Kharazmi University

Mathematical Researches

Print ISSN: 2588-2546
Online ISSN: 2588-2554

Homepage: https://jemr.khu.ac.ir/

Solving Permutation Flow Shop Problem by the Regulations of Columnar Entries in the Processing Times Matrix

Shahriar Farahmand Rad*

Assistant Professor, Department of Mathematics, Payame Noor University, Tehran, Iran. E-mail: sh_fmand@pnu.ac.ir

Article Info

Article type:
Research Article

Article history:
Received: 30 March 2021
Received in revised form: 15 June 2021
Accepted: 5 March 2022
Published online: 29 February 2024

Keywords:
Scheduling,
Permutation flow shop,
Heuristics,
Initial sequences,
Time matrix,
Makespan, NEH,
Taillard's benchmark.

ABSTRACT

Introduction

In the permutation flow shop problem (PFSP), n jobs must be processed in a suitable order on m machines. The objective is to minimize the makespan $(C_{\max})$, the time to complete all jobs. There exists an exact solution for $m=2$, but for $m>2$ the problem is completely NP-hard [1]. Many authors proposed heuristic algorithms for solving PFSP. There are many assumptions such as infinite buffering space between jobs, the sameness of the processing orders of jobs on machines, jobs not passing each other at the processing etc. Matrix $A=\left[a_{ij}\right]_{n\times m}$ is the matrix of the processing times of n jobs on m machines. The a_{ij} is the processing time of the i th job on the j th machine.

Material and Methods

Among many proposed algorithms for solving PFSP, NEH, method is the most effective [2].

The jobs are sorted according to their decreasing completion times. The order of the first two jobs is chosen after finding the best makespan from permutation of them. The third job is inserted before, between or after the fixed two jobs in the previous step. The best makespan shows the best ordering. The above processes are repeated for the fourh and the rest jobs.

The regulations of the columnar entries (RCE) is an algorithm similar to the algorithms that are finding a suitable initial sequences of jobs. The desired order is obtained after regulating the columnar entries in the time matrix A. As the object is $\min C_{\max}$, the permutation of rows and a new order in A must be determined. These rows will be ordered on the basis of RCE in such a form that:

1) entries on the main hypothetical diagonal are small.
2) entries that approach the border of A are greater.
3) the greatest entries on the border of A are placed on the secondary hypothetical diagonal. Then they will decrease by approaching the center of A.

The following figure is a plot for the mentioned design.

[Downloaded from mmr.khu.ac.ir on 2024-10-12]

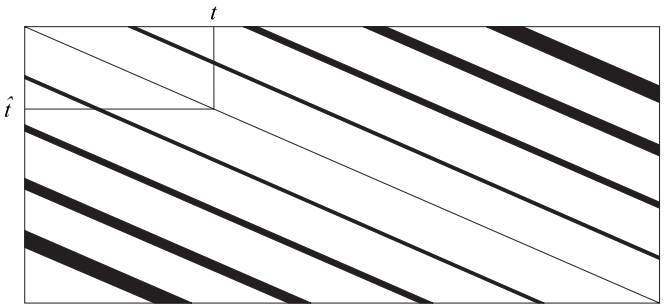


Figure 1. Plot for final design of A

The intersection of the t th column and the main hypothetical diagonal is called $\hat{t}$. Matrix A is not the square, thus there is probably no entry in this intersection. We have

$$\hat{t} = \left[\frac{n(t-1)+m-t}{m-1} + \frac{1}{2} \right] \ , \ t = 1,2,\ldots,m$$

The entries of the t th column are sorted in ascending order. The new ordering of the rows i.e. $\sigma(1),\sigma(2),\ldots,\sigma(n)$ with the transformation $\sigma : \{1,2,\ldots,n\} \to \{1,2,3,\ldots,n\}$ is obtained. If $\sigma(p) = q$, then the q th row of matrix A will be newly located in the p th position. We have $a_{\sigma(1)t} \le a_{\sigma(2)t} \le \ldots \le a_{\sigma(n)t}$. If some of the entries equal each other, then we will choose one of the orders randomly.

Consider the transformation $\varphi : \{1,2,\ldots,n\} \to \{1,2,\ldots,n\}$ with

$$\varphi(i) = \begin{cases} \hat{t} + \dfrac{i}{2} & ; \ 1 \le i \le 2\hat{t} \ , \ i \text{ is even} \\ \hat{t} - \dfrac{i+1}{2} + 1 & ; \ 1 \le i \le 2\hat{t} \ , \ i \text{ is odd} \\ i & ; \ 2n - 2\hat{t} + 1 \le i \le n \end{cases}$$

for $\hat{t} \le \dfrac{n}{2}$ and with

$$\varphi(i) = \begin{cases} \hat{t} - \dfrac{i}{2} + 1 & ; \ 1 \le i \le 2n - 2\hat{t} \ , \ i \text{ is even} \\ \hat{t} + \dfrac{i+1}{2} & ; \ 1 \le i \le 2n - 2\hat{t} \ , \ i \text{ is odd} \\ n - i + 1 & ; \ 2n - 2\hat{t} + 1 \le i \le n \end{cases}$$

for $\hat{t} > \dfrac{n}{2}$.

The last position for the i th row is denoted by $\gamma_t(i)$ when $\gamma_t(i) = (\varphi \circ \sigma^{-1})(i)$.

[Downloaded from mmr.khu.ac.ir on 2024-10-12]

For every t in the matrix K_t, the entries are defined such that:

- for every i, the entry in the i th row and the $\gamma_t(i)$ th column of K_t is equal to a_{it} of A,
- all other entries are zero.

Now $K = K_1 + K_2 + \ldots + K_m = \left[k_{ij}\right]_{n\times n}$ is a matrix in which every k_{ij} denotes the fitness of the location of the i th old row in the new position j.

i) The greatest entry in K, named $k_{i_0 j_0}$, is selected, and for the i_0 th row in A the location j_0 will be specified.

ii) The i_0 th row and the j_0 th column in K are deleted. In the new matrix that obtained from K the same operations of step (i) are run.

iii) The previous steps are repeated and finally the desired initial good order of jobs is resulted.

Bellman, Esogbue and Nabeshima theorem [3] gives the makespan in which

1) the movement is just allowed in two directions right and down from a_{11} to a_{nm} in A,

2) the sum of the entries that are passed by is found.

Results and discussion

The new heuristic algorithm was tested on benchmark problems from Taillard Page [4]. Relative percentage deviation (RPD) which refers to the difference between the heuristic and NEH solutions was calculated by:

$$\mathrm{RPD} = \frac{\mathrm{Heu}_{\mathrm{sol}} - \mathrm{NEH}_{\mathrm{sol}}}{\mathrm{NEH}_{\mathrm{sol}}} \times 100$$

Following table shows these percentages.

Table 1. Average percentage increase over the NEH solutions

J/M	20×5	20×10	20×20	50×5	50×10	50×20	100×5	100×10	100×20	200×10	200×20	500×20	Average
RCE	16	18	14	13	16	18	11	13	15	12	13	11	14

Conclusion

The new heuristic algorithm is found to generate better solutions than some of the proposed heuristics in the literature. Moreover, it can be as good as NEH on an average.

How to cite: Farahmand Rad, Shahriar. (2023). Solving Permutation Flow Shop Problem by the Regulations of Columnar Entries in the Processing Times Matrix. *Mathematical Researches*, 9 (4), 1-23.

© The Author(s). Publisher: Kharazmi University

[Downloaded from mmr.khu.ac.ir on 2024-10-12]

شاپا چاپی: ۲۵۴۶-۲۵۸۸
شاپا الکترونیکی: ۲۵۵۴-۲۵۸۸

پژوهش‌های ریاضی

Homepage: https://mmr.khu.ac.ir/

حل مسئلهٔ کارگاه جریان جایگشتی به‌وسیله تنظیمات درایه‌های ستونی در ماتریس زمان‌های پردازش

شهریار فرهمند راد ✉

استادیار گروه ریاضی، دانشگاه پیام نور، تهران، ایران. رایانامه: sh_fmand@pnu.ac.ir

اطلاعات مقاله

نوع مقاله: مقاله پژوهشی

تاریخ دریافت: ۱۴۰۰/۱/۱۰
تاریخ بازنگری: ۱۴۰۰/۳/۲۵
تاریخ پذیرش: ۱۴۰۰/۱۲/۱۴
تاریخ انتشار: ۱۴۰۲/۱۲/۱۰

واژه‌های کلیدی:
زمان‌بندی،
کارگاه جریانی جایگشتی،
روش‌های ابتکاری،
دنباله اولیه،
ماتریس زمان‌ها،
حداکثر زمان در جریان،
الگوریتم NEH،
مسائل محک تیلارد.

چکیده

مسئله کارگاه جریانی جایگشتی یکی از مسایل مهم و به روز تحقیق در عملیات گسسته است. در این مقاله الگوریتم ابتکاری جدیدی با استفاده از تنظیم درایه‌های ستونی ماتریس زمان‌ها برای حل مسأله کارگاه جریانی جایگشتی پیشنهاد می‌شود. n کار روی m ماشین با زمان‌های قطعی پردازش می‌شوند و هدف اصلی می‌نیمم کردن زمان کل تکمیل کارهاست. مسأله، در زمان چندجمله‌ای قابل حل نیست. مانند بیشتر روش‌های ابتکاری حل مسأله، ابتدا ترتیب اولیه مناسبی از دنباله کارها پیدا می‌شود. برای این منظور ماتریس $K=\left[k_{ij}\right]_{n\times n}$ چنان ساخته می‌شود که هر k_{ij} نشان‌دهنده اندازه مناسب بودن جای سطر قدیم iام در مکان جدید jام باشد. سپس قضیه بلمن، اسوگبو و نابشیما مورد استفاده قرار می‌گیرد. روش ارائه شده با الگوریتم NEH که بهترین روش شناخته موجود است مقایسه می‌شود. مقایسه روی مسایل محک و استاندارد تیلارد انجام می‌گیرد. نتایج محاسباتی نشان می‌دهند الگوریتم ابتکاری بهتر از بعضی روش‌های پیشنهاد شده قبلی می‌باشد و نسبت به بقیه در تعدادی از مثال‌های تیلارد برتر است. به عنوان نتیجه الگوریتم ابتکاری تقریباً به خوبی NEH و امیدبخش می‌باشد. بر اساس ساختار ارائه شده، الگوریتم ابتکاری پیشنهادی می‌تواند به خوبی نقش یک روش ابتکاری موفق را ایفا کند.

