Sistema de soporte de decisiones de producción en un entorno flexible job shop basado en un modelo predictivo-reactivo sujeto a perturbaciones

Sebastian Mateo Meza Villalba

Escuela Colombiana de Ingeniería Julio Garavito
Decanatura de Ingeniería Industrial
Maestría en Ingeniería Industrial
Bogotá D.C., Colombia
2020

Sistema de soporte de decisiones de producción en un entorno flexible job shop basado en un modelo predictivo-reactivo sujeto a perturbaciones

Sebastian Mateo Meza Villalba
Ingeniero Industrial – Ingeniero Mecánico

Trabajo de investigación para optar al título de
Magister en Ingeniería Industrial

Director
Carlos Rodrigo Ruiz Cruz
Ingeniero Industrial, M. Sc.

Codirector
Jose Fernando Jimenez Gordillo
Ingeniero Industrial, PhD.

Escuela Colombiana de Ingeniería Julio Garavito
Decanatura de Ingeniería Industrial
Maestría en Ingeniería Industrial
Bogotá D.C., Colombia
2020

Publicado en 2020 por la Escuela Colombiana de Ingeniería Julio Garavito. AK.45 No.205-59 (Autopista Norte)
Bogotá, Colombia
TEL: +57 – 1 668 36 00

Agradecimientos

A la Escuela Colombiana de Ingeniería Julio Garavito por apoyar mis estudios, mi desarrollo académico y profesional constantemente; por permitirme seguir siendo parte de la institución como estudiante completamente becado en la Maestría y como asistente graduado de investigación.

A mis tutores M. Sc. Carlos Ruiz y PhD. Jose Fernando Jimenez por confiar en todo momento en mis capacidades; por su guía para la realización de este trabajo desde que fue una idea hasta que llego a ser un producto; por su disponibilidad, apoyo y paciencia durante todo el proceso.

A la ingeniera M. Sc. Sonia Jaimes por su infinita paciencia; por su apoyo, no solo a nivel académico, durante toda la Maestría; por su siempre insistencia en que confíe una vez más en mí mismo.

A mi familia por su apoyo incondicional en mi vida académica; por enseñarme el valor de trabajo duro, el valor del amor y el valor de la alegría; por ser el impulso para hacer las cosas cada vez mejor.

A mis amigos que hicieron parte del proceso por su interés en el desarrollo general del trabajo, avances y dificultades; por sus ánimos para seguir adelante.

Resumen

La programación de producción en presencia de eventos en tiempo real es un tema de importancia para el rendimiento de sistemas de programación de operaciones. La mayoría de los sistemas de manufactura operan en entornos dinámicos vulnerables a varios eventos no planeados que continuamente obligan a la revisión y reconsideración de programaciones preestablecidas. En un entorno variable, las formas eficientes de adaptar las soluciones actuales a eventos inesperados son preferibles a las soluciones que pronto se vuelven obsoletas. Esta realidad motivó el desarrollo de un sistema de soporte de decisiones que intenta reducir la brecha entre la teoría y la práctica de la programación de operaciones. El prototipo desarrollado en esta investigación utiliza metaheurística para generar una programación predictiva. Luego, cuando ocurren perturbaciones, tales como la llegada de nuevas tareas o la cancelación de otras, el sistema de soporte de decisiones comienza a actualizar la programación previa través de un módulo reactivo que utiliza heurística basada en principios de reglas de despacho. El sistema propuesto se probó en un escenario simulado de un sistema de manufactura flexible real ubicado en Valenciennes (Francia), llamado AIP-PRIMECA Valenciennes. Igualmente, el sistema se validó una perturbación durante la ejecución en el sistema de manufactura para demostrar la efectividad del modelo propuesto.

Abstract

Production scheduling under real-time events has high importance for the successful performance of real-world scheduling systems. Most manufacturing systems operate in dynamic environments vulnerable to various non-programmed real-time events which continuously forces revision and reconsideration of pre-established schedules. In an uncertain environment, efficient ways to adapt current solutions to unexpected events, are preferable to solutions that soon become obsolete. This situation motivated the development of a decision support system that attempts to fill the gap between scheduling theory and practice. The developed prototype uses metaheuristics to generate a predictive schedule for an initial solution before execution. Then, whenever disruptions happen, like arrival of new tasks or cancelation of others, the decision support system starts updating the schedule through a reactive module that uses heuristics based on dispatching rules principles. The proposed system was tested in a simulated scenario of a real flexible manufacturing system located in Valenciennes (France), called AIP-PRIMECA Valenciennes. A disruption was carried out during the execution in the manufacturing system in order to demonstrate the effectiveness of the proposed model.

Tabla de Contenidos

Introducción **1**

Problemática y justificación 1

Pregunta de investigación 5

Objetivos 5

Objetivo general 5

Objetivos específicos 6

Alcances y limitaciones 6

Alcances 6

Limitaciones 6

Capítulo 1

Sistemas de soporte de decisión: programación de operaciones en manufactura **7**

Introducción 7

La programación de operaciones a nivel empresarial 7

La presencia de perturbaciones en operaciones de manufactura 8

Proceso de toma de decisión ante perturbaciones 10

Teoría de sistemas de soporte de decisión 11

Definición de sistemas de soporte de decisión 11

Características y capacidades de sistema de soporte de decisión 11

Componentes de sistemas de soporte de decisión 13

Capítulo 2

Programación de operaciones dinámica: Revisión de la literatura **15**

Introducción 15

Contexto de investigación de la programación de operaciones 15

Problema de programación de operaciones *job shop* y *flexible job shop* 18

Clasificación de los modelos de programación tipo *job shop* 22

Problema de programación de operaciones dinámico 24

Perturbaciones 24

Estrategias para la toma de decisión 25

Reprogramación de operaciones en presencia de perturbaciones 26

Métodos de solución para el problema de programación de operaciones 29
Métodos exactos de optimización 30
Métodos de aproximación 31
Revisión de la literatura 33
Capítulo 3
Sistema de soporte de decisión de manufactura: programación de operaciones dinámica ..43
Introducción 43
Especificaciones del sistema de soporte de decisión 43
Descripción del sistema de soporte de decisión 46
Subsistema de gestión de datos 47
Módulo de información maestra 48
Módulo de usuario 48
Módulo de generación de orden 49
Módulo de parámetros del sistema 49
Módulo de ejecución inicial 49
Módulo de monitoreo 50
Módulo de ejecución final 53
Subsistema de gestión de modelos predictivos 54
Módulo de definición del problema 54
Módulo de algoritmo genético 58
Módulo de programación inicial 65
Subsistema de gestión de modelos reactivos 66
Módulo de política de reprogramación 67
Módulo de método de reprogramación 69
Subsistema de interfaz de usuario 76
Módulo de orden 77
Módulo de interfaz *offline* 78
Módulo de interfaz *online* 81
Capítulo 4
Sistema de soporte de decisiones – AIP Un caso de estudio experimental 89
Introducción 89

Descripción del sistema de manufactura 89
Diseño de la célula AIP 90
Especificaciones de producción de la célula AIP 91
Protocolo experimental 95
Experimentos 95
Instancias 96
Perturbación 97
Implementación del DSS al sistema de manufactura AIP 98
Implementación técnica del caso de estudio 98
Generación de la programación 98
Relación con el piso de manufactura 100
Actualización de la programación 103
Resultados 112
Conclusiones y trabajo futuro 122
Bibliografía 126
Apéndice A 134
Apéndice B 138

Lista de Tablas

Tabla 1. Revisión de la literatura de las técnicas metaheurísticas más utilizadas para resolver el FJSSP. 32
Tabla 2. Técnicas utilizadas en problemas de programación. 35
Tabla 3. Ventajas y desventajas de las técnicas de simulación en la toma de decisión (adaptado de Leusin et al., 2018). 36
Tabla 4. Medidas de desempeño utilizadas en problemas de programación. 37
Tabla 5. Revisión de la literatura de programación dinámica. 38
Tabla 6. Ejemplo de FJSSP – tiempo de procesamiento. 58
Tabla 7. Ejemplo de FJSSP – tiempo de transporte entre máquinas 58
Tabla 8. Representación de un cromosoma factible. 59
Tabla 9. Ciclo de asignación de operaciones. 61
Tabla 10. Medidas de desempeño sustitutas para la reprogramación de operaciones. 68
Tabla 11. Decisiones asociadas al problema flexible job shop. 70
Tabla 12. Información necesaria para la técnica FAM. 70
Tabla 13. Información necesaria para la técnica SCM. 74
Tabla 14. Correspondencia de atributos en el DSS. 84
Tabla 15. Tiempo en segundos (s) de procesamiento de las operaciones de manufactura (adaptado de Trentesaux et al., 2013) 94
Tabla 16. Secuencia de producción de cada trabajo (adaptado de Trentesaux et al., 2013) 94
Tabla 17. Tiempo de transporte entra máquinas. 95
Tabla 18. Instancias de experimentos. 97
Tabla 19. Resultados mrj_151 – Experimento B. 102
Tabla 20. Resultados mrj_151 – Experimento C. 104
Tabla 21. Resultados mrj_151 – Experimento D. 105
Tabla 22. Resultados mrj_151 – Experimento E. 107
Tabla 23. Actualización de la programación: mrj_151 – Experimento E. 108
Tabla 24. Resultados generales del Experimento A. 113
Tabla 25. Resultados generales del Experimento B. 114
Tabla 26. Resultados generales del Experimento C. 115
Tabla 27. Resultados generales del Experimento D. 116

Tabla 28. Resultados generales del Experimento E. 117
Tabla 29. Ecuaciones de los experimentos realizados - makespan 120
Tabla 30. Arreglo factorial 134
Tabla 31. Análisis de varianza. 135
Tabla 32. Comparación con otros algoritmos – instancias de prueba. 137

Lista de Figuras

Figura 1. Diagrama de flujo de información en un sistema de manufactura (adaptado de Pinedo, 2016).....8
Figura 2. Ciclo para la toma de decisiones ante perturbaciones y sistemas de soporte para cada etapa (adaptado de Arica et al., 2014).....10
Figura 3. Características y capacidades clave de un DSS (adaptado de Turban et al., 2004).....12
Figura 4. Vista esquemática de un DSS (adaptado de Turban et al., 2004).....13
Figura 5. Funciones de tiempo de finalización (Pinedo, 2016).....20
Figura 6. Ejemplo de un sistema de manufactura *flexible job shop*.....21
Figura 7. Parámetros del ejemplo de sistema de manufactura *flexible job shop*.....22
Figura 8. Tipos de entornos tipo *job shop* (J. Zhang et al., 2017).....23
Figura 9. Línea de tiempo de la programación predictiva-reactiva.....28
Figura 10. Métodos de optimización para programación de operaciones (J. Zhang et al., 2017).....30
Figura 11. Reparación de la programación en la fase de ejecución.....44
Figura 12. Características y capacidades del DSS.....45
Figura 13. Esquema general del sistema de soporte de decisiones propuesto.....47
Figura 14. Esquema general del subsistema de gestión de datos.....48
Figura 15. Esquema detallado del subsistema de gestión de datos.....53
Figura 16. Esquema general del subsistema de gestión de modelos predictivos.....54
Figura 17. Diagrama de Gantt de un cromosoma factible del algoritmo genético.....61
Figura 18. Operación de entrecruzamiento del algoritmo genético.....63
Figura 19. Operación de mutación del algoritmo genético.....63
Figura 20. Operación de remplazo del algoritmo genético.....64
Figura 21. Diagrama de flujo del algoritmo genético.....65
Figura 22. Esquema detallado del subsistema de gestión de modelos predictivos.....66
Figura 23. Esquema general del subsistema de gestión de modelos reactivos.....67
Figura 24. Diagrama de flujo de la técnica FAM.....73
Figura 25. Diagrama de flujo de la técnica SCM.....75
Figura 26. Esquema detallado del subsistema de gestión de modelos reactivos.....76
Figura 27. Esquema general del subsistema de interfaz de usuario.....76
Figura 28. Interfaz de usuario principal.....77

Figura 29. Centro de comandos de NetLogo. 78
Figura 30. Esquema general de salida de texto – programación. 79
Figura 31. Esquema general de salida de texto – información de trabajo. 80
Figura 32. Esquema general de salida grafica – diagrama de Gantt. 80
Figura 33. Esquema general de salida gráfica – desempeño de GA. 81
Figura 34. Interfaz de usuario secundaria. 82
Figura 35. Interfaz de usuario secundaria - ejecución. 82
Figura 36. Interfaz de usuario secundaría – información de monitoreo. 83
Figura 37. Esquema general de salida gráfica – robustez de la reprogramación. 86
Figura 38. Esquema detallado del subsistema de interfaz de usuario. 86
Figura 39. Esquema detallado del sistema de soporte de decisiones propuesto. 88
Figura 40. AIP-PRIMECA: Sistema de manufactura flexible ubicado en la Universidad de Valenciennes (Jiménez, 2017). 90
Figura 41. Diseño del sistema de manufactura flexible AIP-PRIMECA (Jiménez, 2017). 92
Figura 42. Tipos de trabajos procesados en la célula AIP (Jiménsez, 2017). 92
Figura 43. Componentes de los trabajos en la célula AIP (Jiménez, 2017). 93
Figura 44. Información de trabajos – mrj_151. 99
Figura 45. Diagrama de Gantt mrj_151 - GA. 100
Figura 46. Diagrama de Gantt mrj_151 – Ejecución en *shop floor*. 101
Figura 47. Diagrama de Gantt mrj_151 – Ejecución con perturbación en *shop floor*. 103
Figura 48. Diagrama de Gantt mrj_151 – Ejecución con perturbación y técnica FAM en *shop floor*. 105
Figura 49. Diagrama de Gantt mrj_151 – Ejecución con perturbación y técnica SCM en *shop floor*. 106
Figura 50. Diagrama de Gantt mrj_151 – Ejecución con perturbación y técnica FAM+SCM en *shop floor*. 108
Figura 51. Progreso de las medidas de desempeño sustitutas – mrj_151. 109
Figura 52. Progreso de la tardanza media – Experimento A y B. 110
Figura 53. Progreso de la tardanza media – Experimento B, C, D y E. 111
Figura 54. Resultados de experimentos – *makespan*. 111
Figura 55. Resultados de experimentos – retraso medio y tardanza media. 112

Figura 56. Resultado de los experimentos - degradación promedio de *makespan* (A%).....118
Figura 57. Resultado de los experimentos - degradación promedio de retraso y tardanza media (A%).....118
Figura 58. Resultado de los experimentos – cambio de la degradación promedio de *makespan* (B%).....118
Figura 59. Resultado de los experimentos – cambio de la degradación promedio de retraso y tardanza media (B%).....119
Figura 60. Resultado de los experimentos – *makespan* de cada instancia: GA, A y B.....119
Figura 61. Resultado de los experimentos – *makespan* de cada instancia: C, D y E.....120
Figura 62. Efectos principales del experimento 135
Figura 63. Efectos de interacción del experimento.....136

Introducción

Problemática y justificación

Las compañías manufactureras enfrentan un desafío fundamental al mantenerse competitivas bajo situaciones dinámicas. Por ello, deben responder rápidamente a los cambios de las necesidades del entorno realizando adaptaciones eficientes en sus procesos internos y alineándolos con los requerimientos a satisfacer. Para enfrentar este desafío se requiere innovación en el diseño y operación de sistemas de manufactura. Un aspecto importante es la implementación de tecnologías de soporte de decisión, en particular de sistemas de soporte de decisiones (DSS: *Decision Support Systems)*. En general, los DSS son sistemas informáticos interactivos que utilizan datos, modelos, documentos, tecnologías de conocimiento y comunicación para apoyar la toma de decisiones y la resolución de problemas complejos (Kasie, Bright, & Walker, 2017).

De hecho, la competencia global lleva a las compañías a encontrar maneras de reducir costos, mejorar el servicio al cliente e incrementar la productividad. Se ha identificado que, en las actividades realizadas a lo largo de cadena de abastecimiento, tanto a nivel interno de una compañía como en la cadena extendida, incluyendo proveedores y clientes, es posible realizar ahorros considerables (Sodhi, 2001). En respuesta a la necesidad de coordinación y comunicación en la cadena de abastecimiento se han desarrollado sistemas de soporte de información integrando tecnologías de información y de soporte de decisión llevando al concepto de planeación de recursos empresariales (ERP: *Enterprise Resource Planning)*. El objetivo principal del ERP es integrar todos los departamentos y funciones de una compañía en un único sistema informático que sirva para satisfacer todas las necesidades de la empresa. En la práctica los sistemas ERP integran procesos o transacciones repetitivas, como pedidos o facturaciones, las cuales pueden ser facilitadas por DSS (Turban, Aronson, & Liang, 2004).

Los sistemas ERP tienen tres componentes principales: base de datos empresarial, módulos de aplicación y un sistema cliente/servidor. Dentro de los módulos de aplicación se encuentra el módulo de manufactura cuyo DSS general se conoce como planeación y control de la producción (MPC: *Manufacturing Planning and Control)*. Los sistemas MPC son un soporte para tomar decisiones adecuadas y proveen la información necesaria para tal fin. Estos sistemas no toman decisiones ni administran las operaciones. Las actividades de soporte de los sistemas MPC se clasifican en tres horizontes de tiempo: largo, mediano y corto plazo. En el largo plazo, se toman decisiones relacionadas con la cantidad necesaria de capacidad, instalaciones, equipos, proveedores, entre otras, para satisfacer la demanda futura del mercado. A mediano plazo, el asunto principalmente abordado es cumplir la demanda en términos de volumen de producción. La distinción con el largo plazo es el enfoque en indicar la cantidad exacta de materiales y capacidad de producción. En el corto plazo se realiza una programación detallada de los recursos para cumplir

con los requisitos de producción. Esto implica tiempo, personas, materiales, equipos e instalaciones (Vollmann, Berry, Whybark, & Jacobs, 2004). En este trabajo, el foco de investigación es la programación de operaciones (*scheduling)* en el piso de manufactura (*shop-floor*), la cual es una actividad de manufactura clasificada en el horizonte de corto plazo.

