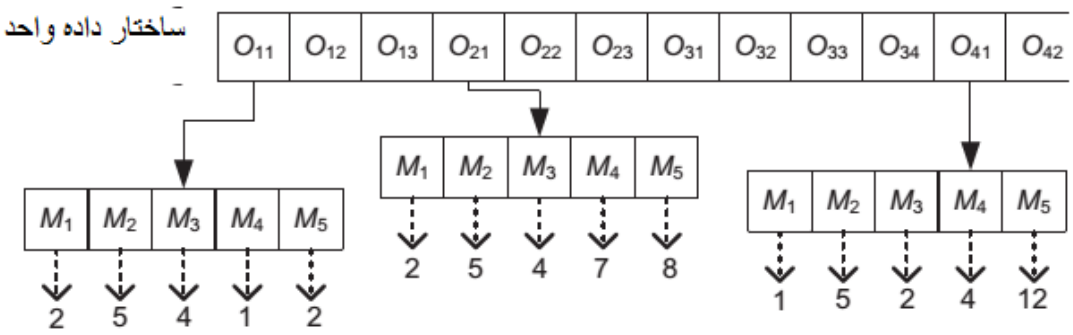
شکل ۲. نمایش راه حل در مقالات (بر گرفته از ژانگ و همکاران، ۲۰۰۹).



شکل ۳. ارتباط بین دستگاه و عملیات.



شکل ۴. ارتباط بین دستگاه و عملیات.



شکل ۵. ساختار داده تحت ارتباطات چند-به-چند.

همانطور که از شکل ۲ دیده می شود، هر راه حل خاص می تواند پس از سپری کردن عملیات کدگشایی دو مرحله ای حاصل آید.

۳,۲. نمایش راه حل شیء گرای FJSP

در رشته مهندسی نرم افزار، رویکردهای شی گرا (OO) روشی قدرتمند در خصوص ساختارهای پیچیده ی تجرید و مدل سازی ثابت شده اند. این رویکردها برای پروژه های بسیاری استفاده شده اند (بغینی و همکاران، ۲۰۱۴، مکی، ۲۰۱۳ و وانگ و همکاران، ۲۰۱۴). متدولوژی برنامه نویسی OO رویکردی بسیار کارآمد برای توسعه ی نرم افزارهایی است که در آن ها تعاملات پیچیده وجود دارد. این متدولوژی شامل ماژولار بودن و نظم منطقی ساختارهای نرم افزار است، و اجازه ی بازکاربردپذیری را می دهد (شیزین و منگوانگ، ۲۰۰۰). متدولوژی OO فرصت های مدل سازی سلسله مراتبی را فراهم می کند.

زبان مدلسازی یکپارچه (UML)، با استفاده از متدولوژی های طراحی OO، می تواند بازکاربردپذیری، توسعه پذیری و اصلاح پذیری را در طراحی نرم افزار فراهم کند (بروکولری، دیگا، و پرونه، ۲۰۰۳). کمک اصلی UML این است که بر روی شکافی که بین حوزه تحلیل و طراحی OO و حوزه برنامه نویسی OO وجود دارد، با ایجاد فرا مدل یکپارچه ای از مفاهیم OO، پل می زند (ون هیلگرسبرگ و کومار، ۱۹۹۹).

این مقاله نمایش شی گرای FJSP با استفاده از دیاگرام کلاس UML را پیشنهاد می دهد و کدگذاری مساله را به یک ساختار داده ی واحد تقلیل می دهد که شی عملیات می تواند داده های خود در مورد دستگاه های جایگزین را در آن نگاه دارد. رویکرد OO پیشنهاد شده می تواند راه حل های عملی را که عملا شبیه به طرح واره های کدگذاری پیشین هستند تولید کند. ارتباط بین دستگاه و عملیات را می توان با یک ارتباط چند-به-چند نمایش داد. ارتباط بین دستگاه و عملیات در FJSP در شکل ۳ ارائه شده است.

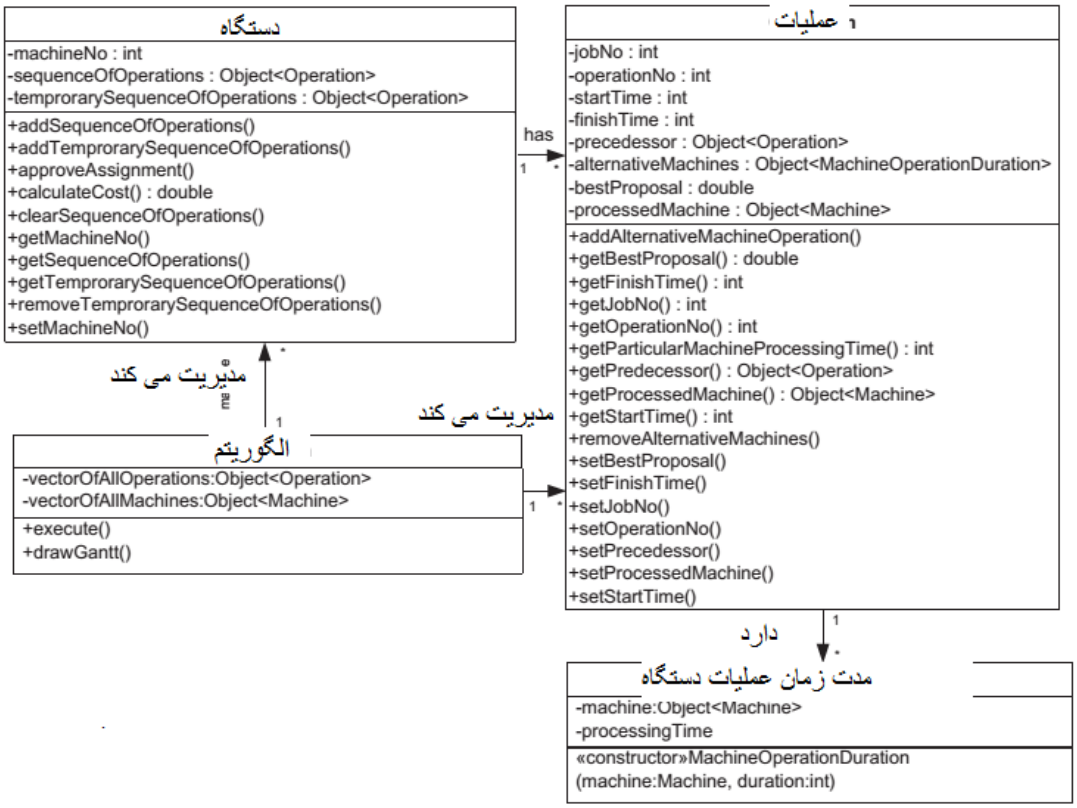
ارتباطات چند-به-چند را می توان با قرار دادن یک کلاس جدید بین دو کلاس مرتبط چند-به-چند، به دو ارتباط یک-به-چند تبدیل کرد. زیرا یک کلاس خاص برای رسیدگی به FJSP می تواند در بین کلاس های دستگاه و عملیات قرار داده شود. کلاس مدت زمان عملیات دستگاه برای رسیدگی به مساله زمان پردازش دستگاه جایگزین در FJSP درج می شود. شکل ۴ دیاگرام ارتباط به روز شده را نشان می دهد. در این دیاگرام ارتباط جدید، دو ارتباط یک-به-چند وجود دارند.

شکل ۴ حاکی از آن است که ممکن است چند مدت زمان عملیات برای یک دستگاه خاص وجود داشته باشد و ممکن است چند مدت زمان عملیات برای یک عملیات خاص وجود داشته باشد. در نتیجه انعطاف پذیری مسیر و جایگشت عملیات ها در یک کلاس واحد نمایش داده می شوند. این استنتاج کاملا با ساختار مساله FJSP مطابق است. در این نمایش مساله، هر عنصر در FJSP با یک کلاس نشان داده می شود. فیلدهای مربوط به عناصر مساله را می توان تا حد لزوم تعریف کرد.

جدول ۱ را می توان برای نشان دادن ارتباط چند-به-چند در طراحی OO استفاده کرد. اشیاء عملیات ها، داده های مربوط به دستگاه های جایگزین و مدت زمان های عملیات مربوطه شان را با استفاده از کلاس مدت زمان عملیات دستگاه نگاه می دارند. شکل ۵ تحقق ارتباط چند به چند را ارائه می دهد.

بر خلاف ساختار داده دودویی در نمایش های راه حل پیشین، شکل ۵ یک ساختار داده واحد برای نمایش FJSP دارد. هر عملیات در جدول ۱ فهرستی از دستگاه های جایگزین دارد و هر دستگاه فهرستی از عملیات ها را داراست (ستون های جدول ۱).

اکنون اطلاعات مربوط به مجموعه دستگاه های جایگزین $M_{i,j}$ هر عملیات $O_{i,j}$ در یک ساختار داده واحد به صورت سلسله مراتبی نگه داشته می شوند. مجددا، طول این آرایه برابر است با مجموع تمام عملیات های همه ی کارها. ساختار داده در شکل ۵ که به صورت O_{11}- O_{12}- O_{13}- O_{21}- O_{22}- O_{23}- O_{31}- O_{32}- O_{33}- O_{34}- O_{41}-O_{42} داده شده است، می تواند به عنوان فهرستی از عملیات هایی که اطلاعات تخصیص دستگاه خودشان را دارند تفسیر شود. به عنوان مثال، شی O_{11} به دستگاه ۳ و O_{41} به دستگاه ۴ تخصیص می یابد. در این مورد، این مساله تنها با فهرستی از عملیات ها (جایگشت عملیات) نمایش داده می شود. این نمایش راه حل سلسله مراتبی برنامه نویسان را قادر به تولید راه حل های جدید صرفا با دستکاری در یک ساختار داده ی واحد می کند.