استناد: فرهمند راد، شهریار (۱۴۰۲). حل مسأله‌ی کارگاه جریان جایگشتی به وسیله تنظیمات درایه‌های ستونی در ماتریس زمان‌های پردازش. *پژوهش‌های ریاضی*، ۹ (۴)، ۱-۲۳.

ناشر: دانشگاه خوارزمی

© نویسندگان.

CC BY NC

[Downloaded from mmr.khu.ac.ir on 2024-10-12]

مقدمه

نظریه زمان‌بندی و کارگاه جریان جایگشتی در آن با در نظر گرفتن حداکثر زمان تکمیل کارها به عنوان تابع هدف از جمله مسایل مهم تحقیق در عملیات گسسته می‌باشد. از آنجایی که می‌نیمم کردن این تابع هدف با ماکزیمم کردن استفاده از تجهیزات معادل است در بسیاری از امور دنیای واقعی ارزشی ویژه دارد. کاربردهای وسیعی از این مفهوم در فرودگاه‌ها (نشست و برخاست هواپیماها)، خطوط تولیدی کارخانه‌ها و کارگاه‌ها، برنامه‌ریزی درسی در محیط‌های آموزشی و غیره مشاهده می‌شود.

در این مسأله بهینه‌سازی ترکیباتی ترتیب مناسبی از n کار که باید روی m ماشین نامرتبط پردازش شوند مورد نظر است. هدف می‌نیمم شدن حداکثر زمان پردازش است. فقط برای حالت $m = ۲$ جواب بهینه توسط جانسون[۱] به دست آمده است و در حالت $m > ۲$، NP - سخت می‌باشد [۱]. به عبارت دیگر روش‌های اندک دقیق مانند روش جانسون و شاخه و کران[۲] دارای اشکالات مهمی مانند کارایی کم، سرعت اندک و افزایش بسیار بالای تعداد حالات ممکن در اثر زیاد شدن تعداد کارها یا ماشین‌ها هستند. به همین دلیل سمت و سو و گرایش زیادی به روش‌های ابتکاری وجود دارد. یعنی رویه‌هایی که جواب شدنی و نه لزوماً بهینه را برای مسأله به دست می‌دهند. در الگوریتم‌های ابتکاری سعی به پیدا کردن ساختمانی ترکیبیاتی و حل مسأله به وسیله ابتکارها می‌باشد. مطالعه روش‌های ابتکاری حل مسأله از دهۀ ۱۹۵۰ میلادی آغاز و همچنان ادامه دارد هر چند تعداد الگوریتم‌های مناسبی که اخیراً ارائه می‌شوند مانند سابق از رشد بالایی برخوردار نیستند. از میان الگوریتم‌های ابتکاری حل مسأله می‌توان به روش‌های پیج[۳] (۱۹۶۱) [۲]، پالمر[۴] (۱۹۶۵) [۳]، پتروف[۵] (۱۹۶۶) [۴]، کمپبل[۶]، دودک[۷] و اسمیت[۸] (۱۹۷۰) [۵]، گوپتا[۹] (۱۹۷۱) [۶]، داننبرینگ[۱۰] (۱۹۷۷) [۷]، کینگ[۱۱] و اسپاچیز[۱۲] (۱۹۸۰) [۸]، استینسون[۱۳] و اسمیت[۱۴] (۱۹۸۲) [۹]، نواز[۱۵]، انسکور[۱۶] و هام[۱۷] (۱۹۸۳) [۱۰]، هوندال[۱۸] و راجگوپال[۱۹] (۱۹۸۸) [۱۱]، ویدمر[۲۰] و هرتز[۲۱] (۱۹۸۹) [۱۲]، تیلارد[۲۲] (۱۹۹۰) [۱۳]، هو[۲۳] و چانگ[۲۴] (۱۹۹۱) [۱۴]، سارین[۲۵] و لفوکا[۲۶] (۱۹۹۳) [۱۵]، موسلین[۲۷] (۱۹۹۵) [۱۶]، لای[۲۸] (۱۹۹۶) [۱۷]، کولاماس[۲۹] (۱۹۹۸) [۱۸]، سولیمان[۳۰] (۲۰۰۰) [۱۹]، فرامینان[۳۱]، لیستن[۳۲] و راجندران[۳۳] (۲۰۰۳) [۲۰]، کورز[۳۴] و اسکین[۳۵] (۲۰۰۴) [۲۱]، لئونگ[۳۶] و پیندو[۳۷] (۲۰۰۵) [۲۲]، اللهوردی[۳۸] و الانزی[۳۹] (۲۰۰۶) [۲۳]، آلایکیران[۴۰]، انجین[۴۱] و دوین[۴۲] (۲۰۰۷) [۲۴]، کالزینسکی[۴۳] و کامبروسکی[۴۴] (۲۰۰۸) [۲۵]، فرهمندراد[۴۵]، روئیز[۴۶] و بروجردیان[۴۷] (۲۰۰۹) [۲۶]، آنکائو[۴۸] (۲۰۱۲) [۲۷]، مالیک[۴۹] و دهینگرا[۵۰]

1. Johnson
2. Branch and Bound
3. Page
4. Palmer
5. Petrov
6. Campbell
7. Dudek
8. Smith
9. Gupta
10. Dannenbring
11. King
12. Spachis
13. Stinson
14. Smith
15. Nawaz
16. Enscore
17. Ham
18. Hundal
19. Rajgopal
20. Widmer
21. Hertz
22. Taillard
23. Hu
24. Changh
25. Sarin
26. Lefoka
27. Mocellin
28. Lai
29. Kolamas
30. Suliman
31. Framinan
32. Leisten
33. Rajendran
34. Kurz
35. Askin
36. Leung
37. Pinedo
38. Allahverdi
39. Al-Anzi
40. Alaykyran
41. Engin
42. Doyen
43. Kalczynski
44. Kamburowski
45. Farahmand
46. Ruiz
47. Boroojerdian
48. Ancau
49. Malik
50. Dhingra

[Downloaded from mmr.khu.ac.ir on 2024-10-12]

(۲۰۱۳) [۲۸]، زو[۱]، یین[۲]، چنگ[۳] و دیگران (۲۰۱۴) [۲۹]، لیو[۴]، جین[۵] و پرایس[۶] (۲۰۱۶) [۳۰]، برام[۷] و ریت[۸] (۲۰۱۸) [۳۱]، نوردیانسیاه[۹] و دیگران (۲۰۱۹) [۳۲] و سائوی[۱۰] و سائور[۱۱] (۲۰۲۰) [۳۳] اشاره کرد.

در تعدادی از روش‌های ابتکاری ذکر شده مستقیماً دنباله‌ای مناسب از کارها به‌دست می‌آیند. برخی مسأله را به حالت ساده‌تر مخصوصاً حالت دوماشینی (برای استفاده از روش جانسون) تبدیل کرده‌اند. در بعضی از آنها از خواص کارها استفاده شده است، گاهی مسأله زمان‌بندی کارگاه جریانی جایگشتی را با استفاده از مسایل دیگر مانند فروشنده دوره‌گرد بررسی کرده‌اند. بخشی از الگوریتم‌ها نیز فن می‌نیمم کردن زمان تلف شده آخرین ماشین را به کار برده‌اند.

در هر کارگاه جریانی جایگشتی فرض‌های زیر وجود دارند:

۱. همهٔ کارها مستقل از هم هستند و در زمان صفر در دسترس می‌باشند.
۲. ترتیب پردازش کارها روی هر ماشینی ثابت است.
۳. هر کار فقط یک بار روی هر ماشین پردازش می‌شود.
۴. هر ماشین در هر زمان فقط یک کار را مورد پردازش قرار می‌دهد.
۵. هیچ‌کاری هم‌زمان روی دو ماشین (یا بیشتر) پردازش نمی‌شود.
۶. ماشین‌های موازی وجود ندارند.
۷. فضای ذخیره بین ماشین‌ها نامتناهی است.
۸. عمل پردازش کارها روی هر ماشین به طور نوبتی است و قابل جلو زدن نیست.
۹. ماشین‌ها همواره آماده و به طور پیوسته در دسترس می‌باشند.

بر اساس فرض‌های بالا اعمال هر یک از n کار باید روی ماشین‌های $۱,۲,\dots,m$ (با همین ترتیب) انجام شود. چون ترتیب پردازش کارها روی هر ماشین یکسان است تعداد جواب‌ها برای زمان‌بندی در یک مسأله $n\times m$ کارگاه جریانی جایگشتی از $(n!)^m$ به $n!$ کاهش می‌یابد.

فرض کنید a_{ij} زمان قطعی پردازش کار iام روی ماشین jام است. ماتریس زمان‌های پردازش n کار روی m ماشین را به صورت زیر در نظر می‌گیریم.

$$M=\begin{bmatrix} a_{۱۱} & a_{۱۲} & a_{۱۳} & \dots & a_{۱m-۱} & a_{۱m} \\ a_{۲۱} & a_{۲۲} & a_{۲۳} & \dots & a_{۲m-۱} & a_{۲m} \\ a_{۳۱} & a_{۳۲} & a_{۳۳} & \dots & a_{۳m-۱} & a_{۳m} \\ \vdots & \vdots & \vdots & \dots & \vdots & \vdots \\ a_{n۱} & a_{n۲} & a_{n۳} & \dots & a_{nm-۱} & a_{nm} \end{bmatrix}_{n\times m}$$

1. Xu
2. Tin
3. Cheng
4. Liu
5. Jin
6. Price
7. Brum
8. Ritt
9. Nurdiansyah
10. Souvey
11. Saur

[Downloaded from mmr.khu.ac.ir on 2024-10-12]

هدف می‌نیمم کردن ماکزیمم زمان تکمیل[1] کارهاست که با $C_{\max}$ نشان داده می‌شود. سطر iام ماتریس متناظر با زمان‌های پردازش کار iام روی ماشین‌های اول، دوم، ... و mام است. دنبال ترتیب مناسبی از سطرها (کارها) مورد نظر است تا $C_{\max}$ می‌نیمم شود. چون ستون‌ها بیانگر ماشین‌های مختلف هستند نمی‌توانند با هم جابه‌جا شوند و یا تغییر جا داشته باشند.