Además, los sistemas de manufactura tradicionales se están transformando en ambientes de manufactura basados en digitalización, redes informáticas, inteligencia artificial y rápida respuesta a través de toda la cadena de abastecimiento (Felsberger, Oberegger, & Reiner, 2017). Estas son características de sistemas flexibles de manufactura (FMS: *Flexible Manufacturing Systems)*. Browne, Dubois, Rathmill, Sethi, & Stecke (1984) definieron un FMS como un sistema informático integrado de dispositivos automatizados de manejo de material y máquinas de control numérico que pueden procesar simultáneamente volúmenes medios de una variedad de tipos de productos. MacCarthy & Liu (1993a) describieron y clasificaron cuatro tipos de FMS de acuerdo con sus características de operación y control. Tipo I – máquina única flexible, tipo II - célula flexible de manufactura, el cual consiste en un conjunto de máquinas que comparten equipos de manejo de material, tipo III – FMS multi-máquina, el cual se diferencia del tipo II al contar con dos o más equipos de manejo de material y tipo IV – FMS multi-célula, el cual consiste en un conjunto de FMS tipo II conectados por un sistema de manejo de material. Asimismo, la teoría clásica de programación de operaciones distingue diferentes entornos de manufactura que se pueden clasificar en la descripción anterior. Máquina única (FMS tipo I), maquinas en paralelo (FMS tipo II), *flow shop* y *job shop* (FMS tipo III) y *flexible flow shop, flexible job shop* y *open shop* (FMS tipo IV) (Pinedo, 2016). En este trabajo, el interés de investigación es el sistema de manufactura *flexible job shop* (FJS). Este entorno se caracteriza por tener un flujo de *n* trabajos a través de *c* células de manufactura cada una con *m* máquinas, en el cual cada trabajo sigue una ruta determinada e independiente de otros trabajos en las máquinas disponibles.

En general, la programación de operaciones es cada vez más importante a nivel operativo, ya que una programación adecuada es condición necesaria para responder a la variabilidad de la demanda, a tiempos de espera más cortos (*lead time*) y a entregas más rápidas y para aprovechar de mejor manera la flexibilidad de sistemas de manufactura apoyados en nuevas tecnologías (Macchi, Polenghi, Sottoriva, Fumagalli, & Negri, 2018). Sin embargo, la mayoría de sistemas de manufactura se encuentran en entornos vulnerables a la ocurrencia de eventos aleatorios en tiempo real, tales como fallo de máquinas, material defectuoso, operarios no disponibles, que convierten con rapidez programaciones previamente realizadas en obsoletas cuando se ejecutan en el piso de manufactura (Elgendy, Hussein, & Elhakeem, 2017). Dichos eventos aleatorios se conocen como perturbaciones. Por ende, las industrias se benefician al entender mejor los efectos de las estrategias de programación en el rendimiento del sistema de manufactura, pues bajo las características antes mencionadas, se tiende a buscar y encontrar formas eficientes y efectivas de adaptar la programación inicial a eventos inesperados, en vez de encontrar una programación de alta calidad que rápidamente quedará desactualizada (Vieira, Herrmann, & Lin, 2003). Bajo estas

condiciones, la mejor programación es la que presenta una baja desviación con respecto a la inicial después de las perturbaciones (Rahmani & Ramezanian, 2016).

De manera teórica, la programación de operaciones es un proceso de toma de decisiones que se ocupan de la asignación de recursos para realización de tareas durante períodos de tiempo determinados y su objetivo es optimizar uno o más objetivos. Las decisiones de programación se ven afectadas por características y limitaciones referentes al sistema de manufactura y a los productos, como los tiempos de procesamiento de operaciones, tiempos de entrega, restricciones de precedencia y disponibilidad de recursos (Madureira, Ramos, & Silva, 2003). Desde el campo de la investigación se ha dedicado bastante esfuerzo en encontrar soluciones óptimas al problema de programación, no obstante, se ha demostrado que este problema es *NP-hard* (Garey & Johnson, 1975), lo que significa que no es posible encontrar una solución óptima para problemas de gran tamaño en un tiempo computacional razonable. Por esta razón, en los últimos años se han propuesto diferentes enfoques heurísticos y metaheurísticos para resolver problemas de programación de operaciones y en particular el *flexible job shop scheduling problem* (FJSSP) (Wang, Luo, & Cai, 2017).

Los problemas de programación pueden ser clasificados en estáticos y dinámicos. Un problema es considerado estático si todos los parámetros son fijos y no cambian durante su ejecución. Por el contrario, un problema es dinámico si uno o más parámetros son variables en el tiempo (French, 1982). Dentro de la programación dinámica se han definido tres categorías: (a) programación totalmente reactiva, (b) programación predictiva-reactiva y (c) programación robusta proactiva. En una programación completamente reactiva, no existe una programación inicial al empezar la ejecución y las decisiones se toman localmente en tiempo real. La programación predictiva-reactiva es un proceso de programación/reprogramación en el que la programación inicial es ejecutada desde el principio y luego es revisada en respuesta a eventos en tiempo real. Los enfoques de programación robusta proactiva se centran en la construcción de programaciones predictivas únicas que satisfacen los requisitos de rendimiento en un entorno dinámico (Ouelhadj & Petrovic, 2009). Al considerar las perturbaciones, a las que los sistemas de manufactura son vulnerables, el foco de este trabajo será la programación dinámica, y por el enfoque usado para encontrar una solución, en programación predictiva-reactiva.

En el marco de la programación predictiva-reactiva, se destacan dos elementos. El primero es la generación de la programación (*schedule generation*), la cual es un mecanismo o modelo de toma de decisión predictivo y sirve como un plan general de ejecución de las tareas a programar. El segundo elemento es la actualización de la programación (s*chedule updating*) que usualmente se considera como la parte reactiva del sistema y busca minimizar el efecto de las perturbaciones en el rendimiento del sistema (Varela & Ribeiro, 2014). La actualización de la programación de operaciones existente en respuesta a perturbaciones u otros eventos no planeados se conoce como reprogramación de operaciones (*rescheduling*) (Vieira et al., 2003).

En este trabajo, se adapta el enfoque dinámico antes mencionado por tres razones principales. La primera es minimizar la brecha existente entre la programación teórica y la programación práctica respecto a su ejecución en el piso de manufactura, pues los modelos y algoritmos de programación no están en capacidad de usar información en tiempo real (Cowling & Johansson, 2002) y actualmente los sistemas de producción con tecnologías de Industria 4.0 han convertido a las máquinas en objetos inteligentes capaces de comunicarse entre ellos y con bases de datos de producción permitiendo el acceso a este tipo de información (Turker et al., 2019). La segunda es reducir la simplificación de supuestos en este tipo de problemas, pues la programación clásica falla para responder a las necesidades en entornos prácticos y en la mayoría de formulaciones de problemas de programación de operaciones se asumen datos determinísticos y estáticos en el tiempo (Larsen & Pranzo, 2019). La tercera razón se basa en las dos anteriores y busca responder a como los métodos tradicionales de optimización en programación de operaciones pueden ser combinados con métodos y técnicas más actuales para tener utilidad y aplicación real en FMS bajo Industria 4.0 expuestos a incertidumbre y ambientes dinámicos.

Teniendo en cuenta todo lo anteriormente expuesto, este trabajo de investigación propone un sistema de soporte de decisiones de producción con un enfoque predictivo-reactivo, cuya finalidad principal es reducir la degradación experimentada de un plan de producción causada por perturbaciones o eventos inesperados. En general, los sistemas de soporte de decisión se pueden categorizar como orientados a los modelos, orientados a la información, orientados a la comunicación y orientados al conocimiento. Las decisiones soportadas por el sistema propuesto se apoyan en algoritmos y modelos de optimización, por ende este se caracteriza como un DSS orientado a los modelos (Felsberger et al., 2017). Para la generación de la programación inicial, se utiliza un módulo predictivo basado en un algoritmo genético. Para la actualización de la programación, necesaria en el caso de que en el sistema de manufactura se presenten perturbaciones, se utiliza un módulo reactivo basado en técnicas heurísticas, cuya importancia radica en su capacidad de resolver los desafíos del enfoque predictivo-reactivo. Dentro del enfoque trabajo se distinguen tres desafíos principales (Vieira et al., 2003). El primero es cuándo: debe ser conocido el momento para ejecutar una reprogramación. El segundo es qué: la reprogramación debe reasignar nuevas decisiones al sistema para cumplir el objetivo establecido. Finalmente, el tercer desafío es, cómo: se deben establecer las técnicas que ejecuten la reprogramación.

Para validar el sistema de soporte de decisiones propuesto, se utiliza un escenario simulado de un sistema de manufactura flexible real, el cual puede ser visto como un entono *flexible job shop* (Trentesaux et al., 2013), ubicado en Valenciennes (Francia), llamado AIP-PRIMECA Valenciennes. La simulación de la célula de manufactura es realizada con la técnica de simulación basada en agentes, cuya flexibilidad y adaptabilidad a los cambios de configuración en el sistema, permiten modelar perturbaciones generadas en el sistema y reprogramar las decisiones de producción generadas por el módulo reactivo. La medida de desempeño principal es el *makespan*

y como medidas sustitutas de reprogramación se establecen el retraso (*lateness*) y la tardanza (*tardiness*).

Este trabajo de investigación se organiza de la siguiente manera. En el Capítulo 1 se presenta una contextualización de la programación de operaciones a nivel empresarial y se presenta la teoría de los DSS. En el Capítulo 2 se presenta el contexto de la programación de operaciones a nivel investigativo. Se presenta una revisión de los métodos de solución al problema de programación y se presenta una revisión de la literatura de problemas de programación de operaciones que se relación con ambientes de manufactura *job shop* sujetos a perturbaciones. El Capítulo 3 muestra en detalle el sistema de soporte de decisiones propuesto y el Capítulo 4 muestra su implementación en la célula de manufactura AIP-PRIMECA. Finalmente, se presentan las conclusiones e investigaciones futuras propuestas.

Pregunta de investigación

Teniendo en cuenta lo mencionado en la sección anterior y el hecho de que la programación de operaciones en manufactura representa aún una oportunidad para tener una cadena de abastecimiento más eficiente y reactiva, la pregunta principal de investigación de este trabajo de grado es: ¿Cómo puede ser reducida la degradación causada por una perturbación con respecto a la programación inicial en ambientes flexibles de manufactura?

Esta pregunta plante tres sub-preguntas específicas:

- ✓ ¿Qué simplificaciones de los modelos clásicos de programación deben ser consideradas para responder a las necesidades de entornos reales?
- ✓ ¿Cómo integrar la ocurrencia de eventos en tiempo real durante la ejecución de una programación inicial bajo un ambiente *flexible job shop*?
- ✓ ¿Cómo pueden ser integrados los métodos tradicionales de programación y los modelos de simulación en una programación predictiva-reactiva?

Objetivos

Objetivo general

Desarrollar un sistema de soporte de decisiones de programación de operaciones predictiva-reactiva bajo una configuración *flexible job shop* sujeto a perturbaciones que reduzca la degradación de las medidas de desempeño asociadas a la programación.

Objetivos específicos

- ✓ Realizar un análisis de las variables que afectan la programación de producción, así como de los eventos en tiempo real y sus efectos en sistemas de manufactura flexibles.
- ✓ Construir un modelo de programación predictivo-reactivo basado en técnicas heurísticas y metaheurísticas para un ambiente de manufactura *flexible job shop*.
- ✓ Modelar escenarios de programación *flexible job shop* para reflejar entornos reales de manufactura por medio de simulación basada en agentes.
- ✓ Evaluar el desempeño del modelo propuesto en relación con las métricas asociadas a los tiempos de terminación de operaciones y retardo y tardanza de operaciones.

Alcances y limitaciones

Alcances

- ✓ Desarrollo de un marco general para un sistema de soporte de decisiones de producción.
- ✓ Programación de un algoritmo genético para resolver el problema clásico de programación de operaciones *flexible job shop scheduling problem* incorporando tiempos de transporte entre máquinas y número máximo de trabajos en el sistema.
- ✓ Programación de dos técnicas heurísticas para resolver el problema de actualización de la programación de operaciones tras la ocurrencia de perturbaciones en el sistema.
- ✓ Programación de la simulación de un piso de manufactura real que integre las programaciones anteriores.

Limitaciones

- ✓ Diseño y desarrollo de software. En este trabajo de investigación se presenta la parte teórica y operativa del sistema de soporte de decisiones. No se presenta su integración a un sistema de manufactura real.
- ✓ Meta-optimización de los parámetros del algoritmo genético.
- ✓ Optimización de parámetros de las técnicas heurísticas.

Capítulo 1
Sistemas de soporte de decisión: programación de operaciones en manufactura

Introducción

El objetivo de este capítulo es presentar el contexto de la programación de operaciones en manufactura a nivel empresarial y el marco teórico de los sistemas de soporte de decisión. La primera sección del capítulo indica la ubicación de la programación dentro del flujo de información de un sistema de manufactura clásico el cual hace parte del sistema global de información de la empresa: ERP (*Enterprise Resource Planning)*. La segunda sección presenta sistemas tecnológicos de soporte y de complemento para el ERP al presentarse perturbaciones durante la ejecución de operaciones en manufactura. La tercera sección introduce el marco del proceso de toma de decisiones para responder a eventos no planeados y muestra la necesidad de un sistema de soporte de decisiones particular para la programación de operaciones. La cuarta sección presenta un marco teórico general de sistemas de soporte de decisión y se divide en definición, características y capacidades y componentes.

La programación de operaciones a nivel empresarial

La programación de operaciones (*scheduling*) es un proceso de toma de decisiones que se ocupan de la asignación de recursos para realización de tareas durante períodos de tiempo determinados y su objetivo es optimizar uno o más objetivos. Este proceso debe interactuar con otras funciones al ser parte de un sistema de producción y tales interacciones toman lugar frecuentemente en el sistema de información global empresarial (Pinedo, 2016).

Actualmente, las empresas cuentan con sistemas de información robustos, los cuales incluyen normalmente una base de datos central, estaciones informáticas de trabajo, computadores personales y terminales de entrada de información conectados entre ellos, que permite tanto la entrada como la solicitud de nueva información. El software que controla el mencionado sistema de información es denominado como planeación de recursos empresarial (ERP: *Enterprise Resource Planning*) (Lee & Lau, 2007). La programación de operaciones se realiza de forma interactiva a través de un sistema de soporte de decisiones (DSS: *Decision Support System*) que está vinculado al sistema ERP (Pinedo, 2016).

Se considera el siguiente marco para entender la ubicación de la programación de operaciones en manufactura. Se tienen las ordenes que se liberan desde la programación maestra de producción (MPS: *Master Production Schedule*) y que se traducen como trabajos (*jobs*) con sus

respectivos tiempos de entrega (*due date*) tras la planeación de requerimientos de material (MRP: *Material Requirement Planning*). Estos trabajos deben ser procesados en máquinas de una célula de manufactura dada una secuencia establecida en el piso de manufactura (*shop-floor*). Se debe tener en cuenta que, el piso de manufactura no es la única parte de la organización que impacta la programación de operaciones, pues esta también se ve afectada por decisiones de largo y mediano plazo. Es decir, decisiones tomadas en un nivel superior de planeación impactan directamente el proceso de programar (Vollmann et al., 2004). La Figura 1 muestra el flujo de información en un sistema de manufactura clásico. Este flujo muchas veces es asociado al marco de la planeación y control de la producción (MPC: *Manufacturing Planning and Control*) el cual es el DSS general del módulo de manufactura en un sistema ERP (Lee & Lau, 2007).

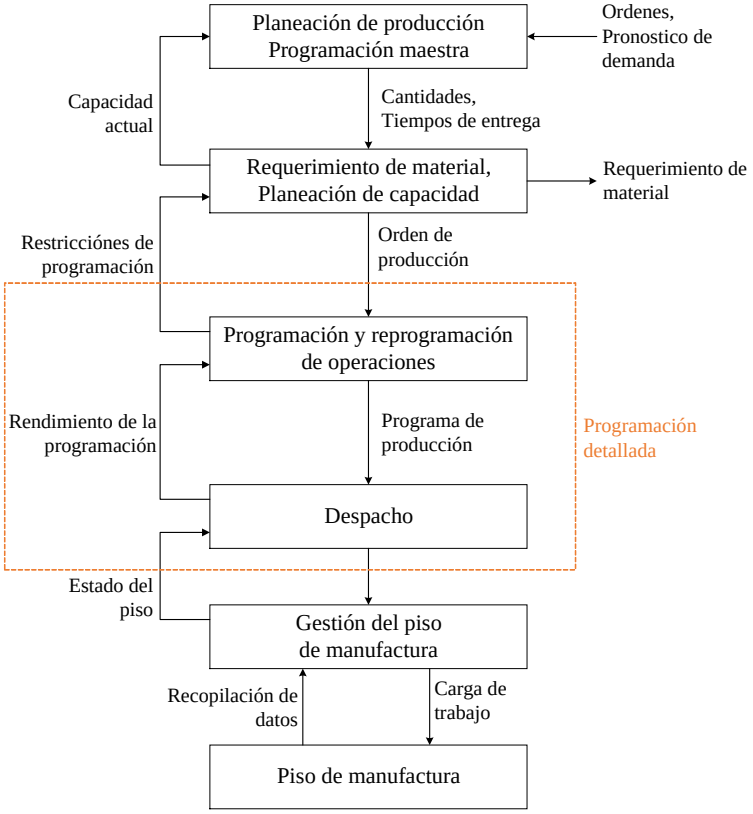

Figura 1. Diagrama de flujo de información en un sistema de manufactura (adaptado de Pinedo, 2016).