شکل ۶. دیاگرام کلاس UML رویکرد پیشنهاد شده.

۳,۲,۱. دیاگرام کلاس FJSP

دیاگرام های کلاس طراحان سیستم را قادر به مشخص کردن روابط ساختاری بین عناصر سیستم می کنند. این کلاس ها بر اساس تحلیل سیستم پیشین تعیین می شوند که نمایش راه حل FJSP را به یک ساختار داده واحد تقلیل داد. شکل ۶ دیاگرام کلاس UML را برای طراحی OO پیشنهاد شده نشان می دهد. سه کلاس وجود دارند که حوزه ی FJSP را نمایش می دهند، که عبارتند از دستگاه، عملیات و مدت زمان عملیات دستگاه. کلاس مدت زمان عملیات دستگاه به منظور رسیدگی به ارتباط بین دستگاه و عملیات ها استفاده می شود، که در آن ها یک عملیات را می توان در دستگاه های جایگزین پردازش کرد.

شی عملیات، مرجع دستگاه های جایگزین را دارد که داده های دستگاه های جایگزین را برای یک عملیات خاص نگاه می دارد. این مرجع از اشیاء مدت زمان عملیات دستگاه که داده های انعطاف پذیری را از جانب شی عملیات نگاه می دارند تشکیل شده است. هر شی مدت زمان عملیات دستگاه خصیصه دستگاه و زمان پردازش را داراست. پیوند دستگاه - زمان پردازش در این نقطه از طراحی انجام می شود. نوع خصیصه دستگاه کلاس مدت زمان عملیات دستگاه، دستگاه است، همانطور که می توان از شکل ۶ دید.

در هنگام تعریف FJSP، اشیاء عملیات ها بر اساس مجموعه مساله داده شده ایجاد می شوند. در هنگام بارگذاری FJSP در حافظه کامپیوتر، سازنده کلاس مدت زمان عملیات دستگاه ها و زمان های پردازش رابطه ای را با اتصال این اطلاعات با استفاده از سازنده اش مقداردهی اولیه می کند. این سازنده دو پارامتر دارد که عبارتند از دستگاه و مدت زمان. آرگومان های این پارامترها از یک مجموعه مساله به دست می آیند (مانند مجموعه ی داده شده در جدول ۱). تعداد دستگاه های جایگزین می تواند برای اشیاء عملیات مختلف متفاوت باشد. تنوع تعداد عملیات توسط سازنده کلاس مدت زمان عملیات دستگاه مورد رسیدگی قرار می گیرد.

شی دستگاه مرجعی را برای عملیات های دارای توالی بر روی فهرست خصیصه های آن دارد. خصیصه ی توالی عملیات های کلاس دستگاه وجود دارد که عملیات های دارای توالی را نگه می دارد. خصیصه توالی عملیات ها داده

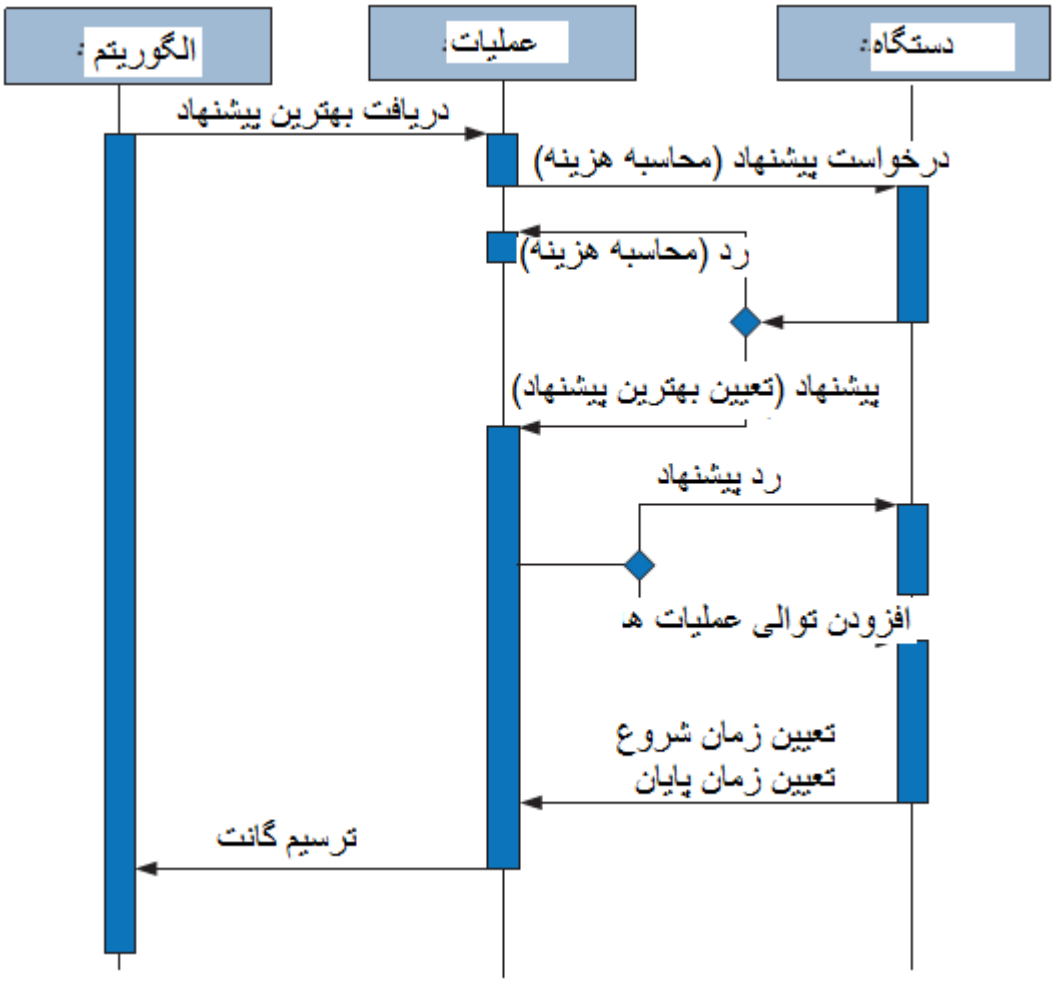
های مربوط به عملیات ها را از نظر نوع عملیات نگاه می دارد. کلاس دستگاه توالی موقتی عملیات ها را نیز دارد که اطلاعات توالی عملیات را به طور موقت به منظور استفاده در الگوریتم راه حل نگاه می دارد.

کلاس های عملیات و دستگاه یک پروتکل تعاملی دارند که در آن اشیاء عملیات به منظور یافتن دستگاه مناسب برای زمان بندی شدن درخواست پیشنهاد می دهند. تعامل بین اشیاء عملیات و دستگاه بر اساس "درخواست پیشنهاد (cfp)" شی عملیات به شی دستگاه است. شی عملیات با استفاده از روش محاسبه هزینه () اشیاء دستگاه، یک cfp را به اشیاء دستگاه، که در سیستم در دسترس هستند، ارسال می کند. شی عملیات از خصیصه دستگاه های جایگزین خود برای یافتن دستگاه های در دسترس استفاده می کند. اشیاء دستگاه نتیجه ای را بر اساس اهداف تعریف شده در کلاس الگوریتم بر می گردانند. کلاس الگوریتم روش اجرا () را دارد که شامل جزئیات الگوریتم مورد استفاده است. کلاس الگوریتم دارای خصیصه های بردار تمام عملیات ها و بردار تمام دستگاه ها است که تمام اطلاعات مربوط به دستگاه ها و عملیات ها را نگاه می دارد. جزئیات مربوط به الگوریتم مورد استفاده برای FJSP در بخش ۴ ارائه می شوند.

۳,۲,۲. دیاگرام تعامل

دیاگرام تعامل همکاری میان نمونه های سیستم را مدل سازی می کند. شکل ۷ دیاگرام تعامل بین نمونه های دستگاه، عملیات و الگوریتم را نشان می دهد. دیاگرام تعامل پس از فراخوانی روش اجرا () شروع می شود. هنگامی که یک عملیات توسط این الگوریتم در نظر گرفته می شود، آنگاه روش محاسبه هزینه () بر روی نمونه خاص دستگاه فراخوانی می شود. همانطور که در شکل ۷ ارائه شده است، نمونه عملیات درخواست پیشنهاد به موارد دستگاه را که به صورت یک فهرست ارائه شده اند می دهد. نمونه های دستگاه هزینه را مطابق با الگوریتم ارائه شده توسط کلاس Algoritm محاسبه می کند. نمونه عملیات یکی از پاسخ های دستگاه ها را می پذیرد. سپس تخصیص عملیات به نمونه دستگاه با فراخوانی روش تعیین بهترین پیشنهاد () نمونه عملیات انجام می شود. نمونه های دستگاه که مورد پذیرش نمونه های عملیات قرار نمی گیرند رد می شوند. نمونه دستگاه انتخاب شده از این

تخصیص با فراخوانی روش افزودن توالی عملیات ها () مطلع می کند. زمان شروع و زمان پایان نمونه های عملیات به ترتیب با فراخوانی روش های تعیین زمان شروع () و تعیین زمان پایان () تعیین می شوند. هنگامی که تمام عملیات ها به یک دستگاه تخصیص یافتند، آنگاه روش ترسیم گانت () نمونه الگوریتم توسط آخرین نمونه عملیات فراخوانی می شود.