برای هر جایگشت π از n کار که $\pi(j)$ کار واقع در سطر jام را نشان می‌دهد در حالت کلی $C_{\max}$ می‌تواند با فرمول زیر محاسبه شود:

$$C_{i,\pi(j)} = \max\left\{C_{i-1,\pi(j)}, C_{i,\pi(j-1)}\right\} + a_{i,\pi(j)} \quad , \quad C_{\max} = C_{m,\pi(n)}$$

در این روابط $C_{j,i}$ نشان‌دهندهٔ زمان تکمیل کار i در ماشین j است. به علاوه برای هر $j = 1,2,\ldots,m$ و هر π داریم:

$$C_{0,\pi(j)} = 0 \ , \ C_{1,\pi(j)} = a_{1,\pi(1)} + \ldots + a_{1,\pi(j)} \quad ,$$

$$C_{i,\pi(0)} = 0 \ , \ C_{i,\pi(1)} = a_{1,\pi(1)} + \ldots + a_{i,\pi(1)}$$

و در نهایت $\min C_{\max}$ هدف مسأله است.

معادل روش بالا می‌توان از قضیه بلمن[2] - اسوگبو[3] - نابشیما[4] [۳۴] استفاده کرد. بر اساس این قضیه $C_{\max}$ مسأله کارگاه جریانی جایگشتی با ماتریس زمان‌های پردازش $M = \left[a_{ij}\right]_{n\times m}$ مساوی وزن سنگین‌ترین مسیر ساده بین رأس‌های متناظر با a_{11} و a_{nm} در گراف جهت‌دار متناظر است.

در گراف جهت‌داری که ذکر شد رأس‌ها a_{ij}ها هستند و مطابق قضیه، تنها حرکت‌های به پایین و یا به راست از رأس a_{11} به a_{nm} مجاز هستند. در هر مسیر طی شده تنها یک بار تمام رأس‌ها وارد مجموعه رأس‌های گراف می‌شوند و در واقع رأس اول و آخر در هیچ مسیری یکی نیست در نتیجه گراف جهت‌دار مربوط دارای دور نیست و الگوریتم محاسباتی قضیه برای تعیین $C_{\max}$ همگرا می‌شود. در هر مسیر با نوع حرکت‌های ذکر شده هر بار $n+m-1$ رأس از گراف جارو شده مجموع زمان‌های این رأس‌ها وزن آن مسیر می‌باشد. تعداد کل این مسیرهای پایین – راست در هر ترتیب ثابت از سطرها برابر $\binom{n+m-2}{n-1}$ است. بعد از پیدا کردن $C_{\max}$ در یک ترتیب ثابت سطرها جایگشتی دیگر از میان تعداد $n!$ جایگشت سطرها $C_{\max}$ بعدی را به همان روش پیمایش ذکر شده بالا نتیجه خواهد داد. با تکرار این فرآیند به تعداد $n!$، $C_{\max}$ یافت می‌شوند که می‌نیمم آنها هدف مسأله کارگاه جریانی جایگشتی n کار و m ماشین خواهد بود.

1. makespan 2. Belman 3. Esogbue 4. Nabeshima

[Downloaded from mmr.khu.ac.ir on 2024-10-12]

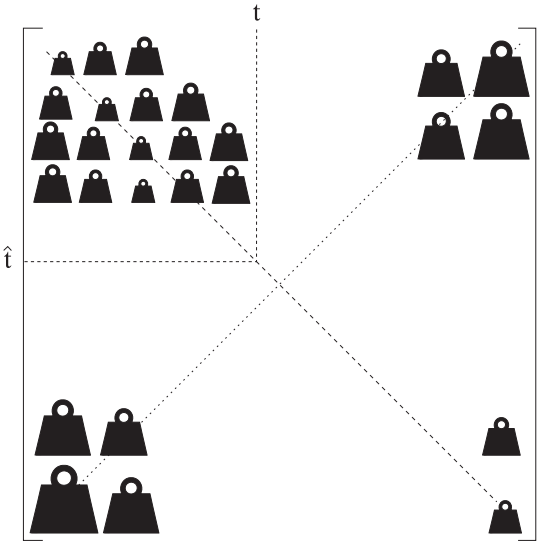
۱. الگوریتم ابتکاری جدید

برای پیدا کردن $C_{\max}$ کوچک‌تر در یک مسئلهٔ $n \times m$ کارگاه جریانی جایگشتی و در نهایت یافتن $\min C_{\max}$ و به منظور دستیابی به جواب نزدیک‌تر به جواب بهینه باید ابتدا ترتیب مناسبی برای سطرهای ماتریس M به‌دست آید.

ماتریس $K = \left[K_{ij}\right]_{n\times m}$ چنان بنا می‌شود که هر K_{ij} نشان‌دهندهٔ اندازه مناسب بودن جای سطر فعلی (قدیم) iام در مکان (سطر) جدید j ام باشد.

۱،۱ روش ساخت ماتریس K

در این روش به این نکته توجه می‌شود که مجموع درایه‌های هر ستون دلخواه j ام از M در اثر جابه‌جایی سطرها تغییر نمی‌کند. برای پیدا شدن $C_{\max}$ و می‌نیمم آن، بهترین حالت قرار گرفتن سطرها آن است که درایه‌های روی قطر اصلی (فرضی) ماتریس M (آنهایی که روی خط رسم شده از $a_{۱۱}$ به a_{nm} قرار می‌گیرند). سبک (کوچک) باشند به علاوه رفته‌رفته با نزدیک شدن به کناره‌های ماتریس، درایه‌های سنگین (بزرگ) قرار بگیرند سنگین‌ترین (بزرگترین) درایه‌ها هم در کناره‌های ماتریس در دو سر قطر فرعی (فرضی) باشند و با نزدیک شدن به مرکز سبک‌تر شوند.



شکل ۱. طرح نهایی مورد نظر قرار گرفتن درایه‌های کوچک و بزرگ در ماتریس زمان‌های پردازش

به ازای هر ستون tام از ماتریس M، محل برخورد قطر اصلی (فرضی) با ستون tام را تعیین می‌کنیم. این محل برخورد که لزوماً در درایه‌های ماتریس رخ نمی‌دهد در سطر با شماره زیر حادث می‌شود:

$$\frac{n\left(t-۱\right)+m-t}{m-۱}$$

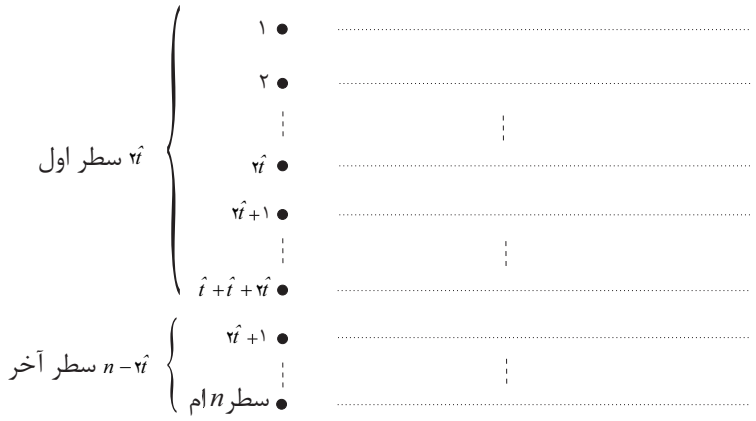
[Downloaded from mmr.khu.ac.ir on 2024-10-12]

این شماره (مقدار) ممکن است عدد صحیح نباشد و گرد شده آن (جزء صحیح مجموع آن با $\frac{۱}{۲}$) را با $\hat{t}$ نشان می‌دهیم.

بر اساس اینکه از نیمی از سطرها رد شده باشیم یا نه دو حالت در نظر می‌گیریم و در هر یک از این دو حالت به شکل خاص خود عمل می‌کنیم.

حالت اول) $\hat{t} \le \frac{n}{۲}$

در این حالت $۲\hat{t}$ سطر اول و $n-۲\hat{t}$ سطر آخر را جداگانه در نظر می‌گیریم.



شکل ۲. آرایش $۲\hat{t}$ سطر اول و $n-۲\hat{t}$ سطر آخر ماتریس زمان‌های پردازش

درایه‌های ستون tام را از سبک به سنگین مرتب می‌کنیم. این مرتب کردن باعث ایجاد ترتیب جدید $\sigma(۱),\sigma(۲),\ldots,\sigma(n)$ می شود. بنابراین داریم:

$$a_{\sigma(۱)t} \le a_{\sigma(۲)t} \le \ldots \le a_{\sigma(n)t}$$

به دلیل مساوی بودن برخی از درایه‌های، ترتیب بالا منحصر به فرد نیست و در این وضعیت یکی از ترتیب‌ها را به تصادف انتخاب می‌کنیم.

اگر فرض کنیم $\sigma(p)=q$ که در آن $q=۱,۲,\ldots,n$، $p=۱,۲,\ldots,n$ آنگاه q یعنی حاصل $\sigma(p)$ جای قرار گرفتن سطری در ترتیب قدیم یا حالت اصلی و ابتدایی M می‌باشد که در حال حاضر در مکان جدید p است.

