Analysis of different methods for combining and improving solutions with evolutionary metaheuristics in flow-shop production problems

Begoña González [1][0000-0002-7915-0655], Daniel A. Rossit [2,4][0000-0002-2381-4352], Mariano Frutos [2,3][0000-0003-2585-4195] and Máximo Méndez[1][0000-0002-7133-7108]

[1] Instituto Universitario SIANI, Universidad de Las Palmas de Gran Canaria (ULPGC), Las Palmas de Gran Canaria (35017), España
[2] Departamento de Ingeniería, Universidad Nacional del Sur (UNS), Bahía Blanca (8000), Argentina
[3] Instituto de Investigaciones Económicas y Sociales del Sur (IIESS UNS-CONICET), Bahía Blanca (8000), Argentina
[4] Instituto de Matemática de Bahía Blanca (INMABB UNS-CONICET), Bahía Blanca (8000), Argentina
bego.landin@ulpgc.es, daniel.rossit@uns.edu.ar, mfrutos@uns.edu.ar, maximo.mendez@ulpgc.es

Abstract. In recent years, there has been growth in the development of algorithms for solving optimization problems in flow-shop production environments involving different methods of combining and improving solutions. On the other hand, it is well known that, for a good performance of an evolutionary algorithm, it is essential to adjust its parameters and methods to a given problem. In this line, and considering *makespan* and *total tardiness* as evaluation criteria, this paper analyzes several methods of combination and improvement of the solutions, with a genetic algorithm and a scatter search algorithm, to evaluate their impact on the diversity of the solutions generated and on the convergence of both metaheuristics, when solving the permutation flow-shop scheduling problem with an instance of 50 jobs and 10 machines. The analysis of the results seeks to improve the understanding of both the behavior of these metaheuristics and the combination and improvement methods considered, so that practical guidelines for real applications can be obtained.

Keywords: Permutation Flow-shop Scheduling Problem, Optimization, Genetic Algorithms, Scatter Search.

Análisis de diferentes métodos de combinación y mejora de soluciones con metaheurísticas evolutivas en problemas de producción flow-shop

Resumen. En años recientes, se ha producido un crecimiento en el desarrollo de algoritmos para resolver problemas de optimización en entornos de producción flow-shop que involucran diferentes métodos de combinación y mejora de las soluciones. Por otro lado, es bien conocido que, para un buen rendimiento de un algoritmo evolutivo, es esencial ajustar sus parámetros y métodos a un problema determinado. En este contexto, y considerando como criterios de evaluación el *makespan* y el *total tardiness*, en este trabajo se analizan varios métodos de combinación y mejora de las soluciones, con un algoritmo genético y un algoritmo de búsqueda dispersa (*scatter search*), para evaluar su impacto en la diversidad de las soluciones generadas y en la convergencia de ambas metaheurísticas, al resolver el problema de programación flow-shop permutacional (*permutation flow-shop scheduling problem*) con una instancia de 50 trabajos y 10 máquinas. El análisis de los resultados busca mejorar la comprensión tanto del comportamiento de estas metaheurística como de los métodos de combinación y mejora considerados, de forma que se pueda obtener directrices prácticas para aplicaciones reales.

Palabras clave: Permutation Flow-Shop Scheduling Problem, Optimización, Algoritmos Genéticos, Scatter Search.

1 Introducción

En los últimos años, se ha observado un aumento significativo en el desarrollo de algoritmos destinados a resolver problemas complejos en entornos de producción flow-shop (Uman et al., 2022; Sharma et al., 2021; Jeen Robert et al, 2017). En este contexto, el presente trabajo se centra en realizar un estudio exhaustivo y comprensivo de diferentes métodos de combinación y mejora de las soluciones, con un algoritmo genético (AG) y un algoritmo de búsqueda dispersa (Scatter Search, SS), empleando como criterios de evaluación el *makespan* (Khatami et al., 2023) y el *total tardiness* (Koulamas, 2020), al ser estos los criterios más estudiados para equilibrar la eficiencia operativa y ofrecer una buena capacidad de discriminación entre soluciones, lo que los hace especialmente útiles en procesos de optimización (Pinedo, 2016).