شکل ۷. دیاگرام تعامل نمونه.

۳،۲،۳. طراحی OO و اثرش بر سیستم های کنترل تولید

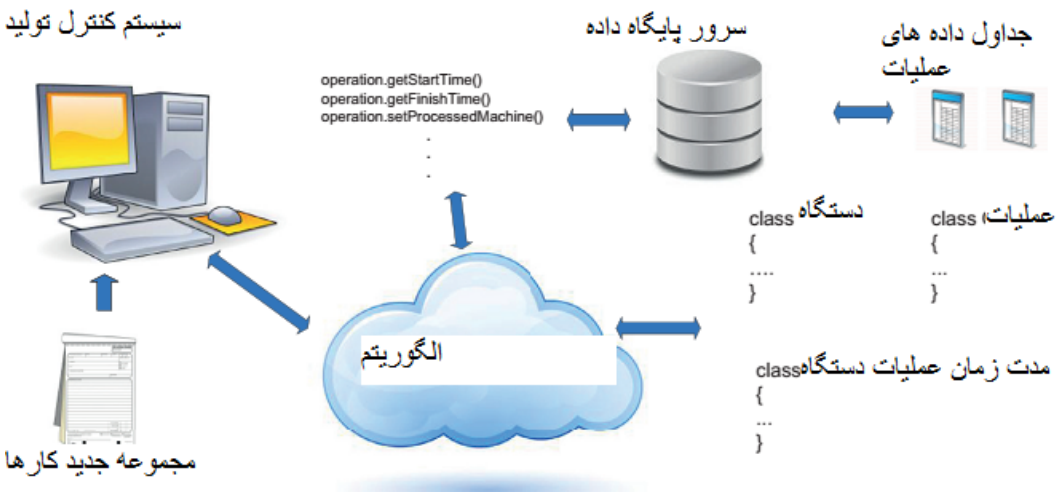
از آنجا که FJSP یکی از شایع ترین مسائل زمان بندی تولید در شرکت های تولیدی است راه حل آن در سیستم کنترل تولید نقش بسیار مهمی را برای توسعه دهندگان سیستم و نرم افزار ایفا می کند. سیستم های کنترل تولید

واقعی مستلزم اقدام برای ادغام هر الگوریتم توسعه یافته هستند. ادغام الگوریتم توسعه یافته در سیستم اطلاعاتی می تواند کار دشواری باشد (پیندو، ۲۰۱۲). توسعه دهندگان سیستم عموما به ادغام یکپارچه ای نیاز دارند که در آن الگوریتم راه حل را می توان به راحتی با سیستم اطلاعاتی بدون ایجاد هر گونه لایه نرم افزاری میانجی یکپارچه کرد.

طراحی FJSP با رویکرد OO قرار است که به توسعه دهندگان نرم افزارها برای ادغام راحت هر الگوریتم توسعه یافته با سیستم کنترل تولید کمک کند. کلاس الگوریتم می تواند فهرست دستگاه ها و عملیات هایی را که برای ضبط / بازیابی اطلاعات دستگاه و کاری به / از DBMS استفاده می شوند نگاه دارد.

DBMS های امروزی که در سیستم های کنترل تولید استفاده می شوند عموما بر اساس اصول برنامه نویسی OO ساخته می شوند. رابطه ای بسیار نزدیک بین برنامه نویسی OO و DBMS وجود دارد. از آنجا که فناوری های OO پرکاربردترین فناوری برنامه نویسی کامپیوتری در سیستم های اطلاعاتی امروزی است، نمایش راه حل FJSP با رویکرد OO می تواند ادغام الگوریتم با آخرین فناوری های اطلاعاتی را برای برنامه نویسان آسان کند.

شکل ۸ طراحی OO پیشنهاد شده را در ارتباط با سیستم اطلاعات تولیدی نشان می دهد. این سیستم اشیاء عملیات و دستگاه را پس از دریافت سفارش تولید می کند. الگوریتم برای این مساله از این کلاس ها به منظور حل این مساله استفاده می کند. پس از اینکه این مساله توسط این الگوریتم حل شد، داده های عملیات (برای مثال، شماره کار، شماره عملیات، زمان شروع، زمان پایان، تقدم و غیره) تنها با فراخوانی روش های اشیاء عملیات در پایگاه داده ثبت می شوند. نیازی به برنامه ی میانجی که نتیجه ی الگوریتم را به جداول پایگاه داده تبدیل می کند وجود ندارد.



شکل ۸. طراحی OO و سیستم اطلاعات تولید پیشنهادی.

مزیت جانبی دیگر طراحی FJSP با رویکرد OO کاربرد بالقوه ی آن در سیستم های خبره و هوشمند است. زیرساخت های فراهم شده توسط طراحی OO به ادغام برخی الگوریتم ها و ابزارهای نرم افزاری برای پشتیبانی از سیستم های کنترل تولید کمک می کنند. به عنوان مثال، تحقیقات چند عامله ی بسیاری بر اساس اصل وا تزویج داده ها، که در آن عناصر سیستم با استفاده از اعضای مستقل نمایش داده می شوند، ایجاد شده اند، همانطور که می توان در مقالات بسیار یافت (نارایاناسوامی و رانگاراج، ۲۰۱۵، کوئلان و همکاران، ۲۰۱۵ و اسونسون و دنیلسون، ۲۰۱۵). اصطلاح تزویج متضاد اصطلاح وا تزویج است و توسط پادغام و وینیکاف (۲۰۱۴) به این صورت تعریف شده است؛ "تزویج یک ویژگی گروهی از اجزاست. تزویج اجزا یعنی اینکه تا چه حد آنها به یکدیگر وابسته هستند. اگر جزء A به جزء B وابسته باشد، آنگاه تغییر B ممکن است مستلزم تغییر A نیز باشد. تزویج بالا سیستمی را حاصل می دهد که در آن تغییر یک جزء احتمالا مستلزم تغییرات در بسیاری از اجزاء دیگر است". بنابراین تزویج داده ها ممکن است به برخی نتایج نامطلوب در طراحی سیستم کنترل تولیدی که در آن الگوریتم های بهینه سازی مختلفی ممکن است قبل از دستیابی به سیستم کنترل خوب امتحان شوند، منجر شود. قابلیت نصب و اجرای رویکرد OO برای تبدیلات نرم و یکپارچه ی الگوریتم های مختلف مناسب به نظر می رسد.

۴. الگوریتم بازپخت شبیه سازی شده برای FJSP شی گرا

طراحی FJSP در رویکرد OO راه را برای نصب و اجرای الگوریتم های مختلف بسیاری برای راه حل مساله هموار می کند. با استفاده از این رویکرد، توسعه دهندگان سیستم و نرم افزار می توانند بدون تغییر ساختار کل سیستم، الگوریتم جدیدی را در سیستم کنترل تولید جاری خود نصب کنند. در مقالات، عموما محققان الگوریتم های ترکیبی را به دلیل مشخصه ی مسیریابی و توالی بندی FJSP، به منظور حل این مساله ترجیح می دهند. با این حال، در این مقاله، یک الگوریتم واحد به عنوان الگوریتم راه حل به منظور نشان دادن اثربخشی رویکرد OO پیشنهاد شده ترجیح داده می شود که در آن نمایش راه حل صرفا با استفاده از یک بردار راه حل واحد صورت می گیرد.

الگوریتم بازپخت شبیه سازی شده (SA) از بدو معرفی اش بر اساس تلاش های مشترک کرک پاتریک، گلات و وچی (۱۹۸۳) نتایج خوبی را برای مسائل بهینه سازی ترکیبی بزرگ تولید کرده است. SA به ویژه در مسائل زمان بندی بسیار موفق است (شیا و وو، ۲۰۰۵). SA را می توان به عنوان مانسته ی الگوریتم مورد استفاده در فیزیک آماری برای شبیه سازی کامپیوتری بازپخت یک جامد به حالتی با حداقل انرژی تلقی کرد (ون لارهوون، آرتس، و لنسترا، ۱۹۹۲). بهینه های محلی می توانند با استفاده از الگوریتم SA حذف شوند. الگوریتم SA جواب های همسایه را که بدتر از جواب کنونی هستند با یک احتمال می پذیرد، بنابراین می تواند حذف شود. این احتمال پذیرش توسط یک پارامتر کنترل (دما) تعیین می شود که در طی رویه SA کاهش می یابد (بایکاسوقلو و گیندی، ۲۰۰۱). با شروع از یک جواب اولیه، بازپخت شبیه سازی شده یک جواب جدید Snew را در همسایگی جواب اصلی S تولید می کند. سپس، تغییر مقدار تابع هدف، $\Delta = f(S_{new}) - f(S)$ محاسبه می شود. برای مساله کمینه سازی، اگر $\Delta < 0$، گذار به جواب جدید پذیرفته می شود. اگر $\Delta > 0$، آنگاه گذار به جواب جدید با یک احتمال پذیرفته می شود، که معمولا با تابع $\exp(-\Delta/T)$ نشان داده می شود، که در آن T پارامتر کنترل به نام دما است. الگوریتم SA عموما از دمای بالایی شروع می شود و سپس دما به تدریج پایین آورده می شود. در هر دما، جستجو برای تعداد معینی تکرار انجام می شود، که طول دوران خوانده می شود. هنگامی که شرط خاتمه برقرار شد، الگوریتم خاتمه می یابد (شیا و وو، ۲۰۰۵). مهم ترین بخش الگوریتم SA برای تولید جواب های

همسایه موثر است. روش عملی طراحی OO به این رویکرد پیشنهاد شده برای تولید جواب های همسایه موثر کمک می کند.