بنابراین $p=\sigma^{-۱}(q)$ مکان جدید سطر qام ماتریس اولیه M می‌باشد. حال باید درایه‌های ذکر شده و مرتب از سبک به سنگین را به گونه‌ای مرتب کنیم که طرح مورد نظر درایه‌های سبک و سنگین حول قطر اصلی تشکیل شود. برای این منظور باید درایهٔ $a_{\sigma(۱)t}$ در سطر با شماره $\hat{t}$ و درایهٔ $a_{\sigma(۲)t}$ در سطر $\hat{t}+۱$ام، درایه $a_{\sigma(۳)t}$ در سطر با شماره $\hat{t}-۱$ و

[Downloaded from mmr.khu.ac.ir on 2024-10-12]

درایه $a_{\sigma(2)t}$ در سطر $\hat{t}+2$ام، ... ، درایه $a_{\sigma(2\hat{t}-1)t}$ در سطر یکم و درایه $a_{\sigma(2\hat{t})t}$ در سطر $2\hat{t}$ قرار گیرند. در نتیجه برای $1\le i\le \hat{t}$ ترتیب سطرهای $\sigma(2i)$ و $\sigma(2i-1)$ به شکل زیر تغییر می‌کنند:

$$\sigma(2i-1)\to \hat{t}-i+1 \quad , \quad \sigma(2i)\to \hat{t}+i$$

سپس سطرهای باقیمانده به ترتیب در مکان‌های $2\hat{t}+1,\dots,n$ قرار داده می‌شوند یعنی برای $2\hat{t}+1\le i\le n$ سطر $\sigma(i)$ در مکان i قرار می‌گیرد.

برای سادگی محاسبات، تبدیل φ به صورت زیر در نظر گرفته می‌شود:

$$\varphi(i)=\begin{cases} \hat{t}+\dfrac{i}{2} & ;\ 1\le i\le 2\hat{t} \\ \hat{t}-\dfrac{i+1}{2}+1 & ;\ 1\le i\le 2\hat{t} \\ i & ;\ 2\hat{t}+1\le i\le n \end{cases}$$

اگر مکان پایانی مورد نظر برای سطر iام با $\gamma_t(i)$ نشان داده شود محاسبات قبلی نشان می‌دهد $\gamma_t\circ\sigma=\varphi$ و در نتیجه

$$\gamma_t(i)=\left(\varphi\circ\sigma^{-1}\right)(i).$$

با توجه به آنچه گذشت $\gamma_t(i)$ نشان‌دهندهٔ مکان مناسب برای سطر iام با توجه به درایه‌های ستون tام است. اما ضرایب اهمیت مناسب بودن سطر iام در مکان $\gamma_t(i)$ به اندازه همان درایه a_{it} است، پس به ازای هر ستون t، ماتریس $n\times n$، K_t را به این صورت تعریف می‌شود که همهٔ درایه‌ها صفر و به ازای هر i، درایه سطر iام و ستون $\gamma_t(i)$ از ماتریس K_t برابر a_{it} از ماتریس M باشد.

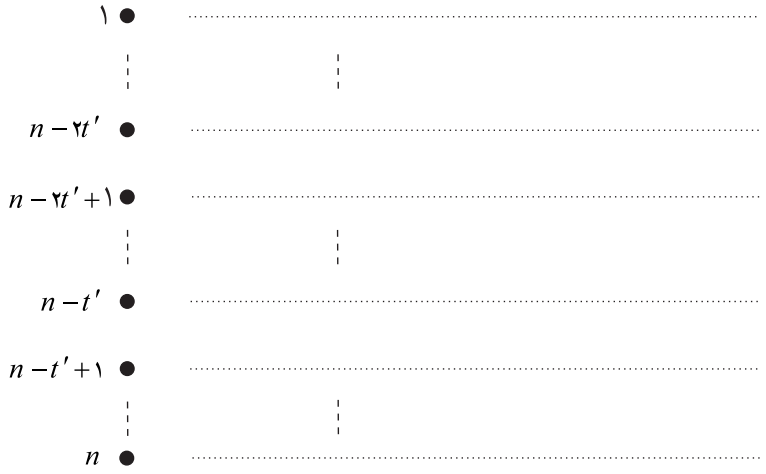
حالت دوم) $\hat{t}>\dfrac{n}{2}$

در این حالت فرض می‌شود $n-\hat{t}=t'$ و $2t'$ سطر آخر و $n-2t'$ سطر اول جدا در نظر گرفته می‌شود (در ستون آخر $\hat{t}=n$ و $t'=0$ خواهد بود)

مانند حالت اول درایه‌های ستون tام $(t=1,2,\dots,m)$ از سبک به سنگین مرتب می‌شوند تا ترتیب جدید زیر برای برای سطرها به‌دست آید:

$$a_{\sigma(1)t}\le a_{\sigma(2)t}\le\dots\le a_{\sigma(n)t}.$$

[Downloaded from mmr.khu.ac.ir on 2024-10-12]



شکل ۳. آرایش $۲t'$ سطر اول و $n-۲t'$ سطر آخر ماتریس زمان‌های پردازش

در صورت مساوی بودن بعضی از درایه‌ها و در نتیجه منحصر به فرد نبودن ترتیب، یکی از ترتیب‌ها به تصادف انتخاب می‌گردد.

در اینجا نیز $\sigma(p)=q$ به همان معنی حالت قبل است و برای تبدیل σ، $\sigma^{-۱}(q)$ نشان‌دهندهٔ مکان جدید سطر qام ماتریس M می‌باشد. دوباره این درایه‌ها به گونه‌ای مرتب می‌شوند که طرح قرار گرفتن درایه‌های سبک و سنگین حول قطر اصلی تشکیل گردد. درایه $a_{\sigma(۱)t}$ در سطر با شماره $n-t'+۱$ و درایه $a_{\sigma(۲)t}$ در سطر $n-t'$، درایه $a_{\sigma(۳)t}$ در سطر با شماره $n-t'+۲$ و درایه $a_{\sigma(۴)t}$ در سطر $n-t'-۱$، ...، درایه $a_{\sigma(۲t'-۱)t}$ در سطر با شماره n و درایه $a_{\sigma(۲t')t}$ در سطر شماره $n-۲t'+۱$ قرار می‌گیرند.

پس برای $۱\le i\le \hat{t}$ در این حالت ترتیب سطرهای $\sigma(۲i)$ و $\sigma(۲i-۱)$ به شکل زیر تغییر می‌کنند:

$$\sigma(۲i)\longrightarrow n-t'-i+۱ \quad \text{یا} \quad \hat{t}-i+۱$$

$$\sigma(۲i-۱)\longrightarrow n-t'+i \quad \text{یا} \quad \hat{t}+i$$

و سطرهای باقیمانده به ترتیب در مکان‌های $۱,\dots,n-۲t'$ قرار داده می‌شوند، یعنی برای $۲t'+۱\le i\le n$ سطر $\sigma(i)$ در مکان $n-i+۱$ قرار می‌گیرد.

مشابه حالت قبل برای سادگی محاسبات تبدیل φ را به صورت زیر در نظر می‌گیریم:

$$\varphi(i)=\begin{cases}\hat{t}-\dfrac{i}{۲}+۱ & ;\ ۱\le i\le ۲n-۲\hat{t}\\ \hat{t}+\dfrac{i+۱}{۲} & ;\ ۱\le i\le ۲n-۲\hat{t}\\ n-i+۱ & ;\ ۲n-۲\hat{t}+۱\le i\le n\end{cases}$$

[Downloaded from mmr.khu.ac.ir on 2024-10-12]

مکان پایانی مورد نظر برای سطر iام باز هم $\gamma_t(i)$ است، طبق محاسبات حالت دوم داریم $\gamma_t \circ \sigma = \varphi$ در نتیجه دوباره

$$\gamma_t(i) = \left(\varphi \circ \sigma^{-1}\right)(i).$$

بحثی مانند حالت اول نشان می‌دهد ضرایب اهمیت مناسب بودن سطر iام در مکان $\gamma_t(i)$ به اندازه همان اندازه درایه a_{it} است و به ازای هر t درایه‌های ماتریس K_t مانند حالت اول تعریف می‌شوند.

۲.۱. تشکیل ماتریس K

با توجه به آنچه گذشت در هر حالت از t و $\hat{t}$ ماتریس K_t به طریقی که گفته شد ساخته می‌شود. میزان اهمیت آن که سطر iام در مکان jام قرار گیرد از به دست آوردن مجموع ماتریس‌های K_t تعیین می‌گردد. بنابراین با ساختن ماتریس $K = K_1 + K_2 + \ldots + K_m$ ترتیب سطرها را می‌توان با طرح مورد نظر گفته شده به شکل زیر مشخص نمود.

۳.۲. روش پیدا کردن ترتیب مناسب سطرهای ماتریس M برای محاسبهٔ $C_{\max}$

۱- بزرگ‌ترین درایه ماتریس K تعیین می‌شود. اگر این درایه در سطر $i_.$ و ستون $j_.$ واقع باشد برای سطر $i_.$ مکان $j_.$ انتخاب می‌گردد. اگر بزرگ‌ترین درایه در چند جا قرار داشته باشد درایه‌ای انتخاب می‌شود که مجموع درایه‌های سطر مربوط به آن کوچک‌ترین باشد. اگر این وضعیت چند جا رخ دهد یکی از آنها به تصادف انتخاب می‌شود.

در این قدم یکی از سطرها مکانش تعیین می‌شود.

۲- با حذف سطر و ستون درایه انتخاب شده در ماتریس K در قدم قبل و ایجاد ماتریس جدید، مرحله قبل تکرار می‌گردد تا مکان یک سطر دیگر مشخص شود.

۳- با تکرار مراحل یک و دو بالا در نهایت همهٔ سطرها و ستون‌های K حذف شده و مکان همهٔ سطرهای ماتریس A معین می‌شوند.

فرآیند بالا ترتیب مناسبی از سطرهای M را به صورت طرحی که مورد نظر بود نتیجه می‌دهد. حال می‌توان به کمک قضیه بلمن - اسوگبو - نابشیما ابتدا $C_{\max}$ و سپس می‌نیمم آن را به‌دست آورد.

از آنجا که در برخی مراحل مانند مرتب کردن درایه‌های ستون M و انتخاب درایه‌های سنگین K، انتخاب تصادفی ممکن است پیش بیاید در صورت بروز این وضعیت می‌توان الگوریتم را چندین بار مثلاً یکصد بار تکرار کرد و از میان جواب‌های به‌دست آمده، بهترین را انتخاب کرد.