La idea principal es analizar en detalle algunos métodos de combinación de soluciones, que dirigen la exploración del espacio de soluciones factibles, y algunos métodos de mejora de las soluciones, que dirigen la explotación de dicho espacio. Se busca evaluar el impacto de estos métodos en términos de diversidad y convergencia, con el fin de comprender mejor su comportamiento y su influencia en el proceso de búsqueda. Se espera que los resultados de este estudio contribuyan a mejorar la comprensión de los mecanismos subyacentes a las metaheurísticas evolutivas aplicadas a problemas de

producción flow-shop, así como a proporcionar directrices prácticas para la selección y configuración de estos algoritmos en aplicaciones reales.

2 Problema del flow-shop

El problema en un sistema productivo de tipo flow-shop requiere que n trabajos (J_1, …, J_n) pasen secuencialmente por una serie de m máquinas (M_1, …, M_m). Cada trabajo debe ser procesado en cada máquina solo una vez, en un orden establecido, primero M_1, luego M_2, después M_3, y así hasta completar todas las máquinas. Cada máquina puede procesar un solo trabajo por vez. El problema consiste en identificar el orden en que deben secuenciarse los trabajos de manera de minimizar algún objetivo que el sistema demande. En este trabajo se perseguirán de manera individual dos objetivos clásicos, el *makespan* y el *total tardiness*. El *makespan* se define como el tiempo que transcurre entre que el primer trabajo se inicia en la primera máquina y el último se completa en la última máquina. El *total tardiness* se refiere a la suma de los retrasos individuales de todos los trabajos en el flujo de producción.

Este problema tiene dos variantes principales: permutativa (PFS, *permutation flow-shop*) y no permutativa (NPFS, *non-permutation flow-shop*). En la variante PFS, todas las tareas siguen una secuencia idéntica en todas las máquinas. Por ejemplo, si el programa es J_1-J_2, en cada máquina se procesa primero J_1 y luego J_2. Por otro lado, la versión NPFS permite que las secuencias de tareas sean diferentes en distintas máquinas. Por ejemplo, en la máquina M_1, la secuencia puede ser J_1- J_2, mientras que en la máquina M_2 podría ser J_2- J_1. Aunque la versión NPFS es más compleja que la PFS, es importante destacar que varios estudios han informado mejoras marginales, generalmente entre el 1% y el 3%, en el *makespan* al emplear la variante NPFS en comparación con la PFS (Rossit et al., 2018).

En este trabajo fue utilizada la versión PFS y la formulación de los objetivos fue extraída para el *makespan* de (Khatami et al., 2023) y para el *total tardiness* de (Koulamas, 2020).

3 Metaheurísticas evolutivas y sus métodos de combinación y mejora de soluciones

Los algoritmos genéticos (Holland, 1992) son métodos de búsqueda inspirados en los procesos naturales de evolución y selección natural, que se utilizan para resolver problemas de optimización en una amplia gama de campos, desde la ingeniería hasta la inteligencia artificial. Los AGs parten de una población inicial de soluciones potenciales para el problema en cuestión, y la van mejorando generación a generación aplicando secuencialmente operadores de selección, cruce y mutación, hasta que se verifica un criterio de parada, por ejemplo, un número máximo de generaciones (iteraciones).

Los algoritmos de búsqueda dispersa (Glover, 1977) son metaheurísticas evolutivas basadas en poblaciones, como los AGs, que se caracterizan por mantener un conjunto de referencia de soluciones diversas, que se combinan para generar nuevas soluciones

candidatas. Los SS consisten en cinco métodos (Laguna et al, 1999), entre los que se encuentran un método de mejora que permite modificar las soluciones para mejorar su calidad (en términos de función objetivo) o su viabilidad, y un método de combinación para crear nuevas soluciones.

En el caso de los AGs, los operadores de cruce se pueden entender como métodos de combinación y los operadores de mutación como métodos de mejora. De hecho, así se han considerado en este trabajo para estudiar el desempeño de algunos de estos métodos tanto con AG como con SS.

4 Experimentaciones y resultados

Para la experimentación se tomó una instancia de 50 trabajos con 10 máquinas. Las soluciones se codificaron como permutaciones del número de trabajos, es decir, cada alelo de la solución contiene un número entero entre 1 y el número de trabajos. Además, se consideraron seis métodos de combinación:

1. Partially-Mapped Crossover (PMX) (Goldberg et al, 1985).
2. Order Crossover (OX) (Davis, 1985).
3. Cycle Crossover (CX) (Oliver et al, 1987).
4. Modified-Cycle Crossover (CX2) (Hussain et al, 2017).
5. Non-Wrapping Order Crossover (NWOX) (Cicirello, 2006).
6. Partially-Mapped Crossover 2 (PMX2) (Hussain et al, 2019).