۴,۱. حرکت همسایگی

همانطور که قبلا گفته شد، هر جواب با یک بردار کدگذاری واحد نمایش داده می شود. در این کار، جواب های همسایه توسط یک رویه ی درج تولید می شوند که در آن امکان پذیری در هر تکرار تضمین می شود. دو نوع حرکت همسایگی در این رویکرد تعریف می شود؛ درج پیش رو و درج پس رو. در هر دو نوع حرکت، فاصله محاسبه می شود و شیء عملیات در نقطه ای در درون بازه ی محاسبه شده وارد می شود. گام های حرکت همسایگی را می توان به صورت زیر تعریف کرد:

گام ۱: انتخاب یک شی عملیات تصادفی از یک کار تصادفی از بردار جواب.

گام ۲: تعیین موقعیت امکان پذیر به صورت تصادفی و درج شیء عملیات در این موقعیت.

امکان پذیری با درج هر عملیاتی که به صورت تصادفی انتخاب شده در موقعیتی که در آن قید تقدم نقض نمی شود به دست می آید (نامعادله ۴). فرض می کنیم که D بازه حداکثر تعداد موقعیت هایی را نشان می دهد که هر عملیات می تواند در آن درج شود و L طول بردار جواب باشد، بنابراین D به صورت متفاوت برای درج های پیش رو و پس رو محاسبه شود. اگر o_{ij} عملیاتی است که به طور تصادفی انتخاب می شود، آنگاه D را می توان توسط شبه کد زیر محاسبه کرد که در آن p_{ij} موقعیت j امین عملیات کار i و رندوم ()، روش مولد شماره تصادفی است ؛(random(L) ∈ (1, L))

برای درج پیش رو، شبه کد زیر برای محاسبه D استفاده می شود:

شروع

// p_{ij} = random(L) عملیاتی که قرار است حرکت داده شود و درج شود انتخاب می شود

مقدار دهی اولیه D با صفر

برای($k = p_{ij} + 1; k <= (L); k++$)

{اگر (اندیس کار موقعیت $k \neq i$) آنگاه D را به میزان یک افزایش بده؛ اگر ($k = i$) حلقه را قطع کن}

پایان

برای درج پس رو، شبه کد زیر برای محاسبه D استفاده می شود:

شروع

// $p_{ij} = random(L)$عملیاتی که قرار است حرکت داده شود و درج شود انتخاب می شود

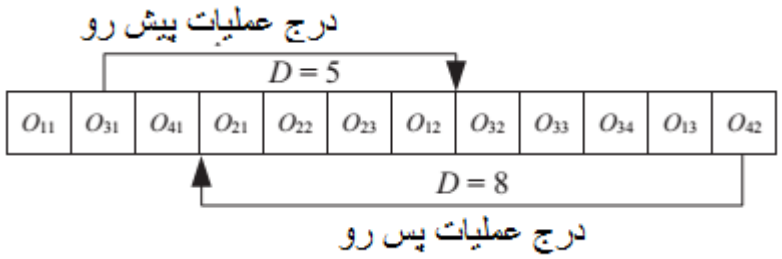
مقدار دهی اولیه D با صفر

برای($k = p_{ij} - 1; k <= (L); k--$)

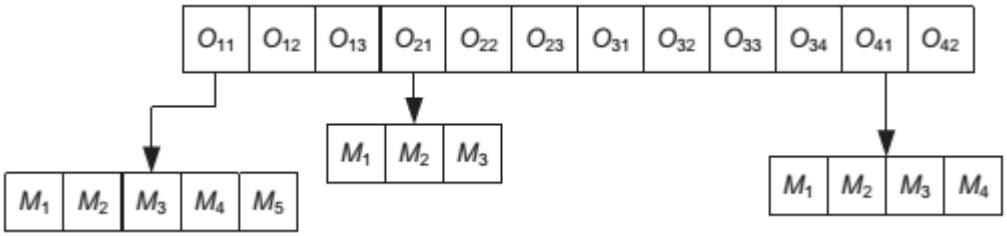
{اگر ($k \neq i$) آنگاه D را به میزان یک افزایش بده؛ اگر ($k = i$) حلقه را قطع کن}

پایان

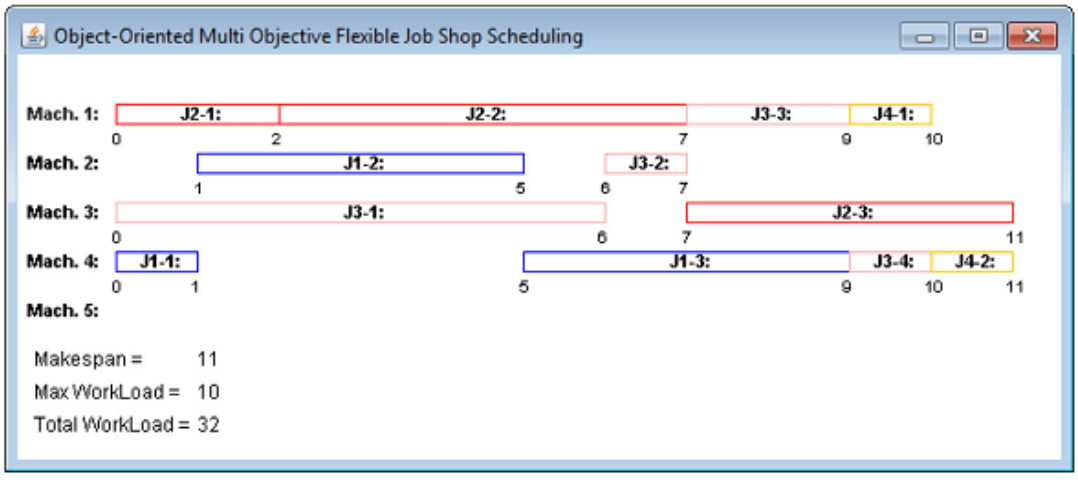
فرض می کنیم $p_{ij}*$ نشان دهنده موقعیت جدید j امین عملیات کار i باشد، که قبلا برای حرکت و درج انتخاب شد؛ نقطه درج پیش رو را می توان با؛ pij*=(pij+random(D)) و نقطه درج پس رو را می توان با pij*=(pij−random(D)) به دست آورد. شکل ۹ یک عملیات حرکت و درج نمونه را برای هر دو رویه ی پیش رو و پس رو نشان می دهد. فرض می کنیم O_{31} برای درج پیش رو انتخاب می شود. به دلیل قاعده تقدم عملیات ها، D = ۵ برای O_{31} یافت می شود. در بردار جواب O_{31} نمی تواند بیش از O_{32} باشد. رویه درج پس رو قاعده یکسانی را اعمال می کند. فرض می کنیم O42 به عنوان عملیات درج پس رو انتخاب می شود. به دلیل قاعده تقدم عملیات ها D = ۸ برای O42 یافت می شود. در بردار جواب O42 نمی تواند بیش از O_{41} باشد.



شکل ۹. درج پیش رو و پس رو.



شکل ۱۰. مقدار دهی اولیه FJSP.



شکل ۱۱. زمان بندی برای مجموعه مساله ۴ × ۵.

جدول ۲. مساله ۸ × ۸.

		M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8
کار۱	$O_{1,1}$	۵	۳	۵	۳	۳	–	۱۰	۹
	$O_{1,2}$	۱۰	–	۵	۸	۳	۹	۹	۶
	$O_{1,3}$	–	۱۰	–	۵	۶	۲	۴	۵
کار۲	$O_{2,1}$	۵	۷	۳	۹	۸	–	۹	–
	$O_{2,2}$	–	۸	۵	۲	۶	۷	۱۰	۹
	$O_{2,3}$	–	۱۰	–	۵	۶	۴	۱	۷
	$O_{2,4}$	۱۰	۸	۹	۶	۴	۷	–	–
کار۳	$O_{3,1}$	۱۰	–	–	۷	۶	۵	۲	۴
	$O_{3,2}$	–	۱۰	۶	۴	۸	۹	۱۰	–
	$O_{3,3}$	۱	۴	۵	۶	–	۱۰	–	۷
کار۴	$O_{4,1}$	۳	۱	۶	۵	۹	۷	۸	۴
	$O_{4,2}$	۱۲	۱۱	۷	۸	۱۰	۵	۶	۹
	$O_{4,3}$	۴	۶	۲	۱۰	۳	۹	۵	۷
کار۵	$O_{5,1}$	۳	۶	۷	۸	۹	–	۱۰	–
	$O_{5,2}$	۱۰	–	۷	۴	۹	۸	۶	–
	$O_{5,3}$	–	۹	۸	۷	۴	۲	۷	–
	$O_{5,4}$	۱۱	۹	–	۶	۷	۵	۳	۶
کار۶	$O_{6,1}$	۶	۷	۱	۴	۶	۹	–	۱۰
	$O_{6,2}$	۱۱	–	۹	۹	۹	۷	۶	۴

	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8
$O_{6,3}$	۱۰	۵	۹	۱۰	۱۱	–	۱۰	–
کار۷ $O_{7,1}$	۵	۴	۲	۶	۷	–	۱۰	–
$O_{7,2}$	–	۹	–	۹	۱۱	۹	۱۰	۵
$O_{7,3}$	–	۸	۹	۳	۸	۶	–	۱۰
کار۸ $O_{8,1}$	۲	۸	۵	۹	–	۴	–	۱۰
$O_{8,2}$	۷	۴	۷	۸	۹	–	۱۰	–
$O_{8,3}$	۹	۹	–	۸	۵	۶	۷	۱
$O_{8,4}$	۹	–	۳	۷	۱	۵	۸	–

نقاط درج پیش رو و پس رو برای O_{31} و O42 را می توان پس از یافتن مقادیر D با استفاده از معادله $p31*=(p31+random(5))$ برای O_{31} و معادله $p42*=(p42-random(8))$ برای O_{42} به دست آورد. ممکن است که SA به زمان های CPU مختلفی برای اجراهای مختلف منجر شود، حتی اگر برای مجموعه مساله یکسانی اجرا شود. اختلاف زمان CPU ناشی از تصادفی بودن در حرکت همسایگی و اندازه تکرارهایی است که این الگوریتم اجرا می کند. به عبارت دیگر، بر اساس $p_{ij}*$ تولید شده توسط الگوریتم، تعداد اشیاء عملیاتی منتقل شونده که توسط SA مورد رسیدگی قرار می گیرند متفاوت است که به اختلاف زمان CPU منجر می شود.