La presencia de perturbaciones en operaciones de manufactura

Los beneficios de los sistemas ERP radican en su capacidad de procesar y registrar transacciones eficientemente, en vez de sus capacidades inherentes de planeación y control (Juan, Barrios, Vallada, Riera, & Jorba, 2014). Sin embargo, cuando un sistema ERP es usado para planear y controlar se implementa la lógica MRP (Koh & Saad, 2003) y se determina la cantidad de material neto requerido y los trabajos a ser procesados en el piso de manufactura con sus

respectivos tiempos de entrega. A pesar de ello, en el piso de manufactura la planeación inicial está expuesta a incertidumbre y a la ocurrencia de eventos aleatorios no planeados. Esto crea problemas en la ejecución de las operaciones, como el retraso de trabajos, la variación de las cargas en máquinas y el cambio de cuellos de botella calculados (K. Chen & Ji, 2007).

Por ello, en las últimas décadas, surgió el sistema APS (*Advanced Planning and Scheduling*) para complementar el sistema ERP y eliminar algunos de los principales supuestos del sistema MRP (Steger-Jensen, Hvolby, Nielsen, & Nielsen, 2011). En un sistema APS se consideran diferentes restricciones para generar una solución óptima, incluyendo disponibilidad de material y máquinas, capacidad de trabajo, tiempos de entrega, niveles de inventario, costo y requisitos de distribución (Lin, Hao, Gen, & Jo, 2012). Una característica importante del sistema APS es la capacidad de simular diferentes escenarios antes de su ejecución (Hvolby & Steger-Jensen, 2010), lo que lleva a cumplir con los plazos de pedidos de forma más eficiente. Haciendo lo anterior, se controlan algunos de los eventos aleatorios que se pueden presentar en el piso de manufactura. Los sistemas APS se han estandarizado y se han presentado como como una tecnología de programación avanzada. El cómo esta tecnología puede manejar las perturbaciones sigue siendo un tema abierto de investigación abierto (Ivert, 2012).

Por otro lado, los sistemas de ejecución de fabricación (MES: *Manufacturing Execution System*) se han desarrollado para apoyar las actividades de ejecución, monitoreo y control de la producción reduciendo los inconvenientes de los sistemas ERP en la explotación de información detallada en tiempo real desde el piso de manufactura. El sistema MES puede describirse como un sistema de información y comunicación en el taller (*shop-floor*) que proporciona retroalimentación en tiempo real (Manetti, 2001). Esto lo convierte en una herramienta eficiente para detectar e identificar perturbaciones. Sin embargo, los sistemas MES carecen de reactividad en la toma de decisiones, particularmente en la evaluación y diagnóstico de acciones correctivas en la programación de operaciones tras la ocurrencia de una perturbación (De Ugarte, Artiba, & Pellerin, 2009).

Además, pueden surgir dificultades de integración de la información cuando se intercambian datos entre sistemas como el ERP y el MES. Por ello, se requieren modelos estándar y esquemas de información de fabricación. Entre tales modelos, el ISA-95 es reconocido en la industria debido a su enfoque en la integración entre estos dos tipos de soluciones (ERP y MES) y su alto nivel de aplicación en el entorno industrial. El ISA-95, proporciona terminologías, modelos e interfaces consistentes para integrar el intercambio de datos entre diferentes niveles en la jerarquía de un sistema de producción. El estándar define la estructura de datos MES y sus servicios asociados con las operaciones de fabricación como (1) definición del producto, (2) planeación de producción, (3) gestión de la capacidad de producción y (4) evaluación del rendimiento de producción (Leusin, Frazzon, Uriona Maldonado, Kück, & Freitag, 2018).

Proceso de toma de decisión ante perturbaciones

El resultado de reportes basados en campo indica que la toma de decisiones en programación de operaciones es un proceso iterativo de identificación y solución de problemas, el cual se realiza recopilando información e interactuando con diferentes partes de la organización (De Snoo, Van Wezel, Wortmann, & Gaalman, 2011). La dinámica del proceso de toma de decisiones, el cual implica identificar qué información se necesita, quién y qué sistemas deberían estar involucrados, qué acciones se pueden tomar, depende de la característica de la perturbación, las cuales se relacionan o con los recursos o con el trabajo (*job*) (Ouelhadj & Petrovic, 2009).

En la Figura 2 se describe gráficamente el proceso básico de toma de decisiones para tratar con eventos no programados (perturbaciones). Este se compone de detectar, identificar, evaluar y responder. Una vez que la causa de la perturbación es detectada a través de sistemas automatizados o humanos (Cowling & Johansson, 2002), los tomadores de decisiones deben identificar la importancia del problema dado para determinar el nivel de atención y que actores están involucrados (máquinas, trabajos o personal). En la siguiente etapa, se evalúan las alternativas de solución en la reprogramación para encontrar la más adecuada dada el estado y las restricciones del sistema. Finalmente, se responde y se ejecuta a partir de la solución seleccionada y se toman medidas adicionales de ser necesarias dadas restricciones impuestas adicionales (Arica, Strandhagen, & Hvolby, 2014).

Como se muestra en la Figura 2 existen diferentes sistemas de información en la identificación y solución de problemas, tales como ERP, APS y MES. Estos soportan las diferentes etapas del proceso de toma de decisiones aprovechando los beneficios que cada sistema provee. Sin embargo, dada la complejidad del proceso programación y reprogramación de operaciones tras perturbaciones, los beneficios de los sistemas APS y MES no son suficientes tal como se mencionó en la sección anterior. Además, otro desafío enfrentado en el proceso de decisión es el de la información. Arica et al. (2014) lo describe y clasifica en representación, disponibilidad, accesibilidad, estructura y visualización de la información.

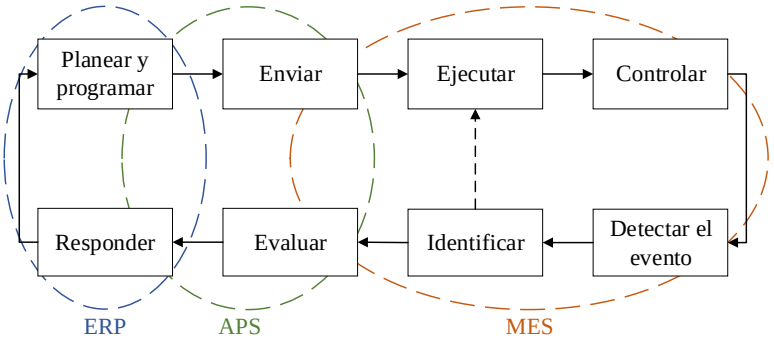

Figura 2. Ciclo para la toma de decisiones ante perturbaciones y sistemas de soporte para cada etapa (adaptado de Arica et al., 2014).

Lo anterior plantea un requerimiento particular de un sistema de soporte de decisiones capaz de integrarse a un sistema ERP y supere las carencias presentadas por los sistemas APS y MES.

Teoría de sistemas de soporte de decisión

Definición de sistemas de soporte de decisión

A pesar de que Turban et al. (2004) afirman que no hay una definición formal para un DSS, ya que cada autor se centra en resaltar las diferentes capacidades desarrolladas por su investigación y aplicación, en este trabajo de investigación se considerarán las siguientes definiciones ya que son el fundamento teórico del cual parten la mayoría de otras definiciones. Gorry & Morton (1971) definieron un DSS como "un sistema informático interactivo, el cual ayuda a los tomadores de decisiones a resolver problemas no estructurados utilizando datos y modelos". Keen & Morton (1978) indicaron otra definición clásica del DSS: "DSS combina los recursos intelectuales de las personas con las capacidades de los computadores para mejorar la calidad de las decisiones. Es un sistema de soporte informático para los responsables de la toma de decisiones que se ocupan de problemas semiestructurados". A partir de las definiciones anteriores, se puede decir que un DSS es una herramienta para extender las capacidades de los tomadores de decisiones y no para remplazar su juicio. Igualmente, aunque no se especifica en las definiciones anteriores, está implícito que un DSS opera de manera interactiva en línea y tiene salidas graficas para facilitar su uso.

En la siguiente subsección se discutirán las características y capacidades generales de un DSS para formalizar y complementar la definición anterior.

Características y capacidades de sistema de soporte de decisión

Turban et al. (2004) definen catorce características ideales que constituyen un DSS. En la Figura 3 se encuentra la numeración correspondiente con las características propias de un DSS y se enlistan de manera general.

Los DSS deben estar en capacidad de:

1. Apoyar problemas semiestructurados y no estructurados, los cuales no pueden resolverse únicamente por medios informáticos. Se debe tener en cuenta tanto el juicio humano como la información computarizada.
2. Apoyar decisiones en todos los niveles de gestión. Desde el nivel estratégico hasta el nivel operativo.
3. Apoyar decisiones individuales o grupales de acuerdo con la dificultad del problema.

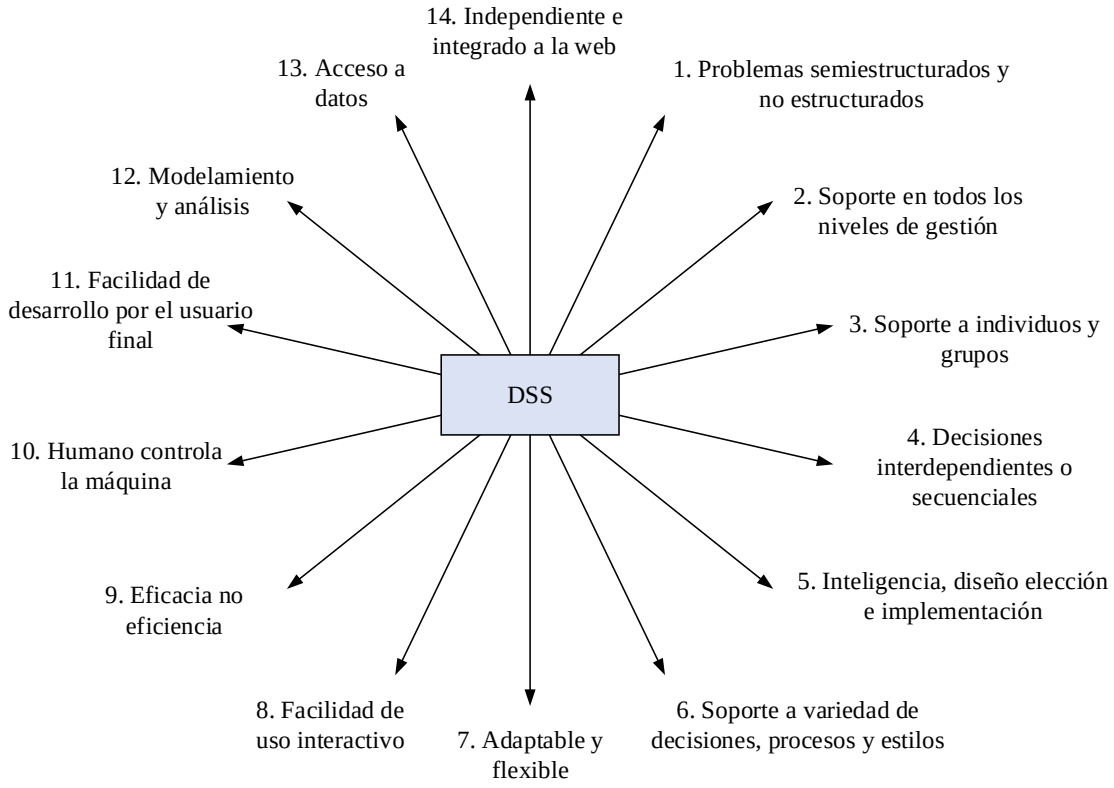

Figura 3. Características y capacidades clave de un DSS (adaptado de Turban et al., 2004).

4. Apoyar decisiones interdependientes o secuenciales. Decisiones que se realizan una sola vez, varias veces o repetidamente.
5. Apoyar todas las etapas del proceso de toma de decisiones: inteligencia, diseño, elección e implementación. Una definición detallada de cada etapa del proceso general de toma de decisiones puede ser consultada en Turban et al. (2004).
6. Apoyar una variedad de procesos de toma de decisiones.
7. Ser reactivo y estar en capacidad de afrontar situaciones de cambio. Asimismo, debe ser flexible permitiendo diferentes opciones al usuario y solucionando otros problemas similares.
8. Ser amigable con el usuario.
9. Mejorar la eficacia de la toma de decisiones (calidad, exactitud, oportuna) en lugar de su eficiencia (costo de tomar una decisión). Bajo esa configuración es posible que le DSS tome más tiempo, pero apoye mejores decisiones.
10. Ser manejador por seres humanos a lo largo de todo el proceso de toma de decisiones en resolución de problemas. El DSS es una herramienta de soporte no de remplazo.

11. Permitir al usuario final desarrollar y modificar sistemas simples. En contraste con sistemas complejos para los cuales se requieren sistemas informáticos más especializados.
12. Utilizar modelos para analizar las situaciones de toma de decisiones.
13. Acceder a diferentes fuentes, formatos y tipos de información.
14. Ser una herramienta usada o por un solo tomador de decisiones o distribuida a través de toda la organización en diferentes niveles con la posibilidad de integrarse a otros DSS.

En el Capítulo 3 se definen las características y capacidades con las que contará el DSS propuesto en este trabajo tomando como referencia el marco presentado en esta subsección.

Componentes de sistemas de soporte de decisión

Para concluir la definición formal de un DSS, se presentan ahora los subsistemas que típicamente lo componen. En la Figura 4 se muestran sombreados los subsistemas típicos y se muestra la relación que tienen cada uno de ellos entre sí y con otros actores de la organización.

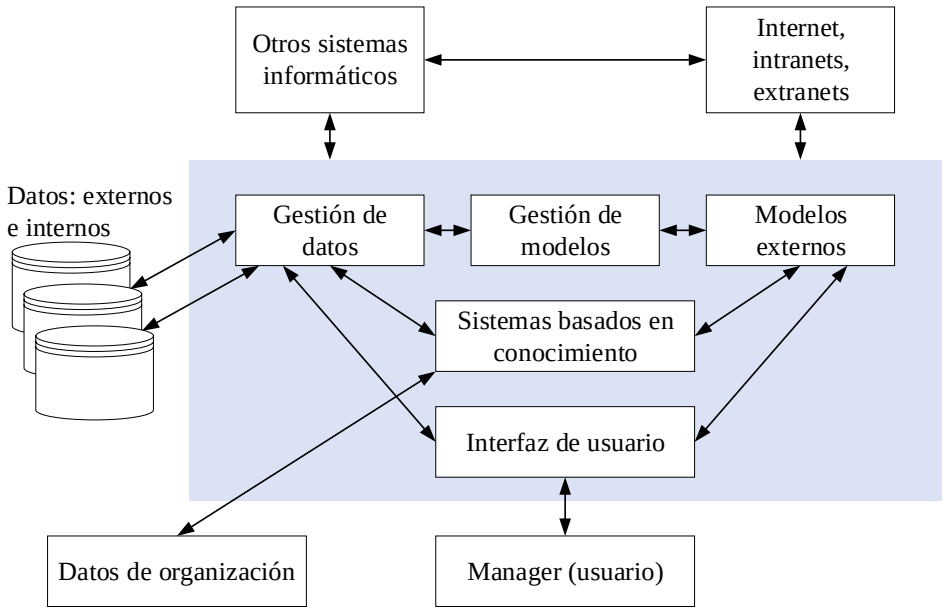

Figura 4. Vista esquemática de un DSS (adaptado de Turban et al., 2004).

Subsistema de gestión de datos

Se compone de una base de datos que contiene toda la información relevante para la situación a soportar. Este subsistema puede estar conectado con bases de datos corporativas y su acceso es comúnmente a través de servidores web.

Subsistema de gestión de modelos

Provee las capacidades analíticas del DSS a través de modelos estadísticos, financieros o cuantitativos. Normalmente es un software que incluye los paquetes mencionados, el cual permite construir modelos propios a partir de programación.

Subsistema de interfaz de usuario

Permite la comunicación entre el usuario y el DSS.

Subsistema de gestión de base de conocimiento

Este subsistema sirve para soportar los tres subsistemas anteriormente mencionados pues provee inteligencia para aumentar la capacidad de toma de decisiones.

En general, los DSS deben incluir al menos los subsistemas de gestión de datos, de gestión de modelos y de interfaz de usuario. El subsistema de gestión de base de conocimiento es opcional, pero puede traer benéficos al integrarse con los otros tres subsistemas.

Capítulo 2
Programación de operaciones dinámica: Revisión de la literatura

Introducción

En el capítulo anterior se indició que el área de manufactura foco de esta investigación es la programación de operaciones. Se presentó una contextualización de la programación de operaciones a nivel empresarial. Se indicó el manejo de perturbaciones a través de las tecnologías de soporte de decisiones y se brindó el marco del proceso de toma de decisiones en presencia de perturbaciones. Finalmente, se introdujeron conceptos generales de sistemas de soporte de decisión. El objetivo de este capítulo es revisar la literatura relacionada con programación de operaciones dinámica y caracterizar los trabajos realizados en este campo. En este capítulo se estudian dichos trabajos, determinando el entorno de manufactura, las perturbaciones consideradas, la estrategia, política, método y objetivo de reprogramación y el nivel de ejecución en el piso de manufactura, usualmente realizada por simulación. Se indican ciertas limitaciones identificadas, las cuales llevan al sistema propuesto en el Capítulo 3.