مثال: ماتریس M یعنی زمان‌های پردازش چهار کار روی سه ماشین را به صورت زیر در نظر می‌گیریم، مراحل تشکیل K و پیدا کردن ترتیب مناسبی از سطرها را انجام می‌دهیم:

[Downloaded from mmr.khu.ac.ir on 2024-10-12]

$$M=\begin{array}{c}\begin{bmatrix} ۹ & ۱۱ & ۵۲ \\ ۸۶ & ۹۹ & ۷۳ \\ ۸۷ & ۱۰ & ۹۵ \\ ۹۳ & ۷۴ & ۹۴ \end{bmatrix}\end{array}\begin{array}{c}(۱)\\(۲)\\(۳)\\(۴)\end{array}$$

$$n=۴ \quad , \quad m=۳$$

$$t=۱ \Rightarrow \frac{n(t-۱)+m-t}{m-۱}=\hat{t}=۱$$ حالت اول

ستون اول $\Rightarrow$ $۹<۸۶<۸۷<۹۳ \xRightarrow[\text{سطرها (همانی)}]{\text{ترتیب جدید}} M=\begin{bmatrix} ۹ & ۱۱ & ۵۲ \\ ۸۶ & ۹۹ & ۷۳ \\ ۸۷ & ۱۰ & ۹۵ \\ ۹۳ & ۷۴ & ۹۴ \end{bmatrix}\begin{array}{c}(۱)\\(۲)\\(۳)\\(۴)\end{array} \Rightarrow$

$$\sigma_۱=Id \Rightarrow \begin{cases} \sigma_۱(۱)=۱ & \Rightarrow \quad \sigma_۱^{-۱}(۱)=۱ \\ \sigma_۱(۲)=۲ & \Rightarrow \quad \sigma_۱^{-۱}(۲)=۲ \\ \sigma_۱(۳)=۳ & \Rightarrow \quad \sigma_۱^{-۱}(۳)=۳ \\ \sigma_۱(۴)=۴ & \Rightarrow \quad \sigma_۱^{-۱}(۴)=۴ \end{cases}$$

$$\gamma_۱=\varphi\circ\sigma^{-۱} \Rightarrow \begin{cases} \gamma_۱(۱) \Rightarrow & \varphi\left(\sigma_۱^{-۱}(۱)\right)=۱ \\ \gamma_۱(۲) \Rightarrow & \varphi\left(\sigma_۱^{-۱}(۲)\right)=۲ \\ \gamma_۱(۳) \Rightarrow & \varphi\left(\sigma_۱^{-۱}(۳)\right)=۳ \\ \gamma_۱(۴) \Rightarrow & \varphi\left(\sigma_۱^{-۱}(۴)\right)=۴ \end{cases}$$

$$\begin{cases} k_{۱۱}=K_۱ \text{ سطر اول و ستون یکم} & =a_{۱۱}=۹ \\ k_{۲۲}=K_۱ \text{ سطر دوم و ستون دوم} & =a_{۱۲}=۸۶ \\ k_{۳۳}=K_۱ \text{ سطر سوم و ستون سوم} & =a_{۱۳}=۸۱ \\ k_{۴۴}=K_۱ \text{ سطر چهارم و ستون چهارم} & =a_{۱۴}=۹۳ \end{cases} \quad , \quad \text{بقیه درایه‌ها}=۰$$

$$\Rightarrow K_۱=\begin{bmatrix} ۹ & ۰ & ۰ & ۰ \\ ۰ & ۸۶ & ۰ & ۰ \\ ۰ & ۰ & ۸۱ & ۰ \\ ۰ & ۰ & ۰ & ۹۳ \end{bmatrix}$$

$t=۲ \Rightarrow \hat{t}=۲$ (یا ۳) حالت اول

[Downloaded from mmr.khu.ac.ir on 2024-10-12]

$$\text{ستون دوم} \Rightarrow ۱۰ < ۱۱ < ۷۴ < ۹۹ = \begin{bmatrix} ۸۷ & ۱۰ & ۹۵ \\ ۹ & ۱۱ & ۵۲ \\ ۹۳ & ۷۴ & ۹۴ \\ ۸۶ & ۹۹ & ۷۳ \end{bmatrix} \begin{matrix} (۳) \\ (۱) \\ (۴) \\ (۲) \end{matrix}$$

$$\Rightarrow \begin{cases} \sigma_۲(۱) = ۳ \\ \sigma_۲(۲) = ۱ \\ \sigma_۲(۳) = ۴ \\ \sigma_۲(۴) = ۲ \end{cases} \Rightarrow \begin{cases} \gamma_۲(۱) = \varphi\left(\sigma_۲^{-۱}(۱)\right) = \varphi(۲) = ۳ \\ \gamma_۲(۲) = \varphi\left(\sigma_۲^{-۱}(۲)\right) = \varphi(۴) = ۴ \\ \gamma_۲(۳) = \varphi\left(\sigma_۲^{-۱}(۳)\right) = \varphi(۱) = ۲ \\ \gamma_۲(۴) = \varphi\left(\sigma_۲^{-۱}(۴)\right) = \varphi(۳) = ۱ \end{cases} \overset{t=۲}{\Rightarrow} \begin{cases} a_{۱۲} = ۱۱ \\ a_{۲۲} = ۹۹ \\ a_{۳۲} = ۱۰ \\ a_{۴۲} = ۷۴ \end{cases}$$

$$\begin{cases} K_۲ \text{ سطر اول و ستون سوم } = k_{۱۳} = a_{۱۲} = ۱۱ \\ K_۲ \text{ سطر دوم و ستون چهارم } = k_{۲۴} = a_{۲۲} = ۹۹ \\ K_۲ \text{ سطر سوم و ستون دوم } = k_{۳۲} = a_{۳۲} = ۱۰ \\ K_۲ \text{ سطر چهارم و ستون اول } = k_{۴۱} = a_{۴۲} = ۷۴ \end{cases} \quad \text{بقیه درایه‌ها} = ۰ ,$$

$$\Rightarrow K_۲ = \begin{bmatrix} ۰ & ۰ & ۱۱ & ۰ \\ ۰ & ۰ & ۰ & ۹۹ \\ ۰ & ۱۰ & ۰ & ۰ \\ ۷۴ & ۰ & ۰ & ۰ \end{bmatrix}$$

$t = ۳ \Rightarrow \hat{t} = ۴ \quad , \quad ۲\hat{t} > n$ حالت دوم

$$\text{ستون سوم} \Rightarrow ۵۲ < ۷۳ < ۹۴ < ۹۵ = \begin{bmatrix} ۹ & ۱۱ & ۵۲ \\ ۸۶ & ۹۹ & ۷۳ \\ ۹۳ & ۷۴ & ۹۴ \\ ۸۷ & ۱۰ & ۹۵ \end{bmatrix} \begin{matrix} (۱) \\ (۲) \\ (۴) \\ (۳) \end{matrix}$$

$$\Rightarrow \begin{cases} \sigma_۳(۱) = ۱ \\ \sigma_۳(۲) = ۲ \\ \sigma_۳(۳) = ۴ \\ \sigma_۳(۴) = ۳ \end{cases} \Rightarrow \begin{cases} \sigma_۳^{-۱}(۱) = ۱ \\ \sigma_۳^{-۱}(۲) = ۲ \\ \sigma_۳^{-۱}(۳) = ۳ \\ \sigma_۳^{-۱}(۴) = ۴ \end{cases}$$

[Downloaded from mmr.khu.ac.ir on 2024-10-12]

$$\begin{cases} \gamma_{۳}(۱)=\varphi\left(\sigma_{۳}^{-۱}(۱)\right)=\varphi(۱)=۴ & , a_{۱۳}=۵۲ \quad , \quad k_{۱۴}=۵۲ \\ \gamma_{۳}(۲)=\varphi\left(\sigma_{۳}^{-۱}(۲)\right)=\varphi(۲)=۳ & , a_{۲۳}=۷۳ \quad , \quad k_{۲۳}=۷۳ \\ \gamma_{۳}(۳)=\varphi\left(\sigma_{۳}^{-۱}(۳)\right)=\varphi(۴)=۱ & , a_{۳۳}=۹۵ \quad , \quad k_{۳۱}=۹۵ \\ \gamma_{۳}(۴)=\varphi\left(\sigma_{۳}^{-۱}(۴)\right)=\varphi(۳)=۲ & , a_{۴۳}=۹۴ \quad , \quad k_{۴۲}=۹۴ \end{cases} \quad \text{, بقیه درایه‌ها} = ۰$$

$$\Rightarrow K_{۳}=\begin{bmatrix} ۰ & ۰ & ۰ & ۵۲ \\ ۰ & ۰ & ۷۳ & ۰ \\ ۹۵ & ۰ & ۰ & ۰ \\ ۰ & ۹۴ & ۰ & ۰ \end{bmatrix}$$

$$K=K_{۱}+K_{۲}+K_{۳}=\begin{bmatrix} ۹ & ۰ & ۰ & ۰ \\ ۰ & ۸۶ & ۰ & ۰ \\ ۰ & ۰ & ۸۱ & ۰ \\ ۰ & ۰ & ۰ & ۹۳ \end{bmatrix}+\begin{bmatrix} ۰ & ۰ & ۱۱ & ۰ \\ ۰ & ۰ & ۰ & ۹۹ \\ ۰ & ۱۰ & ۰ & ۰ \\ ۷۴ & ۰ & ۰ & ۰ \end{bmatrix}+\begin{bmatrix} ۰ & ۰ & ۰ & ۵۲ \\ ۰ & ۰ & ۷۳ & ۰ \\ ۹۵ & ۰ & ۰ & ۰ \\ ۰ & ۹۴ & ۰ & ۰ \end{bmatrix}$$

$$=\begin{bmatrix} ۹ & ۰ & ۱۱ & ۵۲ \\ ۰ & ۸۶ & ۷۳ & ۹۹ \\ ۹۵ & ۱۰ & ۸۷ & ۰ \\ ۷۴ & ۹۴ & ۰ & ۹۳ \end{bmatrix}$$

۹۹ ⇒ سطر دوم و ستون چهارم ⇒ سطر دوم را در جای چهارم قرار می‌دهیم ⇒ در A ⇒ $\begin{bmatrix} & & \\ & & \\ & & \\ ۸۶ & ۹۹ & ۷۳ \end{bmatrix}$