۴,۲. تابع برازش

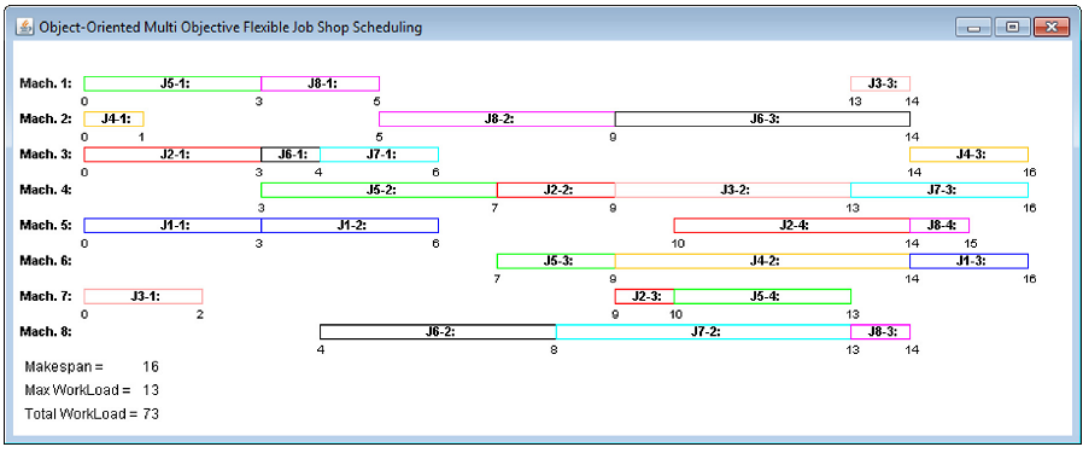
همانطور که در بخش تعریف مساله ارائه شد، FJSP چند هدفه سه هدف زیر را دارد؛

$$min\ f_1 = max_{1\leq k\leq m}(C_k)$$

$$min\ in\ f_2 = max_{1\leq k\leq m}(W_k)$$

$$min\ f_3 = \sum_{k=1}^{m} W_k$$

نمایش راه حل OO که در این مقاله پیشنهاد شده است به سیستم اجازه می دهد تا دو تا از اهداف (f1 و f2) FJSP را بر روی اشیاء عملیات و هدف دیگر را بر روی بردار جواب ارزیابی کند. ارتباط بین اشیاء عملیات و دستگاه شامل کمینه سازی f1 و f2 است. شی دستگاه دارای روش محاسبه هزینه () است که یک مقدار نوع دابل را به شی عملیات باز می گرداند، این روش f1 و f2 را به طور همزمان محاسبه می کند. اشیاء عملیات برای همه دستگاه هایی که در خصیصه دستگاه های جایگزین شان تعریف می شوند درخواست پیشنهاد می دهند (شکل ۶ را ببینید). استفاده از این خصیصه اطلاعات مسیریابی را به سمت اشیاء عملیات می کند. بر اساس این ساختار فوق الذکر، توابع برازش را می توان به صورت زیر ارائه داد:



شکل ۱۲. زمان بندی برای مجموعه مساله ۸ × ۸.

جدول ۳. مساله ۱۰ × ۱۰.

M_1 M_2 M_3 M_4 M_5 M_6 M_7 M_8 M_9 M_{10}

		M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}
کار۱	$O_{1,1}$	۱	۴	۶	۹	۳	۵	۲	۸	۹	۵
	$O_{1,2}$	۴	۱	۱	۳	۴	۸	۱۰	۴	۱۱	۴
	$O_{1,3}$	۳	۲	۵	۱	۵	۶	۹	۵	۱۰	۳
کار۲	$O_{2,1}$	۲	۱۰	۴	۵	۹	۸	۴	۱۵	۸	۴
	$O_{2,2}$	۴	۸	۷	۱	۹	۶	۱	۱۰	۷	۱
	$O_{2,3}$	۶	۱۱	۲	۷	۵	۳	۵	۱۴	۹	۲
کار۳	$O_{3,1}$	۸	۵	۸	۹	۴	۳	۵	۳	۸	۱
	$O_{3,2}$	۹	۳	۶	۱	۲	۶	۴	۱	۷	۲
	$O_{3,3}$	۷	۱	۸	۵	۴	۹	۱	۲	۳	۴
کار۴	$O_{4,1}$	۵	۱۰	۶	۴	۹	۵	۱	۷	۱	۶
	$O_{4,2}$	۴	۲	۳	۸	۷	۴	۶	۹	۸	۴
	$O_{4,3}$	۷	۳	۱۲	۱	۶	۵	۸	۳	۵	۲
کار۵	$O_{5,1}$	۷	۱۰	۴	۵	۶	۳	۵	۱۵	۲	۶
	$O_{5,2}$	۵	۶	۳	۹	۸	۲	۸	۶	۱	۷
	$O_{5,3}$	۶	۱	۴	۱	۱۰	۴	۳	۱۱	۱۳	۹
کار۶	$O_{6,1}$	۸	۹	۱۰	۸	۴	۲	۷	۸	۳	۱۰
	$O_{6,2}$	۷	۳	۱۲	۵	۴	۳	۶	۹	۲	۱۵
	$O_{6,3}$	۴	۷	۳	۶	۳	۴	۱	۵	۱	۱۱
کار۷	$O_{7,1}$	۱	۷	۸	۳	۴	۹	۴	۱۳	۱۰	۷
	$O_{7,2}$	۳	۸	۱	۲	۳	۶	۱۱	۲	۱۳	۳

		M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}
	$O_{7,3}$	۵	۴	۲	۱	۲	۱	۸	۱۴	۵	۷
کار۸	$O_{8,1}$	۵	۷	۱۱	۳	۲	۹	۸	۵	۱۲	۸
	$O_{8,2}$	۸	۳	۱۰	۷	۵	۱۳	۴	۶	۸	۴
	$O_{8,3}$	۶	۲	۱۳	۵	۴	۳	۵	۷	۹	۵
کار۹	$O_{9,1}$	۳	۹	۱	۳	۸	۱	۶	۷	۵	۴
	$O_{9,2}$	۴	۶	۲	۵	۷	۳	۱	۹	۶	۷
	$O_{9,3}$	۸	۵	۴	۸	۶	۱	۲	۳	۱۰	۱۲
کار۱۰	$O_{10,1}$	۴	۳	۱	۶	۷	۱	۲	۶	۲۰	۶
	$O_{10,2}$	۳	۱	۸	۱	۹	۴	۱	۴	۱۷	۱۵
	$O_{10,3}$	۹	۲	۴	۲	۳	۵	۲	۴	۱۰	۲۳

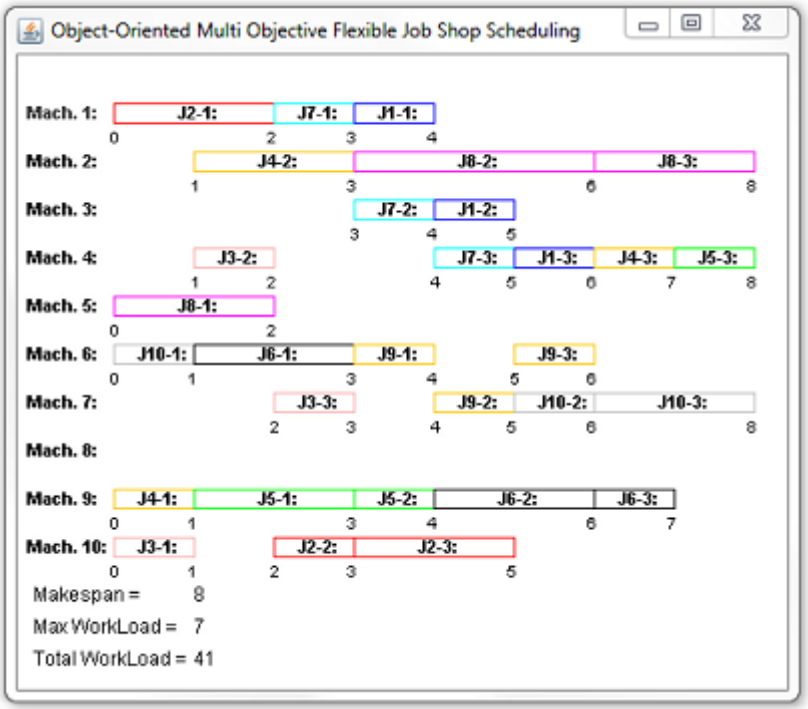
تابع برازش اشیاء عملیات = $\min (w_1 \times f_1 + w_2 \times f_2)$