Con el fin de presentar una caracterización de los enfoques existentes en programación dinámica, la primera sección del capítulo presenta el contexto de investigación de la programación de operaciones desde sus inicios hasta el desarrollo más reciente. La segunda sección presenta la definición formal del problema de programación de operaciones y presenta conceptos relacionados con la medición del desempeño del sistema. Una clasificación de los modelos o entornos *job shop* y las características y limitaciones de cada uno de ellos en frente de las tecnologías de Industria 4.0. se presentan en la tercera sección. La cuarta sección introduce los conceptos claves en programación dinámica divididos en ambiente, política estrategia y métodos; esta sección también introduce la definición y clasificación de las perturbaciones. En la quinta sección se realiza una revisión de los métodos analíticos de solución al problema de programación de operaciones divididos principalmente en métodos exactos y métodos de aproximación. Finalmente, se presenta la caracterización propuesta y se realiza la revisión de literatura en programación dinámica.

Contexto de investigación de la programación de operaciones

En esta sección se introducen el problema de programación de operaciones como problema de investigación. Se indica la brecha existente entre la programación teórica y la programación práctica respecto a su ejecución en el piso de manufactura y se dan algunas razones de esta. Se presenta el concepto de programación de operaciones dinámica y algunos de los trabajos de revisión realizados en este campo. Finalmente, se indican las direcciones de investigación en

programación de operaciones integrando tecnologías de la cuarta revolución industrial: Industria 4.0.

El problema de programación de operaciones fue presentado por primera vez por Johnson (1954). A partir de ese momento un gran número de investigaciones que abordan dicho problema se han desarrollado para encontrar una solución óptima, en particular en entornos *job shop* (JS), debido a su importancia y alta demanda en la industria y al ser la configuración más común en pisos de manufactura (Fuchigami & Rangel, 2018). En las investigaciones realizadas se han usado diferentes métodos para resolver este problema, las cuales, por lo general, se clasifican en métodos exactos y métodos de aproximación. En este trabajo se profundizarán dichos métodos en la sección "Métodos de solución para el problema de programación de operaciones: generación de la programación" del presente capítulo.

Además, la mayoría de las investigaciones realizadas en este campo son teóricas y presentan problemas analíticos bien definidos, simplificados y orientados al objetivo, que en entornos estables, pueden producir resultados de rendimiento esperados (Arica et al., 2014). Sin embargo, los sistemas de manufactura funcionan en entornos dinámicos, en los cuales eventos impredecibles en tiempo real causan cambios en las programaciones realizadas y las convierten en no factibles una vez son ejecutadas en el piso de manufactura (Ouelhadj & Petrovic, 2009). Los eventos en tiempo real pueden ser fallas en la máquina, llegada de trabajos urgentes, cambios en los tiempos de entrega, entre otros. Estos eventos en tiempo real se conocen como perturbaciones.

Lo anterior, genera una brecha entre la teoría y la práctica en la programación de operaciones, pues las formulaciones teóricas enfrentan de manera inadecuada las complejidades de los entornos de manufactura reales realizando suposiciones y aproximaciones que reducen el problema a uno que, se puede resolver por métodos tradicionales pero que, está alejado de la realidad. MacCarthy & Liu, (1993b) abordaron esta brecha, indicaron el fracaso de la teoría de programación clásica al tratar de responder a las necesidades en entornos prácticos y la tendencia en programación de operaciones para hacerla más relevante y aplicable. Shukla & Chen (1996) mostraron a través de una comparación que hay muy poca correspondencia entre la práctica y la teoría en la ejecución de programación de operaciones. Cowling & Johansson (2002) indicaron también la brecha antes mencionada y afirmaron que los modelos y algoritmos de programación no están en capacidad de usar información en tiempo real. En el caso del entorno *job shop* se consideran las siguientes como causa de la brecha mencionada (Arisha, Young, & El Baradie, 2001):

(1) Énfasis en problemas de pequeña escala: En general, las investigaciones se concentran en problemas con *n* trabajos y *m* máquinas, sin embargo, hay grandes esfuerzos en problemas con cuatro o menos máquinas.

(2) Simplificación de las restricciones del problema: En general se consideran los tiempos de procesamiento y transporte conocidos; máquinas con capacidad de procesar un solo trabajo a le vez y las operaciones en las máquinas no pueden ser interrumpidas. Todos los supuestos consideraros en la formulación clásica se indican en la siguiente sección.

(3) Funciones objetivo simples: Se considera un único parámetro de optimización como objetivo global de la programación. Sin embargo, en la práctica se pueden considerar más objetivos según el grado de dificultad del entorno. Por ejemplo, se requiere tener utilización de máquinas balanceada y tiempos de flujo en el sistema bajos.

El problema de programación de operaciones en presencia de eventos en tiempo real es denominando programación dinámica (*dynamic scheduling*). Existe literatura que aborda esta programación proponiendo métodos y técnicas avanzadas que pueden predecir o reaccionar a la ocurrencia de eventos aleatorios no planeados. Los trabajos presentados por (Arisha et al., 2001), Ouelhadj & Petrovic (2009), (Chaari, Chaabane, Aissani, & Trentesaux, 2014), (Çaliş & Bulkan, 2015), Chaudhry & Khan (2016), J. Zhang, Ding, Zou, Qin, & Fu (2017), (Fazel Zarandi, Sadat Asl, Sotudian, & Castillo, 2018), Mohan, Lanka, & Rao (2019) presentan revisiones sobre dichos métodos. Se debe tener en cuenta que la mayoría de estas investigaciones incorporan eventos aleatorios identificados con su respectiva estimación de impacto sobre la programación final y no toman esta información en tiempo real.

La presencia de perturbaciones en entornos de manufactura afecta el proceso de toma de decisiones en programación (como se indicó en el capítulo anterior) e involucra diferentes aspectos de carácter humano, tecnológico y organizacional (HTO: *humanas, technology and organization*). Por ello, se deben identificar las características más importantes de los aspectos antes mencionados en la práctica de la programación y control de operaciones para de esa manera reducir la brecha y mejorar la aplicabilidad e implementación de sistemas de soporte de decisión (DSS) en manufactura (Arica et al., 2014). Berglund & Karltun (2007) presentan una descripción de los aspectos HTO que sirven como conceptos generales y modelo de análisis en programación de operaciones. En cuanto al aspecto tecnológico distinguen dos sistemas, el primero es el sistema tecnológico primario que incluye los equipos de producción para mantener la capacidad de la empresa; el segundo es el sistema que ayuda a la administración y los procedimientos de la empresa pero que no está directamente relacionado con las actividades de valor agregado. En el marco del aspecto tecnológico se sitúa actualmente la Industria 4.0. Ésta lidera la cuarta revolución industrial, que es la dirección futura de la automatización y las tecnologías de la información en manufactura (J. Zhang et al., 2017).

Con la comunicación de información e intercambio entre trabajos, máquinas, herramientas, personas y otros recursos, el cambio de control centralizado a procesos de producción descentralizados o distribuidos marca una característica importante en la Industria 4.0

(Kagermann, 2015). Generalmente la cuarta revolución industrial consta de varios elementos como internet de las cosas (IoT: *Internet of the Things*), sistemas ciber-físicos (CPS: *Cyber-Physical Systems*), fábricas inteligentes (*smart factories*) y nubes informáticas (*cloud computing*) (Hermann, Pentek, & Otto, 2016). Uno de los beneficios de los CPS es la posibilidad de vincular directamente el piso de manufactura con un DSS a través de información en tiempo real, lo que le da al piso de manufactura la capacidad de adaptarse rápidamente a la salida del DSS (Rossit, Tohmé, & Frutos, 2019). Liu, Wang, Wang, Xu, & Zhang, 2019 presentan una revisión de literatura centrada en programación de operaciones con soporte de *cloud computing* y discuten las características típicas de los problemas de programación en *cloud manufacturing*.

Problema de programación de operaciones *job shop* y *flexible job shop*

Esta sección presenta de manera formal la definición del problema *job shop* y *flexible job shop* (FJS). Indica los supuestos que se asumen normalmente para la solución de este problema e introduce conceptos clave relacionados con funciones objetivo usuales para la medición del rendimiento de las soluciones obtenidas.

El *job shop scheduling problem* (JSSP) es un problema clásico de investigación de operaciones que ha sido considerado como un problema complejo de optimización combinatoria y *NP-hard* en términos de complejidad computacional (Garey & Johnson, 1979). Por lo tanto, incluso para instancias pequeñas no se puede asegurar una solución óptima. En el JSSP, hay un número finito *n* de trabajos a ser procesados en un numero finito *m* de máquinas. Cada trabajo está compuesto por un grupo de operaciones que deben ser realizadas en una máquina diferente y en tiempos de procesamiento específicos siguiendo el orden de cada trabajo, los cuales pueden tener una secuencia independiente de procesamiento. El *flexible job shop scheduling problem* (FJSSP) es una extensión del JSSP, la cual permite que una operación sea procesada por cualquier máquina dado un grupo alternativo de máquinas (Chaudhry & Khan, 2016). Existen $(n_1)!\ (n_2)!...\ (n_m)!$ posibles soluciones teóricas, aunque no todas son factibles, siendo n_k el número de operaciones a ser realizadas en la máquina *k*, por lo que el FJSSP se considera también *NP-hard*. La mejor solución debe satisfacer dos condiciones: respetar las restricciones de precedencia del trabajo y optimizar la medida de referencia (Sule, 2018).

En general, el FJSSP puede ser definido de la siguiente manera (Xia & Wu, 2005):

(1) Hay un número *i* de trabajos independientes denotados como $J = \{J_1, J_2, \ldots, J_i\}$.

(2) Un trabajo J_i está conformado por una secuencia *j* de operaciones denotadas como $(O_{i1}, O_{i2}, \ldots, O_{ij})$.

(3) Hay un número *k* de máquinas denotadas como $M = \{M_1, M_2, \ldots, M_k\}$.

(4) Para cada operación O_{ij} hay un grupo de máquinas capaces de realizarla denotado por $M_{ij} \subseteq M$.

(5) El tiempo de procesamiento de cada operación es dependiente de la máquina. Se denota p_{ijk} para el tiempo de procesamiento de la operación O_{ij} realizada en la máquina M_k.

(6) Se consideran, adicionalmente, los siguientes supuestos en el FJSSP (Chaudhry & Khan, 2016):

   (a) Todas las máquinas están disponibles en el tiempo t = 0.

   (b) Todos los trabajos están disponibles en el tiempo $t = 0$.

   (c) Cada operación puede ser procesada por una sola máquina al tiempo.

   (d) No hay restricciones de precedencia entre las operaciones de los diferentes trabajos (trabajos independientes).

   (e) No hay preferencia de operaciones (*pre-emption*). Una vez una operación es iniciada no puede ser interrumpida.

   (f) Tiempo de transporte de un trabajo entre máquinas y tiempo de alistamiento de la máquina para el procesamiento de una operación está incluido en el tiempo de procesamiento.

Por lo anterior, el FJSSP es un problema en el que se debe tanto secuenciar operaciones, como asignar operaciones a máquinas adecuadas para realizar la operación (Pezzella, Morganti, & Ciaschetti, 2008). El FJSSP se puede categorizar entonces en dos subproblemas:

(1) Un subproblema de ruteo en el que se debe seleccionar una máquina apropiada de un grupo de máquinas disponibles para realizar la operación.

(2) Un subproblema de secuenciación en el que las operaciones asignadas a una máquina son secuenciadas con el fin de minimizar la función objetivo determinada.

Debido a los numerosos y complejos objetivos que pueden ser definidos para el FJSSP, no es fácil definir solamente uno. Una de las principales dificultades para medir la efectividad de una secuenciación realizada es que en muchas ocasiones no hay una medida común de valor para todas las propiedades deseables de la programación (Mellor, 1966). Mellor (1966) presenta veintisiete objetivos usados en programación de operaciones, que pueden ser aplicados en un contexto más amplio al netamente analítico y matemático (French, 1982).

Con el fin de definir las funciones objetivo usadas en el FJSSP en términos matemáticos se debe primero indicar algunas definiciones y notaciones (Pinedo, 2016):

(1) El tiempo de finalización de la operación O_{ij} es denotado por C_{ij}.

(2) El tiempo que un trabajo permanece en el sistema, esto es, el tiempo de finalización de la última operación procesada, es denotado por C_j.

(3) El retraso (*lateness*) de un trabajo *j* se define como, $L_j = C_j – d_j$, siendo d_j el tiempo de entrega del trabajo (*due date*).

(4) La tardanza (*tardiness*) de un trabajo *j* se define como $T_j = max\ (C_j – d_j, 0) = max\ (L_j, 0)$.

   (a) La diferencia entre el retraso y la tardanza es que la tardanza nunca toma un valor negativo.

Se define también el tiempo de inactividad de una máquina (French, 1982):

(5) El tiempo de inactividad (*idle time*) de una máquina M_k, se denota por $I_k = C_{max} - \sum p_{ijk}$.

La Figura 5 ilustra la forma de las funciones anteriormente mencionadas.

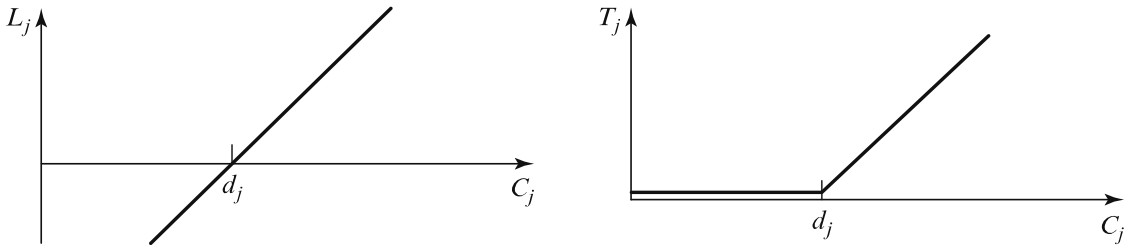

Figura 5. Funciones de tiempo de finalización (Pinedo, 2016).

Además, es importante mencionar que las funciones objetivo se pueden clasificar como regulares y no regulares. Una función objetivo es regular si su óptimo se da tanto iniciando como acabando trabajos tan pronto como sea posible (Chaudhry & Khan, 2016). El ejemplo más común de este tipo de funciones es el *makespan* (C_{max}). El *makespan* definido como el $max\ (C_1, C_2, \ldots, C_n)$ es equivalente al tiempo de finalización del último trabajo que deja el sistema. Un *makespan* bajo usualmente implica una buena utilización de las máquinas (Pinedo, 2016). Una función objetivo irregular es por lo general una función cuyo valor no depende únicamente del tiempo de finalización de un trabajo, por ejemplo, en ambientes de trabajo justo a tiempo (JIT: *Just in Time*) acabar trabajos tempranamente implica mayor cantidad de trabajo en proceso (WIP: *Work in Procesas*) (Chaudhry & Khan, 2016).

Chaudhry & Khan (2016) presentan las funciones objetivo más usadas en los últimos 25 años en el campo de la investigación del FJSSP.

El diseño (*layout*) del entorno FJS se puede considerar como un grafo *G = (S, V)*, que forma la topología de una célula de manufactura flexible. Los vértices (nodos) son máquinas o puntos de giro dentro de la célula. Los segmentos son conexiones que transfieren los trabajos entre vértices. La matriz de adyacencia *|V| x |V|* representa la dirección y las conexiones entre vértices del grafo *G*. La célula de manufactura está compuesta de dos tipos de recurso: las máquinas *M* y los trabajos *J*. Las máquinas *M* se encuentran dentro de la célula como un subconjunto de los vértices *V*. Los trabajos *J* son transportados dentro de la célula de manufactura guiados por los segmentos *S*.

A partir de lo anterior, se definen los siguientes paramentos:

G	Representación de la topología de la célula de manufactura (piso de manufactura).
V	Conjunto de vértices de *G {1, ..., v, ...V}*
S	Conjunto de segmentos de *G {1, ..., s, ...S}*
M	Conjunto de máquinas *{1, ..., m, ...M} M ⊆ V*
PG	Conjunto de puntos de giro *{1, ..., pg, ...PG} PG ⊆ V*
J	Conjunto de trabajos independientes *{1, ..., j ...J}*

La Figura 6 ilustra un ejemplo de un piso de manufactura *flexible job shop*. La topología de este ejemplo es un grafo de 10 vértices y 11 segmentos, definidos en la matriz *V x V*, mostrada en la Figura 7. El sistema tiene cuatro máquinas en capacidad de realizar todas las operaciones que requieren los trabajos a procesar.

