نشریه علمی ـ پژوهشی
روش های عددی در مهندسی

# کمینه‌کردن مجموع وزنی دیرکرد در محیط کارگاه جریانی منعطف با ماشین‌های پردازش دسته‌ای

**نیلوفر فتاحی، محمد رئیسی نافچی* و قاسم مصلحی**

**دانشکده مهندسی صنایع و سیستم‌ها، دانشگاه صنعتی اصفهان، اصفهان**

(دریافت مقاله: ۱۳۹۷/۱۰/۱۹ – دریافت نسخه نهایی: ۱۳۹۸/۳/۱۱)

**چکیده**– زمان‌بندی در محیط‌های تولیدی به‌عنوان یک ابزار رقابتی در جهت بهبود کارایی و پاسخ به نیاز مشتریان به‌کار می‌رود. در این مقاله یک مسئله زمان‌بندی در محیط کارگاه جریانی منعطف سه مرحله‌ای با در نظر گرفتن انسداد و پردازش دسته‌ای بررسی می‌شود. این مسئله با الهام از خط شارژ و بسته‌بندی یک تولید کننده بزرگ باتری خودرو طراحی شده است. در این محیط، مرحله اول و سوم شامل یک ماشین پردازشگر تکی و مرحله دوم شامل m ماشین موازی پردازش دسته‌ای یکسان است. هدف، کمینه کردن مجموع دیرکرد وزنی سفارشات دریافتی است. با توجه به عدم مشاهده بررسی این مسئله در ادبیات موضوع، ابتدا یک مدل برنامه‌ریزی ریاضی برای آن ارائه شده است. همچنین با توجه به NP-hard بودن مسئله، یک الگوریتم فراابتکاری جستجوی همسایگی متغیر و یک الگوریتم فراابتکاری ممتیک برای حل آن توسعه داده شده است. نتایج محاسباتی نشان می‌دهد الگوریتم جستجوی همسایگی متغیر قادر است مسائل تا ابعاد ۱۲۰۰ سفارش و ۱۵ ماشین را با میانگین خطای حدود ۱/۹ درصد نسبت به بهترین جواب به‌دست آمده از بین دو روش، حل کند. الگوریتم ممتیک قادر است مسائل تا ابعاد ۱۲۰۰ سفارش و ۱۵ ماشین را با میانگین خطای حدود ۷/۸ درصد نسبت به بهترین جواب به‌دست آمده از بین دو روش، حل کند. در کل نتایج محاسباتی نشان از کارایی بهتر الگوریتم جستجوی همسایگی متغیر نسبت به الگوریتم ممتیک دارد.

**واژه‌های کلیدی:** زمان‌بندی، کارگاه جریانی منعطف، انسداد، پردازش دسته‌ای، الگوریتم جستجوی همسایگی متغیر، الگوریتم ممتیک.

# Minimizing Total Weighted Tardiness in a Flexible Flowshop Environment Considering Batch Processing Machines

N. Fattahi, M. Reisi-Nafchi*, and G. Moslehi

Department of Industrial and Systems Engineering, Isfahan University of Technology, Iran.

**Abstract**: ***Scheduling in production environments is used as a competitive tool to improve efficiency and respond to customer requests. In this paper, a scheduling problem is investigated in a three-stage flexible flowshop environment with the consideration of blocking and batch processing. This problem has been inspired by the charging and packaging line of a large battery manufacturer. In this environment, the first and third stages involve a single processor machine, and the second one consists of m identical parallel batch processing machines. The objective is to minimize the total weighted tardiness of the received orders.Given the lack of consideration of this problem in the literature, first, a mathematical programming model is presented for***

* : مسئول مکاتبات، پست الکترونیکی:reisi.m@iut.ac.ir

***the problem. Also, due to the NP-hardness of the problem, a variable neighborhood search algorithm and a memetic algorithm are developed to solve it. The computational results show that the variable neighborhood search algorithm can solve instances up to 1200 orders and 15 machines with an average deviation of about 1.9%, relative to the best solution of the two algorithms, and the memetic algorithm can solve instances up to 1200 orders and 15 machines with an average deviation of about 7.8%, as compared e to the best solution of the two algorithms. In general, computational results show the better performance of the variable neighborhood search algorithm in comparison to the memetic algorithm.***

**Keywords:** ***Scheduling, Fexible flowshp, blocking, Batch processing, Variable neighbourhood search algorithm, Memetic algorithm.***

## فهرست علائم

| | | | |
|---|---|---|---|
| $AO'_{kt}$ | سفارشاتی که در زمان t عضو دسته k هستند ولی عضو بقیه دسته‌هایی که پردازش آنان در زمان t تکمیل نشده‌اند، نیستند. $k=1,\ldots,K \quad t=1,\ldots,T$ | m | تعداد ماشین موازی در مرحله ۲ |
| $B_k$ | اگر دسته k تشکیل شود، مقدار آن یک و در غیر این صورت صفر است. $k=1,2,\ldots,K$ | M | عدد مثبت بزرگ |
| $Ct_{sk}$ | زمان تکمیل دسته k در مرحله $s=1,2,3$ $k=1,2,\ldots,K;$ | MaxCap | حداکثر ظرفیت دسته‌ها |
| $Co_{ik}$ | زمان تکمیل سفارش i در دسته $k=1,2,\ldots,K$ $i=1,\ldots,O;$ | $n_i$ | تعداد کارهای سفارش i یا مقدار سفارش $i=1,\ldots,O$ |
| $Cap_f$ | ظرفیت دسته متعلق به خانواده $f=1,\ldots,F$ | $ng_f$ | تعداد نوع کار در خانواده $f=1,\ldots,F$ |
| $Dt_{sk}$ | زمان خروج دسته k از مرحله $k,j=1,2,\ldots,K$ | O | تعداد سفارشات |
| $d_i$ | موعد تحویل سفارش $i=1,\ldots,O$ | $P_{sk}$ | زمان انجام عملیات دسته k در مرحله $s=1,2,3$ $k=1,2,\ldots,K;$ |
| F | تعداد خانواده‌ها | $Q_{ik}$ | تعداد کارهای اختصاص داده شده از سفارش i در دسته k $i=1,\ldots,O; k=1,2,\ldots,K$ |
| $f(i)$ | خانواده سفارش $i=1,\ldots,O$ | $q'_k$ | حداقل تعداد کاری که می‌توان به دسته k انتقال داد بدون اینکه جواب موجود بدتر شود. $k=1,\ldots,K$ |
| $g(i)$ | نوع کار سفارش $i=1,\ldots,O$ | $q''_k$ | حداقل تعداد کاری که می‌توان از دسته k انتقال داد بدون اینکه جواب موجود بدتر شود. $k=1,\ldots,K$ |
| $HU_{hgk}$ | اگر نوع کار g در نیمه h ام دسته k تولید شود، مقدار یک و در غیر این صورت برابر است با صفر $g=1,..,\sum_{f=1}^{F} ng_f \quad k=1,2,\ldots,K; h=1,2$ | $sg_f$ | مجموعه نوع کارهای عضو خانواده $f=1,\ldots,F$ |
| K | حد بالای تعداد دسته‌ها | $Sk_{st}$ | دسته‌هایی که در زمان t یکی از ماشین‌های مرحله s را اشغال کردند. $t=1,\ldots,T$ |

| | | | |
|---|---|---|---|
| $seq_k$ | دسته قرار گرفته در جایگاه k در توالی ورود به مرحله اول $k = ۱,\ldots,K$ | $V_{ik}$ | اگر کاری از سفارش i در دسته k تولید شود مقدار آن یک و در غیر این صورت برابر است با صفر $k = ۱,۲,\ldots,K \ , i = ۱,\ldots.,O$ |
| $T$ | انتهای افق زمانی | $V'_{jki}$ | اگر کاری در مکان j ام دسته k به سفارش i اختصاص یابد مقدار یک می‌گیرد و در غیر این‌صورت برابر است با صفر $k = ۱,۲,\ldots,K$ $j = ۱,\ldots,MaxCap \ , i = ۱,\ldots.,O,$ |
| $T_i$ | میزان دیرکرد سفارش $i = ۱,\ldots,O$ | $w_i$ | وزن سفارش $i = ۱,\ldots,O$ |
| $t_{sf}$ | مدت زمان پردازش خانواده f در مرحله $s = ۱,۲,۳; f = ۱,\ldots,F$ | $X_{lk}$ | اگر دسته k در مرحله ۲ روی ماشین l انجام شود، مقدار آن یک و در غیر این صورت برابر است با صفر $l = ۱,\ldots,m \ , k = ۱,۲,\ldots,K$ |
| $U_{fk}$ | اگر عضو خانواده f در دسته k تولید شود، مقدار آن یک و در غیر این صورت برابر است با صفر $f = ۱,\ldots,F \ , k = ۱,۲,\ldots,K$ | $Y_{skj}$ | اگر دسته k قبل از دسته j در مرحله s پردازش شود، مقدار آن یک و در غیر این‌صورت برابر است با صفر $s = ۱,۲,۳ \ , k, j = ۱,۲,\ldots,K$ |

## ۱- مقدمه

صنعت تولید باتری خودرو به‌دلیل افزایش تعداد کاربران رشد چشمگیری داشته است. مسئله مورد بررسی این مقاله از یک کارخانه‌ی باتری‌سازی که از چهار واحد تولیدی شامل واحدهای صفحه‌سازی، آب مقطرسازی، مونتاژ و شارژ و بسته‌بندی تشکیل شده الهام گرفته شده است. واحدهای آب مقطرسازی، صفحه‌سازی و مونتاژ تا حدود زیادی از تقاضا مستقل بوده و به‌عبارت دیگر تنوع آنها در مقابل تنوع نوع باتری بسیار محدود است. لذا این واحدها تولید خود را بر اساس پیش‌بینی فروش، آماده و انبار می‌کنند. از آنجا که تفاوت باتری‌ها به نحوه تولید آنها در واحد شارژ و بسته‌بندی برمی‌گردد، تمرکز این مقاله بر نحوه زمان‌بندی این واحد است. لذا هدف کمینه‌کردن مجموع وزنی دیرکرد سفارشات است.

در خط تولید باتری خودرو، ماشین‌های موازی متعددی وجود دارد که باتری به‌صورت دسته در این ماشین‌ها پردازش می‌شود. تحقیقات گسترده‌ای در حوزه مسئله زمان‌بندی ماشین‌های پردازش دسته‌ای (BPM)[1] انجام شده است، با این وجود تنوع این مسائل از نظر محیط ماشینی، محدودیت‌ها و تابع هدف متفاوت است. ایکورو و گیمپل [۱] در سال ۱۹۸۶ یک الگوریتم کارآمد برای مسئله زمان‌بندی یک ماشین پردازش دسته‌ای ارائه دادند. لی و همکاران [۲] در سال ۱۹۹۳ یک روش ابتکاری برای مسئله زمان‌بندی ماشین‌های موازی یکسان پردازش دسته‌ای با هدف کمینه کردن مجموع زمان تکمیل ارائه دادند. در ادامه یوزسوی [۳] در سال ۱۹۹۵ چند مسئله در ارتباط با مسئله زمان‌بندی ماشین پردازش دسته‌ای را بررسی کردند. یوزسوی [۳] برای کارهای این مسائل، خانواده ناسازگار[2] در نظر گرفت. دمدان و همکاران [۴] مسئله کمینه‌سازی بیشینه زمان تکمیل با خانواده ناسازگار و هالت و ولز-گالیگو [۵] مسئله کمینه‌سازی مجموع وزنی دیرکرد کارها با خانواده ناسازگار را در نظر گرفتند. آنها یک الگوریتم بهینه سازی ازدحام ذرات[3] برای مسئله زمان‌بندی در یک کارخانه

تولیدات الکترونیکی با محیط ماشین‌های موازی پردازش دسته‌ای ارائه دادند که مسائلی با ابعاد ۳۶۰۰ کار و ۶ ماشین را حل می‌کند.

لو و همکاران [۶] در سال ۲۰۰۹ به بررسی مسئله زمان‌بندی در یک شرکت فلزکاری با محیط کارگاه جریانی منعطف[۴] دو مرحله‌ای با هدف کمینه کردن حداکثر زمان تکمیل دسته‌های کارها پرداختند. مرحله اول شامل چندین ماشین موازی پردازش دسته‌ای است و مرحله دوم فقط یک ماشین پردازشگر تکی (DPM)[۵] دارد. هر کار زمان آماده‌سازی[۶] دارد که به‌کار قبلی آن وابسته است و بین مرحله اول و دوم امکان بروز انسداد[۷] وجود دارد. آنان برای حل این مسئله یک الگوریتم ژنتیک[۸] ارائه دادند.

بالاسوبرامانیان و همکاران [۷] به بررسی مسئله زمان‌بندی در یک کارخانه تولیدات نیمه‌هادی با محیط ماشین‌های موازی پردازش دسته‌ای و وجود تفاوت بین خانواده کارها پرداختند. آنها یک الگوریتم ژنتیک برای کمینه کردن مجموع وزنی دیرکرد کارها ارائه دادند. ماهیراجان و سیکومار [۸] به بررسی مسئله زمان‌بندی بر اساس محیط صنعت ریخته‌گری فولاد با محیط ماشین‌های موازی پردازش دسته‌ای و خانواده متفاوت برای کارها پرداختند. آنها یک روش ابتکاری حریصانه برای کمینه کردن مجموع وزنی دیرکرد کارها ارائه دادند. آلمادور و مونچ [۹] به بررسی مسئله زمان‌بندی بر اساس محیط کارخانه تولیدات نیمه‌هادی با محیط ماشین‌های موازی پردازش دسته‌ای و خانواده متفاوت کارها پرداختند. آنها سه الگوریتم ژنتیک، کلونی مورچگان[۹] و جستجوی همسایگی متغیر[۱۰] برای کمینه کردن مجموع وزنی دیرکرد کارها ارائه دادند. آنها بیان کردند از نظر زمان و کیفیت، الگوریتم جستجوی همسایگی متغیر بر دو الگوریتم دیگر غلبه داشته و مسائلی با ابعاد ۳۶۰۰ کار و ۶ ماشین را حل می‌کند.

حل مسئله زمان‌بندی دسته‌ای با روش‌های فراابتکاری در مقالات اخیر مورد توجه بیشتری قرار گرفته که به‌عنوان نمونه می‌توان به مقالات چیانگ و همکاران [۱۰]، چو [۱۱]، ماهیراجان و همکاران [۱۲] و شی و همکاران [۱۳] اشاره کرد. در این مقالات یک مسئله زمان‌بندی با محیط ماشین‌های موازی پردازش دسته‌ای با محدودیت زمان آماده به‌کار و اندازه غیریکسان کارها بررسی شده و برای حل آن روش‌های ابتکاری یا فراابتکاری ارائه شده است.

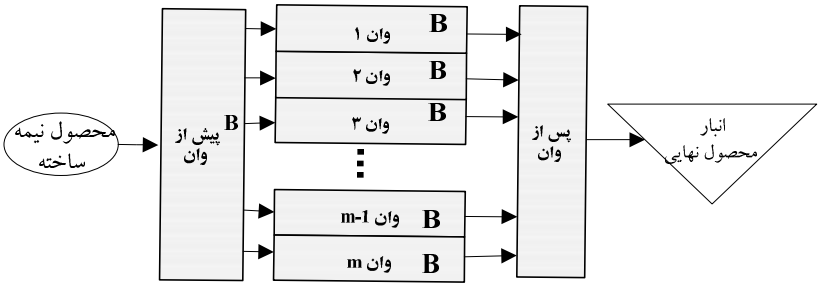
ویوک و همکاران [۱۴] مسئله کمینه‌سازی دامنه عملیات در یک محیط دو مرحله‌ای کارگاه جریانی منعطف را مطالعه کردند. آنها برای حل مسئله یک مدل ریاضی و یک الگوریتم ابتکاری توسعه دادند. زنگ و همکاران [۱۵] مسئله زمان‌بندی دو هدفه کارها در یک محیط کارگاه جریانی منعطف را با الهام از یک کارخانه تولید رول‌های کاغذ با دو مرحله در تولید مطرح و مورد بررسی قرار دادند. آنها برای حل مسئله یک مدل برنامه‌ریزی ریاضی، یک الگوریتم جستجوی ممنوع[۱۱] و یک الگوریتم ژنتیک مرتب‌سازی غیر مغلوب II (NSGAII) ارائه کردند. توابع هدف آنها دامنه عملیات و میزان انرژی مصرفی ماشین‌ها بوده است.

تان و همکاران [۱۶] در سال ۲۰۱۸ مسئله زمان‌بندی محیط کارگاه جریانی منعطف دو مرحله‌ای با ماشین‌های پردازش دسته‌ای در هر مرحله و کارهایی که زمان آماده‌سازی نابرابر دارند را با هدف کمینه کردن مجموع وزنی دیرکرد کارها بررسی کردند. آنان یک روش تجزیه با ترکیب تکنیک‌های جستجوی محلی[۱۳] و یک الگوریتم جستجوی همسایگی متغیر ارائه دادند که مسائلی تا ابعاد ۴۳۲۰ کار و مجموع ۱۶ ماشین در دو مرحله را حل می‌کند.