تابع برازش بردار جواب = $\min (f_3)$

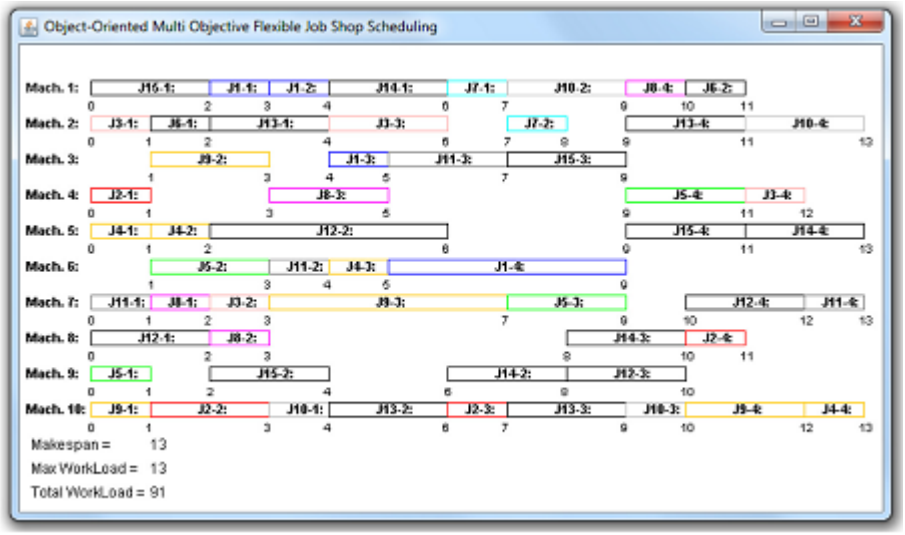
اشیاء دستگاه هزینه عملیات را محاسبه می کنند و نتیجه را به شی عملیات که درخواست پیشنهاد می دهد باز می گردانند. اشیاء عملیات نتیجه ی محاسبه هزینه را از اشیاء دستگاه دریافت می کنند و سپس هزینه های دستگاه های مختلف را مرتب می کنند. آنها پس از مرتب سازی نتایج هزینه، دستگاهی را تخصیص دادن انتخاب می کنند. اشیاء عملیات بهترین پیشنهاد را پس از مرتب سازی پیشنهادات اشیاء دستگاه می یابند. سپس، اشیاء عملیات به طور تصادفی پیشنهادی را از مجموعه پیشنهادات ارائه شده انتخاب می کنند. اگر شیء دستگاه که به طور تصادفی انتخاب شده است شیئی باشد که قبلا حداقل هزینه را پیشنهاد کرده بود، آنگاه انتخاب می شود، در غیر این صورت، این پیشنهاد با احتمال $\exp(-\Delta/T)$ مطابق با الگوریتم SA پذیرفته می شود.

در هر دما، یک جستجو برای تعداد معینی تکرار، به نام طول دوره، انجام می شود. اشیاء عملیات این تصمیم را با استفاده از پارامتر T همزمان با ارزیابی تابع برازش بردار جواب می گیرند. به عبارت دیگر، تابع برازش بردار جواب با استفاده از پارامتر T یکسانی همزمان با تابع برازش اشیاء عملیات ارزیابی می شود. تغییر مقدار (f_3) تابع هدف، $\Delta = f(S_{new}) - f(S)$ محاسبه می شود. اگر $\Delta < 0$، گذار به جواب جدید پذیرفته می شود. اگر $\Delta > 0$، آنگاه گذار به جواب جدید با یک احتمال طبق تابع $\exp(-\Delta/T)$ پذیرفته می شود و سپس دما به تدریج کاهش می یابد.

پذیرش جواب همسایه به پذیرش جواب همسایه در هر دو تابع برازش وابسته است. اگر دو جواب متوالی با بردارهای S_1 و S_2 نمایش داده شوند، آنگاه S_2 می تواند به عنوان جواب جدید پذیرفته شود، اگر با توجه به هر دوی تابع برازش اشیاء عملیات و تابع برازش بردار جواب پذیرفته شود.



شکل ۱۳. زمان بندی برای مجموعه مساله ۱۰ × ۱۰.



شکل ۱۴. زمان بندی برای مجموعه مساله ۱۵ × ۱۰.

۴،۳. زمان بندی خنک سازی

الگوریتم SA عموما از دمای بالایی شروع می شود و سپس دما به تدریج کاهش داده می شود. دمای اولیه باید به منظور پذیرش جواب هایی که به طور تصادفی تولید شده اند بالا باشد (با نزدیک تر شدن به احتمال پذیرش ۱). برآورد مقدار اولیه و نهایی دما را می توان با استفاده از معادلات زیر انجام داد (بایکاسوقلو و گیندی، ۲۰۰۱)؛

$$T_{in} = (f_{min} - f_{max})/(\ln P_c) \quad (7)$$

$$T_f = (f_{min} - f_{max})/(\ln P_f) \quad (8)$$

که در آن f_{min} و f_{max} کران پایینی و فوقانی برای توابع برازش هستند، P_c احتمال اولیه ی پذیرش است و P_f احتمال نهایی پذیرش است.

دما با ثابت α کاهش می یابد که با معادله زیر از بناج و دینگرا (۱۹۹۵) محاسبه می شود؛

$$\alpha = \left(\frac{\ln P_c}{\ln P_f}\right) 1/el_{max} - 1 \qquad (9)$$

که در آن el_{max} تعداد چرخه های کاهش دما (حداکثر تعداد تکرارها) است. هنگامی که تعداد el_{max} فرا برسد، آنگاه الگوریتم خاتمه می یابد. دمای نهایی را می توان با معادله ۱۰ محاسبه کرد. اگر دمای نهایی متفاوت از دمای یافت شده در معادله ۱۰ باشد، آنگاه تغییرات را می توان برای el_{max} انجام داد

$$T_f = T_{in}\alpha^{el_{max}} \qquad (10)$$

جدول ۴. مساله ۱۵ × ۱۰.

		M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}
کار۱	$O_{1,1}$	۱	۴	۶	۹	۳	۵	۲	۸	۹	۴
	$O_{1,2}$	۱	۱	۳	۴	۸	۱۰	۴	۱۱	۴	۳
	$O_{1,3}$	۲	۵	۱	۵	۶	۹	۵	۱۰	۳	۲
	$O_{1,4}$	۱۰	۴	۵	۹	۸	۴	۱۵	۸	۴	۴
کار۲	$O_{2,1}$	۴	۸	۷	۱	۹	۶	۱	۱۰	۷	۱
	$O_{2,2}$	۶	۱۱	۲	۷	۵	۳	۵	۱۴	۹	۲
	$O_{2,3}$	۸	۵	۸	۹	۴	۳	۵	۳	۸	۱
	$O_{2,4}$	۹	۳	۶	۱	۲	۶	۴	۱	۷	۲
کار۳	$O_{3,1}$	۷	۱	۸	۵	۴	۹	۱	۲	۳	۴
	$O_{3,2}$	۵	۱۰	۶	۴	۹	۵	۱	۷	۱	۶
	$O_{3,3}$	۴	۲	۳	۸	۷	۴	۶	۹	۸	۴
	$O_{3,4}$	۷	۳	۱۲	۱	۶	۵	۸	۳	۵	۲

		M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}
کار۴	$O_{4,1}$	۶	۲	۵	۴	۱	۲	۳	۶	۵	۴
	$O_{4,2}$	۸	۵	۷	۴	۱	۲	۳۶	۵	۸	۵
	$O_{4,3}$	۹	۶	۲	۴	۵	۱	۳	۶	۵	۲
	$O_{4,4}$	۱۱	۴	۵	۶	۲	۷	۵	۴	۲	۱
کار۵	$O_{5,1}$	۶	۹	۲	۳	۵	۸	۷	۴	۱	۲
	$O_{5,2}$	۵	۴	۶	۳	۵	۲	۲۸	۷	۴	۵
	$O_{5,3}$	۶	۲	۴	۳	۶	۵	۲	۴	۷	۹
	$O_{5,4}$	۶	۵	۴	۲	۳	۲	۵	۴	۷	۵
کار۶	$O_{6,1}$	۴	۱	۳	۲	۶	۹	۸	۵	۴	۲
	$O_{6,2}$	۱	۳	۶	۵	۴	۷	۵	۴	۶	۵
کار۷	$O_{7,1}$	۱	۴	۲	۵	۳	۶	۹	۸	۵	۴
	$O_{7,2}$	۲	۱	۴	۵	۲	۳	۵	۴	۲	۵
کار۸	$O_{8,1}$	۲	۳	۶	۲	۵	۴	۱	۵	۸	۷
	$O_{8,2}$	۴	۵	۶	۲	۳	۵	۴	۱	۲	۵
	$O_{8,3}$	۳	۵	۴	۲	۵	۴۹	۸	۵	۴	۵
	$O_{8,4}$	۱	۲	۳۶	۵	۲	۳	۶	۴	۱۱	۲
کار۹	$O_{9,1}$	۶	۳	۲	۲۲	۴۴	۱۱	۱۰	۲۳	۵	۱
	$O_{9,2}$	۲	۳	۲	۱۲	۱۵	۱۰	۱۲	۱۴	۱۸	۱۶
	$O_{9,3}$	۲۰	۱۷	۱۲	۵	۹	۶	۴	۷	۵	۶
	$O_{9,4}$	۹	۸	۷	۴	۵	۸	۷	۴	۵۶	۲