Y tres métodos de mejora de las soluciones:

1. Exchange (EXC): se seleccionan dos alelos aleatoriamente y se intercambian.
2. Insertion (INS): se seleccionan dos alelos aleatoriamente y se coloca el segundo justo después del primero.
3. Inversion (INV): se seleccionan dos alelos aleatoriamente y se invierte el orden de los alelos comprendidos entre ellos.

En el caso de AG se llevó a cabo un diseño factorial con $6\times3^2\times5$ tratamientos, que involucró dos factores más: *probabilidad de cruce*: 0.80 (1), 0.85 (2) y 0.90 (3); y *probabilidad de mutación*: 0.10 (1), 0.20 (2), 0.3 (3), 0.35 (4) y 0.40 (5). En el caso del SS se añadió el factor *tamaño de la búsqueda local*:10 (1), 20 (2), 30 (3); obteniendo así un diseño factorial con 6×3^2 tratamientos. Se realizaron 30 ejecuciones por cada tratamiento. En el caso del AG se usó la selección por torneo y se aplicó elitismo (los 2 mejores individuos de cada generación pasaron a la siguiente generación). En el caso de SS se usaron los métodos de generación de diversidad, de actualización del conjunto de referencia y de generación de subconjuntos utilizados en (Rossit et al, 2024). En ambos casos se fijó como criterio de parada de los algoritmos un número máximo de 100 000 evaluaciones de la función objetivo.

Para el análisis de los resultados se procedió a representar gráficamente los tratamientos con gráficas de cajas y bigotes (ver Figs. 1–6), y se realizaron test de hipótesis adecuados para confirmar (o desmentir) las conclusiones visuales. Así, se pudo concluir en términos de convergencia:

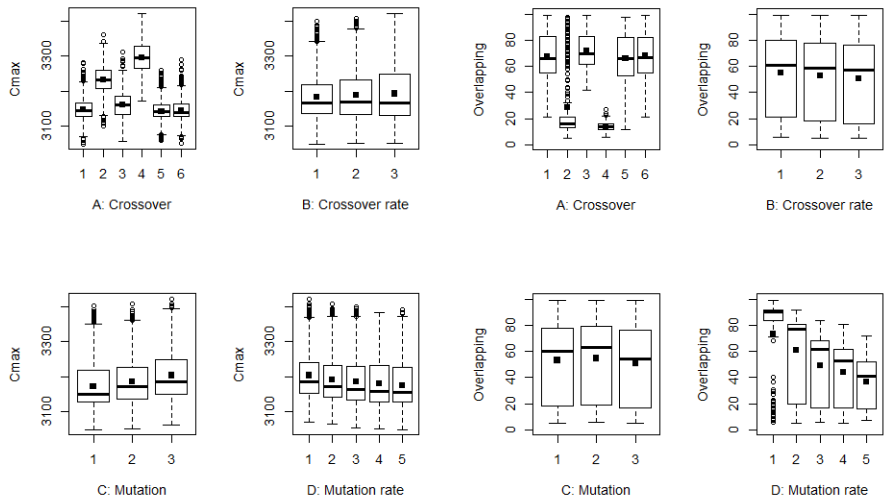

Fig. 1. AG con 100 individuos y 1000 generaciones. Makespan (Cmax) y porcentaje de solapamientos (Overlapping).