مسئله‌ای که در این مقاله مطرح شده، مسئله زمان‌بندی در محیط کارگاه جریانی منعطف سه مرحله‌ای با در نظر گرفتن انسداد و پردازش دسته‌ای[۱۴] است. مرحله اول و سوم در این مسئله شامل یک ماشین پردازشگر تکی و مرحله دوم شامل m ماشین موازی پردازش دسته‌ای یکسان است. قابل ذکر است که بررسی این مسئله تاکنون در ادبیات موضوع مشاهده نشده است. جدول (۱) مقایسه‌ای بین مطالعات فوق و مقاله حاضر دارد.

**جدول ۱- بررسی ویژگی‌های مطالعات ادبیات موضوع و تفاوت آن با مقاله حاضر.**

| نویسندگان | محیط | | | تابع هدف | | | | محدودیت | | | | ابعاد حل | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | تک ماشین | ماشین موازی | کارگاه جریانی | کارگاه جریانی منعطف | سایر | دامنه عملیات | مجموع وزنی دیرکرد | سایر | زمان آماده به‌کار | ناسازگاری | اندازه غیریکسان کارها | انسداد | سایر | روش حل | تعداد کار | تعداد مراحل | تعداد ماشین |
| ایکورو و گیمپل [۱] | ✓ | | | | | ✓ | ✓ | | ✓ | | | | | الگوریتم ابتکاری | | | ۱ |
| لی و همکاران [۲] | | ✓ | | | | ✓ | | | | | | | | الگوریتم ابتکاری | | | |
| یوزسوی [۳] | ✓ | | | | | ✓ | | | | ✓ | | | | الگوریتم ابتکاری | ۱۵۰ | | ۱ |
| دمدان و ولز-گالیگلو [۴] | | ✓ | | | | ✓ | | | ✓ | ✓ | | | | روش ابتکاری | ۲۰۰ | | ۵ |
| هالت و همکاران [۵] | | ✓ | | | ✓ | | ✓ | | | ✓ | | | | مدل ریاضی و الگوریتم بهینه‌سازی ازدحام ذرات | ۱۰۰ | ۴ | |
| لو و همکاران [۶] | | | ✓ | | | ✓ | | | | | | ✓ | ✓ | مدل ریاضی و الگوریتم ژنتیک | | ۲ | |
| بالاسوبرامانیان و همکاران [۷] | | ✓ | | | | | ✓ | | | ✓ | | | | الگوریتم ژنتیک | ۳۰۰ | | ۵ |
| ماهیراجان و سیواکومار [۸] | ✓ | | | | | | ✓ | | | ✓ | | | | روش ابتکاری حریصانه | ۸۶۱ | ۲۰ | ۱ |
| آلمادور و مونج [۹] | | ✓ | | | | | ✓ | | | ✓ | | | | الگوریتم ژنتیک، الگوریتم کلونی مورچگان و جستجوی همسایگی متغیر | ۳۶۰۰ | | ۶ |
| چیانگ و همکاران [۱۰] | | ✓ | | | | | ✓ | | ✓ | | ✓ | | ✓ | روش فراابتکاری ممتیک | ۳۰۰ | ۵ | |
| چو [۱۱] | | ✓ | | | | | ✓ | | ✓ | | ✓ | | | مدل ریاضی و الگوریتم شبیه‌سازی تبرید | ۲۰۰ | ۴ | |
| ماهیراجان و همکاران [۱۲] | ✓ | | | | | | ✓ | | ✓ | | ✓ | | | روش ابتکاری حریصانه، الگوریتم شبیه‌سازی تبرید | ۱۰۰ | | ۱ |
| شی و همکاران [۱۳] | ✓ | ✓ | | | | ✓ | | ✓ | | ✓ | | | | روش ابتکاری | ۴۰۰ | | ۱۰ |
| ویوک و همکاران [۱۴] | | | | ✓ | | ✓ | | | | | | | ✓ | مدل ریاضی، الگوریتم ابتکاری | | ۲ | |
| زنگ و همکاران [۱۵] | | | | ✓ | | ✓ | | ✓ | | | | | ✓ | مدل ریاضی، الگوریتم جستجو ممنوع و الگوریتم NSGAII | ۱۲۰ | ۲ | ۱۳ |
| تان و همکاران [۱۶] | | | | ✓ | | | ✓ | | ✓ | ✓ | | | | مدل برنامه‌ریزی عددصحیح مختلط، روش تجزیه بر اساس مرحله با ترکیب تکنیک‌های جستجوی محلی و یک الگوریتم جستجوی همسایگی | ۴۳۲۰ | ۲ | ۱۴ |
| مقاله حاضر | | | | ✓ | | | ✓ | | | ✓ | | ✓ | | مدل ریاضی، الگوریتم ممتیک و الگوریتم جستجوی همسایگی متغیر | ۵۴۰۰ | ۳ | ۱۵ |




**شکل ۱- مراحل تولید باتری**

در ادامه به بیان تعریف مسئله پرداخته شده و پس از آن یک مدل برنامه‌ریزی ریاضی برای آن ارائه می‌شود. سپس دو روش فراابتکاری جستجوی همسایگی متغیر و ممتیک[15] برای حل این مسئله ارائه می‌شود. نتایج محاسباتی حاصل از حل مدل و روش‌های فراابتکاری بر روی مسائل نمونه نیز در ادامه آمده است. در انتها جمع‌بندی و پیشنهاداتی برای مطالعات آتی بیان شده است.

## ۲- تعریف مسئله و مفاهیم

باتری‌های مونتاژ شده از انبار مونتاژ وارد واحد شارژ و بسته بندی می‌شوند. فرایند شارژ با تزریق الکترولیت به باتری در دستگاه فیلینگ[16] آغاز می‌شود. سپس باتری‌ها بر روی نقاله حرکت کرده و توسط یک دستگاه ترولی[17] به یکی از وان‌های شارژ موازی انتقال داده می‌شوند تا جریان الکتریکی به باتری‌ها اعمال شود. در این وان‌ها تعداد زیادی از باتری‌ها قرار می‌گیرند و به‌عبارت دیگر پردازش در این وان‌ها به‌صورت دسته‌ای است. پس از اتمام شارژ، باتری‌ها توسط یک دستگاه ترولی دیگر از وان‌ها خارج می‌شوند.

در هر وان حداکثر دو نوع باتری با ویژگی‌های مشترک امکان شارژ دارد. اگر در وان فقط باتری‌هایی از یک نوع وجود داشته باشند می‌توان از کل ظرفیت وان استفاده کرد. اما اگر همزمان دو نوع باتری از یک خانواده در یک وان شارژ شوند حداکثر تعداد هر نوع باتری می‌تواند به اندازه نصف ظرفیت وان باشد.

پس از طی مدت زمان مشخص، عمل شارژ باتری‌ها در وان به اتمام رسیده و تمام باتری‌های درون وان آماده خروج می‌شوند. دستگاه ترولی طی چندین دفعه و در هر دفعه یک ردیف باتری را از وان خارج می‌کند. در ادامه، باتری روی نقاله حرکت می‌کند تا به دستگاه تخلیه اسید برسد. از آنجا که مدت زمان بین رسیدن دو باتری متوالی به دستگاه تخلیه اسید از مدت زمان پردازش این دستگاه کمتر است، امکان دارد برخی باتری‌ها در این مرحله، در داخل وان و یا حتی وان‌های دیگر متوقف شوند و انسداد به‌وجود آید.

در ادامه فرایند تولید، باتری‌ها بر روی ایستگاه‌های تخلیه اسید، فیلینگ دوم، همسطح سازی[18]، کنترل سطح اسید، نصب درپوش، شستشو، تست ولتاژ، تست نشتی و پلمپ درب دوم پردازش می‌شوند. در انتها باتری‌ها، بسته‌بندی و آماده تحویل به مشتری می‌شوند. به‌طور کلی تولید باتری‌ها دارای سه مرحله شامل پیش از وان، وان و پس از وان است که در شکل (۱) این مراحل قابل مشاهده است. علامت B در شکل (۱)، امکان به‌وجود آمدن انسداد را نشان می‌دهد.

هر سفارش یک مشتری شامل تعدادی از یک نوع باتری است. سفارشات هر مشتری دارای موعد تحویل است که سعی می‌شود هر سفارش در موعد مقرر به مشتری تحویل داده شود. هدف کمینه‌کردن دیرکرد در تحویل سفارشات مشتریان است. از آنجا که مشتریان متنوع هستند و می‌توانند تولید کنندگان خودرو و یا فروشگاه‌های فروش باتری باشند، میزان اهمیت و وزن آنها با یکدیگر متفاوت است.

با توجه به تحلیل انجام شده که از شبیه‌سازی این خطوط به‌دست آمده است می‌توان کل مراحل تولید قبل از وان که

به‌صورت سری هستند را به یک مرحله تبدیل کرد و به روش مشابه مراحل تولید بعد از وان را نیز می‌توان در قالب یک مرحله جمع کرد. بنابراین می‌توان گفت که روند تولید باتری در واحد شارژ و بسته‌بندی از سه مرحله تشکیل شده که در مرحله دوم باتری‌ها در چند وان موازی شارژ می‌شوند. لذا فرایند تولید باتری در واحد شارژ و بسته‌بندی، یک محیط کارگاه جریانی منعطف ($FF_m$) سه مرحله‌ای است که در مرحله‌ی دوم آن چند ماشین موازی وجود دارد. از آنجا که در وان بیش از یک باتری می‌تواند قرار بگیرد و باتری‌های داخل هر وان به‌طور همزمان پردازش می‌شوند، از این‌رو می‌توان باتری‌های در حال پردازش همزمان در یک وان را به‌عنوان یک دسته در نظر گرفت. از این رو وان‌ها دارای ویژگی پردازش دسته‌ای هستند.

با توجه به مطالب بیان شده، مسئله مورد بررسی شامل یک محیط کارگاه جریانی منعطف با در نظر گرفتن انسداد و پردازش دسته‌ای است که در مرحله اول و سوم یک ماشین پردازشگر تکی و در مرحله دوم m ماشین موازی پردازش دسته‌ای یکسان حضور دارد. این مسئله کارگاه جریانی منعطف با در نظر گرفتن انسداد و پردازش دسته‌ای یا به اختصار FFBBP[19] نام گذاری شده است.

یک تولید کننده، دارای O سفارش و هر سفارش شامل $n_i$ ($i = ۱,\ldots,O$) کار و مجموع تمام کارها در تمام سفارش‌ها برابر N ($N = \sum_{i=۱}^{O} n_i$)است که همگی باید تولید شوند. هر سفارش دارای وزنی است که اهمیت آن سفارش را در رعایت موعد تحویل نشان می‌دهد. همه کارهای یک سفارش دارای موعد تحویل یکسان بوده و همگی باید با هم تحویل شوند.

هدف این مسئله کمینه‌کردن مجموع دیرکرد وزنی سفارشات است و با نماد سه جزئی گراهام و همکاران [۱۷] به‌صورت $FF_۳\left(۱^{(1)}(\delta), PM^{۲}(\beta), ۱^{(3)}(\delta)\right)\left|\text{block, batch, incompatible}\right|\sum_j w_j T_j$ نشان داده می‌شود. در این نماد، $\delta$ نشان دهنده ماشین پردازشگر تکی، $PM^{۲}(\beta)$ نشانگر ماشین‌های موازی پردازش دسته‌ای مرحله دوم، block نشان دهنده انسداد بین ماشین‌های موازی، batch نماد دسته و incompatible نشان دهنده ناسازگاری یا متفاوت بودن خانواده‌ها است. از آنجایی که مسئله فوق قابل کاهش به مسئله $۱||\sum_j w_j T_j$ با پیچیدگی به‌شدت NP-hard است، لذا پیچیدگی این مسئله نیز به‌شدت NP-hard است. در ادامه دو مفهوم خانواده و دسته، دقیق‌تر معرفی شده است.

**خانواده:** چند نوع باتری که ویژگی‌های مشترکی دارند، به‌طوری که امکان شارژ همزمان آنها در یک وان وجود داشته باشد، متعلق به یک خانواده هستند. هر باتری فقط می‌تواند در یک خانواده قرار گیرد.

**دسته:** در مرحله دوم تولید (پردازش در وان) این امکان وجود دارد که ترکیبی از کارهای سفارشات مختلف با هم شارژ شوند، که به این ترکیب دسته گفته می‌شود. در هر دسته، حداکثر دو نوع کار از یک خانواده وجود دارد. هر دسته به‌طور پیوسته و بدون انقطاع روی ماشین‌های مرحله دوم پردازش می شود. اگر در یک دسته فقط از یک نوع کار وجود داشته باشد، اندازه دسته حداکثر می‌تواند به اندازه ظرفیت ماشین مرحله دوم باشد و اگر دو نوع کار در یک دسته باشد اندازه هر دسته می تواند حداکثر به اندازه نصف ظرفیت ماشین مرحله دوم باشد. ظرفیت ماشین مرحله دوم بر اساس خانواده کار، متفاوت است.

## ۳- مدل برنامه‌ریزی ریاضی

در این بخش یک مدل برنامه‌ریزی ریاضی عدد صحیح مختلط برای حل مسئله FFBBP و با نام BSS توسعه داده شده است. فرمول‌بندی مدل BSS به قرار زیر است:

$$\text{Min} \sum_{i=۱}^{O} w_i \times T_i \qquad (۱)$$

$$\sum_{k=۱}^{K} Q_{ik} = n_i \qquad i = ۱,\ldots,O \qquad (۲)$$

$$\sum_{i=1}^{O} Q_{ik} \le MaxCap \times B_k \qquad k = 1,2,\ldots,K \tag{۳}$$

$$B_k - B_{k+1} \ge 0 \qquad k = 1,\ldots,K-1 \tag{۴}$$

$$Cap_{f(i)} \times V_{ik} \ge Q_{ik} \qquad k = 1,2,\ldots,K \;; i = 1,\ldots,O \tag{۵}$$

$$\sum_{h=1}^{2} HU_{hg(i)k} \ge V_{ik} \qquad k = 1,2,\ldots,K \;; i = 1,\ldots,O \tag{۶}$$

$$\sum_{f=1}^{F} U_{fk} \le 1 \qquad k = 1,2,\ldots,K \tag{۷}$$

$$\sum_{g \epsilon sg_f} \sum_{h=1}^{2} HU_{hgk} \le 2U_{fk} \qquad k = 1,2,\ldots,K; f = 1,\ldots,F \tag{۸}$$

$$\sum_{g \epsilon sg_f} HU_{hgk} \le 1 \qquad k = 1,2,\ldots,K; f = 1,\ldots,F; h = 1,2 \tag{۹}$$

$$\sum_{j=1}^{0/5 Cap_{f(i)}} V'_{jki} \le 0/5 Cap_{f(i)} \times HU_{1g(i)k} \qquad k = 1,2,\ldots,K \;; i = 1,\ldots,O \tag{۱۰}$$

$$\sum_{j=0/5 Cap_{f(i)}+1}^{Cap_{f(i)}} V'_{jki} \le 0/5 Cap_{f(i)} \times HU_{2g(i)k} \qquad k = 1,2,\ldots,K \;; \quad i = 1,\ldots,O \tag{۱۱}$$

$$\sum_{j=1}^{Cap_{f(i)}} V'_{jki} = Q_{ik} \qquad k = 1,2,\ldots,K \;; i = 1,\ldots,O \tag{۱۲}$$

$$\sum_{j=1}^{0/5 Cap_{f(i)}} \sum_{i=1}^{O} V'_{jki} \le 1 \qquad k = 1,2,\ldots,K \tag{۱۳}$$

$$P_{2k} = \sum_{f=1}^{F} \left( t_{2f} \times U_{fk} \right) \qquad k = 1,2,\ldots,K \tag{۱۴}$$

$$P_{sk} = \sum_{i=1}^{O} \left( t_{sf(i)} \times Q_{ik} \right) \qquad k = 1,\ldots,K; s = 1,3 \tag{۱۵}$$

$$\sum_{l=1}^{m} X_{lk} = B_k \qquad k = 1,2,\ldots,K \tag{۱۶}$$

$$Ct_{1k} \ge P_{1k} \qquad k = 1,2,\ldots,K \tag{۱۷}$$

$$Ct_{sk} \ge Ct_{s-1k} + P_{sk} \qquad k = 1,\ldots,K; s = 2,3 \tag{۱۸}$$

$$Y_{skj} + Y_{sjk} = 1 \qquad k,j = 1,\ldots,K; k \ne j; s = 1,2,3 \tag{۱۹}$$

$$Ct_{sj} + M\left(1 - Y_{skj}\right) \ge P_{sj} + Dt_{sk} \qquad k,j = 1,\ldots,K; s = 1,3 \tag{۲۰}$$

$$Ct_{2j} + M\left(3 - Y_{2kj} - X_{lk} - X_{lj}\right) \ge P_{2j} + Dt_{2k} \qquad k,j = 1,\ldots,K \;\; l = 1,2,..,m \tag{۲۱}$$

$$Ct_{sk} = Dt_{(s-1)k} + P_{sk} \qquad k = 1,\ldots,K \;; s = 1,3 \tag{۲۲}$$

$$Ct_{3k} = Dt_{3k} \qquad k = 1,2,\ldots,K \tag{۲۳}$$

$$Co_{ik} \ge Dt_{2k} + j \times t_{3f(i)} \times V'_{jki} \qquad k = 1,2,\ldots,K \;; i = 1,\ldots,O; j = 1,\ldots,Cap_{f(i)} \tag{۲۴}$$

$$T_i \ge Co_{ik} - d_i - M\left(1 - V_{ik}\right) \qquad k = 1,2,\ldots,K \;; i = 1,\ldots,O \tag{۲۵}$$

$$Ct_{sk}, P_{sk}, Dt_{sk} \ge 0 \qquad s = 1,2,3; k = 1,2,\ldots,K \tag{۲۶}$$

$$X_{lk} \in \{0,1\}, B_k \in \{0,1\} \qquad l = 1,..,m; k = 1,2,\ldots,K \tag{۲۷}$$

$$Y_{skj} \in \{0,1\} \qquad s = 1,2,3; j,k = 1,2,\ldots,K \tag{۲۸}$$

$$U_{fk} \in \{0,1\} \qquad k = 1,2,\ldots,K \;; f = 1,\ldots,F \tag{۲۹}$$

$$V'_{jki}, V_{ik} \in \{0,1\}; T_i, Q_{ik}, Co_{ik} \ge 0 \qquad k = 1,2,\ldots,K \;; i = 1,\ldots,O; j = 1,..,Cap_{f(i)} \tag{۳۰}$$

$$HU_{hgk} \in \{0,1\} \qquad h = 1,2; k = 1,2,\ldots,K \;; g = 1,\ldots,\sum_{f=1}^{F} ng_f \tag{۳۱}$$

رابطه (۱) مجموع وزن‌دار مدت زمان دیرکرد سفارشات را به‌عنوان تابع هدف حداقل می‌کند. رابطه (۲) تا (۴) تضمین می کند که کل سفارشات در دسته‌های تشکیل شده تولید شوند. رابطه (۵) تعیین می‌کند در هر دسته کدام سفارشات تولید شود، به‌طوری که مجموع کارهای سفارشاتی که در هر دسته تولید می‌شود از ظرفیت آن بیشتر نشود. رابطه (۶) تا (۹) تضمین می کند کارهای یک دسته حداکثر از دو نوع کار متعلق به یک خانواده باشند،که در هر نصف دسته یک نوع کار تولید شود. رابطه (۱۰) و (۱۱) به‌ترتیب تضمین می‌کند نصف اول و دوم ظرفیت دسته از یک نوع کار تشکیل شود. رابطه (۱۲) و (۱۳) بیان می‌کند تعداد مکان‌هایی که یک سفارش در یک دسته اشغال می‌کند برابر تعداد کارهای تخصیص داده شده از آن

سفارش در آن دسته است.