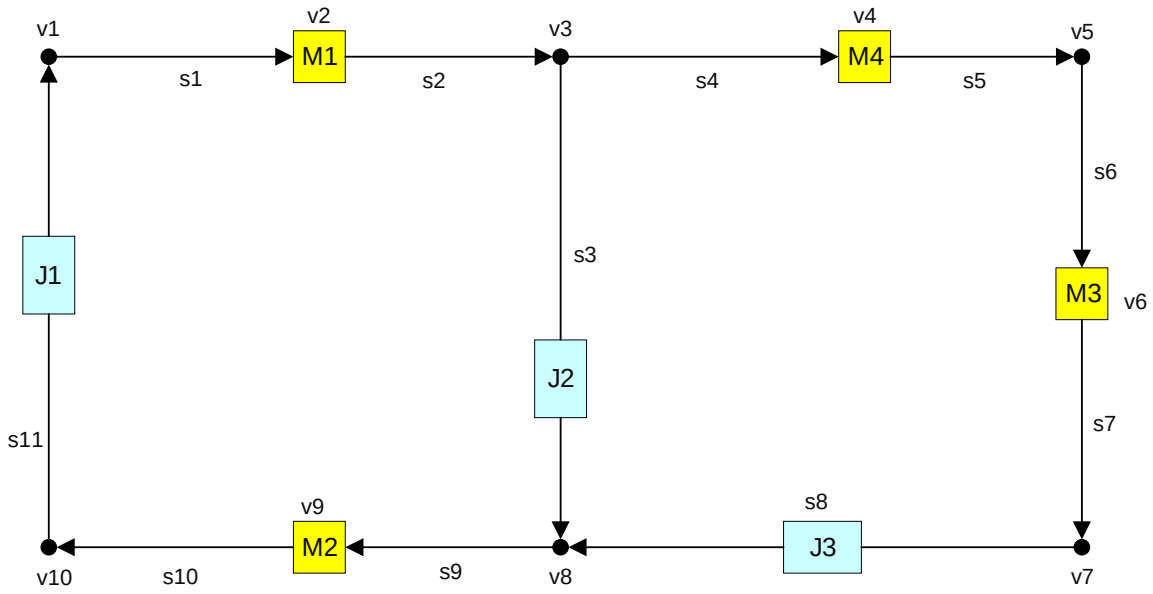

Figura 6. Ejemplo de un sistema de manufactura *flexible job shop*.

Los parámetros del ejemplo anterior, se muestran en la Figura 7.

$$\text{VxV} = \begin{array}{c} \\ \text{v1} \\ \text{v2} \\ \text{v3} \\ \text{v4} \\ \text{v5} \\ \text{v6} \\ \text{v7} \\ \text{v8} \\ \text{v9} \\ \text{v10} \end{array} \begin{array}{c} \begin{array}{cccccccccc} \text{v1} & \text{v2} & \text{v3} & \text{v4} & \text{v5} & \text{v6} & \text{v7} & \text{v8} & \text{v9} & \text{v10} \end{array} \\ \begin{pmatrix} - & 1 & - & - & - & - & - & - & - & - \\ - & - & 1 & - & - & - & - & - & - & - \\ - & - & - & 1 & - & - & - & - & - & - \\ - & - & - & - & 1 & - & - & - & - & 1 \\ - & - & - & - & - & 1 & - & - & - & - \\ - & - & - & - & - & - & 1 & - & - & - \\ - & - & - & - & - & - & - & 1 & - & - \\ - & - & - & - & - & - & - & - & 1 & - \\ 1 & - & - & - & - & - & - & - & - & - \\ - & - & - & - & - & - & - & - & 1 & - \end{pmatrix} \end{array}$$

G = {V, S}
V = {v1, ..., v10}
S = {s1, ..., s11}
M = {M1, M2, M3, M4} = {v2, v9, v6, v4}
PG = {v1, v3, v5, v7, v8, v10}
J = {J1, J2, J3}

Figura 7. Parámetros del ejemplo de sistema de manufactura *flexible job shop*.

Clasificación de los modelos de programación tipo *job shop*

En la sección anterior, se definió que el entorno *flexible job shop* es una extensión del entorno clásico *job shop*. De acuerdo con la teoría clásica de programación de operaciones (Pinedo, 2016), el *flexible job shop* es un entorno de manufactura y puede ser clasificado dentro de un sistema flexible de manufactura (FMS: *Flexible Manufacturing System*) tipo IV de acuerdo a lo presentado por MacCarthy & Liu (1993a).

Esta sección presenta una clasificación de los modelos o entornos *job shop* con el fin de determinar la ubicación del entorno *flexible job shop*. Esta clasificación se realiza a partir de las variables que afectan principalmente la programación de operaciones y permite identificar las características y limitaciones de cada modelo en entornos de manufactura con tecnologías de Industria 4.0.

Los problemas de programación de operaciones pueden ser clasificados de acuerdo con las principales variables que los afectan. Estas son: fuentes de producción de demanda, número de máquinas, complejidad del sistema de producción, índice de rendimiento, características del ambiente de producción y restricciones de los recursos (Lin et al., 2012). Con base en las características anteriores y considerando también la cantidad de plantas con que cuenta un sistema, existen quince modelos de programación tipo job shop mostrados en la Figura 8.

Con el fin de entender cada modelo tipo *job shop* según su alcance e identificar sus características principales, la clasificación presentada en la Figura 8 se reorganiza en cinco grandes estructuras o modelos siguiendo el marco presentado por J. Zhang et al. (2017). Estos se clasifican en *job shop*: tipo básico, tipo múltiples máquinas, tipo múltiples recursos, tipo múltiples plantas y tipo planta inteligente.

(1) Tipo básico (JS): en este modelo, un trabajo específico se procesa en una máquina determinada y no es posible escoger entre otras máquinas para las operaciones del

trabajo. La decisión de este modelo es secuenciar las operaciones con el objetivo de minimizar costos. Las restricciones asociadas son las mismas consideradas en un sistema APS.

(2) Tipo múltiples máquinas (FJS): en este modelo es posible escoger más de una máquina para el procesamiento de operaciones de los diferentes trabajos. La decisión de este modelo es secuenciar las operaciones y asignar las máquinas para procesarlas de acuerdo con su disponibilidad. Se consideran como objetivos la minimización de costos y el balance de carga de las máquinas. Las restricciones son las mismas del modelo FJS.

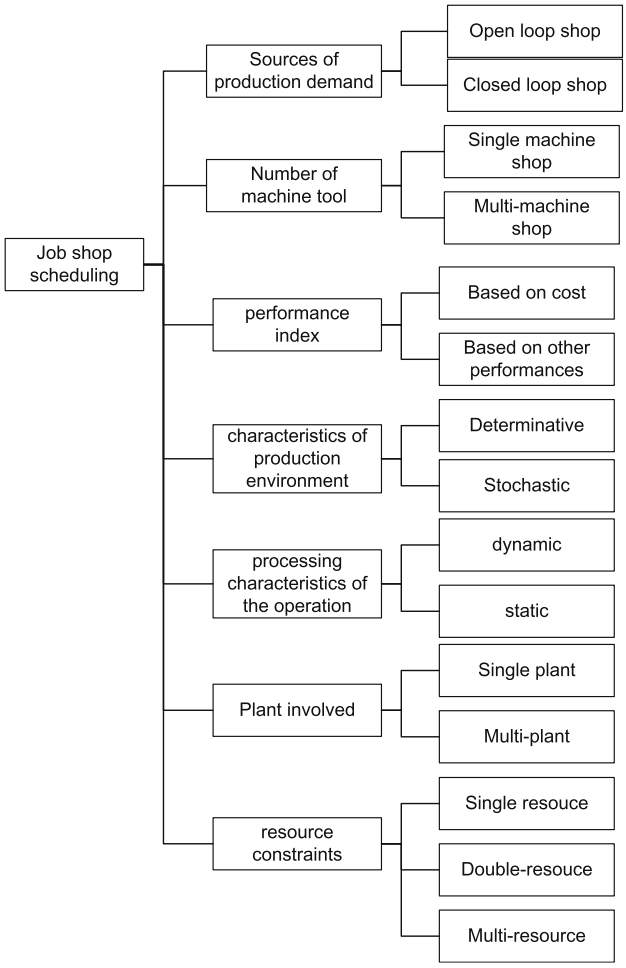

Figura 8. Tipos de entornos tipo *job shop* (J. Zhang et al., 2017).

(3) Tipo múltiples recursos o múltiples recursos FJS (MrFJS): en este modelo, además de la disponibilidad de la máquina, se contempla la disponibilidad de recursos para la asignación. En general, la capacidad de producción está restringida por máquinas, herramientas, accesorios, operadores, vehículos, robots y otros recursos de manufactura. La decisión de este modelo es secuenciar las operaciones y asignar

diferentes recursos disponibles a la vez, incluyendo máquinas y otros recursos para una operación. Se consideran como objetivos la minimización de costos y el balance de carga de las máquinas. Además de las restricciones del FJS, la información de recursos en el piso de manufactura es considerada en el modelo. Esto último lo hace más sofisticado y dinámico que el FJS.

(4) Tipo múltiples plantas (MpFJS): este modelo se basa en el MrFJSP y contempla además plantas múltiples y los transportes entre ellas. La decisión de este modelo es secuenciar operaciones y asignar diferentes recursos disponibles a la vez, incluyendo máquinas, otros recursos de operación y la planta de procesamiento. Se consideran como objetivos la minimización de costos, el balance de carga de las máquinas y la minimización de longitud recorrida por vehículos. En adición al modelo anterior, entre las restricciones se debe contemplar la cadena de abastecimiento de las diferentes plantas, las recogidas y las entregas de material.

(5) Tipo planta inteligente (SFFJS): este modelo contempla una programación inteligente distribuida a partir de agentes autoorganizados y con poder autónomo de toma de decisiones. Este modelo considera como agentes inteligentes las máquinas, los trabajos, los recursos asociados a la producción y las plantas de manufactura. Lo anterior se logra con base en la utilización de tecnologías de Industria 4.0, la cual usualmente consta de varios elementos como IoT, CPS, fábricas inteligentes y nubes informáticas (Hermann et al., 2016).

Problema de programación de operaciones dinámico

Esta sección presenta el marco teórico que sirve como línea base para definir los conceptos claves de la programación dinámica predictiva-reactiva. Se presenta una clasificación de las perturbaciones ocurridas en entornos de manufactura. Se presentan las estrategias para la toma de decisión para enfrentar las perturbaciones en modelos de programación. Se presentan los conceptos claves de la reprogramación de operaciones y se hace la introducción a la siguiente sección de este capítulo referente a los métodos de solución de problemas de programación.

En términos generales, un problema de programación de operaciones se considera dinámico si uno o más parámetros son variables en el tiempo (French, 1982). La anterior definición considera la presencia de eventos en tiempo real.

Perturbaciones

Los eventos en tiempo real o perturbaciones se han clasificado en dos categorías (Vieira et al., 2003):

(1) Relacionados con los recursos: falla de la máquina, contratiempos con el operador, no disponibilidad o falla de herramientas, límite de capacidad instalada, escasez de material, material defectuoso entre otros.

(2) Relacionados con el trabajo: aparición de trabajos urgentes, cancelación de trabajos, cambio de tiempos de entrega, llegada temprana o tardía de trabajos a las máquinas, cambios en la prioridad, cambios en el tiempo de procesamiento entre otros.

A lo largo de este trabajo se han referido y se referirán las perturbaciones como eventos en tiempo real, eventos impredecibles, eventos no planeados, eventos no programados, eventos aleatorios, eventos aleatorios no planeados.

Estrategias para la toma de decisión

La programación de operaciones dinámica se ha clasificado bajo tres categorías (Ouelhadj & Petrovic, 2009).

Programación completamente reactiva

Bajo esta estrategia no se realiza ninguna programación previa, en vez de ello las decisiones se toman de manera local en tiempo real. Sin embargo, no se logran medidas de desempeño óptimas en el piso de manufactura ya que, el tiempo de ejecución en el proceso de decisión se centra en la rapidez y no en la calidad de la respuesta. En este enfoque se utilizan con frecuencia reglas de prioridad o algoritmos voraces (*greedy algorithm)*, como por ejemplo en (Zahmani, Atmani, Bekrar, & Aissani, 2015) y (Bekkar et al., 2016).

Programación robusta proactiva

Bajo esta estrategia se realizan programaciones predictivas que incluyen en impacto en la medida de desempeño en entornos dinámicos. En este enfoque se aplican generalmente los métodos descritos en la sección “Métodos de solución para el problema de programación de operaciones: generación de la programación” del presente capitulo, los cuales buscan medidas óptimas de desempeño. La principal dificultad de este enfoque es la determinación del impacto de los eventos en el desempeño del piso de manufactura (previsibilidad) y el supuesto de disponibilidad total de la información. Por esta razón, la ejecución de programaciones robustas se usa para procesos *offline*, es decir, los realizadas antes de la ejecución.

Programación predictiva-reactiva

En esta estrategia se realiza un proceso de programación/reprogramación en el que la programación inicial es revisada en respuesta a los eventos en tiempo real. En general, en este enfoque se destacan dos fases. La primera es la generación de la programación (*schedule generation*), la cual es un mecanismo predictivo y sirve como un plan general para otras actividades en la operación. Esta fase también se conoce como fase de planeación o fase *offline*. La segunda fase es la actualización de la programación (*schedule updating*) que usualmente se

considera como la parte reactiva del sistema e intenta minimizar el efecto de las perturbaciones en el rendimiento del sistema (Varela & Ribeiro, 2014). Esta fase también se conoce como fase de ejecución o fase *online*.

La actualización de la programación de operaciones existente en respuesta a perturbaciones u otros eventos no planeados se conoce como reprogramación de operaciones (*rescheduling*). Se destacan tres tipos de estudios. El primero se centra en reparar una programación que ha sido afectada por una perturbación. El segundo propone crear una programación robusta en respuesta a las perturbaciones ocurrentes. Finalmente, el tercero estudia cómo las políticas de reprogramación afectan el desempeño de un sistema dinámico de manufactura (Vieira et al., 2003).

A lo largo de este trabajo se han referido y se referirán los términos reprogramación de operaciones (*rescheduling*) y actualización de la programación (*schedule updating*) como términos análogos. La actualización de operaciones propuesta en este trabajo de grado se basa en el primer estudio, es decir en reparar la programación inicial tras la ocurrencia de una perturbación con el fin de garantizar la continuidad de la producción.

Reprogramación de operaciones en presencia de perturbaciones

En esta sección se toman los conceptos de los marcos teóricos presentados por Larsen & Pranzo (2019) y Vieira et al. (2003). El marco teórico presentado en esta subsección incluye ambientes de reprogramación, políticas de reprogramación, estrategias de reprogramación y métodos de reprogramación.

Ambientes de reprogramación

Identifican el conjunto de trabajos que deben ser programados, la capacidad de las máquinas para procesar operaciones y el nivel de variabilidad del sistema. La teoría clásica de programación de operaciones distingue diferentes entornos o ambientes de manufactura. Máquina única, maquinas en paralelo, *flow shop* y *job shop* y *flexible flow shop*, *flexible job shop* y *open shop* (Pinedo, 2016).

Políticas de reprogramación

Especifican cuando un nuevo proceso de reprogramación debe ser iniciado. Es necesario definir otros términos para describir aspectos de la política.

(1) Punto de programación (reprogramación). Es el punto en el tiempo en el que una decisión de programación (reprogramación) es ejecutada.

(2) Periodo de reacción. Es el tiempo entre la ocurrencia de una perturbación y el inicio de un proceso de reprogramación.

(3) Periodo congelado (*frozen period*). Es la duración de la programación que no puede ser cambiada y debe ser mantenida. Se consideran las operaciones que están en ejecución u operaciones con tiempos de inicio inminente.

(4) Periodo de reprogramación. Es el tiempo entre dos puntos de reprogramación consecutivos.

(5) Objetivo de la reprogramación. Se distingue entre la optimización completa, en la cual se optimiza el mismo objetivo usado en la fase *offline* y la optimización parcial, en la cual el objetivo es minimizar una función objetivo sustituta. En la reprogramación parcial o reparación de la programación el objetivo es minimizar la desviación con respecto a la programación inicial teniendo en cuenta una medida de desempeño diferente a la inicial.

En general, los sistemas que monitorean, administran y controlan pisos de manufactura de operaciones requieren de ciertas capacidades para lograr los objetivos anteriormente descritos. Entre ellas se distinguen la capacidad de optimalidad, reactividad y adaptabilidad. La optimalidad indica que el sistema debe ser capaz de lograr un funcionamiento óptimo medido por métricas predefinidas de eficiencia y efectividad. La reactividad indica que el sistema debe tener la capacidad de actuar en respuesta a los cambios dentro de su entorno. La adaptabilidad indica que el sistema debe ser capaz de cambiar el comportamiento y el funcionamiento en respuesta a cambios en el medio ambiente (Jiménez, 2017). Se debe notar, por las definiciones anteriores que, la adaptabilidad se compone por reactividad y optimalidad.

Para lograr el objetivo de optimización completa, el sistema debe tener la capacidad de adaptabilidad. Para lograr el objetivo de reprogramación parcial, el sistema debe tener la capacidad de reactividad. Además, el sistema de control que guía la toma de decisiones de reprogramación en ambos casos es diferente. En el objetivo de optimización completa, el sistema es centralizado y se toman decisiones a nivel global. En el objetivo de reprogramación parcial, el sistema es heterárquico y se toman decisiones a nivel local.

(6) Periodo de cálculo. Es el tiempo que toma el método de solución para encontrar una nueva programación si se tiene como objetivo una optimización completa. Es el tiempo que toma el método de solución para encontrar nuevas decisiones asociadas a los recursos si se tiene como objetivo una reparación de la programación.

(7) Estabilidad de la programación. Mide el número de revisiones o cambios que una programación sufre durante su ejecución.

(8) Nerviosismo de la programación (*scheduling nervousness*). Es la medición contraria a la estabilidad de la programación. Se considera como cambios frecuentes en la programación y baja predictibilidad.