		M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}
كار١٠	$O_{10,1}$	٥	٨	٧	٤	٥٦	٣	٢	٥	٤	١
	$O_{10,2}$	٢	٥	٦	٩	٨	٥	٤	٢	٥	٤
	$O_{10,3}$	٦	٣	٢	٥	٤	٧	٤	٥	٢	١
	$O_{10,4}$	٣	٢	٥	٦	٥	٨	٧	٤	٥	٢
كار١١	$O_{11,1}$	١	٢	٣	٦	٥	٢	١	٤	٢	١
	$O_{11,2}$	٢	٣	٦	٣	٢	١	٤	١٠	١٢	١
	$O_{11,3}$	٣	٦	٢	٥	٨	٤	٦	٣	٢	٥
	$O_{11,4}$	٤	١	٤٥	٦	٢	٤	١	٢٥	٢	٤
كار١٢	$O_{12,1}$	٩	٨	٥	٦	٣	٦	٥	٢	٤	٢
	$O_{12,2}$	٥	٨	٩	٥	٤	٧٥	٦٣	٦	٥	٢١
	$O_{12,3}$	١٢	٥	٤	٦	٣	٢	٥	٤	٢	٥
	$O_{12,4}$	٨	٧	٩	٥	٦	٣	٢	٥	٨	٤
كار١٣	$O_{13,1}$	٤	٢	٥	٦	٨	٥	٦	٤	٦	٢
	$O_{13,2}$	٣	٥	٤	٧	٥	٨	٦	٦	٣	٢
	$O_{13,3}$	٥	٤	٥	٨	٥	٤	٦	٥	٤	٢
	$O_{13,4}$	٣	٢	٥	٦	٥	٤	٨	٥	٦	٤
كار١٤	$O_{14,1}$	٢	٣	٥	٤	٦	٥	٤	٨٥	٤	٥
	$O_{14,2}$	٦	٢	٤	٥	٨	٦	٥	٤	٢	٦
	$O_{14,3}$	٣	٢٥	٤	٨	٥	٦	٣	٢	٥	٤
	$O_{14,4}$	٨	٥	٦	٤	٢	٣	٦	٨	٥	٤

		M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}
کار۱۵	$O_{15,1}$	۲	۵	۶	۸	۵	۶	۳	۲	۵	۴
	$O_{15,2}$	۵	۶	۲	۵	۴	۲	۵	۳	۲	۵
	$O_{15,3}$	۴	۵	۲	۳	۵	۲	۸	۴	۷	۵
	$O_{15,4}$	۶	۲	۱۱	۱۴	۲	۳	۶	۵	۴	۸

۴،۴. مقداردهی اولیه

OO FJSP پیشنهاد شده با بارگذاری مجموعه مساله در حافظه کامپیوتر با توالی بندی کارها به ترتیب، مقداردهی اولیه می شود. عملیات های شان نیز به منظور تولید جواب اولیه امکان پذیر توالی بندی می شوند. شکل ۱۰ مقداردهی اولیه ی ساده را برای چهار کار نشان می دهد. تعداد عملیات های کارها و تعداد دستگاه های جایگزین برای هر عملیات ممکن است برای مجموعه مساله متفاوت باشند.

این پارامترها به صورت زیر تعیین می شوند؛ P_c و P_f به ترتیب برابر با ۰،۹۹ و 10^{-15} فرض می شوند. مقدار $(f_{min}-f_{max})$ برابر با -۱۰۰ فرض می شود. el_{max} برابر با ۲۰۰۰۰ فرض می شود. T_{in} و T_f به ترتیب با استفاده از معادلات ۷ و ۸ برابر با ۹۹۴۹،۹ و ۲،۸۹۵ یافت می شوند. α برابر با ۰،۹۹ (با استفاده از معادله ۹) محاسبه می شود. هر دو پارامتر w_1 و w_2 برابر با ۰،۵ تعیین می شوند. این الگوریتم می تواند برای ۲۰۰۰۰ بار برای مجموعه های مسائل اجرا شود.

جدول ۵. مقایسه عملکردهای الگوریتم برای مجموعه مساله ۸ × ۸.

زمان CPU (میلی ثانیه)	رویکرد OO			hGA[e]			hPSO[d]			MOGA[c]			PSO + SA[b]			AL[a]		
	f_3	f_2	f_1	f_3	f_2	f_1	f_3	f_2	f_1	f_3	f_2	f_1	f_3	f_2	f_1	f_3	f_2	f_1
2300	73	13	16	75	12	15	77	12	14	75	13	16	75	12	15	75	13	16
2700	75	12	15				75	12	15				73	13	16			
							77	11	16									
							73	13	16									

a: قاسم و همکاران (a۲۰۰۲)، (b۲۰۰۲)

b: PSO + SA: شیا و وو (۲۰۰۵).

c: MOGA: سعد و همکاران (۲۰۰۸).

d: hPSO: شائو و همکاران (۲۰۱۳).

e: hGA: گائو و همکاران (۲۰۰۷).

جدول ۶. مقایسه ی عملکردهای الگوریتم برای مجموعه مساله ۱۰ × ۱۰.

زمان CPU (میلی ثانیه)	رویکرد OO			hGA			hPSO			MOGA			PSO + SA			AL		
	f_3	f_2	f_1	f_3	f_2	f_1	f_3	f_2	f_1	f_3	f_2	f_1	f_3	f_2	f_1	f_3	f_2	f_1
3100	41	7	8	43	5	7	43	5	7	44	5	7	44	6	7	45	5	7
2823	42	5	8				42	6	7							42	5	8
3158	43	7	7				42	5	8							41	7	8
							41	7	8									

جدول ۷. مقایسه عملکردهای الگوریتم برای جموعه مسائل ۱۵ × ۱۰.

CPU زمان (میلی ثانیه)	OOرویکرد			hGA			hPSO			MOGA			PSO + SA			AL		
	f_3	f_2	f_1	f_3	f_2	f_1	f_3	f_2	f_1	f_3	f_2	f_1	f_3	f_2	f_1	f_3	f_2	f_1
5452	91	13	13	91	11	11	91	11	11	99	11	23	91	12	11	95	11	23
5820	91	12	14				93	10	12							91	11	24
							95	10	11									

جدول ۸. تحلیل آماری و مقایسه زمان CPU رویکرد پیشنهاد شده.

زمان متوسط CPU (میلی ثانیه)	انحراف معیار جواب ها			میانگین جواب ها			بدترین جواب ها			بهترین جواب ها			شماره نمونه	مجموعه مساله
	f_3	f_2	f_1	f_3	f_2	f_1	f_3	f_2	f_1	f_3	f_2	f_1		
1912	0	0	0	32	10	11	32	10	11	32	10	11	20	4×5^f
2514	1.45	0.5	1.5	74.3	12.5	15.8	73	13	19	73	13	16	20	8×8^f
							77	12	14	75	12	15		
8200		–			–			–		77	12	14	–	8×8^d
										75	12	15		
										77	11	16		
										73	13	16		
3012	0.8	0.8	0.6	41.6	6.6	8	41	7	9	41	7	8	20	10×10^f
										42	5	8		
										43	7	7		
11,700		–			–			–		43	5	7	–	10×10^d
										42	6	7		
										42	5	8		
										41	7	8		
5525	1	0.9	1.2	91.8	12	12.4	95	10	25	91	13	13	20	15×10^f
										91	12	14		
										93	11	11		
81,600		–			–			–		91	11	11	–	15×10^d
										93	10	12		
										95	10	11		

d: جواب های hPSO - شائو و همکاران (۲۰۱۳)، f: جواب های رویکرد OO. نماد "-" بدان معنی است که هیچ مقداری در مقاله ی مربوطه داده نشده است.

۵. نتایج آزمایش

به منظور نشان دادن اثربخشی و عملکرد رویکرد OO پیشنهاد شده، این الگوریتم با چند مجموعه مساله که از قاسم، حمادی، و بورن (a۲۰۰۲) (b۲۰۰۲) گرفته شده اند پیاده سازی می شود. موارد مساله با تعداد کارها (n)، تعداد دستگاه ها (m) و هر عملیات $O_{i,,j}$ کار i شناسایی می شوند که در آن nxm به مجموعه ای از n کار و m دستگاه اشاره دارد. چهار نمونه انتخاب می شوند که عبارتند از؛ مساله ۴ × ۵، مساله ۸ × ۸، مساله ۱۰ × ۱۰ و مساله ۱۵ × ۱۰. همه ی موارد مساله در کامپیوتری با پیکربندی Intel Core i5 CPU، □□□ □□۴، و ۲,۶۷ GHz اجرا می شوند. جاوا SE 7 به عنوان پلت فرم و زبان برنامه نویسی استفاده می شود.