مدت زمان پردازش هر دسته در هر مرحله وابسته به خانواده آن دسته است که در مرحله دوم باتوجه به رابطه (۱۴) برابر با مدت زمان پردازش خانواده f در این مرحله است و در مرحله اول و سوم با توجه به رابطه (۱۵) با مدت زمان پردازش مقداری از کار خانواده f که دسته k را تشکیل می‌دهد برابر است. رابطه (۱۶) بیان می‌کند هر دسته در صورت تشکیل باید در مرحله دوم روی یکی از ماشین‌های موازی انجام شود. با استفاده از روابط (۱۷) تا (۲۳) زمان تکمیل و زمان خروج دسته‌ها در مراحل محاسبه می‌شود. در رابطه (۲۴) زمان تکمیل هر سفارش از هر دسته به‌دست می‌آید. رابطه (۲۵) میزان دیرکرد هر سفارش بر اساس زمان تکمیل کارهای آن سفارش در دسته‌هایی که آن کارها را تأمین می کند، محاسبه می‌کند. روابط (۲۶) تا (۳۱) نیز دامنه متغیرهای مدل را نشان می‌دهد. قابل ذکر است که روابط (۱۷) تا (۲۳) با الهام از مدل ساویک [۱۸] و بر اساس مسئله مورد بررسی تطبیق داده شده و سایر روابط مخصوص این مقاله است.

## ۴- خواص و ویژگی‌های مسئله

در این بخش ویژگی‌هایی برای مسئله FFBBP در قالب چند لم بیان شده است.

لم ۱: اگر در زمان t یک دسته مرحله سوم را ترک کند و رابطه (۳۲) برای دو دسته k و j برقرار باشد، آنگاه ابتدا دسته k وارد مرحله سوم می‌شود به‌عبارت دیگر داریم $(Dt_{2j} > Dt_{2k})$.

$$Ct_{2j} \geq p_{3k} + \max(Ct_{2k}, t) \qquad (۳۲)$$

**اثبات:** اگر یک دسته در زمان t مرحله سوم را ترک کند، در زمان t می‌توان پردازش دسته‌ی جدیدی را در مرحله سوم آغاز کرد. همچنین اگر دو دسته k و j در مرحله دوم وجود دارند که دسته k در زمان t در مرحله دوم در حال پردازش بوده و یا پردازش آن در مرحله دوم تکمیل شده است ولی پردازش دسته j بعد از زمان t طبق رابطه (۳۲) تکمیل می‌شود، اگر در زمان t ابتدا دسته k برای خروج از مرحله دوم انتخاب شود در نتیجه رابطه (۳۳) برقرار است:

$$Dt_{3k} = \max(t, Ct_{2k}) + p_{3k} \qquad (۳۳)$$

پس دسته k، در زمان $\max(t, Ct_{2k}) + p_{3k}$ مرحله سوم را ترک می‌کند و از آنجا که $Ct_{2k'} \geq p_{3k} + \max(Ct_{2k}, t)$ است قبل از اتمام پردازش دسته k در مرحله سوم هنوز پردازش دسته j در مرحله دوم تمام نشده است. ازاین‌رو اثبات کامل است.

در یک توالی اگر دسته j بلافاصله پس از دسته k در مرحله سوم پردازش شود، مجموع تابع هدف دو دسته طبق رابطه (۳۴) محاسبه می‌شود. مجموع تابع هدف دو دسته k و j وقتی دسته k در زمان t تکمیل می‌شود با $f_{k(t)\to j}$ نشان داده می‌شود. در ادامه پارامترهای استفاده شده در رابطه (۳۴) تعریف شده است.

$$f_{k(t)\to j} = \sum_{i=1}^{AO'_{kt}} \max\left(0, q_{ik} \times t_{3f_{(i)}} + \max(Ct_{2k}, t) - d_i\right) + \sum_{i=1}^{AO'_{jt}} \max\left(\begin{matrix}\max\left(p_{3k} + \max(Ct_{2k}, t), Ct_{2j}\right) \\ +q_{ij} \times t_{3f_{(i)}} - d_i, 0\end{matrix}\right) \qquad (۳۴)$$

مجموعه دسته‌هایی که در زمان t یکی از ماشین‌های مرحله s را اشغال کردند با $Sk_{st}$ نشان داده می‌شود و با توجه به تعریف $f_{k(t)\to j}$ لم ۲ ارائه می‌شود.

لم ۲: دو دسته k و j را در نظر بگیرید که عضو مجموعه $sk_{2t}$ هستند و ماشین مرحله سوم در زمان t آزاد می‌شود. در صورتی که روابط (۳۵) تا (۳۸) برای دو دسته k و j برقرار باشد و $f_{k(t)\to j} < f_{j(t)\to k}$ باشد ابتدا دسته k وارد مرحله سوم می‌شود.

$$p_{3j} + \max(Ct_{2j}, t) \geq p_{3k} + \max(Ct_{2k}, t) \qquad (۳۵)$$

$$\max(Ct_{2j}, t) \geq \max(Ct_{2k}, t) \qquad (۳۶)$$

$$Ct_{2i} \geq p_{3j} + \max\left(Ct_{2j}, p_{3k} + \max(Ct_{2k}, t)\right) \quad i \epsilon Sk_{2\max(Ct_{2k}, t)} / (j, k) \qquad (۳۷)$$

$$\max\left(\max(Ct_{2k}, t), Ct_{1i}\right) + p_{2i} \geq p_{3j} + \max\left(Ct_{2j}, p_{3k} + \max(Ct_{2k}, t)\right) \quad i \epsilon Sk_{1\max(Ct_{2k}, t)} \qquad (۳۸)$$

**اثبات:** اگر فرض شود در زمان t، ماشین سوم آزاد شده است و

دو دسته $k$ و $j$ در مرحله دوم وجود دارند. رابطه (۳۵) بیان می‌کند اگر دسته $k$ زودتر از دسته $j$ در مرحله سوم پردازش شود، ماشین مرحله سوم زودتر آزاد می‌شود. رابطه (۳۶) بیان می‌کند اگر دسته $k$ ابتدا پس از زمان $t$ در مرحله سوم پردازش شود زودتر مرحله دوم را ترک می‌کند تا این‌که ابتدا دسته $j$ پس از زمان $t$ در مرحله سوم پردازش شود.

با توجه به برقراری رابطه $f_{k(t)\to j} < f_{k(t)\to j}$، مجموع تابع هدف دو دسته $k$ و $j$ وقتی دسته $j$ بلافاصله قبل از دسته $k$ تکمیل می‌شود بیشتر از مجموع تابع هدف دو دسته $k$ و $j$ وقتی دسته $k$ بلافاصله قبل از دسته $j$ تکمیل می‌شود نیست. از این‌رو پردازش دسته $j$ قبل از دسته $k$ تابع هدف سفارشات دو دسته را بهتر نمی‌کند و باعث پردازش زودتر دسته‌های بعدی نمی‌شوند. طبق روابط (۳۷) و (۳۸) پردازش بقیه دسته‌ها در مرحله دوم تا اتمام پردازش دو دسته $k$ و $j$ در مرحله سوم تکمیل نخواهد شد. لذا اثبات کامل است.

لم ۳: اگر سفارشات دسته $k$ از خانواده $f$ باشد، در صورتی که دسته $k'$ بلافاصله پس از دسته $k$ به مرحله سوم وارد شود، می‌توان تعداد $q'$ کار به دسته $k$ اضافه کرد، بدون این که جواب موجود بدتر شود. $q'$ بر اساس رابطه (۳۹) محاسبه می‌شود.

$$\min\left(\frac{Ct_{۲k'}-Dt_{۳k}}{t_{۳f}},\frac{Dt_{۱k}-Ct_{۱k}}{t_{۱f}}\right)\geq q' \qquad (۳۹)$$

**اثبات:** رابطه $Dt_{۱k}\geq t_{۱f}\times q'+Ct_{۱k}$ نشان می‌دهد که پردازش دسته $k$ در زمان $Ct_{۱k}$ روی ماشین مرحله اول به پایان می‌رسد و این دسته حداقل به اندازه مدت زمان $t_{۱f}\times q'$ بر روی ماشین مرحله اول مسدود می‌شود. پس اضافه شدن $q'$ کار از خانواده $f$ با مدت زمان $t_{۱f}\times q'$ به دسته $k$ باعث دیرتر ترک‌کردن دسته $k$ از مرحله اول نمی‌شود.

زمان پردازش دسته $k$ در مرحله دوم در هر صورت $p_{۲k}$ است و افزایش یا کاهش تعداد کار دسته تغییری در آن حاصل نمی‌کند. در صورت اضافه شدن $q'$ کار به دسته $k$، دسته $k$ به اندازه مدت زمان $t_{۳f}\times q'$ بیشتر نیاز به پردازش در مرحله سوم دارد، پس در صورت اضافه شدن $q'$ کار به دسته $k$ زمان ترک دسته $k$ برابر $t_{۳f}\times q'+Dt_{۳k}$ می‌شود. بر اساس رابطه $Ct_{۲k'}\geq t_{۳f}\times q'+Dt_{۳k}$ زمان تکمیل مرحله دوم دسته $k'$ و دیگر دسته‌هایی که پس از دسته $k$ به مرحله سوم وارد می‌شوند بیشتر از مدت زمان $t_{۳f}\times q'$ است. بنابراین تفاوتی ندارد که دسته $k$، به اندازه مدت زمان $t_{۳f}\times q'$ در مرحله سوم بیشتر بماند و همین‌طور از آنجا که ماشین مرحله سوم ماشین پردازشگر تکی است، کارهای یک دسته تک تک پردازش شده و مرحله سوم را ترک می‌کنند. پس اضافه شدن $q'$ کار به دسته $k$ در زمان تکمیل بقیه کارهای این دسته در مرحله سوم تأثیری ندارد.

لم ۴: فرض کنید بدون اینکه تابع هدف بدتر شود $q'$ کار را می‌توان به دسته $k$ اضافه کرد. یک توالی شامل دو دسته $k$ و $k'$ از یک نوع و از خانواده $f$ وجود دارد که دسته $k'$ قبل از دسته $k$ در مرحله اول پردازش می‌شود. بدون اینکه تابع هدف بدتر شود $q''$ کار را می‌توان از دسته $k'$ به دسته $k$ انتقال داد که $q''$ برابر با $\min\left(q',\max_{o=۱,\ldots,O}\left(۰,\left\lfloor\frac{d_oV_{ok'}-Dt_{۳k}}{t_{۳f}}\right\rfloor\right)\right)$ است.

**اثبات:** فرض کنید $q'$ کار را می‌توان به دسته $k$ انتقال داد و در توالی موجود دسته $k'$ قبل از دسته $k$ وجود دارد که کارهای دسته $k'$ و دسته $k$ از یک نوع است. در دسته $k'$ کارهای دو سفارش $o$ و $o'$ تولید می‌شود که موعد تحویل سفارش $o'$ بزرگ‌تر مساوی موعد تحویل $o$ است. حالت (۱) $Q_{ok'}\geq Q_{o'k'}$ و $d_{o'}\geq t_{۳f}\times q'+Dt_{۳k}$:

۱- الف: $Q_{o'k'}\geq q'$: با انتقال $q'$ کار از سفارش $o'$ به دسته $k$ همچنان دیرکرد سفارش $o'$ صفر است و زمان تکمیل دسته‌های ما بین $k$ و $k'$ یا کمتر شده یا تغییری نمی‌کند.

۱- ب: $Q_{ok'}\geq q'>Q_{o'k'}$:

۱- ب- ۱: اگر $d_o\geq t_{۳f}\times q'+Dt_{۳k}$: می‌توان تعدادی کار از سفارش $o'$ و تعدادی کار از سفارش $o$ که در مجموع مساوی $q'$ است را به دسته $k$ انتقال داد.

۱-ب-۲: اگر $d_o < t_{۳f} \times q' + Dt_{۳k}$: ابتدا کلیه کارهای سفارش $o'$ که در دسته $k'$ بوده است به دسته $k$ انتقال داده می‌شود و سپس به اندازه $\max\left(۰, \min\left(q' - q_{o'}, \left\lfloor \frac{d_o - Dt_{۳k}}{t_{۳f}} \right\rfloor\right)\right)$ کار از کارهای سفارش $o$ که در دسته $k$ بوده‌اند به دسته $k$ منتقل می‌شود، به‌طوری که همچنان دیرکرد سفارش $o'$ و $o$ صفر است و زمان تکمیل دسته‌های ما بین $k$ و $k'$ یا کمتر شده یا تغییری نمی‌کند.

حالت ۲: $Q_{ok'} < Q_{o'k'}$: با انتقال $\max\left(۰, \min\left(q', \left\lfloor \frac{d_{o'} - Dt_{۳k}}{t_{۳f}} \right\rfloor\right)\right)$ کار از سفارش $o'$ از دسته $k'$ به دسته $k$ همچنان دیرکرد سفارش $o'$ صفر است و زمان تکمیل دسته‌های ما بین $k$ و $k'$ یا کمتر شده یا تغییری نمی‌کند.

در حالت کلی در صورتی که برخی از کارهای سفارش $o$ در دسته $k'$ باشد، تعداد کارهایی که از سفارش $o$ در دسته $k'$ می‌توان به دسته $k$ انتقال داد به‌طوری که تابع هدف بدتر نشود برابر با $\max\left(۰, \min\left(q', \max\left(۰, \left\lfloor \frac{d_o V_{ok'} - Dt_{۳k}}{t_{۳f}} \right\rfloor\right)\right)\right)$ است پس برای تمامی سفارشات داخل دسته $k$، $q''$ برابر $\max\left(۰, \min\left(q', \max_{o=۱,..,O}\left(۰, \left\lfloor \frac{d_o V_{ok'} - Dt_{۳k}}{t_{۳f}} \right\rfloor\right)\right)\right)$ است.

## ۵- روش ابتکاری

در این بخش یک الگوریتم ابتکاری برای حل مسئله به نام "تخصیص کارها در دسته و تعیین توالی و زمان‌بندی دسته‌ها" (HJABSS) [۲۰] ارائه می‌شود. از این الگوریتم می‌توان به‌صورت تنها و یا به‌عنوان جواب اولیه روش‌های فراابتکاری استفاده نمود. این الگوریتم شامل چهار رویه تعیین توالی اولیه سفارشات، تخصیص کارهای سفارشات به دسته‌ها، زمان‌بندی دسته‌ها با روش حریصانه و زمان‌بندی مجدد دسته‌ها با روش ابتکاری است. قسمت‌های مختلف روش ابتکاری HJABSS در ادامه شرح داده خواهد شد.