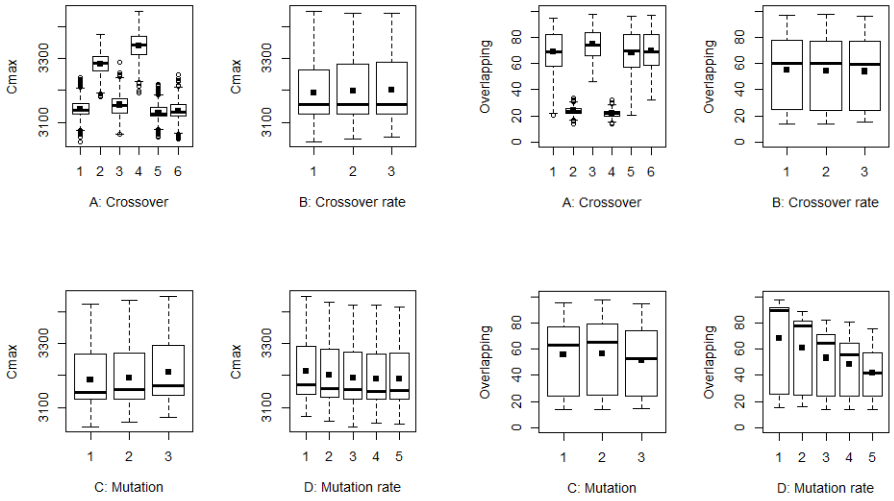

Fig. 2. AG con 200 individuos y 500 generaciones. Makespan (Cmax) y porcentaje de solapamientos (Overlapping).

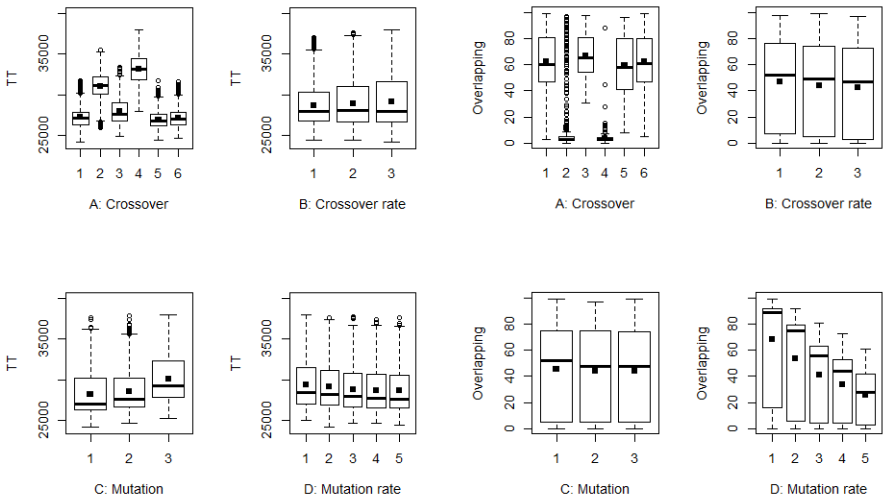

Fig. 3. AG con 100 individuos y 1000 generaciones. Total tardiness (TT) y porcentaje de solapamientos (Overlapping).

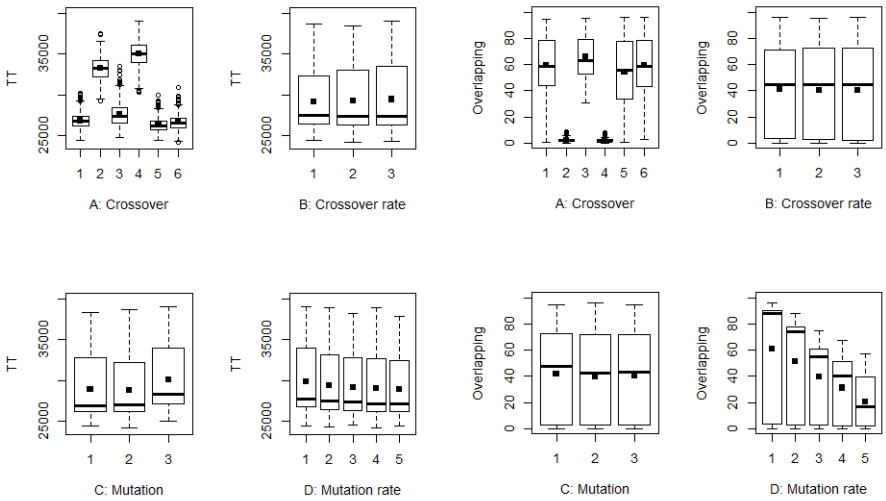

Fig. 4. AG con 200 individuos y 500 generaciones. Total tardiness (TT) y porcentaje de solapamientos (Overlapping).