(9) Robustez de la programación. Mide que tanto se degrada la medida de desempeño en la ejecución en el piso de manufactura tras la ocurrencia de perturbaciones.

Las políticas de reprogramación pueden clasificarse en periódicas, continuas o basadas en eventos (Ouelhadj & Petrovic, 2009). Bajo una política periódica el proceso de reprogramación empieza después de un período fijo de tiempo. En la reprogramación continua, el proceso de reprogramación se lleva a cabo en cada avance de tiempo. En una política de reprogramación basada en eventos (también conocida como predictiva-reactiva) se establece que el proceso comienza en respuesta a algunos eventos específicos (Larsen & Pranzo, 2019).

La Figura 9 muestra los términos anteriormente descritos. Una vez termina la fase de planeación, se procede con la fase de ejecución. En esta fase, una perturbación puede ocurrir en el punto t_p. De acuerdo con la política establecida, en el punto t_r inicia la reprogramación (denotado también como t_{start}). Al mismo tiempo, comienza el periodo de cálculo, cuya duración es hasta el punto t_c. Una vez termina el periodo de recálculo y de acuerdo con el objetivo de reprogramación, se envía la nueva información al piso de manufactura para su ejecución - objetivo de optimización completa - o se ejecutan las nuevas decisiones en el piso de manufactura - objetivo de reprogramación parcial -. Dado que alguna parte de la programación previa no puede ser cambiada inmediatamente (periodo congelado), se considera la decisión de reprogramación completamente ejecutada en el punto t_{rs} (punto de reprogramación).

En el objetivo de reprogramación parcial, una vez termina el periodo de recálculo, la ejecución en el piso de manufactura no requiere de envío de información, dado que las decisiones se toman a nivel local.

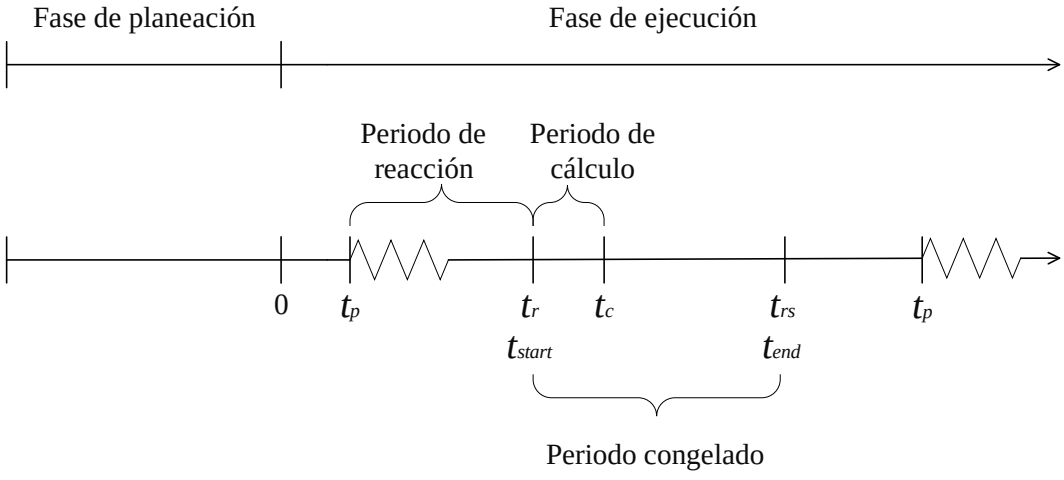

Figura 9. Línea de tiempo de la programación predictiva-reactiva.

Estrategias de reprogramación

Describen de qué manera las decisiones de programación y reprogramación son realizadas. Se distinguen tres estrategias en ambientes dinámicos de manufactura: programación completamente reactiva, programación robusta proactiva y programación predictiva-reactiva. Estas fueron explicadas en la subsección anterior.

Métodos de reprogramación

Describen como las programaciones y reprogramaciones son generadas y actualizadas. En las siguientes secciones se describen los métodos usados en la estrategia predictiva-reactiva, la cual es el foco de esta investigación.

Métodos de solución para el problema de programación de operaciones

La sección anterior presentó el marco teórico de la reprogramación de operaciones en presencia de perturbaciones y concluyó con los métodos de reprogramación. Esta sección retoma dicho concepto y se centra en los métodos de solución para el problema de programación de operaciones. Estos métodos se pueden usar tanto en la primera, como en la segunda fase del enfoque predictivo-reactivo, es decir, en la generación de la programación inicial y la actualización de la programación. Se presenta un marco de los métodos usados para la solución del FJSSP clásico, se presenta revisiones literarias y bibliografía asociada a los métodos presentados y se hace énfasis en el método metaheurísticos. En particular, en el algoritmo genético -técnica metaheurística- que será utilizado en este trabajo para la generación de la solución inicial.

Existen diferentes maneras de resolver el problema clásico de programación *flexible job shop*. Algunos investigadores han realizado recopilaciones de esta temática. Por ejemplo, Jain & Meeran (1998) clasificaron y compararon varios algoritmos de solución del momento. Estudios posteriores y recientes se centran en el desarrollo de algoritmos inteligentes. Por ejemplo, Çaliş & Bulkan (2015) presentan algoritmos inteligentes, metaheurísticas y formas especiales del JSSP.

Los métodos de solución y optimización para programación de operaciones están divididos en métodos de optimización exacta y métodos de aproximación. Los métodos exactos incluyen reglas eficientes, programación matemática y algoritmos ramificación y poda (B&B: *Branch and Bound*). Los métodos de aproximación incluyen métodos constructivos, inteligencia artificial, métodos de busca local y algoritmos metaheurísticos. La Figura 10 muestra los métodos anteriormente indicados y algunas técnicas de solución de cada uno.

Métodos exactos de optimización

A causa del continuo desarrollo de la tecnología y la computación, los métodos exactos se desarrollaron aproximadamente hasta 1980 para dar paso a los métodos de aproximación. Por ende, esta revisión comprende los años entre 1950 y 1980.

Los métodos basados en reglas de eficiencia son las primeras aproximaciones en este campo. De acuerdo con la información de entrada, se establecen reglas de procesamiento para determinar la secuencia exacta de operaciones en busca de un óptimo definido. Johnson (1954) propuso una serie de reglas para resolver un problema *flow shop* con dos máquinas. El criterio de minimización fue el *makespan*, el cual tiene una gran influencia incluso en investigaciones recientes. Wagner (1959) encontró técnicas de programación matemática que podían resolver el JSP de manera óptima. Sin embargo, también se encontró que estas técnicas son deficientes debido al excesivo tiempo computacional requerido. Por ello, se desarrollaron los métodos enumerativos, especialmente el de ramificación y poda (B&B). Brooks & White (1965) y Lomnicki (1965) aplicaron B&B para encontrar una solución óptima en un JSP. Hefetz & Adiri (1982) presentaron una solución óptima para un problema JSP con dos máquinas, en el cual cada operación es realizada con tiempos de procesamiento diferentes.

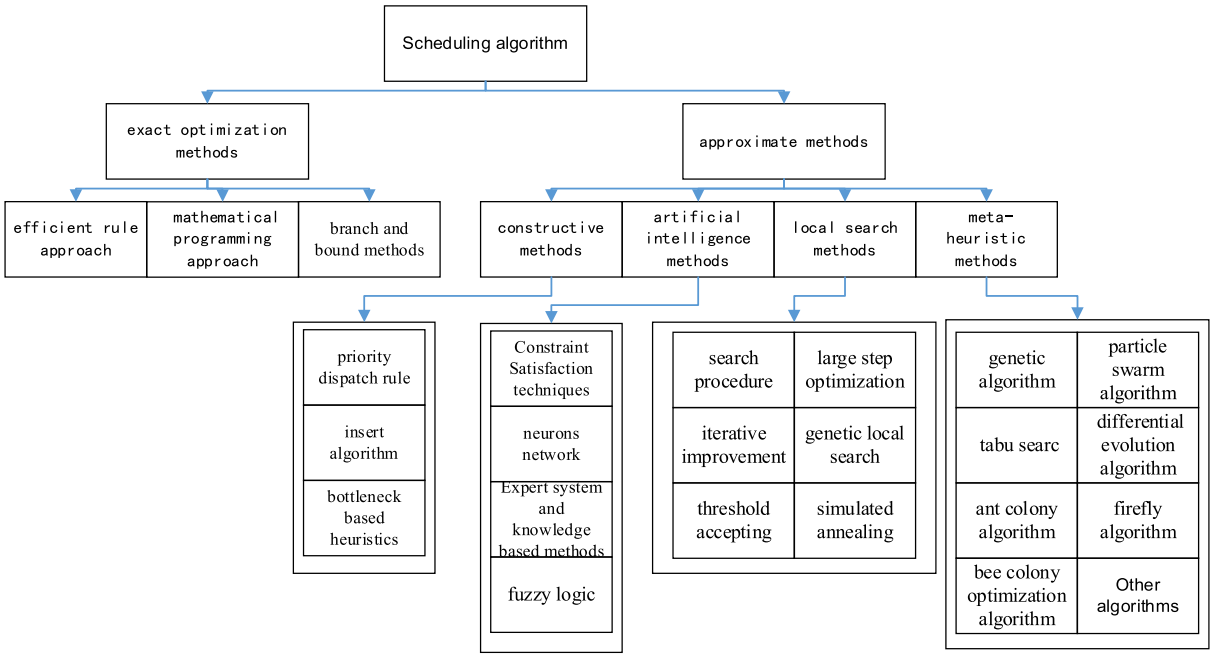

Figura 10. Métodos de optimización para programación de operaciones (J. Zhang et al., 2017).

Manne (1960) mezcló enfoques de programación lineal y entera discreta (MIP: *Mixed-Integer Programming*) y propuso una forma común de formulación matemática, que incluía una función objetivo lineal, una serie de restricciones lineales y las variables de decisión enteras y binarias. Esta formulación involucró menos variables y calculó más eficientemente que en Wagner (1959). Por otro lado, debido a que el método de B&B puede calcular el límite inferior de algunos

subconjuntos, es un método eficaz para resolver el problema de programación con una mejor solución. McMahon & Florian (1975) presentaron una aplicación exitosa, la cual se construyó al identificar el trabajo crítico con el máximo retraso y luego determinar los otros trabajos. Sarin, Ahn, & Bishop (1988) y Potts & Van Wassenhove (1985) mejoraron los métodos de B&B por separado. Los métodos son diferentes, pero ambos se centran en reglas de análisis, mecanismo de delimitación y generación del límite superior.

Métodos de aproximación

Métodos constructivos

Este método incluye tres técnicas típicas: reglas de prioridad de despacho (PDR: *priority dispatching rules*), algoritmos de inserción y heurística basada en cuello de botella. J. Zhang et al (2017) presentan una amplia revisión de estas técnicas bajo la siguiente estructura: primeros estudios, estudios recientes y combinación con algoritmos

Métodos de inteligencia artificial

Fazel Zarandi et al. (2018) presentan la revisión más reciente de programación inteligente cubriendo varias herramientas y técnicas categorizadas en lógica difusa, sistemas expertos en programación (*experta systems*), *Machine Learning*, algoritmos de optimización estocásticos de búsqueda local y programación con restricciones. Çaliş & Bulkan (2015) clasifican y presentan diferentes enfoques para la solución del JSSP entre los cuales se destacan sistemas basados en agentes, lógica difusa y redes neuronales artificiales. Akyol & Bayhan (2007) presentaron una revisión de la literatura en las aplicaciones de redes neuronales (NN: *Neural Networks*) en problemas de programación y encontraron principalmente cuatro categorías: redes de Hopfield, redes perceptrón multicapa, redes basadas en competición y enfoques híbridos.

Métodos de búsqueda local

Este método consiste en la búsqueda de un conjunto finito de soluciones, una función o conjunto de funciones a optimizar y una estrategia de búsqueda. De acuerdo con la estrategia de búsqueda, las técnicas típicas incluyen procedimiento GRASP, mejoramiento iterativo, valor de aceptación, optimización de paso largo, búsqueda genética local y recocido simulado (SA: *simulated annealing*) (Jain & Meeran, 1998). Entre las técnicas indicadas anteriormente el recocido simulado es el algoritmo de búsqueda local más usado en la programación de operaciones (J. Zhang et al., 2017). Çaliş & Bulkan (2015) presentan aplicaciones de diferentes técnicas para solucionar el JSSP, entre las que se destacan la búsqueda de haz (BS: *beam search*) y búsqueda tabú (TS: *tabu search*). Además, Chaudhry & Khan (2016) en su revisión de las técnicas usadas para el FJSSP tiene tres subcapítulos dedicados a recocido simulado, búsqueda tabú, GRASP y heurísticas determinísticas.

Finalmente, es importante notar que los métodos heurísticos raramente son usados de manera independiente y en muchas investigaciones de aplicación han sido combinados con métodos metaheurísticos. Por eso, trabajos citados en esta sección, serán citados en la siguiente presentando revisiones literarias de ambos métodos.

Métodos metaheurísticos

Una gran cantidad de trabajos se han realizado con revisiones literarias de los métodos metaheurísticos usados para resolver el FJSSP. Los métodos más frecuentes en la literatura comprenden:

(a) Algoritmo de colonia de abejas artificiales (ABC: *Artificial Bee Colony*).
(b) Optimización por colonia de hormigas (ACO: *Ant Colony Optimization*).
(c) Sistema inmunitario artificial (AIS: *Artificial Immune System*).
(d) Algoritmo de murciélago (BA: *Bat Algorithm*).
(e) Evolución diferencial (DE: *Differential Evolution*).
(f) Algoritmo de luciérnagas (FA: *Firefly Algorithm*).
(g) Algoritmo genético (GA: *Genetic Algorithm*).
(h) Algoritmo de campeonato (LCA: *League Championship Algorithm*).
(i) Optimización por enjambre de partículas (PSO: *Particle Swarm Optimization*).
(j) Técnicas híbridas.

La Tabla 1 resume las técnicas metaheurísticas utilizando la convención anterior y presenta bibliografía para profundizar en cada una de ellas.

Tabla 1. Revisión de la literatura de las técnicas metaheurísticas más utilizadas para resolver el FJSSP.

Referencia	**(a)**	**(b)**	**(c)**	**(d)**	**(e)**	**(f)**	**(g)**	**(h)**	**(i)**	**(j)**
Çaliş & Bulkan (2015)	X	X					X		X	
Kalra & Singh (2015)		X		X			X	X	X	
Chaudhry & Khan (2016)	X	X	X				X		X	X
J. Zhang et al. (2017)		X			X	X	X		X	
Fazel Zarandi et al. (2018)		X					X		X	
Mohan et al. (2019)		X	X				X		X	

Algoritmo genético. El algoritmo genético pertenece a la clase de algoritmos evolutivos y su desarrollo está inspirado en el proceso natural de evolución genética y el principio de supervivencia del más apto "*survival of the fittest*" para obtener soluciones en ingeniería. El primer

trabajo sobre un GA fue realzado por Holland (1975) y fue posteriormente extendido por Goldberg (1989).

Çaliş & Bulkan (2015) indican que el 26,4 % de las investigaciones realizadas en la solución del JSSP han sido desarrolladas con algoritmos genéticos (GA's), siendo este el porcentaje más alto para una técnica metaheurística y siendo seguida por técnicas de redes neuronales con 18,9 %. Por lo anterior, el GA ha probado ser una de las técnicas metaheurísticas más eficientes en la solución del JSSP y como consecuencia del FJSSP. Amjad et al. (2018) presentan la revisión más extensa de algoritmos genéticos aplicados a la solución del FJSSP. En la revisión presentan el concepto de algoritmo genético, sus elementos básicos y su adaptación para la solución del FJSSP; presentan el estado del arte de las técnicas GA clásico, GA avanzado, y GA híbrido (hGA: *hybrid GA*) para la solución del FJSSP.

El desarrollo del algoritmo genético propuesto en este trabajo se muestra en la sección "Subsistema de gestión de modelos" del Capítulo 3.

Revisión de la literatura

Esta sección presenta la literatura de los problemas de programación de operaciones que se relación con ambientes de manufactura *job shop* sujetos a perturbaciones. Esta revisión pretende identificar publicaciones con las características enunciadas a lo largo de este capítulo con el fin de presentar las contribuciones en esta área, construir una caracterización de la programación dinámica y plantear las características propias del sistema más adelante propuesto.

La revisión incluyó una búsqueda de artículos publicados recientemente en revistas internacionales que están indexadas en bases de datos reconocidas. Se realizó una investigación en bases de datos de ingeniería y tecnología, incluyendo *IEEE Xplore*, *Science Direct*, *Springerlink*, *Taylor & Francis* y *Wiley Online Library*. Además, se realizó una investigación en *Google Scholar* y la base de datos *JSTOR*. La búsqueda incluyó las siguientes palabras clave: *production scheduling*, *job shop*, *flexible job shop*, *dynamic scheduling*, *dynamic job shop*, *decision support systems*, *real-time production scheduling*, *online scheduling*, *predictive-reactive scheduling*, *rescheduling* y *scheduling in Industry 4.0*.

Los artículos presentados se analizaron de acuerdo con las siguientes características.

(a) Ambiente de programación. Se consideran solo los ambientes de manufactura job shop y flexible job shop. Se identificó durante la búsqueda una gran cantidad de literatura relacionada con ambientes flow shop pero estos no se consideraron dada la diferencia fundamental de la no flexibilidad en la ruta de cada trabajo.

(b) Perturbaciones. Se identifica si las perturbaciones consideradas se relacionan con el recurso, con el trabajo o con ambas. Algunos artículos que no presentaban claridad en este aspecto no fueron incluidos en la revisión.