### ۵-۱- تعیین توالی اولیه سفارشات

در این قسمت حداکثر شش مجموعه مرتب شده از سفارشات با نام $SO^j$ $(j=۱,...,۶)$ به‌دست آورده می‌شود. ابتدا برای هر سفارش، یک موعد تحویل داخلی (ID)[۲۱] بر اساس رابطه (۴۰) محاسبه می‌شود. موعد تحویل داخلی هر سفارش، حداکثر زمانی است که پردازش یک سفارش می‌تواند آغاز شود تا تکمیل آن دیرکرد نداشته باشد:

$$ID_i = d_i - \left(t_{۱f(i)} \times n_i + t_{۲f(i)} + t_{۳f(i)} \times n_i\right) \qquad (۴۰)$$

سپس سفارشات را بر اساس سه ترتیب غیرنزولی موعد تحویل، اختلاف موعد تحویل از مدت زمان پردازش مرحله دوم و اختلاف موعد تحویل از مدت زمان پردازش مرحله سوم مرتب کنید. سفارشاتی که اولویت یکسان در ترتیب‌های به‌دست آمده دارند، بر اساس دو ترتیب غیرصعودی وزن سفارش ($w_i$) و $\frac{w_i}{ID_i}$ مرتب کنید.

### ۵-۲-رویه تخصیص کارهای سفارشات به دسته‌ها (H_JAB)[۲۲]

پس از تعیین توالی سفارشات، رویه ابتکاری H_JAB مشخص می‌کند که چند دسته تولید شود و در هر دسته چند کار وجود داشته باشد. رویه ابتکاری H_JAB از دو فاز تشکیل شده و برای هر شش مجموعه $SO^j$ $(j=۱,...,۶)$ اجرا می‌شود. در فاز اول ظرفیت اولیه هر دسته تعیین می‌شود. در فاز دوم کارهای سفارش‌ها به دسته‌ها تخصیص داده می‌شود. در رویه ابتکاری H_JAB، نمادهای $CapF_f$، $CapO_i$ و $CapB_k$ به‌ترتیب ظرفیت اولیه دسته‌های شامل کارهای خانواده $f$، ظرفیت اولیه دسته‌های شامل سفارش $i$ و ظرفیت دسته $k$ تعریف شده است.

#### فاز اول: تعیین ظرفیت

**گام ۱:** تعداد نیم‌دسته‌هایی که هر نوع کار نیاز دارد را به‌دست آورید. برای محاسبه‌آن، تعداد کل کارهای متعلق به هر

نوع کار را محاسبه و بر ظرفیت خانواده همان نوع کار تقسیم کنید.

**گام ۲:** مجموع نیم‌دسته‌های نوع کاری‌های محاسبه شده در گام ۱ برای هر خانواده را به‌دست آورید.

**گام ۳:** به‌طور پیش فرض حداکثر تعداد دسته موجود را برابر تعداد سفارشات قرار دهید. ظرفیت اولیه برای دسته‌هایی که شامل کارهای خانواده f هستند ( $CapF_f$ ) برابر با $\left\lceil \frac{\text{تعداد دسته مورد نیاز برای خانواده } f \times \text{ظرفیت خانواده } f}{\text{حداکثر تعداد دسته}} \right\rceil$ است.

**فاز دوم: تخصیص سفارشات به دسته‌ها**

**گام ۱:** ظرفیت اولیه دسته‌های شامل iامین سفارش مجموعه so را بر اساس رابطه (۴۱) محاسبه کنید.

$$CapO_{so(i)} = \begin{cases} \left( \frac{ID_{so(i)}}{t_{۱f(so(i))} + t_{۳f(so(i))}} \right), & i = ۱ \\ \left( \frac{ID_{so(i)} - ID_{so(i-۱)}}{t_{۱f(so(i))} + t_{۳f(soi)}} \right), & i \neq ۱ \end{cases} \qquad (۴۱)$$

**گام ۲:** اولین سفارش مرتب شده (i) که در آن کاری وجود دارد که در دسته قرار نگرفته را در نظر بگیرید. اگر دسته k وجود داشت که نوع یکی از نیم‌دسته‌های آن با سفارش i یکسان بود به اندازه حداقل تعداد ظرفیت باقی مانده در دسته k و تعداد کار سفارش مرتب شده i از این سفارش به دسته k انتقال داده می‌شود. در غیر این صورت اگر دسته k وجود داشت که خانواده آن با خانواده سفارش مرتب شده i یکسان و نوع کار نیم‌دسته دوم آن تعیین نشده بود، نوع کار در نیم‌دسته دوم را برابر نوع کار سفارش i قرار دهید. به اندازه حداقل نصف ظرفیت دسته k و تعداد کار سفارش مرتب شده i از این سفارش به دسته k انتقال داده می‌شود.

این گام را تا قرار دادن تمامی کارهای سفارشات در دسته‌ها ادامه دهید. اگر کاری از سفارشی وجود داشت که در دسته‌های موجود قرار نگرفت به گام ۳ بروید در غیر این‌صورت این فاز پایان می‌یابد.

**گام ۳:** بر اساس اولین سفارش مرتب شده‌ای (i) که در آن کاری وجود دارد، یک دسته جدید ایجاد کنید. خانواده این دسته را برابر با خانواده سفارش i قرار دهید. ظرفیت دسته ( $CapB_k$ ) به $\min\left(Cap_{f(so(i))}, \max\left(CapO_{so(i)}, CapF_{f(so(i))}\right)\right)$ تغییر داده شود. اگر کارهای سفارش بیش از $Cap_{f(so(۱))}/۲$ باشد نوع کار در هر دو نیم‌دسته را برابر نوع کار اولین سفارش مرتب شده قرار دهید. در غیر این‌صورت نوع کار فقط در نیم‌دسته اول را برابر نوع کار اولین سفارش مرتب شده قرار دهید. نوع کار در نیم‌دسته دوم در ادامه تعیین می‌شود. سپس به گام ۲ بروید.

### ۵-۳- زمان‌بندی دسته‌ها (U_BSC)[۲۳]

در قسمت سوم، پس از تعیین دسته‌ها و توالی ورود به ماشین اول در مورد زمان‌بندی دسته‌ها (BSC) تصمیم گرفته می‌شود. در این روش بر اساس الگوریتم costB، یک دسته، از بین دسته‌هایی که آمادگی خروج از مرحله دوم را در زمان t دارند به‌عنوان دسته‌ای که مرحله دوم را ترک می‌کند انتخاب می‌شود. در نتیجه یک ماشین در مرحله دوم بیکار می‌شود و دسته‌ای جدید بر اساس توالی ورودی مرحله اول می‌تواند وارد مرحله دوم شود. این روند ادامه داده می‌شود تا توالی خروجی تمام دسته‌ها از مرحله دوم به‌دست آید و زمان‌بندی مسئله تکمیل شود. الگوریتم costB دارای دو گام است که در ادامه بیان می‌شود.

**گام ۱:** مجموع وزنی دیرکرد سفارشات درون دسته i که در زمان t آمادگی خروج از مرحله دوم را دارند درصورت ترک دسته i از مرحله دوم در زمان t محاسبه می‌شود.

**گام ۲:** دسته‌ای که کمترین مجموع دیرکرد وزنی سفارشات مثبت را دارد از بین دسته‌هایی که در زمان t آمادگی خروج از مرحله دوم را دارند انتخاب می‌شود. در صورت عدم وجود چنین دسته‌ای، دسته‌ای که کمینه زمان تکمیل را در مرحله دوم دارد انتخاب می‌شود.

### ۵-۴- رویه ابتکاری زمان‌بندی دسته‌ها (H_BSC)[۲۴]

برای بهبود جواب به‌دست آمده، در زمان‌بندی بهترین توالی محاسبه شده در قسمت سوم تجدید نظر می‌شود. بدین صورت که بهترین توالی محاسبه شده در قسمت سوم به‌عنوان جواب موجود در نظر گرفته می‌شود. زمان‌بندی دسته‌ها در این قسمت با استفاده از توالی ورودی دسته‌ها به مرحله اول و توالی کارهای درون دسته‌ها در جواب موجود با یک الگوریتم ابتکاری به نام جست‌وجوی شعاعی[۲۵] انجام می‌شود.

در هر سطح تصمیم‌گیری این الگوریتم برای این‌که کدام یک از m ماشین موازی تخلیه شود m حالت مختلف وجود دارد. در سطح kام برای دسته kام تصمیم‌گیری می‌شود که این دسته بر روی کدام یک از m ماشین موازی پردازش می‌شود. هر گره نماینده یک توالی جزئی از ماشین‌ها است. زیر شاخه هر گره فرزند آن نامیده می‌شود و گره مورد نظر با نام والد آن فرزند شناخته می‌شود. فرزند یک گره والد، عبارت است از یک توالی جزئی از ماشین‌های گره والد به اضافه یک ماشین که در انتهای آن قرار می‌گیرد.

با استفاده از الگوریتم U_BSC یک حد بالا برای مسئله محاسبه شده است. اگر تابع هدف توالی جزئی یک گره، از حد بالا بیشتر باشد گره مورد نظر قطع می‌شود. استفاده از لم ۱ و لم ۲ تعداد حالات امکان‌پذیر در الگوریتم جست‌وجوی شعاعی را کاهش داده به‌طوری که جواب بهینه در حالاتی که تشکیل می‌شوند همچنان وجود داشته باشد. هنگامی که تعداد گره‌های آخرین سطح مورد بررسی به ۲۰۰ عدد رسید از فیلتر ۱ و هنگامی که تعداد گره‌های آخرین سطح مورد بررسی به ۳۰۰ عدد رسید از فیلتر ۲ برای کاهش حالات امکان‌پذیر استفاده شده است. طبیعی است که استفاده از دو فیلتر ۱ و ۲ ممکن است باعث از دست دادن برخی از جواب‌ها شود. قابل ذکر است که اعداد ۲۰۰ و ۳۰۰ بر اساس آزمایشات مقدماتی در راستای ایجاد توازن بین مدت زمان حل و کیفیت جواب به‌دست آمده‌اند و برای تمامی مسائل با ابعاد مختلف نیز یکسان در نظر گرفته شده‌اند.

**فیلتر ۱:** فرض کنید t زمانی است که یک دسته مرحله سوم را ترک کند. همچنین دو دسته k و j را در نظر بگیرید که عضو مجموعه $sk_{۲t}$ هستند. در صورتی که روابط (۳۵) و (۳۶) برای دو دسته k و j برقرار باشد و مجموع تابع هدف این دو دسته وقتی دسته j بلافاصله قبل از دسته k تکمیل می‌شود بیشتر از مجموع تابع هدف این دو دسته وقتی دسته k بلافاصله قبل از دسته j تکمیل می‌شود نباشد، به‌عبارتی داشته باشیم $f_{k(t)\to k'} < f_{k(t)\to k'}$، ابتدا دسته k وارد مرحله سوم می‌شود.

**فیلتر ۲:** فرض کنید t زمانی است که یک دسته مرحله سوم را ترک کند و به‌عبارتی ماشین مرحله سوم در زمان t آزاد شود. همچنین دو دسته k و j را در نظر بگیرید که عضو مجموعه $sk_{۲t}$ هستند. در صورتی که روابط (۳۵) و (۳۶) برای دو دسته k و j برقرار باشد، ابتدا دسته k وارد مرحله سوم می‌شود.

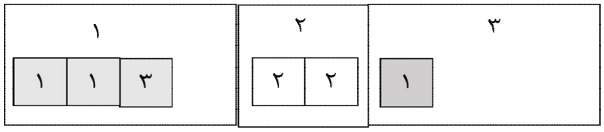
### ۵-۵- الگوریتم جستجوی همسایگی متغیر

الگوریتم جستجوی همسایگی متغیر که به‌اختصار VNS نامیده می‌شود، یک الگوریتم فراابتکاری مبتنی بر جستجوی محلی است [۱۹]. ایده اصلی این الگوریتم توانمند کردن روش جستجوی محلی یا به‌عبارتی دیگر افزایش توانایی در فرار کردن از جواب محلی است [۹]. این الگوریتم توسط میلادنوویک و هانسن [۱۹] در سال ۱۹۹۷ ارائه شده است.

فرض کنید $N_l$ که در آن $l = \{۱,۲,۳,\ldots,l_{max}\}$ است ساختار همسایگی یا تکانش[۲۶] lام است. الگوریتم با استفاده از جواب اولیه تولید شده که در بخش (۶-۲) نحوه به‌دست آوردن آن بیان شده است، کار خود را آغاز می‌کند و تا زمانی که معیار خاتمه برقرار شود، حلقه اصلی الگوریتم تکرار می‌شود. حلقه اصلی الگوریتم VNS دارای دو فاز اصلی تکانش و جستجوی محلی (Local_search) است. در فاز تکانش، الگوریتم از جواب فعلی (x) با استفاده از lامین ساختار همسایگی (تکانش)، به یک جواب همسایه ($x'$) حرکت

```
VNS(input: x,N_l,(l=1,2,.., l_max ),output:x)
x=generate initial solution ();
Repeat the following until the stopping criteria (current_time ≥ nonImp_time and current_time ≥ total_time) are met:
Set l =1;
While (l ≤ l_max )
x′=Shaking (x, l)
 x "=Local_search (x′)
 If f ( x ") < f( x ' )
x = x ";
l= 1;
Else
l= l + 1;
End If
End while
```

**شکل ۲- شبه کد الگوریتم VNS پیشنهادی**




**شکل ۳- نمایش رشته جواب اول**

می‌کند. در فاز جستجوی محلی نیز بر روی جواب $x'$ با استفاده از روش‌های جستجوی همسایگی، جستجو انجام می‌گیرد تا به بهینه محلی $x''$ برسد. اگر بهینه محلی به‌دست آمده از جواب فعلی (x) بهتر باشد جایگزین آن می‌شود و در غیر این صورت از ساختار همسایگی بعد ($N_{l+1}$) برای ادامه جستجو استفاده می‌شود. این جستجو تا زمانی که $l \le l_{max}$ باشد ادامه می‌یابد. ساختار همسایگی، جستجوی محلی و شرط توقف مورد استفاده در الگوریتم VNS ارائه شده در این مقاله، در بخش‌های بعد توضیح داده شده است. پارامتر Current_time نشان دهنده مدت زمان اجرای فعلی، nonImp_time نشان‌دهنده حداکثر مدت زمان مجاز عدم بهبود و Total_time معرف حداکثر مدت زمان کل در این الگوریتم است. شکل (۲) شبه کد الگوریتم جستجوی همسایگی متغیر VNS پیشنهادی، را نشان می‌دهد.

### ۶-۱- نمایش جواب

برای نمایش جواب، دو رشته جواب در نظر گرفته شده است. رشته جواب اول ساختار هر دسته که شامل چند کار و هر کار عضو کدام سفارش است را نمایش می‌دهد. ساختار جواب اول از K قسمت تشکیل شده است که K تعداد کل دسته‌ها را نشان می‌دهد. در هر قسمت یک سطر $Q_k$ درایه‌ای وجود دارد که $Q_k$ تعداد کارها در دسته k است. این رشته جواب نشان می‌دهد کاری که متعلق به سفارش i است در دسته k در چه جایگاهی تولید می‌شود. برای بیان بهتر یک مثال با سه سفارش و شش کار تعریف شده‌است. در این مثال به‌ترتیب تعداد سه، دو و یک کار متعلق به سفارش ۱، ۲ و ۳ است. همچنین دو سفارش ۱ و ۳ از یک خانواده هستند و بنابراین کارهای این دو سفارش می‌توانند در یک دسته قرار بگیرد. نمایش رشته جواب اول این مثال در شکل (۳) نشان داده شده است. شماره هر دسته در بالای هر آرایه قرار گرفته است. فضای خالی در هر دسته نشان‌دهنده مقدار ظرفیت خالی آن دسته است.

رشته جواب دوم توالی دسته‌ها در مرحله سوم را نشان می‌دهد. ساختار جواب یک سطر K درایه است که K تعداد

| ۲ | ۱ | ۳ |
|---|---|---|

**شکل ۴- نمایش رشته جواب دوم**

دسته‌ها را نشان می‌دهد. نمایش رشته جواب دوم برای این مثال در شکل (۴) نمایش داده شده است.

### ۶-۲- جواب اولیه

در بسیاری از الگوریتم‌ها جواب اولیه به‌صورت تصادفی تولید می‌شود. در این مقاله یک جواب اولیه توسط روش ابتکاری HJABSS که در بخش (۵) ارائه شد به‌دست آمده است.

### ۶-۳- مرحله تکانش

در این مرحله نیاز است که در ابتدا انواع مختلف همسایگی‌هایی که استفاده خواهند شد تعریف شوند. با توجه به نوع مسئله، چهار تکانش بر روی جابه‌جایی دسته‌ها و بر روی جابه‌جایی کارها تعریف می‌شود. در ادامه پارامترهایی که در الگوریتم VNS مورد نیاز است بیان شده است. نمادهای تعریف شده $O$، $K$، $d_i$، $Ct_{sk}$، $P_{sk}$، $V_{ik}$ و $Q_{ik}$ در مدل برنامه‌ریزی ریاضی، در الگوریتم VNS پیشنهادی نیز به‌کار رفته‌اند.