مجموعه مساله ۴ × ۵ در جدول ۱ داده شده است و بهترین زمان بندی پیدا شده برای این مجموعه مساله در شکل ۱۱ داده شده است. جواب پیدا شده برای این مساله عبارت است از $f_1 = 11, f_2 = 10$ و $f_3 = 32$. بهترین جواب پیدا شده برای این مجموعه مساله در مقالات عبارت است از $f_1 = 11, f_2 = 10$ و $f_3 = 32$ که با جواب الگوریتم پیشنهاد شده برابر است، رویکرد پیشنهاد شده جواب جایگزینی را برای این مجموعه مساله ایجاد می کند، که در آن دستگاه ۵ استفاده نمی شود. زمان CPU برای این اجرا ۱۸۸۷ میلی ثانیه است. رویکرد پیشنهاد شده همین جواب را با استفاده از دستگاه های کمتر در سیستم پیدا می کند.

مجموعه مساله ۸ × ۸ در جدول ۲ داده شده است، و بهترین زمان بندی پیدا شده برای این مجموعه مساله در شکل ۱۲ داده شده است. جواب های غیر مغلوب یافته شده توسط رویکرد OO برای این مجموعه مساله عبارتند از $(f_1 = 16, f_2 = 13, f_3 = 73)$ و $(f_1 = 15, f_2 = 12, f_3 = 75)$.

مجموعه مساله ۱۰ × ۱۰ در جدول ۳ داده شده است، و بهترین زمان بندی پیدا شده برای این مجموعه مساله در شکل ۱۳ داده شده است. جواب های غیر مغلوب یافته شده توسط رویکرد OO برای این مجموعه مساله عبارتند از $(f_1 = 8, f_2 = 7, f_3 = 41)$ $(f_1 = 8, f_2 = 7, f_3 = 41)$, و $(f_1 = 8, f_2 = 7, f_3 = 41)$.

مجموعه مساله ۱۰ × ۱۵ در جدول ۴ داده شده است، و بهترین زمان بندی پیدا شده برای این مجموعه مساله در شکل ۱۴ داده شده است. جواب های غیر مغلوب یافته شده توسط رویکرد OO برای این مجموعه مساله عبارتند از $(f_1 = 13, f_2 = 13, f_3 = 91)$ و $(f_1 = 14, f_2 = 12, f_3 = 91)$.

جواب هایی که برای FSJP چند هدفه در مقالات پیدا شده اند با نتایج حاصل از رویکرد OO پیشنهاد شده مقایسه می شوند. جدیدترین نتایج در حوزه ی FJSP توسط قاسم و همکاران، b۲۰۰۲، شیا و وو، ۲۰۰۵، گائو و همکاران، ۲۰۰۷ و سعد و همکاران، ۲۰۰۸، و شائو و همکاران (۲۰۱۳) پیدا شدند. جداول ۵، ۶، و ۷ نتایج حاصل از رویکرد پیشنهاد شده را با تحقیقات موجود در مقالات مقایسه می کنند. زمان های CPU برای جواب های OO پیشنهاد شده ارائه شده اند. زمان های CPU مختلف برای رویکرد ارائه شده امکان پذیر هستند، حتی برای همان مجموعه مساله ای که می توان در جداول ۵ و ۶ و ۷ دید. اختلاف زمان CPU ناشی از تصادفی بودن در حرکت همسایگی و اندازه ی تکرارهایی که این الگوریتم اجرا می کند است. نتایج نشان می دهند که رویکرد OO قادر به تولید زمان بندی هایی است که معمولا توسط سایر تکنیک های الگوریتم و جواب دیگر مغلوب نمی شوند.

تحلیل آماری نیز برای نشان دادن توانایی و سازگاری طراحی پیشنهاد شده ی FJSP انجام می شود. تعداد اجرا (تعداد نمونه ها)، مقادیر کمینه (بهترین جواب ها)، مقادیر بیشینه (بدترین جواب ها)، میانگین جواب ها، انحراف معیارهای جواب ها، و زمان محاسباتی متوسط برای هر مجموعه مساله در جدول ۸ ارائه می شوند. همانطور که نتایج جدول ۸ نشان می دهند جواب های طراحی پیشنهاد شده برای مجموعه مساله های مختلف سازگار به نظر می رسند. جدول ۸ نیز مقایسه ی زمان CPU متوسط رویکرد پیشنهاد شده را با نتایج الگوریتم به روز موجود در مقالات نشان می دهد.

۶. نتیجه گیری

مساله FJSP چند هدفه با استفاده از رویکرد OO در این مقاله طراحی و حل می شود. سه هدف FJSP در این تحقیق در نظر گرفته می شوند: کمینه کردن حداکثر زمان تکمیل تمام عملیات ها، کمینه کردن بار کاری دستگاه

دارای بیشترین بار، و کمینه کردن بار کاری کل تمام دستگاه ها. اگر چه رویکردهای جواب در مقالات عموما از طرح های کدگذاری دو رشته ای برای نمایش مسیریابی و جایگشت جواب شان استفاده می کنند، این مقاله طرح کدگذاری واحدی را برای نمایش و حل همین مساله پیشنهاد می کند. در رویکرد پیشنهاد شده، عملیات ها و دستگاه ها به عنوان اشیاء نرم افزاری طراحی می شوند که به نمایش مساله به شیوه ی هوشمند و کارآمد کمک می کنند.

این مقاله نمایش OO برای FJSP را با استفاده از دیاگرام کلاس UML پیشنهاد می دهد، و کدگذاری مساله را به یک ساختار داده واحد تقلیل می دهد که در آن ساختار داده سلسله مراتبی برای نشان دادن اشیاء عملیات FJSP استفاده می شود. مزیت اصلی رویکرد پیشنهاد شده توانایی اش برای تقلیل تعداد ساختارهای داده ای مورد استفاده در طی فرایند حل مساله و افزایش تطبیق پذیری الگوریتم ها با سیستم های کنترل تولید واقعی است. رویکرد پیشنهاد شده می تواند مستقیما جواب های های امکان پذیر را به طور موثر تولید کند. با این حال، تعداد عناصر سیستم در هنگام ساخت OO سیستم افزایشی است. این ویژگی این رویکرد را می توان به عنوان محدودیت مقاله حاضر در نظر گرفت. سطح واتزویج داده ها نقش مهمی را در طراحی سیستم OO ایفا می کند، زیرا تعداد اشیاء مختلف در این سیستم ممکن است پیچیدگی سیستم را افزایش دهد. طراحان سیستم باید سطح وا تزویج را در چنین شرایطی حفظ کنند.

تصور می شود که این مطالعه هر دوی کمک های نظری و عملی را برای مساله FJSP، که چند بار در مقالات با الگوریتم های مختلف بسیاری حل شده است، ارائه می دهد. رویکرد کنونی با ارائه مشخصه تزویج داده ها به مسائل سیستم های تولیدی انعطاف پذیر کمک می کند. تقسیم بندی مساله در قطعات کوچکتر و ارائه فرصت اولویت الگوریتم، از لحاظ نظری از این فرایند حل مساله پشتیبانی می کنند. این رویکرد به کارورزان با ارائه مکانیزم نصب و اجرای آسان الگوریتم های بهینه سازی مختلف کمک می کند.

علاوه بر اثربخشی در نمایش جواب، طراحی FJSP با رویکرد OO ادغام الگوریتم های زمان بندی با سیستم های کنترل تولید واقعی را تسهیل می کند، زیرا اکثر DBMS های مدرن در سیستم های کنترل تولید از فناوری OO و عامل گرا استفاده می کنند. محققان این ادغام را کاری دشوار تلقی می کنند. این کمک راندمان توسعه نرم افزار

را افزایش می دهند، زیرا هیچ تلاشی برای توسعه نرم افزار میانجی برای تبدیل نتایج الگوریتم های زمان بندی به سیستم کنترل تولید نیاز نیست.

نتایج تجربی نشان می دهند که رویکرد OO پیشنهاد شده تنها با استفاده از یک الگوریتم واحد بر خلاف اکثر مطالعات قبلی که از الگوریتم ترکیبی برای یافتن جوابی برای FJSP چند هدفه استفاده کردند قادر به یافتن جوابِ غیر مغلوبِ اکثر مجموعه های مساله است.

مساله FJSP به شیوه ی رویکرد پیشنهاد شده ای که قبلا در مقالات مورد بحث واقع شد، بررسی نشده است. در مدل پیشنهاد شده، مکانیسم وا تزویج داده های مربوط به عناصر سیستم وجود دارد که در آن هر عنصر می تواند به طور مستقل وجود داشته باشد. مکانیسم مستقل ارائه شده در این رویکرد ممکن است به برخی جهات آتی در مورد سیستم های خبره و هوشمند منجر شود. نخست اینکه، تبدیل رویکرد OO پیشنهاد شده به سیستم خبره و هوشمند با در نظر گرفتن تقریب های سیستم عامل گرا و خدمت گرا ممکن به نظر می رسد. عناصر سیستم را می توان به عنوان عوامل نرم افزاری که تصمیمات خود را به طور مستقل می گیرند، طراحی کرد. دوم اینکه، رویدادهای محیط تولیدی پویا می توانند به عنوان موضوع تحقیقاتی آتی در نظر گرفته شوند. در این حوزه ی تحقیقاتی، در نظر گرفتن رویدادهای بازار پویای ورودها و تغییرات سفارش ممکن خواهد بود. این کار شامل ورودهای کاری جدید در محیط های تولیدی انعطاف پذیر خواهد بود. خرابی های دستگاه ها و مسیریابی مجدد عملیات نیز در این حوزه تحقیقاتی در نظر گرفته خواهند شد. سوم اینکه، قابلیت های زمان بندی مجدد آنی و یادگیری را می توان بررسی کرد. در نهایت، الگوریتم های فرا ابتکاری مختلف را می توان به رویکرد پیشنهاد شده وصل کرد.