۹۵ ⇒ سطر سوم و ستون اول ⇒ سطر سوم را در جای اول قرار می‌دهیم ⇒ در A ⇒ $\begin{bmatrix} ۸۷ & ۱۰ & ۹۵ \\ & & \\ & & \\ ۸۶ & ۹۹ & ۷۳ \end{bmatrix}$

۹۴ ⇒ سطر چهارم و ستون دوم ⇒ سطر چهارم را در جای دوم قرار می‌دهیم ⇒ در A ⇒ $\begin{bmatrix} ۸۷ & ۱۰ & ۹۵ \\ ۹۳ & ۷۴ & ۹۴ \\ & & \\ ۸۶ & ۹۹ & ۷۳ \end{bmatrix}$

[Downloaded from mmr.khu.ac.ir on 2024-10-12]

$$\begin{bmatrix} ۸۷ & ۱۰ & ۹۵ \\ ۹۳ & ۷۴ & ۹۴ \\ ۹ & ۱۱ & ۵۲ \\ ۸۶ & ۹۹ & ۷۳ \end{bmatrix} \Rightarrow$$ در A سطر اول را در جای سوم قرار می‌دهیم $\Rightarrow$ سطر اول و ستون سوم $\Rightarrow$ ۱۱

۳. الگوریتم NEH

الگوریتم NEH در سال ۱۹۸۳ توسط نواز، انسکور و هام پیشنهاد شد و در حال حاضر بهترین روش ابتکاری برای می‌نیمم کردن $C_{\max}$ در مسأله کارگاه جریانی جایگشتی است [۲۶]. در توضیحی ساده می‌توان این الگوریتم را به صورت زیر ارائه کرد:

۱. کارها بر اساس ترتیب نزولی زمان پردازش آن‌ها روی همه ماشین‌ها مرتب می‌شوند.

۲. در دنباله مرحله قبل دو کار اول در نظر گرفته می‌شوند و ترتیب مناسب بین آن دو برای پیدا شدن می‌نیمم $C_{\max}$ انتخاب می‌شود.

۳. بقیه کارها یکی‌یکی در قبل از کار اول، بین کار اول و دوم و بعد از کار دوم در ترتیب ثابت مرحله قبل به منظور پیدا کردن می‌نیمم $C_{\max}$ حاصل از سه کار جای داده می‌شوند.

روند بالا برای الحاق بقیه کارها به دنباله انتخابی ثابت سه کار تکرار و تا پایان ترتیب حاصل از n کار و یافتن می‌نیمم $C_{\max}$ ادامه می‌یابد. این الگوریتم نیز مانند الگوریتم تنظیم درایه‌های ستونی ماتریس زمان‌ها دنباله ابتدایی مناسبی برای اجرای قضیه بلمن – اسوگبو – نابشیما و پیدا کردن جواب بهینه یا نزدیک به آن نتیجه می‌دهد.

۴. ارزیابی محاسباتی

در این بخش الگوریتم ارائه شده مورد ارزیابی قرار می‌گیرد. برای این منظور نتایج الگوریتم ابداع شده تنظیمات ستونی با نتایج الگوریتم NEH در مسأله کارگاه جریانی جایگشتی با هدف می‌نیمم کردن $C_{\max}$ مورد مقایسه قرار می‌گیرد. این مقایسه روی مسایل آزمایشگاه تیلارد انجام می‌شود. مسایل تیلارد ۱۲۰ مسأله $n \times m$ در دوازده گروه با اندازه مختلف $n \in \{۲۰,۵۰,۱۰۰,۲۰۰,۵۰۰\}$ و $m \in \{۵,۱۰,۲۰\}$ و ده مسأله در هر اندازه می‌باشند. مؤلفین نظریه کارگاه جریانی جایگشتی به طور وسیعی این مسایل را در الگوریتم‌های ابتکاری خود به کار می‌برند. تعدادی از مسأله‌ها هنوز حل نشده باقی مانده‌اند. تا حال حاضر سه مورد در ۲۰×۵۰۰، شش مورد در ۲۰×۲۰۰، نه مورد در ۲۰×۱۰۰ و همه ده مورد در ۲۰×۵۰ مسأله باز می‌باشند. برای بقیه جواب بهینه و دقیق به دست آمده است.

[Downloaded from mmr.khu.ac.ir on 2024-10-12]

الگوریتم در زبان پایتون[1] اجرا و همه آزمایش‌ها روی یک رایانه Quad-core Intel core i7 با سرعت ۲/۶ گیگا هرتز و حافظه ۱۶ گیگابایتی انجام شد.

اندازه اجرای به کار رفته درصد انحراف نسبی روی جواب‌های NEH در هر یک از ۱۲۰ مورد است که بر اساس آن

$$\text{درصد انحراف نسبی} = \frac{(\text{جواب NEH}) - (\text{جواب روش ابتکاری})}{(\text{جواب NEH})} \times ۱۰۰$$

.

در جدول زیر می‌نیمم زمان حداکثر در جریان حاصل از الگوریتم NEH روی مسایل آزمایشگاه تیلارد [۲۷] و حداقل $C_{\max}$ الگوریتم REC گردآوری شده‌اند.

جدول ۱. می‌نیمم $C_{\max}$ برای الگوریتم ابتکاری و NEH و درصد انحراف نسبی

اندازه مسأله	شماره مسأله	جواب NEH	جواب RCE	درصد انحراف نسبی الگوریتم ابتکاری	اندازه مسأله	شماره مسأله	جواب NEH	جواب RCE	درصد انحراف نسبی الگوریتم ابتکاری
۲۰×۵	۱	۱۲۸۶	۱۴۶۲	۱۳	۱۰۰×۵	۶۱	۵۵۱۹	۶۱۰۸	۱۰
	۲	۱۳۶۵	۱۵۲۷	۱۱		۶۲	۵۳۴۸	۵۷۹۰	۸
	۳	۱۱۵۹	۱۴۱۰	۲۱		۶۳	۵۲۱۹	۵۷۰۸	۹
	۴	۱۳۲۵	۱۶۴۳	۲۴		۶۴	۵۰۲۳	۵۴۲۱	۷
	۵	۱۳۰۵	۱۴۱۹	۸		۶۵	۵۲۶۶	۵۹۱۷	۱۲
	۶	۱۲۲۸	۱۵۶۲	۲۷		۶۶	۵۱۳۹	۵۶۴۳	۹
	۷	۱۲۷۸	۱۴۷۰	۱۵		۶۷	۵۲۵۹	۶۰۵۳	۱۵
	۸	۱۲۲۳	۱۵۴۹	۲۶		۶۸	۵۱۲۰	۶۰۱۳	۱۷
	۹	۱۲۹۱	۱۳۸۸	۷		۶۹	۵۴۸۹	۵۹۸۲	۸
	۱۰	۱۱۵۱	۱۲۴۷	۸		۷۰	۵۳۴۱	۶۱۶۶	۱۵
۲۰×۱۰	۱۱	۱۶۸۰	۱۹۶۷	۱۷	۱۰۰×۱۰	۷۱	۵۸۴۶	۶۶۷۳	۱۴
	۱۲	۱۷۲۹	۲۲۲۱	۲۸		۷۲	۵۴۵۳	۶۴۶۶	۱۸
	۱۳	۱۵۵۷	۱۸۴۳	۱۸		۷۳	۵۸۲۴	۶۲۹۰	۸
	۱۴	۱۴۳۹	۱۶۸۸	۱۷		۷۴	۵۹۲۹	۶۹۱۳	۱۶
	۱۵	۱۵۰۲	۱۸۱۳	۲۰		۷۵	۵۶۷۹	۶۳۷۰	۱۲
	۱۶	۱۴۵۳	۱۴۷۳	۱۹		۷۶	۵۳۷۵	۶۴۵۲	۲۰
	۱۷	۱۵۶۲	۱۷۶۵	۱۲		۷۷	۵۷۰۴	۶۴۳۳	۱۲
	۱۸	۱۶۰۹	۱۹۰۰	۱۸		۷۸	۵۷۶۰	۶۵۶۲	۱۳
	۱۹	۱۶۴۷	۱۸۹۴	۱۴		۷۹	۶۰۳۲	۶۵۴۵	۸
	۲۰	۱۶۵۳	۱۹۸۴	۲۰		۸۰	۵۹۱۸	۶۷۸۶	۱۴

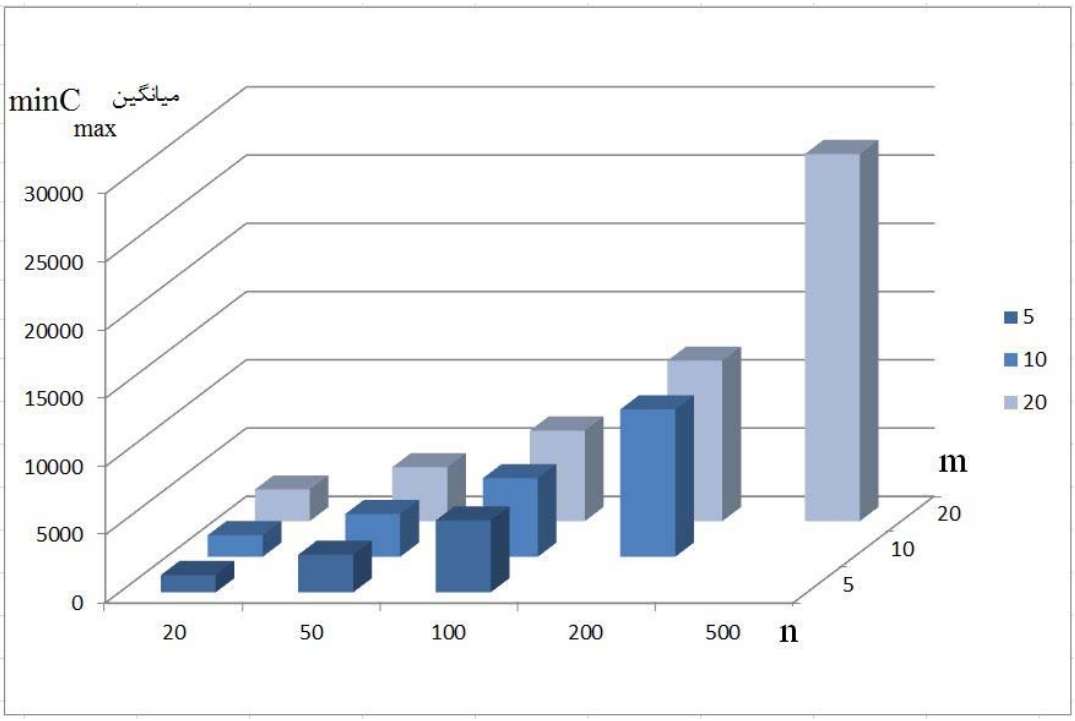
1. Python

[Downloaded from mmr.khu.ac.ir on 2024-10-12]