در ادامه تکانش $N_1$ که مربوط به جابه‌جایی دسته است و تکانش‌های $N_2$، $N_3$ و $N_4$ که مربوط به تکانش‌های جابه‌جایی کارها می‌باشند، توضیح داده می‌شود. در این تکانش‌ها، دسته‌ای که در آن سفارشی وجود دارد که دچار دیرکرد شده، دسته دیرکرددار نام‌گذاری شده و همچنین یک رشته جواب درصورتی امکان‌پذیر است که محدودیت ظرفیت دسته، خانواده و نوع کار در آن رعایت شود. دو دسته با نام‌های انتقال دهنده و گیرنده نیز تعریف شده که در صورت امکان‌پذیری، تعدادی کار به‌طور تصادف از دسته انتقال‌دهنده به دسته گیرنده منتقل می‌شود.

**جابه‌جایی دسته ($N_1$):** در این ساختار به‌صورت تصادفی یکی از دسته‌های دیرکرددار انتخاب می‌شود (مانند دسته‌ای که در جایگاه $k$ قرار دارد) و اگر رابطه (۴۲) بین دسته $seq_k$ و دسته $seq_{k+j}$ برقرار بود دسته $seq_k$ با دسته $seq_{k+j}$ جابه‌جا می‌شود. در غیر این صورت دسته $seq_k$ با زودترین دسته (مانند دسته‌ای که در جایگاه $k'$ قرار دارد) که قبل از آن وارد مرحله اول شده و در رابطه (۴۳) صدق می‌کند در توالی ورودی جابه‌جا می‌شود.

$$Ct_{۳seq_{k+j}} - p_{۳seq_{k+j}} - p_{۲seq_{k+j}} + p_{۳seq_k} + p_{۲seq_k} \le v_{iseq_k} \times d_i \quad i = ۱,\dots,O;\ j = ۱,\dots,m \qquad (۴۲)$$

$$Ct_{۳seq_k} - p_{۳seq_k} - p_{۲seq_k} + p_{۳seq_{k'}} + p_{۲seq_{k'}} \le v_{iseq_{k'}} \times d_i \quad i = ۱,\dots,O \qquad (۴۳)$$

**روش اول انتقال کارها به دسته‌های موجود ($N_2$):** بر اساس لم ۳ تعداد $q'$ کار که می‌تواند به هر دسته انتقال یابد بدون اینکه جواب بدتر شود و بر اساس لم ۴ تعداد $q''$ کار که می‌تواند از هر دسته انتقال باید بدون اینکه جواب بدتر شود، محاسبه می‌شود. سپس دسته انتقال دهنده به‌صورت تصادفی از بین دسته‌هایی که تعداد $q''$ آنها بیشتر از صفر است انتخاب شده و دسته گیرنده از بین دسته‌هایی که تعداد $q'$ آنها بزرگ‌تر از صفر است و کارهای آن هم نوع و هم خانواده دسته انتقال دهنده است، به‌صورت تصادفی انتخاب می‌شود.

**روش دوم انتقال کارها به دسته‌های موجود ($N_3$):** دسته انتقال‌دهنده به‌صورت تصادفی از بین دسته‌های دیرکرددار و دسته‌هایی که در توالی ورودی بلافاصله قبل از این دسته‌ها قرار دارند انتخاب شده و دسته گیرنده از بین دسته‌هایی که در توالی ورودی قبل از دسته انتقال‌دهنده وجود دارند و کارهای آن هم نوع و هم خانواده دسته انتقال دهنده است، انتخاب می‌شود. تعدادی کار، حداکثر به اندازه ظرفیت خالی دسته گیرنده، از دسته انتقال‌دهنده به دسته گیرنده انتقال داده می‌شود.

**انتقال کار به دسته جدید ($N_4$):** دسته انتقال‌دهنده از بین دسته‌های دیرکرددار، دسته‌هایی که در توالی ورودی بلافاصله

قبل از این دسته‌ها وجود دارند و دسته‌هایی که زمان تکمیل آنها از مرحله اول و زمان ترک آنها از مرحله اول یکسان است به‌صورت تصادفی انتخاب می‌شود. تعدادی کار به‌صورت تصادفی از دسته انتقال‌دهنده انتخاب می‌شود و به دسته جدید در توالی ورودی پس از دسته انتقال‌دهنده انتقال داده می‌شود.

لازم به‌ذکر است که مطابق آزمایشات عددی اولیه، تکانش‌های معرفی شده در بالا به همان ترتیب معرفی شده، یعنی $N_1$، $N_2$، $N_3$ و $N_4$ در الگوریتم مورد استفاده قرار گرفته‌اند.

### ۶-۴- جستجوی محلی

در الگوریتم VNS پیشنهادی برای تخصیص کارها به دسته‌ها، از سه عملگر mergebatch_splitbatch و swaphalfbatch، برای تعیین توالی دسته‌ها از عملگر movebatch و swapbatch و برای زمان‌بندی دسته‌ها از عملگر movebatch استفاده شده است. در ادامه این عملگرها شرح داده شده‌اند.

**عملگر** mergebatch_splitbatch: ابتدا یک عدد تصادفی بین صفر و یک به نام $\alpha$ تولید می‌شود. اگر این عدد تصادفی بزرگ‌تر از ۰/۲ (مقدار ۰/۲ بر اساس روش تاگوچی تنظیم شده است) بود، از بین دسته‌هایی که ظرفیت آنها پر نشده است یک دسته انتخاب شده و دسته گیرنده نامیده می‌شود. از میان دسته‌هایی که پس از آن دسته وجود دارد و امکان‌پذیری را رعایت کنند، یک دسته به‌صورت تصادفی انتخاب و انتقال‌دهنده نامیده می‌شود. تعدادی تصادفی کار، حداکثر به اندازه ظرفیت خالی دسته گیرنده، از دسته انتقال‌دهنده به دسته گیرنده انتقال داده می‌شود.

اگر $\alpha$ بزرگ‌تر از ۰/۲ نبود به‌صورت تصادفی از بین دسته‌هایی که زمان تکمیل آنها از مرحله اول و زمان ترک آنها از مرحله اول یکسان است و دسته‌های دیرکردار، دسته‌هایی که قبل از دسته‌های دیرکردار وارد مرحله اول شده‌اند دسته‌ای انتخاب می‌شود. این دسته، دسته انتقال‌دهنده نامیده می‌شود. یک دسته جدید ساخته می‌شود. به‌صورت تصادفی تعدادی کار از دسته انتقال‌دهنده به دسته جدید انتقال داده می‌شود. دسته جدید در توالی ورودی در فاصله‌ای کمتر از دو برابر تعداد ماشین موازی پس از دسته انتقال‌دهنده قرار می‌گیرد.

**عملگر** swapbatch: بر اساس لم ۳، $q'$ هر دسته محاسبه می‌شود. سپس به‌صورت تصادفی از بین دسته‌هایی که $q'$ آن بزرگ‌تر از صفر است یک دسته انتخاب شده و به‌صورت تصادفی با دسته دیگر که هم خانواده با آن نیست جا به‌جا می‌شود.

**عملگر** movebatch1: به‌صورت تصادفی از بین دسته‌هایی که زمان تکمیل آنها از مرحله اول و زمان ترک آنها از مرحله اول یکسان است، دسته‌های دیرکردار و دسته‌هایی که قبل از دسته‌های دیرکردار وارد مرحله اول شده‌اند، دسته‌ای انتخاب می‌شود. جایگاه این دسته در توالی ورودی مرحله اول به‌صورت تصادفی به جایگاهی که کمتر از دو برابر تعداد ماشین موازی با آن فاصله دارد منتقل می‌شود.

**عملگر** swaphalfbatch: به‌صورت تصادفی یک دسته انتخاب می‌شود. کارهایی که در نیمه دوم این دسته قرار دارد با کارهایی که در نیمه اول دسته‌ای دیگر که در توالی ورودی مرحله اول پس از این دسته قرار دارد به شرطی که این دو نیم دسته از یک خانواده باشند ولی از یک نوع کار نباشند، جا به‌جا می‌شود.

**عملگر** movebatch2: ابتدا برای جایگاه هر دسته در توالی خروجی یک بازه محاسبه می‌شود. این بازه تضمین می‌کند توالی خروجی جدید، با توالی ورودی همخوانی داشته باشد. در واقع توالی خروجی امکان‌پذیر شود. اولین جایگاهی که دسته k بتواند در آن قرار گیرد $a_k$ و آخرین جایگاهی که دسته k بتواند در آن قرار گیرد $b_k$ نامگذاری شده است. برای هر دسته (k) که کاری از نوع کار j در آن وجود دارد، اولین دسته که پس از آن نیز کاری از نوع کار j در آن وجود دارد پیدا کرده و مقدار $b_k$ به حداقل مقدار $b_k$ و جایگاه اولین دسته پس از دسته k که نوع کاری مشابه دارد، تغییر می‌کند. برای مقدار $a_k$ نیز به حداکثر مقدار $a_k$ و جایگاه اولین دسته قبل از دسته k

```
Local search(input: x′ ∈ S, E ,output: x″)
Set q = 0 and k =1
While (iteration_local ≤IL_max)
for k=1,…,3 do
 x″ =Local search (x′,k)
While (iteration_localOut ≤ILO_max)
 x″ =Local search (x′,4)
iteration_localOut= iteration_localOut+1;
End while
If f( x″) - f( x′)<E
 x′ = x″
End If
iteration_local = iteration_local +1;
End while
 x″ = x′
```

**شکل ۵- شبه کد جستجوی محلی**

که نوع کاری مشابه دارد، تغییر می‌کند. سپس به‌صورت تصادفی یک دسته انتخاب شده و جایگاه این دسته در توالی خروجی مرحله سوم به‌صورت تصادفی به جایگاهی در بازه $[a_k, b_k]$ منتقل می‌شود. این فرایند برای $\ln(k \times m)$ دسته (که بر اساس روش تاگوچی تنظیم شده است) انجام می‌شود.

در جستجوی محلی اول ابتدا عملگر mergebatch_ splitbatch و سپس عملگر swapbatch بر روی رشته جواب اجرا می‌شود. همچنین جستجوی محلی دوم ترکیب دو عملگر movebatch1 و mergebatch_splitbatch است و جستجوی محلی سوم ترکیب دو عملگر swaphalfbatch و movebatch1 است. در جستجوی محلی چهارم ابتدا با استفاده از رویه UBSC یک توالی برای پردازش دسته‌ها در مرحله سوم و بر اساس توالی دسته‌ها در مرحله اول در جواب موجود محاسبه می‌شود، سپس عملگر movebatch2 اجرا می‌شود.

در جستجوی محلی مقدار تابع هدف جواب جدید با جواب فعلی مقایسه می‌شود، اگر جواب جدید بهتر بود جایگزین جواب فعلی می‌شود و اگر جواب جدید بدتر بود با احتمال E جایگزین جواب فعلی می‌شود. لذا این روش منجر به گیرافتادن در بهینه محلی نمی‌شود. برای تعیین احتمال پذیرش یک جواب بدتر، از پارامتر E استفاده می‌شود. پارامتر MaxE به‌عنوان حداکثر تعداد تکرار بدون بهبود تعیین می‌شود و $MaxE_0$ مقدار اولیه حداکثر تعداد تکرار بدون بهبود است. در ابتدای فرایند، مقدار MaxE برابر $MaxE_0$ قرار می‌گیرد. پس از اینکه تعداد تکرارهای بدون بهبود از مقدار MaxE بیشتر شد مقدار پارامتر E به $E \times \gamma$ تغییر داده می‌شود و MaxE برابر مجموع MaxE و $MaxE_0$ قرار می‌گیرد. لازم به‌ذکر است که $\gamma$ یک ضریب تعدیل است و در بخش تنظیم پارامترهای الگوریتم مقدار آن تعیین می‌شود. شبه کد رویه جستجوی محلی در شکل (۵) قابل مشاهده است.

### ۶-۵- شرط توقف

دو شرط توقف برای این الگوریتم در نظر گرفته شده است. در صورتی که مدت زمان اجرا از زمان total_time گذشت به ساختار همسایگی بعد نرفته و الگوریتم متوقف می‌شود. شرط توقف دیگر آن است که در صورت عدم بهبود جواب موجود تا مدت زمان مشخص، الگوریتم متوقف می‌شود.

## ۷- الگوریتم ممتیک

الگوریتم ممتیک (MA) روشی برای یافتن جواب در مسائل بهینه‌سازی است که زیرمجموعه‌ای از الگوریتم‌های تکاملی[۲۷] به شمار می‌رود. در این مقاله الگوریتم ممتیک پیشنهادی از ترکیب الگوریتم ژنتیک و جستجوی محلی توسعه داده شده است. این

VNS(input: popsize, $P_m, P_c$ ,output:best chromosome)

**For** i=1 to popsize

$pop_i = randomsolution(n)$

$pop_i = localsearch(pop_i)$

Evaluate( $pop_i$ )

**End for**

**Repeat** the following until the stopping criteria (Current_time ≥ nonImp_time and Current_time ≥ total_time) are met:

**While** ($i \le popsize$)

**If** ($p < P_c$) // p is a random number between 0 and 1

$parent_1 = select(pop)$

$parent_2 = select(pop)$

$offspring_i, offspring_{i+1} = crossover(parent_1, parent_2)$

**Else**

$offspring_i = select(pop)$

**End If**

If ($p < P_m$)

$offspring_i = mutate(offspring_i)$

**End If**

$offspring_i = localsearch(offspring_i)$

$Evaluate(offspring_i)$

**End while**

Update population and best current chromosome

**شکل ۶- شبه کد الگوریتم MA پیشنهادی**

الگوریتم یک الگوریتم مبتنی بر جمعیت است، ار این‌رو به‌طور تصادفی یا با یک یا چند روش ابتکاری جمعیت اولیه تولید می‌شود. این الگوریتم از عملگرهای تقاطع، جهش و انتخاب جهت جستجوی فضای جواب مسئله استفاده می‌کند. عملگرهای ژنتیک، جستجوی محلی و شرط توقف مورد استفاده در این مقاله، در ادامه توضیح داده شده است. همچنین مجموعه پارامترهای مورد نیاز برای توضیح این الگوریتم به‌صورت زیر تعریف شده‌اند.

| | |
|---|---|
| Popsize | اندازه جمعیت |
| $P_c$ | نرخ تقاطع |
| $P_m$ | نرخ جهش |
| $pop_i$ | جواب i ام |
| Parents | والدین |
| Offsprings | فرزندان |
| Chromosome | رشته جواب |
| Mutate | جهش |
| Local_search | جستجوی محلی |
| Evaluate | ارزیابی جواب موجود |
| Select | انتخاب رشته جواب از جمعیت موجود |
| Current_time | مدت زمان اجرای فعلی |
| nonImp_time | حداکثر مدت زمان مجاز عدم بهبود |
| Total_time | حداکثر مدت زمان کل |

نمای کلی الگوریتم ممتیک پیشنهادی مشابه شبه کد نشان داده شده در شکل (۶) است.

| ۱ | ۲ | ۶ | -۱ | ۴ | ۵ | -۳ | ۳ | -۲ |
|---|---|---|---|---|---|---|---|---|

**شکل ۷- نمایش رشته جواب اول ممتیک**

| ۲ | ۱ | ۳ |
|---|---|---|

**شکل ۸- نمایش رشته جواب دوم**

### ۷-۱- نمایش جواب

ساختار کروموزوم پیشنهادی از دو رشته جواب تشکیل شده است. رشته اول مربوط به ساختار دسته‌ها و توالی پردازش آنها در مرحله اول است. ساختار رشته جواب اول، یک سطر K+N درایه‌ای است که اعداد ۱ تا N نشان‌دهنده تعداد کارها است. همچنین k عدد منفی در این ساختار وجود دارد که نشان‌دهنده شماره دسته‌ها و جدا کننده کارهای دسته‌ها است. جهت مشخص شدن ساختار یک رشته جواب، مثالی با سه سفارش و شش کار تعریف شده است. در این مثال کار ۱، ۲ و ۳ متعلق به سفارش ۱، کار ۴ و ۵ متعلق به سفارش ۲ و کار ۶ متعلق به سفارش ۳ است. همچنین دو سفارش ۱ و ۳ از یک خانواده هستند. در شکل (۷) رشته جواب اول این مثال مشاهده می‌شود در این رشته، کار ۱، ۲ و ۶ در دسته ۱، کار ۳ در دسته ۲ و کار ۴ و ۵ در دسته ۳ پردازش می‌شوند.

رشته جواب دوم توالی دسته‌ها در مرحله سوم را نشان می‌دهد. ساختار این رشته جواب یک سطر با K درایه است که K تعداد دسته‌ها را نشان می‌دهد. در شکل (۸) رشته جواب دوم مثال نمایش داده شده است. در این رشته جواب ابتدا دسته ۲ از مرحله دوم خارج شده و در مرحله سوم پردازش می‌شود. سپس دسته ۱ و ۳ به‌ترتیب در مرحله سوم پردازش می‌شوند.