(c) Estrategia para la toma de decisión. Se consideran las tres estrategias para la toma de decisión en programación dinámica. Se hizo esto con la intención de ampliar la perspectiva sobre las técnicas de solución y las medidas de desempeño utilizadas.

(d) Política de reprogramación. Se distingue entre las políticas periódica, continua y basada en eventos antes enunciadas. A partir de las definiciones presentadas de estrategias para la toma de decisión, se entiende que las programaciones completamente reactiva y predictiva-reactiva tienen implícita una política de reprogramación; una programación robusta proactiva no considera política de reprogramación -ya que la programación en su totalidad de hace de manera *offline*-.

(e) Método de reprogramación. Se distingue entre programación *offline* y programación *online*. A partir de las definiciones presentadas, se entiende que la programación completamente reactiva se compone de un solo método *online*; la programación predictiva-reactiva tiene implícita tanto un método *offline* como *online*; y la programación robusta proactiva se compone de un solo método *offline*. Conforme a la revisión realizada, la Tabla 2 indica los métodos más utilizados y presenta los acrónimos utilizados en la Tabla 5.

(f) Ejecución. Se evalúa como es la relación con el piso de manufactura. A partir de las definiciones presentadas, se entiende que se debe desarrollar un modelo de simulación, que represente el piso de manufactura de la mejor manera y considere los eventos dinámicos que se pueden presentar. En general, las programaciones robustas proactivas generan, en el algoritmo de programación, las perturbaciones a las que está sujeto el sistema de manera discreta pero no usan simulación para validar sus efectos. Las programaciones completamente reactivas y predictivo-reactivas utilizan simulación para responder a las perturbaciones en tiempo real. La simulación es una técnica utilizada en investigación de operaciones (OR: *Operations Research*) como herramienta de análisis para los escenarios "*what-if*". Las técnicas predominantes de simulación en OR son la simulación de eventos discretos (DES: *Discrete Event Simulation*) y simulación de sistemas dinámicos (SDS: *Systems Dynamics Simulation*). La simulación basada en agentes (ABS: *Agent Based Simulation*) ha tomado más importancia en los últimos años para simular diferentes comportamientos (Majid, Fakhreldin, & Zuhairi, 2016).

Tabla 2. Técnicas utilizadas en problemas de programación.

Abreviación	**Técnica**
ACO	Optimización por colonia de hormigas (*Ant Colony Optimization*)
ANN	Red neuronal artificial (*Artificial Neural Network*)
DE	Evolución diferencial (*Differential Evolution*)
GA	Algoritmo genético (*Genetic Algorithm*)
GRASP	Greedy randomized adaptive search procedure
HEU	Heurísticas
MIP	Programación entera mixta (*Mixed-Integer Programming*)
NPA	Protocolo de negociación entre agentes
PDR	Reglas de prioridad de despacho (*Priority Dispatching Rules*)
SA	Recocido simulado (*Simulated Annealing*)
TS	Búsqueda tabú (*Tabu Search*)
VNS	Búsqueda de vecindad variable (*Variable Neighbourhood Search*)

Bangsow (2012) presenta aplicaciones de la DES en diferentes campos de investigación y brinda una perspectiva de como los tomadores de decisión pueden usar esta herramienta para planear y optimizar. Semini, Fauske, & Strandhagen (2006) presentan una revisión sobre aplicaciones de la DES en la toma de decisiones de lógica de manufactura y Smith (2003) presenta una revisión y clasificación de la literatura sobre el uso de la DES para el diseño de sistemas de manufactura y problemas de operaciones. Macal (2016) presenta los antecedentes y el estado actual de la ABS e identifica diferentes campos de investigación en los cuales se ha empleado indicando literatura relacionada. Leitão (2009) presenta literatura en sistemas de control de manufactura utilizando técnicas de inteligencia artificial distribuida, es decir, sistemas multi-agente (MAS) y Monostori, Váncza, & Kumara (2006) presentan softwares de agentes y sistemas multi-agente y describen sus posibles aplicaciones en manufactura. de fabricación; prestan especial atención a las cuestiones metodológicas y a sistemas industriales desplegados. La SDS no es considerada en este documento ya que, esta técnica se centra en el modelado a nivel agregado y, por lo tanto, no es adecuado para modelar una población heterogénea a nivel individual, en el caso de este trabajo de grado, productos y máquinas. Siebers, MacAl, Garnett, Buxton, & Pidd (2010) y Brailsford (2014) enuncian las principales diferencias entre DES y ABS y los campos de aplicación en investigación en los cuales ambas técnicas tienen mayores ventajas de ejecución. Discuten porque la DES es la técnica de simulación más utilizada en OR y porque la ABS, a pesar de ser utilizada en otros campos como

ciencias sociales o económicas, no ha tenido la misma aceptación y aplicación en OR.

En cuanto a manufactura, la respuesta del sistema debe ser rápida y por ende el tiempo de respuesta es de interés para las dos técnicas discutidas (DES y ABS). La ABS tiene la ventaja de responder rápido a cambios dinámicos a través de decisiones locales. La Tabla 3 presenta las principales ventajas y desventajas de estas técnicas en programación de operaciones.

(g) Objetivo de la reprogramación. Se considera la optimización completa o la reparación de la programación inicial.

(h) Objetivo de programación. En la sección "Problema de programación de operaciones job shop y flexible job shop" se indicaron algunos conceptos relacionados para la definición del objetivo de programación y se presentó literatura para revisar las medidas de desempeño comúnmente utilizadas. Conforme a la revisión realizada, la Tabla 4 resume y enuncia las funciones objetivo utilizadas.

(i) Carácter. Se considera el tipo de investigación realizada. Se indica si el articulo considerado realizó una aplicación de manera teórica o si fue una aplicación a nivel practico.

Tabla 3. Ventajas y desventajas de las técnicas de simulación en la toma de decisión (adaptado de Leusin et al., 2018).

Técnica de simulación	**Ventajas**	**Desventajas**
Simulación de eventos discretos	Fácil y barato de implementar. Útil para responder preguntas tipo "*what-if*".	Difícil para validar. Difícil adaptación a cambios en tiempo real del sistema. Los resultados pueden ser difíciles de interpretar. Depende de sistemas centralizados.
Simulación basada en agentes	Rápida respuesta a cambios en el ambiente de manufactura. Altamente flexible en cambios de las configuraciones del sistema. Modularidad, reconfigurabilidad, adaptabilidad, tolerancia a fallas y características de extensibilidad. Soporta sistemas distribuidos.	Resultados subóptimos Difícil de implementar Altamente dependiente de la comunicación entre agentes.

Tabla 4. Medidas de desempeño utilizadas en problemas de programación.

Medida	**Notación**	**Formula**	**Definición**	**Impacto**
Makespan	C_{max}	$\max_{1 \le j \le n} C_j$	Tiempo necesario para completar todos los trabajos	Minimiza directamente los costos.
Máximo tiempo de flujo	F_j	$\max_{1 \le j \le n} F_j$	El tiempo que un trabajo *j* permanece dentro del piso de manufactura.	Aumenta los costos si el tiempo es mayor.
Tiempo de flujo medio	$\bar{F}$	$\frac{\sum_{j=1}^{n} F_J}{n}$	Tiempo promedio que todos los trabajos pasan en el sistema.	Aumenta los costos si el tiempo es mayor.
Tardanza total	T	$\sum_{j=1}^{n} T_j$	La diferencia positiva entre el tiempo de finalización de un trabajo y el tiempo de entrega.	A usar cuando trabajos retrasados son penalizados.
Tardanza media	$\bar{T}$	$\frac{\sum_{j=1}^{n} T_J}{n}$	El promedio de la tardanza de todos los trabajos.	A usar cuando se estipula un tiempo de entrega para la producción.
Numero de trabajos tardíos	n_T	$\sum_{j=1}^{n} U_j$	Número de trabajos con tardanza.	Afecta directamente los costos y la disponibilidad de máquinas.
Carga total de las máquinas	W_T	$\sum_{j=1}^{n} W_j$	Carga total de trabajo de todas las máquinas.	Asegura una máxima utilización de máquinas.
Consumo de energía	EC_T	$\sum_{j=1}^{n} W_j e_j$	Energía total consumida por todas las máquinas.	Permite realizar la producción minimizando el consumo energético.

Tabla 5. Revisión de la literatura de programación dinámica.

	(a)		(b)		(c)			(d)			(e)		(f)		(g)		(h)									(i)	
(a) Ambiente (b) Perturbaciones (c) Estrategia (d) Política (e) Método (f) Ejecución (g) Objetivo de programación (h) Criterio de optimización (i) Carácter	Job shop	Flexible job shop	Con los recursos	Con el trabajo	Completamente reactiva	Robusta proactiva	Predictiva-reactiva	Continua	Periódica	Basada en eventos	Offline	Online	DES	MAS	Optimización completa	Reparación	C_{max}	F_j	$\bar{F}$	T	$\bar{T}$	n_T	W_T	Consumo de energía	Teórico	Aplicado
(Pfeiffer, Kádár, & Monostori, 2007)	X		X	X			X		X	X	GA	GA	X			X	X								X	X
(Bidot, Vidal, Laborie, & Beck, 2009)		X	X				X			X	MIP	MIP	X		X		X								X	
(Adibi, Zandieh, & Amiri, 2010)	X		X	X		X		-	-	-	VNS + ANN	-	-	-	X		X			X					X	
(Mahdavi, Shirazi, & Solimanpur, 2010)		X		X	X					X	-	PDR	X		-	-	X						X		X	
(Parthanadee & Buddhakulsomsiri, 2010)	X			X	X					X	-	PDR	X		-	-			X	X						X
(Horng, Lin, & Yang, 2012)	X			X		X		-	-	-	DE	-	-	-	X					X					X	
(Sharma & Jain, 2014)	X		X			X		-	-	-	PDR	-	-	-	X		X	X	X	X	X	X			X	
(Y. Zhang, Huang, Sun, & Yang, 2014)	X		X				X			X	GA	GA		X	X	X	X								X	
(Kulkarni & Venkateswaran, 2015)	X			X		X		-	-	-	MIP	-	-	-	X		X								X	
(Ahmadi, Zandieh, Farrokh, & Emami, 2016)		X	X			X		-	-	-	GA	-	-	-	X		X								X	

Tabla 5. Revisión de la literatura de programación dinámica.

	(a)		(b)		(c)			(d)			(e)		(f)		(g)		(h)								(i)	
(a) Ambiente (b) Perturbaciones (c) Estrategia (d) Política (e) Método (f) Ejecución (g) Objetivo de programación (h) Criterio de optimización (i) Carácter	Job shop	Flexible job shop	Con los recursos	Con el trabajo	Completamente reactiva	Robusta proactiva	Predictiva-reactiva	Continua	Periódica	Basada en eventos	Offline	Online	DES	MAS	Optimización completa	Reparación	C_{max}	F_j	$\bar{F}$	T	$\bar{T}$	n_T	W_T	Consumo de energía	Teórico	Aplicado
(Kück, Ehm, Freitag, Frazzon, & Pimentel, 2016)	X		X	X			X			X	HEU	HEU	X			X	No especifica								X	
(Kundakci & Kulak, 2016)	X		X	X		X		-	-	-	GA + TS	-	-	-	X		X								X	
(Sharma & Jain, 2016)	X		X			X		-	-	-	PDR	-	-	-	X				X		X				X	
(Yang, Zeng, Wang, & Sun, 2016)		X		X		X		-	-	-	GA	-	-	-	X		X							X		X
(Baykasoğlu & Karaslan, 2017)	X		X	X	X				X	X	-	GRASP	X			X	X		X		X				X	
(Elgendy et al., 2017)		X	X	X		X		-	-	-	GA	-	-	-		X	X									
(S. Zhang & Wong, 2017)		X	X	X			X		X	X	ACO	ACO		X	X	X	X								X	
(Wang et al., 2017)		X	X	X		X		-	-	-	GA	-	-	-	X		X								X	
(Xiong, Fan, Jiang, & Li, 2017)	X		X			X		-	-	-	PDR	-	-	-	X					X					X	
(Leusin et al., 2018)	X		X	X	X					X	-	PDR		X	-	-	X						X		X	X
(Macchi et al., 2018)	X		X			X		-	-	-	GA	-	-	-	X		X						X			X

Tabla 5. Revisión de la literatura de programación dinámica.

	(a)		(b)		(c)			(d)			(e)		(f)		(g)		(h)								(i)	
(a) Ambiente (b) Perturbaciones (c) Estrategia (d) Política (e) Método (f) Ejecución (g) Objetivo de programación (h) Criterio de optimización (i) Carácter	Job shop	Flexible job shop	Con los recursos	Con el trabajo	Completamente reactiva	Robusta proactiva	Predictiva-reactiva	Continua	Periódica	Basada en eventos	Offline	Online	DES	MAS	Optimización completa	Reparación	C_{max}	F_j	$\bar{F}$	T	$\bar{T}$	n_T	W_T	Consumo de energía	Teórico	Aplicado
(Ortíz, Betancourt, Negrete, De Felice, & Petrillo, 2018)		X	X			X		-	-	-	PDR	-	-	-	X						X				X	X
(Xie & Chen, 2018)		X		X		X		-	-	-	GA	-	-	-	X		X								X	
(Larsen & Pranzo, 2019)	X		X	X			X			X	MIP	MIP	X		X		X								X	X
(Turker et al., 2019)	X		X				X			X	PDR	PDR	X		X						X		X		X	
(Alves et al., 2020)		X	X				X			X	GA	NPA		X		X	X								X	
(Ferreirinha, Santos, Madureira, Varela, & Bastos, 2020)		X	X	X			X		X	X	SA	PDR	X			X			X						X	

La revisión de literatura presentada muestra que la programación dinámica es aún un tema en desarrollo y con oportunidades de investigación. Además, se muestra que la mayoría de los trabajos realizados se enfocan en entornos de manufactura *job shop* y que la estrategia para la toma de decisión más utilizada es la robusta proactiva, con una gran atención en el desarrollo de algoritmos que consideran diferentes eventos aleatorios durante su desarrollo y diferentes medidas de desempeño.

Los trabajos relacionados con estrategias predictivas-reactivas utilizan la misma técnica de solución tanto en la fase *offline* como *online* y, por ende, consideran la misma medida de desempeño en la programación en ambas fases. Más específicamente, los trabajos en esta revisión se centran en métodos de solución sencillos, en los que se consideran las reglas de despacho y métodos heurísticos, y en métodos de solución complejos, en los que se consideran la programación lineal y métodos metaheurísticos. Utilizar la misma técnica de solución implica convertir el problema dinámico, en uno estático tras la ocurrencia de perturbaciones, es decir, en una captura instantánea del estado del sistema, para ser solucionado nuevamente de acuerdo con los cambios presentados. La generación de la nueva solución o reprogramación, requiere de periodos de cálculo computacionales considerables, los cuales varían según la complejidad del sistema estudiado. Como consecuencia, la ejecución de esta nueva solución, en el punto de reprogramación, puede no corresponder con la situación actual del sistema. Adicionalmente, el periodo de cálculo es mayor si la técnica de solución considera más de un solo objetivo.

En el enfoque de métodos de solución sencillos, se utilizan diferentes reglas y técnicas de búsqueda para asignar las operaciones, por ende se consideran las reglas de despacho y los métodos heurísticos. Por ejemplo, Kück et al. (2016) proponen un sistema que genera la programación inicial mediante una heurística de optimización acoplado con un modelo de simulación, el cual se adapta continuamente a los datos en tiempo real del piso de manufactura. En el caso de haber perturbaciones, la heurística de optimización ajusta la programación existente o genera una nueva y actualiza el modelo de simulación. Turker et al. (2019) proponen un DSS diseñado para aumentar el rendimiento de las reglas de despacho en la programación dinámica utilizando datos en tiempo real. Para analizar sus efectos, se ejecutan reglas de despacho seleccionadas de la literatura en un modelo de simulación creado en Arena. Cuando el número de trabajos que esperan en la cola de una máquina toma un valor crítico, el DSS puede cambiar el orden de la programación para alimentar otra máquina lo antes posible. Para ello, encuentra el trabajo con mayor prioridad de acuerdo con la regla de despacho activa y coloca ese trabajo en la primera posición de la cola de la máquina actual o de otra máquina. Ferreirinha et al. (2020) proponen un DSS conectado a un sistema MRP que utiliza recocido simulado para la generación de la solución inicial. El sistema propuesto incluye un módulo de eventos dinámicos que utiliza diferentes reglas de despacho, que ajustan la programación a los cambios identificados y que se ajustan a la medida de desempeño seleccionada.

En el enfoque de métodos de solución complejos, se consideran espacios de solución más amplios, por ende la búsqueda es más exhaustiva y se tiene como objetivo encontrar mejores soluciones que con los métodos de la clasificación anterior. Se han clasificado dentro de este enfoque la programación lineal y los métodos metaheurísticos. Por ejemplo, Pfeiffer et al. (2007) proponen una técnica basada en simulación para la evaluación de métodos de la programación. El modelo de simulación es generado con información de producción. El generador de la programación, un algoritmo genético, calcula el plan de producción a ejecutar en la simulación. Tras la ocurrencia de perturbaciones, se decide reparar la programación o hacer una reprogramación completa con el mismo módulo de programación, es decir, el algoritmo genético. Y. Zhang et al. (2014) proponen un modelo multi-agente para la ejecución y monitoreo de la programación de operaciones. En el modelo se diseñaron cuatro tipos de agentes para cumplir con la programación en tiempo real en el piso de manufactura: máquinas, evaluación de capacidad, programación en tiempo real y monitor de procesos. El agente de programación en tiempo real utiliza un algoritmo genético para la programación y reprogramación de operaciones. Larsen & Pranzo (2019) proponen un marco general de reprogramación para abordar problemas de programación dinámica. La arquitectura propuesta se compone de un controlador, un simulador, un solucionador, un escritor de instancias y un módulo lector de soluciones. Además, el marco debe interactuar tanto con la realidad (mundo real o simulación) y con un supervisor (usuario). Como solucionador, utiliza un modelo de programación de restricciones estándar para el problema clásico *job shop*.