درصد انحراف نسبی الگوریتم ابتکاری	جواب RCE	جواب NEH	شماره مسأله	اندازه مسأله	درصد انحراف نسبی الگوریتم ابتکاری	جواب RCE	جواب NEH	شماره مسأله	اندازه مسأله
۱۵	۷۵۷۵	۶۵۴۱	۸۱	۱۰۰×۲۰	۹	۲۶۴۳	۲۴۱۰	۲۱	۲۰×۲۰
۱۴	۷۴۸۵	۶۵۲۳	۸۲		۱۶	۲۴۹۹	۲۱۵۰	۲۲	
۱۴	۷۶۲۵	۶۶۳۹	۸۳		۱۳	۲۷۳۹	۲۴۱۱	۲۳	
۱۶	۷۶۲۷	۶۵۵۷	۸۴		۲۳	۲۷۹۸	۲۲۶۲	۲۴	
۱۷	۷۸۴۰	۶۶۹۵	۸۵		۹	۲۶۲۳	۲۳۹۷	۲۵	
۱۷	۷۸۰۸	۶۶۶۴	۸۶		۱۸	۲۷۹۰	۲۳۴۹	۲۶	
۱۵	۷۶۳۹	۶۶۳۲	۸۷		۱۲	۲۶۵۳	۲۳۶۲	۲۷	
۱۶	۷۸۶۴	۶۷۳۹	۸۸		۱۵	۲۵۸۸	۲۲۴۹	۲۸	
۱۳	۷۶۰۸	۶۶۷۷	۸۹		۱۸	۲۷۴۰	۲۳۲۰	۲۹	
۱۴	۷۶۳۴	۶۶۷۷	۹۰		۱۵	۲۶۴۰	۲۲۷۷	۳۰	
۱۰	۱۲۰۱۷	۱۰۹۱۲	۹۱	۲۰۰×۱۰	۱۵	۳۱۵۸	۲۷۳۳	۳۱	۵۰×۵
۱۳	۱۲۱۹۴	۱۰۷۱۶	۹۲		۱۶	۳۳۱۷	۲۸۴۳	۳۲	
۱۰	۱۲۱۵۸	۱۱۰۲۵	۹۳		۱۳	۳۰۰۸	۲۶۴۰	۳۳	
۶	۱۱۷۶۳	۱۱۰۵۷	۹۴		۱۸	۳۲۹۳	۲۷۸۲	۳۴	
۱۲	۱۱۹۸۶	۱۰۶۴۵	۹۵		۶	۳۰۵۱	۲۸۶۸	۳۵	
۲۰	۱۲۵۵۹	۱۰۴۵۸	۹۶		۱۲	۳۲۰۰	۲۸۵۰	۳۶	
۱۲	۱۲۴۰۵	۱۰۹۸۹	۹۷		۱۴	۳۱۵۲	۲۷۵۸	۳۷	
۱۴	۱۲۳۴۹	۱۰۸۲۹	۹۸		۱۴	۳۱۰۸	۲۷۲۱	۳۸	
۱۴	۱۲۱۲۳	۱۰۵۷۴	۹۹		۱۸	۳۰۴۹	۲۵۷۶	۳۹	
۱۲	۱۲۱۷۰	۱۰۸۰۷	۱۰۰		۱۲	۳۱۳۶	۲۷۹۰	۴۰	
۱۳	۱۳۲۰۵	۱۱۶۲۵	۱۰۱	۲۰۰×۲۰	۱۵	۳۶۰۷	۳۱۳۵	۴۱	۵۰×۱۰
۱۱	۱۳۰۲۶	۱۱۶۷۵	۱۰۲		۱۷	۳۵۶۳	۳۰۳۲	۴۲	
۱۲	۱۳۳۶۰	۱۱۸۵۲	۱۰۳		۲۰	۳۵۹۹	۲۹۸۶	۴۳	
۱۳	۱۳۴۳۵	۱۱۸۰۳	۱۰۴		۱۴	۳۶۶۹	۳۱۹۸	۴۴	
۱۳	۱۳۳۰۸	۱۱۶۸۵	۱۰۵		۲۳	۳۸۸۷	۳۱۶۰	۴۵	
۱۷	۱۳۶۴۳	۱۱۶۲۹	۱۰۶		۱۴	۳۶۳۸	۳۱۷۸	۴۶	
۱۳	۱۳۴۱۴	۱۱۸۳۳	۱۰۷		۹	۳۵۸۹	۳۲۷۷	۴۷	
۱۳	۱۳۵۰۸	۱۱۹۱۳	۱۰۸		۱۸	۳۶۹۹	۳۱۲۳	۴۸	
۱۴	۱۳۴۰۷	۱۱۶۷۳	۱۰۹		۲۱	۳۶۵۳	۳۰۰۲	۴۹	
۱۴	۱۳۵۶۸	۱۱۸۶۹	۱۱۰		۱۳	۳۶۸۲	۳۲۵۷	۵۰	

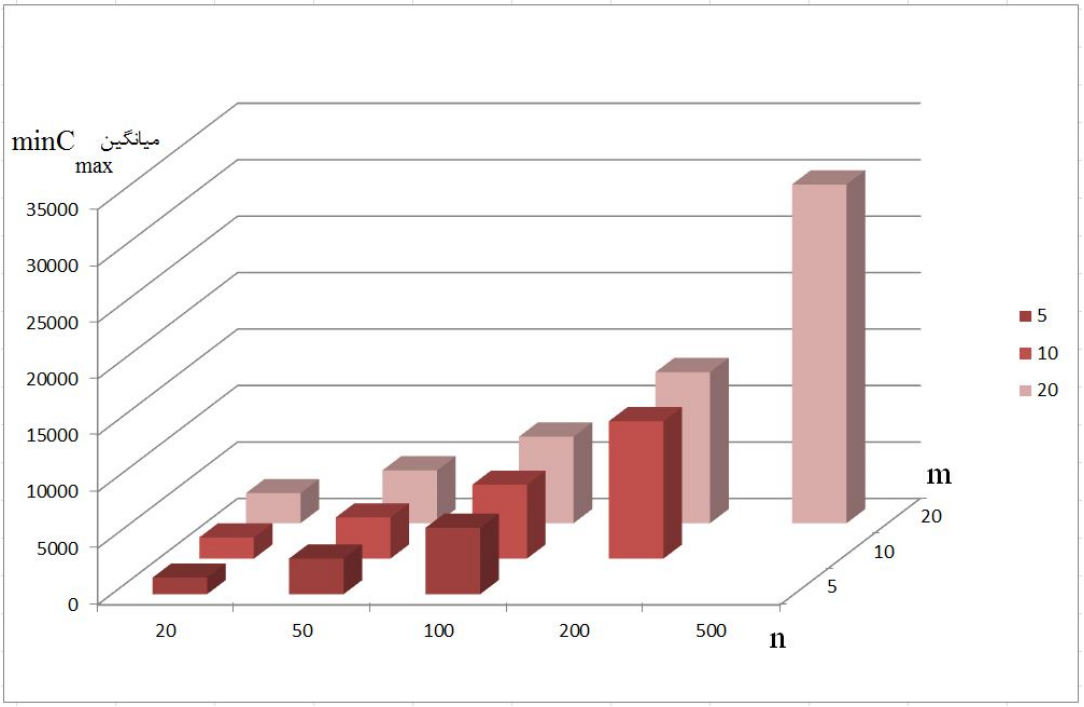
[Downloaded from mmr.khu.ac.ir on 2024-10-12]

درصد انحراف نسبی الگوریتم ابتکاری	جواب RCE	جواب NEH	شماره مسأله	اندازه مسأله	درصد انحراف نسبی الگوریتم ابتکاری	جواب RCE	جواب NEH	شماره مسأله	اندازه مسأله
۱۵	۳۰۷۴۱	۲۶۶۷۰	۱۱۱	۵۰۰×۲۰	۱۹	۴۸۹۷	۴۰۸۲	۵۱	۵۰×۲۰
۹	۲۹۷۶۵	۲۷۲۳۲	۱۱۲		۱۸	۴۶۴۲	۳۹۲۱	۵۲	
۱۱	۲۹۹۸۹	۲۶۸۴۸	۱۱۳		۱۵	۴۵۳۴	۳۹۲۷	۵۳	
۱۵	۲۹۹۰۶	۲۷۰۵۵	۱۱۴		۲۲	۴۸۵۶	۳۹۶۹	۵۴	
۱۳	۳۰۲۹۷	۲۶۷۲۷	۱۱۵		۲۰	۴۶۲۷	۳۸۳۵	۵۵	
۹	۲۹۴۳۴	۲۶۹۹۲	۱۱۶		۱۷	۴۵۹۱	۳۹۱۴	۵۶	
۱۱	۲۹۷۸۰	۲۶۷۹۷	۱۱۷		۱۶	۴۵۹۸	۳۹۵۲	۵۷	
۹	۲۹۷۹۸	۲۷۱۳۸	۱۱۸		۱۳	۴۴۵۰	۳۹۳۸	۵۸	
۱۳	۳۰۱۹۲	۲۶۶۳۱	۱۱۹		۲۱	۴۸۰۵	۳۹۵۲	۵۹	
۱۱	۳۰۱۳۲	۲۶۹۸۴	۱۲۰		۱۹	۴۸۸۴	۴۰۷۹	۶۰	