### ۷-۲- تولید جواب‌های اولیه

برای به‌دست آوردن جواب اولیه از سه روش تصادفی، HJABSS و ترتیب کمترین موعد تحویل (EDD)[۲۸] استفاده می‌شود. در روش تصادفی ابتدا سفارشات به‌صورت تصادفی مرتب می‌شود و در روش EED سفارشات به‌ترتیب EDD مرتب می‌شود. سپس در هر دو روش به‌ترتیب، کارهای سفارشات بر اساس خانواده، نوع کار و ظرفیت خانواده هر سفارش به دسته‌ها تخصیص داده می‌شود. مقدار ظرفیت هر دسته، برابر با ضریبی از ظرفیت خانواده قرار داده می‌شود. هر چه تعداد جواب‌های اولیه بیشتر می‌شود، مقدار این ضریب از ۱ به ۰/۷۵ کاهش می‌یابد.

### ۷-۳- عملگرهای ممتیک

**عملگر انتخاب:** برای انتقال کروموزوم‌ها به نسل بعد از روش چرخ رولت[۲۹] استفاده شده است. بر اساس این روش برای هر کروموزوم یک احتمال انتخاب در نظر گرفته می‌شود. این احتمال بر اساس برازندگی آن محاسبه می‌شود. در این مقاله برازندگی هر کروموزوم برابر با معکوس تابع هدف آن کروموزوم در نظر گرفته شده است.

**عملگر تقاطع:** تقاطع فرایندی است که در آن نسل قدیمی کروموزوم‌ها با یکدیگر ترکیب می‌شوند تا نسل تازه‌ای از کروموزوم‌ها به‌وجود بیایند. در الگوریتم ممتیک پیشنهادی با احتمال ۰/۲ از عملگر تقاطع تک نقطه‌ای، با احتمال ۰/۴ از عملگر تقاطع دو نقطه‌ای و با احتمال ۰/۴ از عملگر تقاطع n نقطه‌ای استفاده می‌شود. مقادیر این احتمالات بر اساس روش تاگوچی محاسبه شده است. پس از اعمال یکی از عملگرهای تقاطع ممکن است برخی فروض مسئله در رشته جواب فرزند نقص شده باشد. به‌طور مثال ممکن است در یک دسته کارهایی از چند خانواده یافت شود و یا اینکه کارهای یک دسته از یک خانواده اما بیش از دو نوع باشد. بنابراین کارهای این دسته در دو یا چند دسته انجام می‌شود. به‌عبارتی یک یا چند ژن منفی ما بین ژن‌های آن دسته جایگزین می‌شود تا رشته جواب به‌دست آمده امکان‌پذیر شود.

**عملگر تقاطع تک‌نقطه‌ای:** در این عملگر کروموزوم‌های والد از یک نقطه برش داده می‌شوند. بنابراین هر کروموزوم والد به دو قسمت تقسیم می‌شود. قسمت اول والد دوم و بقیه کروموزوم‌های والد اول که در قسمت اول والد دوم وجود

ندارد، کروموزوم جدید اول را می‌سازد. همچنین قسمت اول والد اول و کروموزوم‌های والد دوم که در قسمت اول والد اول وجود ندارد، کروموزوم جدید دوم را می‌سازد. نقطه برش بلافاصله پس از یک ژن با مقدار منفی انتخاب می‌شود. یادآوری می‌شود که هر ژن با مقدار منفی نشان دهنده شماره دسته است. برای انتخاب نقطه برش برای هر ژن با مقدار منفی یک احتمال انتخاب در نظر گرفته می‌شود. این احتمال با مجموع دیرکرد ژن‌های درون دسته مورد نظر و دسته‌های قبل از آن رابطه معکوس دارد.

**عملگر تقاطع دو نقطه‌ای**: در این عملگر کروموزوم‌های والد از دو نقطه برش داده می‌شوند. بنابراین هر کروموزوم والد به سه قسمت تقسیم می‌شود. با جابه‌جایی قسمت میانی دو کروموزوم والد، دو فرزند به‌دست می‌آید، به‌طوری که ژن‌هایی که در قسمت میانی وجود دارد در قسمت اول و سوم تکرار نمی‌شوند. انتخاب نقطه برش در عملگر تقاطع دونقطه‌ای مشابه عملگر تقاطع تک نقطه‌ای است با این تفاوت که هرکدام از ژن‌ها می‌توانند برای برش انتخاب شوند.

**عملگر تقاطع n نقطه‌ای**: در این عملگر کروموزوم‌های والد از n نقطه برش داده می‌شوند. بنابراین هر کروموزوم والد به n+1 قسمت تقسیم می‌شود. با انتخاب n موقعیت ترکیب و چیدن ژن‌ها مشابه آنچه در تقاطع تک نقطه‌ای و دو نقطه‌ای گفته شد، ترکیب n نقطه‌ای خواهیم داشت. نقطه‌های برش، تنها بعد از ژن‌های مشخص‌کننده دسته یعنی ژن‌های با مقدار منفی، انتخاب می‌شود. برای انتخاب نقطه برش برای هر ژن والد اول یک احتمال انتخاب در نظر گرفته می‌شود. این احتمال با مجموع دیرکرد ژن‌ها از ابتدا تا ژن مورد نظر رابطه معکوس دارد.

**عملگر جهش**: در این مرحله به‌ترتیب با احتمال ۰/۳، ۰/۲ و ۰/۵ از عملگر جهش جانشینی، عملگر جهش جابه‌جایی تصادفی و عملگر جهش جابه‌جایی استفاده می‌شود. مقادیر این احتمالات بر اساس روش تاگوچی محاسبه شده است.

**عملگر جهش جانشینی**: ابتدا یک ژن به تصادف انتخاب می‌شود. مجموعه‌ای از ژن‌های هم علامت آن که مطابق با فروض مسئله قابلیت جابه‌جایی با ژن انتخاب شده را دارند و از یک سفارش نیستند تعریف می‌شود. از این مجموعه یک ژن انتخاب شده و با ژن انتخاب شده اول جابه‌جا می‌شود.

**عملگر جهش جابه‌جایی تصادفی**: یک ژن با مقدار کمتر و مساوی N از کروموزوم انتخاب می‌شود. یک مجموعه از ژن هایی که می‌توانند بعد از آنها جایگذاری شود به شرطی که جواب امکان‌پذیر باشد تعریف می‌شود. از این مجموعه یک ژن انتخاب شده و ژن انتخاب شده اول پس از آن جایگذاری می شود.

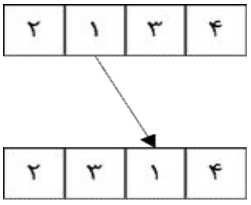
**عملگر جهش جابه‌جایی**: بر اساس لم ۴ تعداد "q" کار که می‌تواند از هر دسته انتقال یابد بدون اینکه جواب بدتر شود محاسبه می‌شود. سپس به‌صورت تصادفی یک ژن از بین دسته‌هایی که تعداد "q" آنها بیشتر از صفر است، انتخاب می‌شود. آن ژن در دسته‌ای که کارهای آن هم نوع و هم خانواده ژن انتخاب شده است، جایگذاری می‌شود.

### ۷-۴- جستجوی محلی

در این بخش مراحل تعریف شده در جستجوی محلی چهارم VNS اجرا می‌شود. با این تفاوت که این فرایند برای $\ln\left(K^{2}\times m\right)$ دسته انجام می‌شود. شکل (۹) نحوه اعمال این جستجوی محلی را نشان می‌دهد.

## ۸- آزمایش‌های عددی

به‌منظور ارزیابی مدل BSS، روش فراابتکاری VNS و روش فراابتکاری MA، عملکرد آنها در حل تعدادی مسئله نمونه بررسی شده است. الگوریتم‌های ذکر شده در محیط Visual C# 2012 کدنویسی شده و مسائل نمونه بر روی یک رایانه با مشخصات Intel(R) Core(TM) i7-6700 CPU @ 4.00 GHz و 8.00 GB RAM در محیط Windows 7-64 bit حل شده است. مدل ریاضی BSS با استفاده از قابلیت Concert Technology نرم‌افزار CPLEX 12.4 در محیط Visual C#




**شکل ۹- نمایش جستجوی محلی**

۲۰۱۲ کدنویسی شده و به‌وسیله رایانه‌ای با مشخصات ذکر شده مورد آزمایش قرار گرفته است. قابل ذکر است که محدودیت زمانی ۳۶۰۰ ثانیه برای حل مدل ریاضی لحاظ شده است.

### ۸-۱- نحوه تولید مسائل نمونه

برای بررسی عملکرد روش‌های ارائه شده در این بخش چگونگی تولید مسائل نمونه تصادفی تشریح شده است. در محیط مورد بررسی، ظرفیت دسته وابسته به خانواده است. برای این‌که مسائل نمونه همه حالت‌ها را پوشش دهد ظرفیت دسته‌ها به‌صورتی که هم کمتر و هم بیشتر از ماشین‌های موازی باشند پیشنهاد شده است.

بر اساس مطالعه تان و همکاران [۱۶] ابعاد سفارش به‌ازای هر خانواده تعیین شده است. از آنجایی که هر سفارش می‌تواند در یک دسته قرار گیرد لذا تعداد کار در یک سفارش برای مسئله مورد بررسی کمتر از ظرفیت دسته در نظر گرفته شده است [۲۰]. تعداد خانواده بر اساس مقاله تان و همکاران [۱۶]، برابر ۴ و ۶ فرض شده است.

در مسئله مورد بررسی علاوه بر خانواده، تعداد نوع وجود دارد که کمتر از تعداد خانواده است. بنابراین دو مقدار ۲ و ۵ برای آن تعریف شده است. همچنین در مرحله اول و سوم فقط یک ماشین وجود دارد. تعداد ماشین‌های موازی در مرحله دوم برای ابعاد کوچک برابر ۲، ۴ و ۶ در نظر گرفته شده است به شرطی که تعداد سفارشات کمتر از ۳ برابر تعداد ماشین موازی در مرحله دوم نباشد. از این رو تعداد سفارشات در ابعاد کوچک اعداد ۶، ۸، ۱۰، ۱۲، ۱۴، ۱۶و ۱۸ در نظر گرفته شده است. تعداد ماشین‌های موازی در مرحله دوم برای ابعاد بزرگ برابر ۴، ۶، ۱۰ و ۱۵ در نظر گرفته شده است.

میزان بارکاری[۳۰] هر مرحله برابر با فاصله زمانی بین خروج دو کار متوالی از آن مرحله است. هر چقدر تعداد ماشین در یک مرحله بیشتر باشد احتمال اینکه آن مرحله گلوگاه باشد کمتر می‌شود. اگر N تعداد کارها، $f(i)$ خانواده کار $i$ و $ms_j$ تعداد ماشین موازی مرحله $j$ باشد میزان بارکاری در مرحله $j$ بر اساس رابطه (۴۴) محاسبه می‌شود [۱۶].

$$wl_j = \frac{1}{(ms_j)} \sum_{i=1}^{N} \frac{t_{jf(i)}}{cap_{jf(i)}} \qquad (۴۴)$$

هنگامی که $\left| \frac{1}{(ms_j)} \sum_{i=1}^{N} \frac{t_{jf(i)}}{cap_{jf(i)}} - \frac{1}{(ms_k)} \sum_{i=1}^{N} \frac{t_{kf(i)}}{cap_{kf(i)}} \right|$ برای هر دو مرحله $j$ و $k$ کمترین مقدار را بگیرد بین دو مرحله تعادل وجود دارد. از آنجا که در مسئله مورد بررسی تعداد ماشین در مرحله اول و سوم برابر یک و ظرفیت دسته هم برابر یک است، در نتیجه زمان دو مرحله تقریباً باید با هم برابر باشند. برای اینکه خط مورد نظر در مسئله بالانس باشد، مابین مرحله اول و دوم رابطه (۴۵) و ما بین مرحله دوم و سوم رابطه (۴۶) در نظر گرفته شده است.

$$\frac{2m(m+1)\sum_{i=1}^{N} t_{1f(i)}}{2m+1} > \sum_{i=1}^{N} \frac{t_{2f(i)}}{cap_{f(i)}} > \frac{2m(m-1)\sum_{i=1}^{N} t_{1f(i)}}{2m-1} \qquad (۴۵)$$

$$\frac{2m(m+1)\sum_{i=1}^{N} t_{3f(i)}}{2m+1} > \sum_{i=1}^{N} \frac{t_{2f(i)}}{cap_{f(i)}} > \frac{2m(m-1)\sum_{i=1}^{N} t_{3f(i)}}{2m-1} \qquad (۴۶)$$

برای این‌که طبق تعریف مسئله، بین مرحله دوم و مرحله سوم انسداد به‌وجود آید باید حداقل مدت زمان پردازش یک کار از یک خانواده در مرحله سوم از مرحله اول بیشتر باشد ولی مدت زمان پردازش یک کار از یک خانواده در مرحله سوم باید از دو برابر مدت زمان پردازش یک کار از یک خانواده در مرحله اول کمتر باشد (یا میانگین آن کمتر باشد). اگر این مقدار بیشتر از دو برابر باشد نشان می‌دهد تعداد ماشین در مرحله سوم باید بیش از یک ماشین باشد. برای این‌که این شرایط حاصل شود هر کار مانند i باید در روابط (۴۵) و (۴۶) صدق کند و همچنین رابطه $t_{1f(i)} \le t_{3f(i)} \le 2t_{1f(i)}$ نیز برقرار باشد. فصل اشتراک این روابط، روابط (۴۷) و $t_{3f(i)} = U\left( t_{1f(i)}, \min\left( \left( \frac{2m^2+m-1}{2m^2-m-1} \right) t_{1f(i)}, 2t_{1f(i)} \right) \right)$ است:

$$t_{2f(i)} = U\left( \frac{2cap_{f(i)} m(m-1) t_{3f(i)}}{2m-1}, \frac{2cap_{f(i)} m(m+1) t_{1f(i)}}{2m+1} \right) \qquad (47)$$

با توجه به مطالب بیان شده مدت زمان‌های پردازش در مراحل در دو حالت به‌صورتی که در جدول (۲) مشاهده می‌شود تولید می‌شوند.

در مسئله مورد بررسی موعد تحویل هر سفارش اهمیت دارد و همان‌طور که در تعریف مسئله ذکر شد، تمامی کارهای یک سفارش باید با هم تحویل داده شوند. موعد تحویل بهتر است ضریبی از $\left( n_i \times t_{1f(i)} + t_{2f(i)} + n_i \times t_{3f(i)} \right)$ باشد. این ضریب طبق مطالعه تان و همکاران [۱۶] باید بزرگ‌تر مساوی یک باشد، پس مقدار یک نیز برای آن در نظرگرفته می‌شود. از دیگر موارد دارای اهمیت در تعیین موعد تحویل در یک مسئله، درصد سفارشاتی است که دیرکرد دارند. پس موعد تحویل باید ضریبی از زمان اتمام سفارشات نیز باشد. در نتیجه فاکتور $R^{31}$ برای کنترل دامنه موعدتحویل و فاکتور $T^{32}$ برای نشان دادن درصد تقریبی کارهای دیرکرددار تعریف شده است [۹]. برای R مقادیر ۱/۰، ۱/۳ و ۱/۷ و برای T مقادیر ۰/۲۵ و ۰/۷۵ در نظر گرفته شده است [۱۶].

برای هر کدام از ابعاد، ۱۰ مسئله نمونه ایجاد شده است. بنابراین در مجموع ۱۹۲۰ (۲۴×۵×۱۰+۲۴×۲×۱۰+۲۴×۱×۱۰) مسئله نمونه تصادفی در ابعاد کوچک مورد آزمایش قرار گرفته و در ابعاد بزرگ ۲۸۸۰ (۲۴×۴×۳×۱۰) مسئله نمونه تصادفی بررسی شده است.

### ۸-۲- تنظیم پارامتر در الگوریتم‌های فراابتکاری

از آنجایی که کیفیت عملکرد الگوریتم‌های فراابتکاری تا حد زیادی به پارامترها و عملگرهای انتخابی بستگی دارد، در این مقاله همه پارامترها و عملگرهای الگوریتم‌های پیشنهادی به‌طور جداگانه با استفاده از روش طراحی آزمایشات تاگوچی تنظیم شده است.

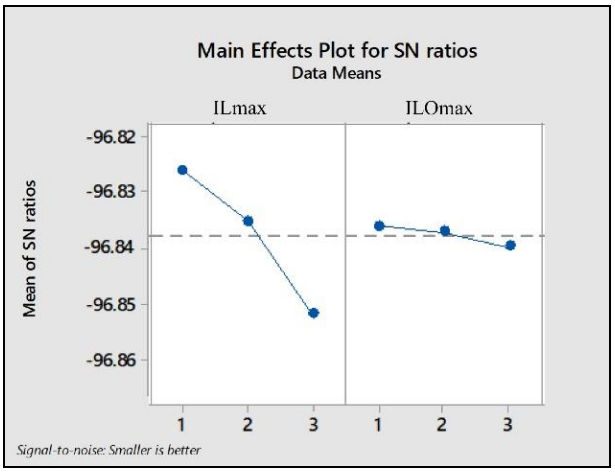
به‌طور معمول الگوریتم VNS دارای پارامترهای بیشینه تعداد تکرار حلقه جستجو محلی ( $IL_{max}$ ) و بیشینه تعداد تکرار حلقه جستجوی محلی توالی خروجی ( $ILO_{max}$ ) است. پارامترهای دیگر همچون احتمال پذیرش جواب بدتر (E) و ضریب احتمال پذیرش جواب بدتر ( $\gamma$ ) در این مقاله معرفی شده است. درنهایت، حداکثر مدت زمان مجاز عدم بهبود (nonImp_time) و حداکثر مدت زمان کل (total_time) مربوط به معیار توقف نیز باید تعیین شوند. پارامترهای در نظر گرفته شده در الگوریتم VNS پیشنهادی به همراه سطوح در نظر گرفته شده برای آنها در جدول (۳) آمده است. با استفاده از روش تاگوچی از بین سطوح مختلف ارائه شده، پارامتر تعداد تکرار حلقه جستجو محلی برابر با $\ln\left(O^2\right)$، بیشینه تعداد تکرار حلقه جستجوی محلی توالی خروجی برابر ۴، احتمال پذیرش جواب بدتر برابر ۰/۰۲، ضریب احتمال پذیرش جواب بدتر برابر با ۱/۵، حداکثر مدت زمان کل برابر با دو برابر تعداد سفارش و حداکثر مدت زمان مجاز عدم بهبود برابر با $\max\left(6, \frac{\text{total time}}{6}\right)$ تنظیم شده است. در شکل (۱۰) به‌عنوان نمونه نسبت سیگنال به نویز برای دو پارامتر از الگوریتم VNS نشان داده شده است. لازم به‌ذکر است که سطح با نسبت سیگنال به نویز بیشتر انتخاب می‌شود.