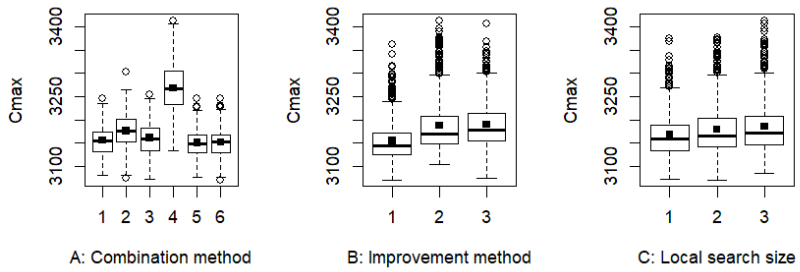

Fig. 5. SS. Makespan (Cmax).

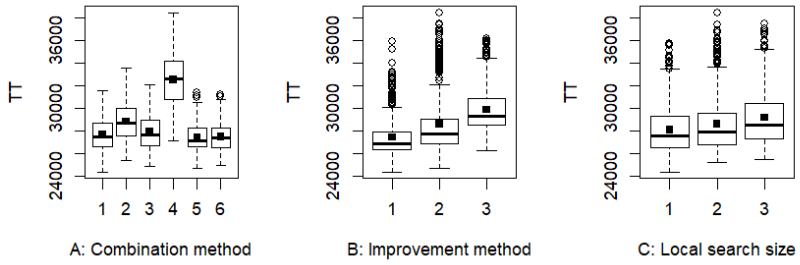

Fig. 6. SS. Total tardiness (TT).

- AG con poblaciones de 100 individuos:
  - *Makespan*: no se observan diferencias significativas entre los métodos de combinación NWOX, PMX y PMX2, que son los que presentan un mejor rendimiento. Lo mismo sucede con las dos probabilidades de cruce más bajas y las dos probabilidades de mutación más altas. El método de mejora que presenta un mejor rendimiento es el EXC.
  - *Total tardiness*: no se observan diferencias significativas entre los métodos de combinación NWOX y PMX2, que son los que presentan un mejor rendimiento. Lo mismo sucede con las dos probabilidades de cruce más bajas y las tres probabilidades de mutación más altas. El método de mejora que presenta un mejor rendimiento es el EXC.
- AG con poblaciones de 200 individuos:
  - *Makespan*: El método de combinación que presenta un mejor rendimiento es NWOX. No se observan diferencias significativas entre las tres probabilidades de cruce ni entre las tres probabilidades de mutación más altas, que son las que presentan un mejor rendimiento. De nuevo, el método de mejora que presenta un mejor rendimiento es el EXC.
  - *Total tardiness*: El método de combinación que presenta un mejor rendimiento es NWOX. No se observan diferencias significativas entre las tres probabilidades de cruce ni entre las dos probabilidades de mutación más altas, que son las que

presentan un mejor rendimiento. Ahora, no se encuentran diferencias significativas entre los métodos de mejora EXC e INS, que son los que presentan un mejor rendimiento.

- SS: con ambas funciones objetivo, no se observan diferencias significativas entre los métodos de combinación NWOX, PMX y PMX2, que son los que presentan un mejor rendimiento. El método de mejora que presenta un mejor rendimiento es el EXC. Lo mismo sucede con el menor tamaño de búsqueda local.

Respecto a la diversidad de las soluciones generadas, el SS ya tiene mecanismo para mantener un conjunto de referencia de soluciones diversas, pero es bien conocido que el AG tiende a converger a poblaciones con soluciones repetidas, que pueden desembocar en un superindividuo que domina la convergencia y, en algunos casos, puede producir una convergencia prematura a un óptimo local. En este sentido, se han revisado las poblaciones finales de todas las ejecuciones del AG en busca de superindividuos. Se observa que, en el caso de *total tardiness*, se producen superindividuos en más del 97% de las ejecuciones, aunque hay que matizar que mientras el porcentaje medio de soluciones repetidas es alto (superior al 50%) con los métodos de combinación PMX, PMX2, NWOX y CX, con los métodos OX y CX2 este porcentaje medio es bajo, del orden del 12% y el 5%, respectivamente, con poblaciones de 100 individuos, e inferior al 4% con poblaciones de 200 individuos. Con el *makespan*, si bien el porcentaje de soluciones repetidas es similar, ahora se observa diversidad en estas soluciones, lo que refleja la alta multimodalidad de esta función objetivo.