شکل ۴. هیستوگرام سه‌بعدی نتایج NEH

[Downloaded from mmr.khu.ac.ir on 2024-10-12]



شکل ۵. هیستوگرام سه‌بعدی نتایج RCE

در مسأله‌های ۳۵ و ۹۴ تیلارد درصد انحراف نسبی ۶٪ به دست آمده است. در مسأله ۹ تیلارد مقدار ۷٪ برای درصد انحراف نسبی حاصل شده است. در مسأله‌های ۵، ۱۰، ۶۲، ۶۹، ۷۳ و ۷۸ درصد انحراف نسبی ۸٪ و در مسأله‌های ۲۱، ۲۵، ۴۷، ۶۳، ۶۶، ۱۱۲، ۱۱۶ و ۱۱۸ این مقدار ۹٪ به دست آمده است. درصد انحراف نسبی در مسأله‌های ۶۱، ۹۱ و ۹۳، ۱۰٪ محاسبه شده است. این نتایج میزان نزدیکی جواب‌های الگوریتم ابتکاری را به نحو مناسبی به جواب‌های NEH نشان می‌دهند. همچنان‌که جدول زیر در مورد میانگین درصد انحراف‌های نسبی مسأله‌های تیلارد نشان می‌دهد الگوریتم RCE برای مسأله‌های بزرگ بسیار مناسب‌تر عمل می‌کند و این خود در مسایل زمان‌بندی برای کارایی الگوریتم‌ها شاخص بسیار مهمی می‌باشد.

جدول ۲. میانگین درصدهای انحراف نسبی RCE روی جواب‌های NEH

ماشین/کار	۲۰×۵	۲۰×۱۰	۲۰×۲۰	۵۰×۵	۵۰×۱۰	۵۰×۲۰	۱۰۰×۵	۱۰۰×۱۰	۱۰۰×۲۰	۲۰۰×۱۰	۲۰۰×۲۰	۵۰۰×۲۰	میانگین کل
RCE	٪۱۶	٪۱۸	٪۱۴	٪۱۳	٪۱۶	٪۱۸	٪۱۱	٪۱۳	٪۱۵	٪۱۲	٪۱۳	٪۱۱	٪۱۴

همچنان‌که جدول شماره ۴ در [۳۵] نشان می‌دهد عملکرد RCE به طور کامل از الگوریتم‌های مشهور جانسون، پیج، گوپتا بسیار بهتر بوده حتی در بعضی از مسأله‌های کوچک تیلارد به عنوان مثال در مسأله ۲۰×۲۰ همانند الگوریتم هوندال و راجگوپال بوده از الگوریتم‌های داننبرینگ و پالمر کاراتر است. در مسأله ۲۰×۵۰ نیز از الگوریتم داننبرینگ بهتر می‌باشد.

[Downloaded from mmr.khu.ac.ir on 2024-10-12]

۵. نتایج

در این مقاله الگوریتمی جدید و ابتکاری (RCE) ارائه گردید که عملکرد بهتری نسبت به تعدادی از الگوریتم‌های ابتکاری دیگر برای حل مسأله کارگاه جریانی جایگشتی با هدف می‌نیمم کردن $C_{\max}$ دارد. این الگوریتم در تعدادی از مسایل استاندارد تیلارد مخصوصاً در مسأله‌های بزرگ جواب‌های نزدیکی به جواب‌های NEH دارد به علاوه در بعضی از مسأله‌های کوچک نیز جواب‌های بهتری از چندین الگوریتم مشهور دیگر نتیجه می‌دهد. با توجه به خاصیت‌های ریاضی الگوریتم پیشنهاد شده می‌توان به خوبی آن را به عنوان یک روش موفق به کار برد.

References

1. M. R. Garey, D. S. Johnson and R. Sethi, The complexity of flowshop and jobshop scheduling, Mathematics of operations research, **1**(2) (1976), 117-129.
2. E. S. Page, An approach to the scheduling of jobs on machines, Journal of the Royal Statistical Society: Series B (Methodological), **23**(2) (1961), 484-492.
3. D. S. Palmer, Sequencing jobs through a multi-stage process in the minimum total time - a quick method of obtaining a near optimum, Journal of the Operational Research Society, **16**(1) (1965), 101-107.
4. V. A. Petrov, Flow Line Group, production planning. Business Publications, London, 1966.
5. H. G. Campbell, R. A. Dudek and M. L. Smith, A heuristic algorithm for the n job, m machine sequencing problem, Management science, **16**(10) (1970), B-630.
6. J. N. Gupta, A functional heuristic algorithm for the flowshop scheduling problem, Journal of the Operational Research Society, **22**(1) (1971), 39-47.
7. D. G. Dannenbring, An evaluation of flow shop sequencing heuristics, Management science, **23**(11) (1977), 1174-1182.
8. J. R. King and A. S. Spachis, Heuristics for flow-shop scheduling, International Journal of Production Research, **18**(3) (1980). 345-357.
9. J. P. Stinson and A. W. Smith, A heuristic proǵramminǵ procedure for sequencinǵ the static flowshop, The International Journal of Production Research, **20**(6) (1982), 753-764.
10. M. Nawaz, E. E. Enscore Jr and I. Ham, A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem, Omega, **11**(1) (1983), 91-95.
11. T. S. Hundal and J. Rajgopal, An extension of Palmer's heuristic for the flow shop scheduling problem, International Journal of Production Research, **26**(6) (1988), 1119-1124.
12. M. Widmer and A. Hertz, A new heuristic method for the flow shop sequencing problem, European Journal of Operational Research, **41**(2) (1989), 186-193.
13. E. Taillard, Some efficient heuristic methods for the flow shop sequencing problem, European journal of Operational research, **47**(1) (1990), 65-74.
14. J. C. Ho and Y. L. Chang, A new heuristic for the n-job, M-machine flow-shop problem, European Journal of Operational Research, **52**(2) (1991), 194-202.

[Downloaded from mmr.khu.ac.ir on 2024-10-12]

15. S. Sarin and M. Lefoka, Scheduling heuristic for the n-jobm-machine flow shop, Omega, **21**(2) (1993), 229-234.
16. J. V. Moccellin, A new heuristic method for the permutation flow shop scheduling problem, Journal of the Operational Research Society, **46**(7) (1995), 883-886.
17. T. C. Lai, A note on heuristics of flow-shop scheduling, Operations Research, **44**(4) (1996), 648-652.
18. C. Koulamas, A new constructive heuristic for the flowshop scheduling problem, European Journal of Operational Research, **105**(1) (1998), 66-71.
19. S. M. A. Suliman, A two-phase heuristic approach to the permutation flow-shop scheduling problem, International Journal of Production Economics, **64**(1) (2000), 143-152.
20. J. M. Framinan, R. Leisten and C. Rajendran, Different initial sequences for the heuristic of Nawaz, Enscore and Ham to minimize makespan, idletime or flowtime in the static permutation flowshop sequencing problem. International Journal of Production Research, **41**(1) (2003), 121-148.
21. M. E. Kurz and R. G. Askin, Scheduling flexible flow lines with sequence-dependent setup times, European Journal of Operational Research, **159**(1) (2004), 66-82.
22. J. Y. T. Leung, H. Li and M. Pinedo, Order scheduling in an environment with dedicated resources in parallel, Journal of Scheduling, **8**(5) (2005), 355-386.
23. A. Allahverdi and F. S. Al-Anzi, A PSO and a Tabu search heuristics for the assembly scheduling problem of the two-stage distributed database application, Computers & Operations Research, **33**(4) (2006), 1056-1080.
24. K. Alaykýran, O. Engin and A. Döyen, Using ant colony optimization to solve hybrid flow shop scheduling problems, The international journal of advanced manufacturing technology, **35**(5) (2007), 541-550.
25. P. J. Kalczynski and J. Kamburowski, An improved NEH heuristic to minimize makespan in permutation flow shops, Computers & Operations Research, **35**(9) (2008), 3001-3008.
26. S. F. Rad, R. Ruiz and N. Boroojerdian, New high performing heuristics for minimizing makespan in permutation flowshops, Omega, **37**(2) (2009), 331-345.
27. M. Ancău, On Solving Flowshop Scheduling Problems, Proceedings of the Romanian Academy. Series A, **13**(1) (2012), 71-79.
28. A. Malik and A. K. Dhingra, Comparative analysis of heuristics for make span minimizing in flow shop scheduling, International Journal of Innovations in Engineering and Technology, **2**(4) (2013), 263-269.
29. J. Xu, Y. Yin, T. C. E. Cheng, C. C. Wu and S. Gu, An improved memetic algorithm based on a dynamic neighborhood for the permutation flowshop scheduling problem, International Journal of Production Research, **52**(4) (2014), 1188-1199.
30. W. Liu, Y. Jin and M. Price, A new Nawaz–Enscore–Ham-based heuristic for permutation flow-shop problems with bicriteria of makespan and machine idle time, Engineering Optimization, **48**(10) (2016), 1808-1822.
31. A. Brum and M. Ritt, Automatic Design of Heuristics for Minimizing the Makespan in Permutation Flow Shops, In 2018 IEEE Congress on Evolutionary Computation (CEC) (pp. 1-8). IEEE.

[Downloaded from mmr.khu.ac.ir on 2024-10-12]

32. R. Nurdiansyah, O. A. W. Rijanto, B. Santosa and S. E. Wiratno, An Improved Differential Evolution Algorithm for Permutation Flow Shop Scheduling Problem, International Journal of Operations Research, **16**(2) (2019), 37-44.

33. C. Sauvey and and N. Sauer, Two NEH Heuristic Improvements for Flowshop Scheduling Problem with Makespan Criterion. Algorithms, **13**(5) (2020), 112.

34. R. Bellman, A. O. Esogbue and I. Nabeshima, Mathematical aspects of scheduling and applications: modern applied mathematics and computer science, Vol. 4, Elsevier, 2014.

35. S. Farahmand Rad, An effective new heuristic algorithm for solving permutation flow shop scheduling problem, Transactions on Combinatorics (in press).

[Downloaded from mmr.khu.ac.ir on 2024-10-12]