**جدول ۲- نحوه تولید داده‌های مسائل نمونه**

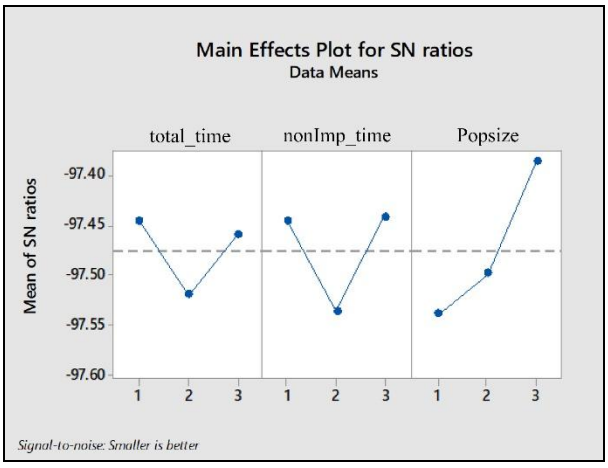
| | فاکتور | مقدار | تعداد |
|---|---|---|---|
| ابعاد کوچک | تعداد سفارشات(**O**) | ۶، ۸، ۱۰، ۱۴ و ۱۸ | ۵ |
| | تعداد کار در سفارش **i** ( $n_i$ ) | $n_i = U\left[1, \min\left(8, cap_{f(i)}\right)\right]$ | ۱ |
| | تعداد خانواده (**F**) | ۴ و ۶ | ۲ |
| | تعداد نوع ( $ng_f$ ) | ۲ و ۵ | ۲ |
| | تعداد ماشین مرحله دوم (**m**) | ۲، ۴ و ۶ ( $m \le \frac{O}{3}$ ) | ۳ |
| ابعاد بزرگ | تعداد سفارش به‌ازای هر خانواده ( $O_f$ ) | ۵۰، ۷۰ و ۱۰۰ | ۳ |
| | تعداد کار هر در سفارش ( $n_i$ ) | $n_i = U\left[1, \min\left(8, cap_{f(i)}\right)\right]$ | ۱ |
| | تعداد خانواده (**F**) | ۶ و ۱۲ | ۲ |
| | تعداد نوع ( $ng_f$ ) | ۲ و ۵ | ۲ |
| | تعداد ماشین مرحله دوم (**m**) | ۴، ۶، ۱۰ و ۱۵ | ۴ |
| | زمان پردازش یک کار خانواده f در مرحله اول ( $t_{1f}$ ) | ۵ با احتمال ۰/۲، ۱۰ با احتمال ۰/۳، ۱۵ با احتمال ۰/۳ و ۲۰ با احتمال ۰/۲ | ۱ |
| | زمان پردازش یک کار خانواده f در مرحله دوم ( $t_{2f}$ ) | $t_{2f} = U\left(\frac{2cap_f m(m-1)t_{3f}}{(2m-1)}, \frac{2cap_f m(m+1)t_{1f}}{(2m+1)}\right)$ | ۱ |
| | زمان پردازش یک کار خانواده f در مرحله سوم ( $t_{3f}$ ) | $t_{3f} = U(t_{1f}, \min\left(\left(\left(\frac{(2m^2+m-1)}{(2m^2-m-1)}\right)t_{1f}, 2t_{1f}\right)\right)$ | ۱ |
| | حداکثر ظرفیت دسته یک خانواده در مرحله دوم ( $cap_f$ ) | ۲، ۴ و ۶ (**m<6**)، m-۲، m و m+۲ ( $6 \le m > 15$ ) و ۱۲، ۱۶ و ۱۸ (m=۱۵) | ۱ |
| | وزن سفارشات ( $w_i$ ) | $U(0,1)$ | ۱ |
| | موعد تحویل هر سفارش ( $d_i$ ) | $d_i = U\left(0, T\left(\sum_{i'=1}^{N} t_{1f(i')} + \frac{1}{m}\sum_{i'=1}^{N}\frac{t_{2f(i')}}{cap_{f(i')}} + \sum_{i'=1}^{N} t_{3f(i')}\right)\right) + R\left(n_i \times t_{1f(i)} + t_{2f(i)} + n_i \times t_{3f(i)}\right); T = 0/25, 0/75; R = 1, 1/3, 1/7$ | ۶ |

**جدول ۳- پارامترها و سطوح در نظر گرفته شده برای آنها در الگوریتم VNS**

| پارامتر | سطح ۱ | سطح ۲ | سطح ۳ |
|---|---|---|---|
| total_time | O | ۱/۵×O | ۲×O |
| nonImp_time | $\max\left(6, \frac{\text{total time}}{8}\right)$ | $\max\left(6, \frac{\text{total time}}{6}\right)$ | $\max\left(6, \frac{\text{total time}}{4}\right)$ |
| E | ۰/۰۱ | ۰/۰۲ | ۰/۰۴ |
| γ | $\frac{f(\text{new}) - f(\text{best})}{f(\text{new})}$ | ۱/۵ | ۲ |
| $IL_{max}$ | $\ln(O^2)$ | $\ln(O^2 m)$ | ۱۶ |
| $ILO_{max}$ | ۴ | $\ln(O^2 m)$ | ۲m |




**شکل ۱۰- نسبت سیگنال به نویز در روش تاگوچی برای الگوریتم VNS (پارامترهای $IL_{max}$ و $ILO_{max}$)**




**شکل ۱۱- نسبت سیگنال به نویز در روش تاگوچی برای الگوریتم MA (پارامترهای total_time، nonImp_time و Popsize)**

**جدول ۴- پارامترها و سطوح در نظر گرفته شده برای آنها در الگوریتم MA**

| پارامتر | سطح ۱ | سطح ۲ | سطح ۳ |
|---|---|---|---|
| total_time | O | ۱/۵×O | ۲×O |
| nonImp_time | $\max\left(۶,\frac{\text{total time}}{۴}\right)$ | $\max\left(۶,\frac{\text{total time}}{۳}\right)$ | $\max\left(۶,\frac{\text{total time}}{۲}\right)$ |
| Popsize | ۳۰ | ۶۰ | ۱۰۰ |
| $P_C$ | ۰/۶ | ۰/۸ | ۱ |
| $P_m$ | ۰/۰۵ | ۰/۱ | ۰/۲ |

یک الگوریتم MA معمولاً دارای پارامترهای اندازه جمعیت (Popsize)، نرخ تقاطع ($P_c$) و نرخ جهش ($P_m$) است و حداکثر مدت زمان مجاز عدم بهبود (nonImp_time) و حداکثر مدت زمان کل (total_time) مربوط به معیار توقف است. پارامترهای در نظر گرفته شده در الگوریتم MA پیشنهادی به همراه سطوح در نظر گرفته شده برای آنها در جدول (۴) آمده است. با استفاده از روش تاگوچی از میان سطوح در نظر گرفته شده برای پارامترها، پارامتر اندازه جمعیت برابر با ۱۰۰، نرخ تقاطع ۰/۶، نرخ جهش ۰/۱، حداکثر مدت زمان کل برابر با تعداد سفارش و حداکثر مدت زمان مجاز عدم بهبود برابر با $\max\left(۶,\frac{\text{total time}}{۲}\right)$ تنظیم شده است. در شکل (۱۱) به‌عنوان نمونه نسبت سیگنال به نویز برای سه پارامتر از الگوریتم MA نشان داده شده است.

## ۸-۳- نتایج حل

کارایی روش فراابتکاری جستجوی متغیر همسایگی (VNS) و روش فراابتکاری ممتیک (MA) با مدل BSS در مسائلی که مدل BSS قادر به حل آنها بوده است با جواب حاصل از این دو روش مقایسه شده است. نتایج این مقایسه در جدول (۵) آمده است. در این جدول ستون "تعداد یکسان"، تعداد مسائلی که جواب روش حل شده با مدل یکسان است را بیان می‌کند.

از جدول (۵) می‌توان نتیجه گرفت با افزایش تعداد ماشین، تعداد مسائلی که مدل BSS قادر به حل آنها بوده است کاهش یافته است و با افزایش مقدار R و T یعنی با افزایش دامنه موعد تحویل سفارشات، تعداد مسائلی که مدل BSS قادر به حل آنها بوده افزایش می‌یابد. نتایج نشان می‌دهد در ابعاد کوچک جواب

**جدول ۵- نتایج حل روش ابتکاری و روش‌های فراابتکاری در ابعاد کوچک**

| پارامتر | مقدار | مدل ریاضیBSS | | روش ابتکاری HJABSS | | VNS | | MA | |
|---|---|---|---|---|---|---|---|---|---|
| | | تعداد بهینه حل شده | مدت زمان (ثانیه) | تعداد بهینه حل شده | مدت زمان (ثانیه) | تعداد یکسان | مدت زمان (ثانیه) | تعداد یکسان | مدت زمان (ثانیه) |
| نوع | ۲ | ۸۷ | ۱۸۶/۶ | ۷۹ | ۰/۰ | ۸۷ | ۰/۵ | ۸۵ | ۰/۳ |
| | ۵ | ۸۰ | ۳۰۱/۵ | ۷۴ | ۰/۰ | ۸۰ | ۰/۵ | ۷۹ | ۰/۱ |
| خانواده | ۴ | ۸۲ | ۲۴۷/۰ | ۷۷ | ۰/۰ | ۸۲ | ۰/۴ | ۸۱ | ۰/۳ |
| | ۶ | ۸۵ | ۲۳۰/۲ | ۷۶ | ۰/۰ | ۸۵ | ۰/۶ | ۸۳ | ۰/۲ |
| تعداد سفارش به‌ازای هر خانواده | ۶ | ۸۳ | ۱۴/۷ | ۷۹ | ۰/۰ | ۸۳ | ۰/۴ | ۸۱ | ۰/۴ |
| | ۸ | ۳۹ | ۵۲/۰ | ۳۹ | ۰/۰ | ۳۹ | ۰/۱ | ۳۸ | ۰/۲ |
| | ۱۰ | ۱۴ | ۱۹۶/۵ | ۱۳ | ۰/۰ | ۱۴ | ۰/۵ | ۱۴ | ۰/۱ |
| | ۱۴ | ۲۲ | ۳۳۹/۷ | ۱۷ | ۰/۰ | ۲۲ | ۱/۰ | ۲۲ | ۰/۰ |
| | ۱۸ | ۹ | ۱۲۹۱/۹ | ۵ | ۰/۰ | ۹ | ۱/۰ | ۹ | ۰/۰ |
| تعداد ماشین | ۲ | ۱۳۶ | ۵۴/۲ | ۱۳۱ | ۰/۰ | ۱۳۶ | ۰/۳ | ۱۳۳ | ۰/۳ |
| | ۴ | ۲۲ | ۳۳۹/۷ | ۱۷ | ۰/۰ | ۲۲ | ۱/۰ | ۲۲ | ۰/۰ |
| | ۶ | ۹ | ۱۲۹۱/۹ | ۵ | ۰/۰ | ۹ | ۱/۰ | ۹ | ۰/۰ |
| R | ۱ | ۲ | ۲۵/۱ | ۱ | ۰/۰ | ۲ | ۱/۰ | ۲ | ۰/۰ |
| | ۳/۱ | ۱۳ | ۱۲/۶ | ۱۲ | ۰/۰ | ۱۳ | ۰/۳ | ۱۱ | ۱/۰ |
| | ۷/۱ | ۱۵۲ | ۲۸۸/۶ | ۱۴۰ | ۰/۰ | ۱۵۲ | ۰/۵ | ۱۵۱ | ۰/۱ |
| T | ۲۵/۰ | ۳۸ | ۳۷/۸ | ۳۷ | ۰/۰ | ۳۸ | ۰/۳ | ۳۶ | ۰/۷ |
| | ۷۵/۰ | ۱۲۹ | ۳۰۳/۱ | ۱۱۶ | ۰/۰ | ۱۲۹ | ۰/۶ | ۱۲۸ | ۰/۱ |

روش ابتکاری HJABSS در ۹۱/۶۱ درصد از مسائل با مدل ریاضی یکسان است. این موضوع نشان دهنده عملکرد خوب روش ابتکاری HJABSS است. همچنین در ۱۰۰ درصد مسائل نمونه، روش فراابتکاری VNS و در ۹۸/۲۰ درصد مسائل، روش MA دارای جواب یکسانی با مدل ریاضی بوده است.

از آنجا که مدل ریاضی BSS در ابعاد بزرگ‌تر کارآیی خود را از دست داده و حل نمی‌شود، باید از روش‌های ابتکاری و فراابتکاری برای حل نمونه‌های واقعی استفاده نمود. اما برای ارزیابی کارایی دو روش ارائه شده VNS و MA در ابعاد بزرگ‌تر درصد خطای هر روش نسبت به بهترین جواب ملاک مقایسه قرار گرفته است. جدول (۶) نتایج حل در ابعاد بزرگ را نشان می‌دهد. از جدول (۶) می‌توان نتیجه گرفت درصد خطای روش VNS با افزایش تعداد سفارش در هر خانواده کاهش یافته و با افزایش مقدار R، تعداد ماشین، نوع و خانواده، درصد خطای آن افزایش یافته است. درصد خطای روش MA با افزایش مقدار R، T و نوع افزایش یافته و با افزایش تعداد خانواده کاهش یافته است. با توجه به ستون تعداد نمونه با تابع هدف بهتر می‌توان نتیجه گرفت که در مسائل هر چه تعداد ماشین‌ها افزایش یافته کارایی روش MA بیشتر شده است و در و هر چه مقدار R یعنی دامنه موعد تحویل سفارشات افزایش یافته کارایی روش VNS بیشتر شده است.