5 Conclusiones

En general, se puede concluir que los métodos de combinación que producen más diversidad en las soluciones (OX y CX2) producen un menor rendimiento en convergencia. Además, en el caso del AG, estos dos cruces obtienen mejores valores medios de las funciones objetivo con 100 individuos que con 200. Con los demás operadores de cruce ocurre exactamente lo contrario, seguramente porque, aunque los porcentajes de solapamientos de las soluciones son similares, el número de soluciones es el doble, lo que redunda en una mayor exploración del espacio de búsqueda.

Como trabajo futuro se propone repetir el experimento con instancias mayores, es decir, que involucren más trabajos y/o más máquinas, para comprobar si las conclusiones se consolidan. También sería interesante diseñar otros experimentos que involucren más métodos de combinación y mejora de los resultados e incluso extender la experimentación a otros problemas combinatorios.

Agradecimientos. Los autores agradecen el soporte económico otorgado por la Consejería de Economía, Industria, Comercio y Conocimiento del Gobierno de Canarias a través de la subvención directa concedida a la ULPGC denominada “Apoyo a la actividad de I+D+i. Campus de Excelencia Internacional CEI CANARIAS-ULPGC”. También agradecen a la Secretaría General de Ciencia y Tecnología de la Universidad Nacional del Sur por financiar parte de este trabajo a través de los proyectos PGI 24/ZJ49 y 24/J086.

Referencias

Cicirello, V.A. (2006). Non-wrapping order crossover: An order preserving crossover operator that respects absolute position. In: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation.

Davis, L. (1985). Applying adaptive algorithms to epistatic domains. In: Proceedings of the 9th International Joint Conferences on Artificial Intelligence, 162–164.

Glover, F. (1977). Heuristics for integer programming using surrogate constraints. Decision Sciences 8, 156–166.

Goldberg, D.E., Lingle, R. (1985). Alleles, loci and the traveling salesman problem. In: Proceedings of the 1st International Conference on Genetic Algorithms and Their Applications, 154–159.

Holland, J. (1992). Adaptation in natural and artificial systems. MIT Press Cambridge.

Hussain, A., Muhammad, Y., Nauman, M., Hussain, I., Mohamd, A., Gani, S. (2017). Genetic algorithm for traveling salesman problem with modified cycle crossover operator. Computational Intelligence and Neuroscience, 1–7.

Hussain, A., Muhammad, Y.S., Sajid, M.N. (2019). A simulated study of genetic algorithm with a new crossover operator using traveling salesman problem. Punjab University Journal of Mathematics 51(5), 61-77.

Jeen Robert, R.B., Rajkumar, R. (2017). An effective genetic algorithm for flow shop scheduling problems to minimize makespan. Mechanical Technologies 23(4), 594-603.

Khatami, M., Salehipour, A., Cheng, T.C.E. (2023). Flow-shop scheduling with exact delays to minimize makespan. Computers & Industrial Engineering 183(1), 109456.

Koulamas, C. (2020). The proportionate flow shop total tardiness problem. European Journal of Operational Research 284(2), 439-444.

Laguna, M., Glover, F. (1999). Scatter Search. Graduate School of Business. University of Colorado, Boulder.

Oliver, I.M., Smith, D.J., Holland, J.R.C. (1987). A study of permutation crossover operators on the traveling salesman problem. In: Proceedings of the Second International Conference on Genetic Algorithms and Their Applications, 224–230.

Pinedo, M. L. (2016). Scheduling: Theory, Algorithms, and Systems (5th ed.). Springer.

Rossit, D.A., Tohmé, F., Frutos, M. (2018). The non-permutation flow-shop scheduling problem: a literature review. Omega, 77, 143-153.

Rossit, D.G., González Landín, B., Frutos, M. and Méndez Babey, M. (2024). Scatter Search Algorithm for a Waste Collection Problem in an Argentine Case Study. Urban Science, 8(4), 240.

Sharma, M., Sharma, M., Sharma, S. (2021). An improved NEH heuristic to minimize makespan for flow shop scheduling problems. Decision Science Letters 10(3), 311-322.

Umam, M.S., Mustafid, M., Suryono, S. (2022). A hybrid genetic algorithm and tabu search for minimizing makespan in flow shop scheduling problem. Journal of King Saud University, Computer and Information Sciences 34(9), 7459-7467.