نتایج جدول (۶) نشان می‌دهد الگوریتم VNS در ابعاد بزرگ به‌طور میانگین ۲۸/۸۶ درصد، جواب اولیه را بهبود داده است و

**جدول ۶- نتایج حل روش‌های فراابتکاری در ابعاد بزرگ**

| پارامتر | مقدار | در صد خطا نسبت به بهترین جواب | | مدت زمان حل (ثانیه) | | درصد بهبود نسبت به جواب اولیه (HJABSS) | | تعداد نمونه با تابع هدف بهتر | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | VNS | MA | VNS | MA | VNS | MA | VNS | مساوی | MA |
| نوع | ۲ | ۰/۵۷ | ۷/۳۵ | ۱۳۵۶/۴ | ۶۲۲/۵ | ۲۷/۴۸ | ۲۳/۱۸ | ۱۱۵۷ | ۰ | ۲۸۳ |
| | ۵ | ۳/۱۹ | ۸/۲۴ | ۱۳۶۵/۱ | ۶۴۸/۸ | ۳۰/۲۴ | ۲۷/۰۰ | ۱۱۰۹ | ۱۴ | ۳۳۲ |
| خانواده | ۶ | ۱/۷۰ | ۸/۰۷ | ۹۰۷/۳ | ۹۳۱/۹ | ۳۰/۰۱ | ۲۶/۰۳ | ۱۱۳۷ | ۱۴ | ۳۰۴ |
| | ۱۲ | ۲/۰۶ | ۷/۵۳ | ۱۸۱۴/۲ | ۳۳۹/۴ | ۲۷/۷۱ | ۲۴/۱۴ | ۱۱۲۹ | ۰ | ۳۱۱ |
| تعداد سفارش | ۵۰ | ۲/۹۹ | ۶/۹۴ | ۹۲۲/۵ | ۳۷۶/۵ | ۳۳/۰۳ | ۳۰/۸۳ | ۷۳۰ | ۸ | ۲۳۱ |
| به‌ازای هر | ۷۰ | ۱/۷۶ | ۹/۵۶ | ۱۳۰۰/۷ | ۶۹۶/۶ | ۲۸/۸۴ | ۲۳/۸۲ | ۷۶۱ | ۶ | ۱۹۹ |
| خانواده | ۱۰۰ | ۰/۸۸ | ۶/۹۰ | ۱۸۵۹/۱ | ۸۳۳/۹ | ۲۴/۷۱ | ۲۰/۶۰ | ۷۷۵ | ۰ | ۱۸۵ |
| تعداد ماشین | ۴ | ۰/۰۱ | ۱۲/۸۲ | ۱۳۶۱/۹ | ۵۴۲/۰ | ۳۰/۹۶ | ۲۳/۱۵ | ۶۴۴ | ۷ | ۷۷ |
| | ۶ | ۰/۹۹ | ۵/۲۱ | ۱۳۴۶/۸ | ۷۵۹/۴ | ۲۶/۸۷ | ۲۳/۹۳ | ۵۸۱ | ۰ | ۱۳۹ |
| | ۱۰ | ۲/۶۲ | ۴/۷۵ | ۱۳۵۷/۸ | ۶۱۶/۳ | ۲۵/۸۲ | ۲۴/۳۶ | ۵۲۹ | ۷ | ۱۹۱ |
| | ۱۵ | ۳/۹۰ | ۸/۴۱ | ۱۳۷۶/۷ | ۶۲۴/۹ | ۳۱/۷۹ | ۲۸/۹۱ | ۵۱۲ | ۰ | ۲۰۸ |
| R | ۱/۰ | ۱/۵۵ | ۶/۲۸ | ۱۳۶۸/۶ | ۶۱۴/۷ | ۲۸/۱۰ | ۲۵/۰۸ | ۷۴۸ | ۰ | ۲۱۲ |
| | ۱/۳ | ۱/۹۰ | ۸/۰۲ | ۱۳۶۳/۶ | ۶۷۲/۵ | ۲۹/۱۲ | ۲۵/۲۵ | ۷۵۷ | ۸ | ۲۰۳ |
| | ۱/۷ | ۲/۱۸ | ۹/۰۹ | ۱۳۵۰/۱ | ۶۱۹/۸ | ۲۹/۳۶ | ۲۴/۹۳ | ۷۶۱ | ۶ | ۲۰۰ |
| T | ۰/۲۵ | ۲/۰۲ | ۳/۶۲ | ۱۳۶۳/۳ | ۶۰۴/۰ | ۲۱/۵۶ | ۲۰/۱۷ | ۱۰۸۰ | ۰ | ۳۶۰ |
| | ۰/۷۵ | ۱/۷۴ | ۱۱/۹۷ | ۱۳۵۸/۲ | ۶۶۷/۳ | ۳۶/۱۶ | ۳۰/۰۰ | ۱۱۸۶ | ۱۴ | ۲۵۵ |

الگوریتم MA در این ابعاد به‌طور میانگین ۲۵/۰۹ درصد، جواب اولیه را بهبود داده است. همچنین در ابعاد بزرگ روش‌های VNS و MA به‌ترتیب ۱/۹ و ۷/۸ درصد نسبت به بهترین جواب خطا دارند. با وجود اینکه الگوریتم VNS نسبت به الگوریتم MA مدت زمان حل بیشتری دارد، اما در اکثر موارد الگوریتم VNS عملکرد بهتری نسبت به الگوریتم MA داشته است.

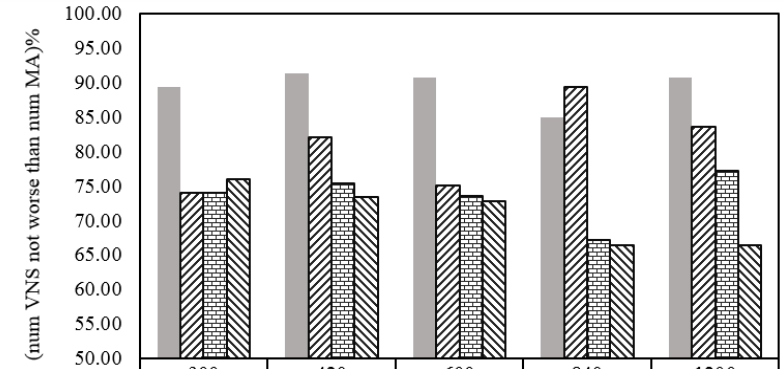
شکل (۱۲) میانگین درصد مسائلی را نشان می‌دهد که در آنها جواب روش VNS بدتر از روش MA نبوده است. این شکل بر اساس تفکیک تعداد کار و تعداد ماشین موازی در مرحله دوم نشان‌داده شده است. همان‌طور که در شکل (۱۲) قابل مشاهده است برای مثال‌های تولید شده در ابعاد ۴ ماشین در حداقل ۸۵/۰۰ درصد از مسائل نمونه جواب VNS بدتر از MA نبوده است. این درصد برای ابعاد ۶، ۱۰ و ۱۵ ماشین به‌ترتیب حداقل ۷۴/۰۰، ۶۷/۱۴ و ۶۶/۴۳ درصد است.

شکل (۱۳) و شکل (۱۴) به‌ترتیب درصد خطای الگوریتم VNS و الگوریتم MA را نسبت به بهترین جواب با تفکیک ابعاد تعداد کار و تعداد ماشین موازی در مرحله دوم نشان داده است.

همان‌طور که در شکل (۱۳) قابل مشاهده است هر چه تعداد کار و ماشین افزایش یابد درصد خطای الگوریتم VNS نیز افزایش یافته است. ولی شکل (۱۴) وابستگی کمتر الگوریتم MA به تعداد کار و تعداد ماشین را نشان می‌دهد.

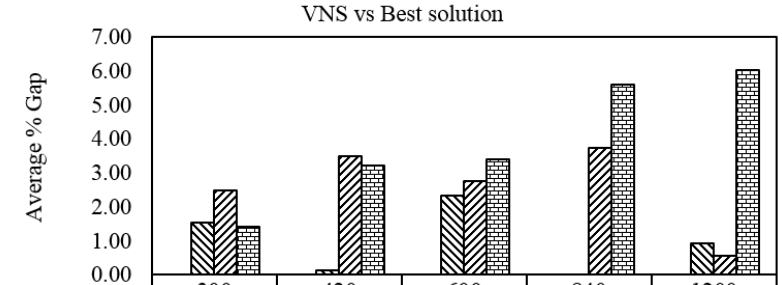
## ۹- نتیجه‌گیری

در این مقاله یک مسئله در محیط کارگاه جریانی منعطف سه



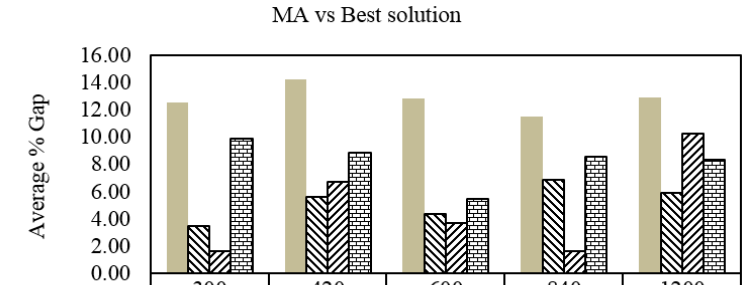

| | 300 | 420 | 600 | 840 | 1200 |
|---|---|---|---|---|---|
| 4 machines | 89.33 | 91.33 | 90.71 | 85.00 | 90.71 |
| 6 machines | 74.00 | 82.00 | 75.00 | 89.29 | 83.57 |
| 10 machines | 74.00 | 75.33 | 73.57 | 67.14 | 77.14 |
| 15 machines | 76.00 | 73.33 | 72.86 | 66.43 | 66.43 |

**شکل ۱۲- میانگین درصد جواب‌های الگوریتم VNS که بدتر از جواب‌های الگوریتم MA نیست**




| | 300 | 420 | 600 | 840 | 1200 |
|---|---|---|---|---|---|
| 4 machines | 0.00 | 0.00 | 0.00 | 0.04 | 0.00 |
| 6 machines | 1.55 | 0.13 | 2.33 | 0.00 | 0.93 |
| 10 machines | 2.48 | 3.50 | 2.77 | 3.74 | 0.56 |
| 15 machines | 1.43 | 3.24 | 3.42 | 5.60 | 6.05 |

**شکل ۱۳- درصد خطای الگوریتم VNS نسبت به بهترین جواب با تفکیک ابعاد تعداد کار**




| | 300 | 420 | 600 | 840 | 1200 |
|---|---|---|---|---|---|
| 4 machines | 12.55 | 14.23 | 12.83 | 11.51 | 12.92 |
| 6 machines | 3.48 | 5.62 | 4.33 | 6.84 | 5.88 |
| 10 machines | 1.60 | 6.72 | 3.67 | 1.60 | 10.25 |
| 15 machines | 9.90 | 8.87 | 5.46 | 8.57 | 8.36 |

**شکل ۱۴- درصد خطای الگوریتم MA نسبت به بهترین جواب با تفکیک ابعاد تعداد کار**

مرحله‌ای با در نظر گرفتن انسداد و پردازش دسته‌ای بررسی شد. مرحله اول و سوم شامل یک ماشین پردازشگر تکی و مرحله دوم شامل m ماشین موازی پردازش دسته‌ای یکسان است. هدف این مسئله کمینه کردن مجموع دیرکرد وزنی سفارشات است. محیط بررسی شده در این مقاله از خط شارژ و بسته‌بندی باتری یک کارخانه باتری‌سازی الهام گرفته شده است.

برای حل مسئله مورد نظر یک مدل ریاضی توسعه داده شد. این مدل در ابعاد ۱۸ سفارش و ۶ ماشین تنها قادر است ۹ مسئله از ۴۰ مسئله نمونه را به‌صورت بهینه حل کند. نتایج محاسباتی نشان می‌دهد امکان استفاده از مدل در ابعاد واقعی وجود ندارد.

با توجه به اینکه این مسئله NP-hard است و فقط برای مسائل کوچک امکان حل بهینه آن وجود دارد، بر همین اساس دو الگوریتم فراابتکاری یکی بر مبنای جستجوی همسایگی متغیر و دیگری بر مبنای ممتیک برای حل مسئله توسعه داده شدند. از آنجایی که کیفیت عملکرد الگوریتم‌های فراابتکاری تاحد زیادی به پارامترها و عملگرهای انتخابی بستگی دارد دراین مقاله همه پارامترها و عملگرهای الگوریتم‌های پیشنهادی بطور جداگانه با استفاده از روش طراحی آزمایشات تاگوچی تنظیم شده است. آزمایشات متعددی برای ارزیابی الگوریتم‌ها انجام شده است. نتایج نشان می‌دهد که الگوریتم VNS کاراتر از الگوریتم MA است. چرا که الگوریتم VNS قادر است مسائل تا ابعاد ۱۲۰۰ سفارش و ۱۵ ماشین را با خطای حدود ۱/۹ درصد نسبت به بهترین جواب، حل کند و الگوریتم MA قادر است مسائل تا ابعاد ۱۲۰۰ سفارش و ۱۵ ماشین را با خطای حدود ۷/۸ درصد نسبت به بهترین جواب حل کند. این در حالی است که مدت زمان حل الگوریتم MA نصف مدت زمان حل VNS است.

جهت مطالعات آتی می‌توان مسئله مورد بررسی را با فرض زمان در دسترس برای هر ماشین بررسی کرد. همچنین می‌توان در این مسئله هزینه مصرف انرژی ماشین‌های موازی مرحله دوم را علاوه بر هزینه دیرکرد به‌عنوان تابع هدف در نظر گرفت. می‌توان مسئله مورد بررسی را در محیط کارگاه جریانی منعطف m مرحله‌ای در نظر گرفت که بین مراحل انسداد وجود داشته باشد.

## واژه‌نامه

1. batch processing machines
2. incompatible
3. partical swarm optimization
4. flexible flow shop
5. discrete processing machines
6. setup time
7. block
8. genetic algorithm
9. ant colony optimization
10. variable neighborhood search
11. tabu search
12. non-dominated sorting genetic algorithm ii
13. local search
14. batch processing
15. memetic
16. filling
17. trolley
18. leveling
19. flexible flow shop with block and batch processing
20. heuristic for job assignment within the batch with batch sequencing and scheduling
21. internal due date
22. heuristic for job assignment within batch
23. upper bound for batch scheduling cost
24. heuristic for batch scheduling
25. beam search
26. shaking
27. evolutionary algorithms
28. earliest due date
29. roulette wheel
30. workload
31. range of due date
32. tightness of schedules

## مراجع


1. Ikura, Y., and Gimple, M., "Efficient Scheduling Algorithms for a Single Batch Processing Machine", *Operations Research Letters*, Vol. 5, pp. 61-65, 1986.
2. Lee, C.-Y., Uzsoy, R., and Martin-Vega, L.A., "Efficient Algorithms for Scheduling Semiconductor

Burn-in Operations", *Operations Research*, Vol. 40, pp. 764-775, 1992.
3. Uzsoy, R., "Scheduling Batch Processing Machines with Incompatible Job Families", *International Journal of Production Research*, Vol. 33, pp. 2685-2708, 1995.
4. Damodaran, P., and Velez-Gallego, M. C., "Heuristics for Makespan Minimization on Parallel Batch Processing Machines with Unequal Job Ready Times", *The International Journal of Advanced Manufacturing Technology*, Vol. 49, pp. 1119-1128, 2010.
5. Hulett, M., Damodaran, P., and Amouie, M., "Scheduling Non-Identical Parallel Batch Processing Machines to Minimize Total Weighted Tardiness Using Particle Swarm Optimization", *Computers & Industrial Engineering*, Vol. 113, pp. 425-436, 2017.
6. Luo, H., Huang, G. Q., Zhang, Y., Dai, Q., and Chen, X., "Two-Stage Hybrid Batching Flowshop Scheduling with Blocking and Machine Availability Constraints Using Genetic Algorithm", *Robotics and Computer-Integrated Manufacturing*, Vol. 25, pp. 962-971, 2009.
7. Balasubramanian, H., Mönch, L., Fowler, J., and Pfund, M., "Genetic Algorithm Based Scheduling of Parallel Batch Machines with Incompatible Job Families to Minimize Total Weighted Tardiness", *International Journal of Production Research*, Vol. 42, pp. 1621-1638, 2004.
8. Mathirajan, M., and Sivakumar, A., "Minimizing Total Weighted Tardiness on Heterogeneous Batch Processing Machines with Incompatible Job Families", *The International Journal of Advanced Manufacturing Technology*, Vol. 28, pp. 1038, 2006.
9. Almeder, C., and Mönch, L., "Metaheuristics for Scheduling Jobs with Incompatible Families on Parallel Batching Machines", *Journal of the Operational Research Society*, Vol. 62, pp. 2083-2096, 2011.
10. Chiang, T.-C., Cheng, H.-C., and Fu, L.-C., "A Memetic Algorithm for Minimizing Total Weighted Tardiness on Parallel Batch Machines with Incompatible Job Families and Dynamic Job Arrival", *Computers & Operations Research*, Vol. 37, pp. 2257-2269, 2010.
11. Chou, F.-D., "Minimising the Total Weighted Tardiness for Non-Identical Parallel Batch Processing Machines with Job Release Times and Non-Identical Job Sizes", *European Journal of Industrial Engineering*, Vol. 7, pp. 529-557, 2013.
12. Mathirajan, M., Bhargav, V., and Ramachandran, V., "Minimizing Total Weighted Tardiness on a Batch-Processing Machine with Non-Agreeable Release Times and Due Dates", *The International Journal of Advanced Manufacturing Technology*, Vol. 48, pp. 1133-1148, 2010.
13. Shi, Z., Huang, Z., and Shi, L., "Customer Order Scheduling on Batch Processing Machines with Incompatible Job Families", *International Journal of Production Research*, Vol. 56, pp. 795-808, 2018.
14. Vivek, P., Saravanan, R., Chandrasekaran, M., Pugazhenthi, R., and Vairavel, M., "A Critical-Machine Based Heuristic for HFS Batch Scheduling", *International Journal of Mechanical Engineering and Technology*, Vol. 9, pp. 105-119, 2018.
15. Zeng, Z., Hong, M., Man, Y., Li, J., Zhang, Y., and Liu, H., "Multi-Object Optimization of Flexible Flow Shop Scheduling with Batch Process — Consideration Total Electricity Consumption and Material Wastage", *Journal of Cleaner Production*, Vol. 183, pp. 925-939, 2018.
16. Tan, Y., Mönch, L., and Fowler, J. W., "A Hybrid Scheduling Approach for a Two-Stage Flexible Flow Shop with Batch Processing Machines", *Journal of Scheduling*, Vol. 21, pp. 209-226, 2018.
17. Graham, R. L., Lawler, E. L., Lenstra, J. K., and Kan, A. R., "Optimization and Approximation in Deterministic Sequencing and Scheduling: A Survey", *Annals of Discrete Mathematics*, Vol. 5, pp. 287-326, 1979.
18. Sawik, T., "Mixed Integer Programming for Scheduling Flexible Flow Lines with Limited Intermediate Buffers", *Mathematical and Computer Modelling*, Vol. 31, pp. 39-52, 2000.
19. Mladenović, N., and Hansen, P., "Variable Neighborhood Search", *Computers & Operations Research*, Vol. 24, pp. 1097-1100, 1997.
20. Parsa, N. R., Karimi, B., and Kashan, A. H., "A Branch and Price Algorithm to Minimize Makespan on a Single Batch Processing Machine with Non-Identical Job Sizes", *Computers & Operations Research*, Vol. 37, pp. 1720-1730, 2010.