Además de las características analizadas anteriormente, se observó que algunos trabajos solo consideraban programaciones mono objetivo. No obstante, en ambientes de manufactura reales, se consideran diferentes objetivos con el fin de alcanzar los requerimientos internos y externos en la producción (Sharma & Jain, 2014).

En general, la revisión presentada en este trabajo indica que, hay aún una necesidad de un sistema que contribuya en la resolución de problemas de programación de operaciones en entornos *flexible job shop* en tiempo real. Este sistema debe responder las siguientes preguntas. (1) Cuándo: debe ser conocido el momento para ejecutar una reprogramación. (2) Qué: la reprogramación debe reasignar nuevas decisiones a máquinas y trabajos para cumplir el objetivos u objetivos establecidos. (3) Cómo: se deben establecer las técnicas que ejecuten la reprogramación que contribuyan al alcance del objetivo o medida de desempeño establecida.

Considerando estas preguntas, el siguiente capítulo describe el sistema de soporte de decisiones propuesto.

Capítulo 3
Sistema de soporte de decisión de manufactura: programación de operaciones dinámica

Introducción

El capítulo anterior presentó una revisión de la literatura relacionada con programación de operaciones dinámica y presentó conceptos claves para el diseño de programaciones predictivas-reactivas con métodos de optimización. Considerando la contextualización del Capítulo 1 y el marco teórico del Capítulo 2, en este capítulo se propone un sistema de soporte de decisión (DSS) de manufactura, particularmente en programación de operaciones. Este capítulo describe en detalle la composición y configuración del DSS.

La siguiente sección describe las especificaciones del DSS, el cual está basado en la ejecución de una reprogramación parcial de operaciones y en las características y capacidades enunciadas en el Capítulo 1. En la segunda sección se presenta una descripción general del sistema de soporte de decisiones propuesto. Esta descripción continua y se profundiza en las cuatro secciones siguientes, correspondientes a cada uno de los subsistemas que hacen parte del DSS: (1) subsistema de gestión de datos, (2) subsistema de gestión de modelos predictivos, (3) subsistema de gestión de modelos reactivos y (4) subsistema de interfaz de usuario. En la última sección se describe la implementación del DSS en sistemas de manufactura.

Las descripciones presentadas en este capítulo se realizan teniendo en cuenta lo indicado en la sección “Alcances y limitaciones” de este documento. Se presenta la parte funcional del DSS con el fin de realizar una prueba experimental de la propuesta.

Especificaciones del sistema de soporte de decisión

El DSS propuesto está diseñado para guiar la ejecución operacional de la programación de operaciones inicial (*offline scheduling*) y para responder a eventos no planeados en tiempo real (*online scheduling*). El objetivo principal del DSS propuesto es la reparación a la programación inicial, debido a ello, la capacidad principal del sistema es la reactividad. El DSS contiene un mecanismo de reprogramación que detecta e identifica la ocurrencia de perturbaciones, que evalúa que se cumplan las condiciones establecidas en la política de reprogramación, que responde y ejecuta nuevas decisiones generadas en la programación parcial.

La Figura 11 muestra el proceso de reprogramación parcial de operaciones o reparación de la programación en la fase de ejecución y su efecto sobre la medida de desempeño analizada. Se

toma como línea de tiempo la propuesta en la Figura 9. En el piso de manufactura se empieza la ejecución de la programación inicial en el instante 0. Desde ese instante hasta el punto de ocurrencia de una primera perturbación, t_p, el valor esperado de la medida de desempeño analizada se mantiene estable (V_e). A partir del punto t_p, el valor de la medida de desempeño se empieza a desviar con respecto al inicial (degradar) hasta el valor D_{p1}, valor en el cual se inicia el proceso de reprogramación en el punto t_r. El inicio del proceso tiene en cuenta también la política de reprogramación establecida. Durante el tiempo transcurrido hasta el punto t_c, en el cual se tiene una nueva decisión de programación, el valor de la medida de desempeño continúa desviándose hasta el valor D_{c1}. Tras la ejecución de las nuevas decisiones de programación en piso de manufactura, la pendiente de la desviación empieza a disminuir hasta alcanzar el valor D_{f1} en el punto t_{rs}. La desviación no se detiene hasta ese punto (t_{rs}), dado que alguna parte de la programación previa no puede ser cambiada inmediatamente (periodo congelado). Se espera que con el proceso de reprogramación, el valor de la medida de desempeño - aunque degradado - se mantenga estable hasta la ocurrencia de una siguiente perturbación.

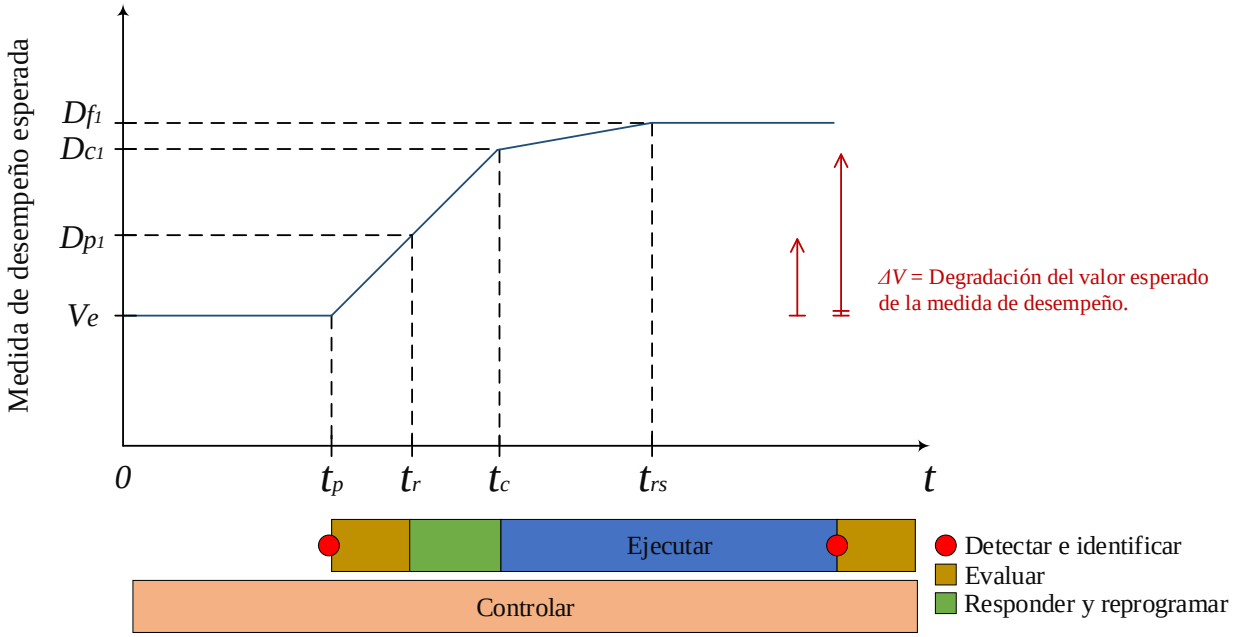

Figura 11. Reparación de la programación en la fase de ejecución.

En la parte inferior de la Figura 11, se muestra también la relación de las etapas del proceso de reprogramación con las etapas del ciclo de toma de decisión indicado en la Figura 2. El punto rojo hace referencia a la detección e identificación de la ocurrencia de una perturbación. Tras la ocurrencia, se evalúa que se cumplan las condiciones establecidas en la política de reprogramación para responder y generar nuevas decisiones de programación. Una vez se establecen las nuevas decisiones de la reprogramación parcial, se ejecutan en el piso de manufactura hasta la ocurrencia de una nueva perturbación, momento en el que el ciclo empieza de nuevo. Durante todo el ciclo se controla la fase de ejecución de la programación.

El DSS propuesto cuenta con las siguientes características y capacidades mostradas en la Figura 12.

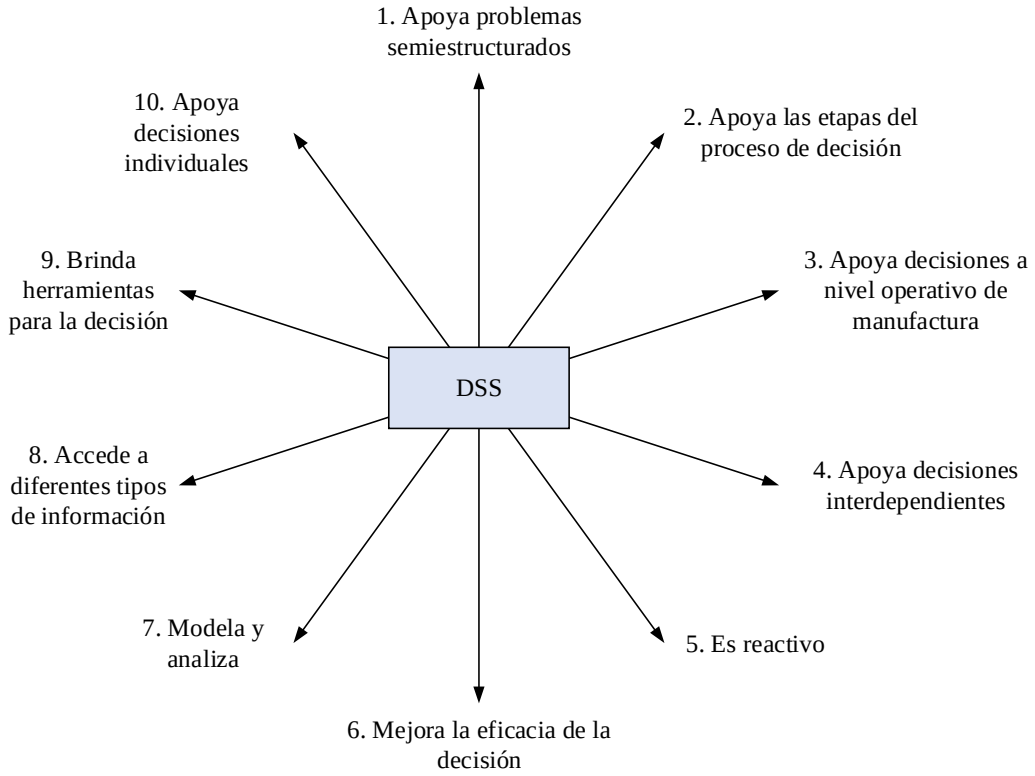

Figura 12. Características y capacidades del DSS.

1. Apoya problemas semiestructurados. La teórica clásica del proceso de toma de decisión distingue tres etapas: inteligencia, diseño y elección; estas pueden ser relacionadas respectivamente con las etapas: identificar, evaluar y responder mostradas en la Figura 2. La programación de operaciones se considera un problema semiestructurado ya que las etapas de identificación y respuesta no son totalmente estructuradas. En la identificación, las condiciones para iniciar una reprogramación no son rutinarias y por ende esta etapa no tiene un proceso estándar. En la respuesta se puede seleccionar un curso de acción de acuerdo con el conocimiento del tomador de decisiones y la información disponible; este proceso tampoco es estándar.
2. Apoya las etapas del proceso de toma de decisiones: inteligencia, diseño y elección.
3. Apoya decisiones en el nivel operativo de la manufactura. La programación de operaciones es una decisión de corto plazo y por ende de nivel operativo.
4. Apoya decisiones interdependientes. Además, dada la naturaleza del programa dinámico y la aleatoriedad de las perturbaciones, las decisiones se realizan repetidamente.

5. Es reactivo y está en capacidad de afrontar situaciones de cambio.
6. Mejora la eficacia de la toma de decisiones (calidad, exactitud, oportuna) en lugar de su eficiencia (costo de tomar una decisión).
7. Utiliza modelos (técnicas metaheurísticas y heurísticas) para analizar las situaciones de toma de decisiones.
8. Accede a diferentes tipos de información. Genera salidas de información tras la ejecución de modelos.
9. Brinda al usuario final herramientas para modificar la programación inicial.
10. Apoya decisiones para un tomador de decisión individual definido como planeador de la producción.

Cada una de estas características se enunciarán a lo largo de este capítulo indicando su función en el DSS.

Descripción del sistema de soporte de decisión

El DSS propuesto es un sistema destinado a apoyar la toma de decisiones a nivel operativo en programación de operaciones de manufactura (característica 3). Es un complemento a las capacidades de los tomadores de decisión (característica 10) para responder a eventos no planeados en tiempo real (característica 5). Está dirigido a decisiones que son apoyadas por algoritmos y modelos de optimización y por ello se centra en los modelos para la toma de decisión (característica 7); el DSS propuesto es orientado a los modelos (Felsberger et al., 2017).

El DSS propuesto se compone de cuatro subsistemas, como se muestra en la Figura 13.

(1) Subsistema de gestión de datos. Gestiona la información requerida para desarrollar la programación inicial de operaciones. Además, se encarga de enviar las decisiones al piso de manufactura en la fase *offline* de la programación. Este subsistema administra los parámetros del piso de manufactura, lo que permite monitorear el estado de los aspectos físicos y de ejecución de la producción. Las salidas de este subsistema son entradas para el subsistema de gestión de modelos y el subsistema de interfaz de usuario.

(2) Subsistema de gestión de modelos predictivos. Gestiona el modelo y la técnica requeridos para la generación de la programación de operaciones inicial. Este subsistema genera la programación inicial a partir de una técnica metaheurística, algoritmo genético, resolviendo los problemas de ruteo y secuenciación del FJSSP, minimizando una medida de desempeño establecida, *makespan*. La programación inicial establece las decisiones predictivas para los trabajos, relacionadas con la

secuencia de entrada al sistema, ruta de procesamiento y tiempo estimado de inicio de cada operación.

(3) Subsistema de gestión de modelos reactivos. Gestiona las modelos y técnicas requeridos para la actualización de la programación de operaciones. El objetivo de los modelos de este subsistema, técnicas heurísticas, es generar decisiones reactivas en la ejecución de la programación en el piso de manufactura que minimicen el efecto de las perturbaciones.

(4) Subsistema de interfaz de usuario. Permite la comunicación entre el usuario y el sistema de soporte de decisiones permitiéndole acceder al subsistema de gestión de datos e interactuar con los subsistemas de gestión de modelos predictivos y reactivos. El componente principal de interacción en la interfaz es el objeto estándar, es decir, botones o menús desplegables. No obstante, la interacción también se puede realizar por medio de lenguaje de programación.

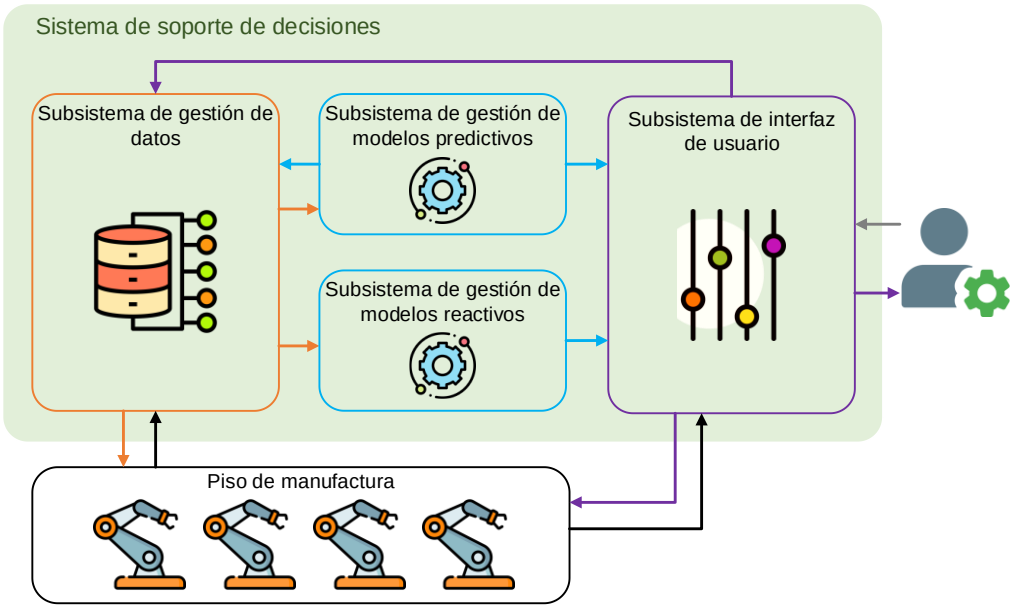

Figura 13. Esquema general del sistema de soporte de decisiones propuesto.

Las siguientes secciones describen los procedimientos y técnicas usadas por los subsistemas del DSS. Se recomienda al lector una lectura apoyada y guiada por la Figura 39.

Subsistema de gestión de datos

El subsistema de gestión de datos se compone de un conjunto de archivos de información, cada uno responsable de una tarea específica dentro del sistema de soporte de decisiones

(característica 8). Esta sección describe la composición y el funcionamiento general de la información manejada y su relación con los otros subsistemas.

La Figura 14 presenta el esquema general del subsistema de gestión de datos y a continuación se describe cada uno de los módulos.

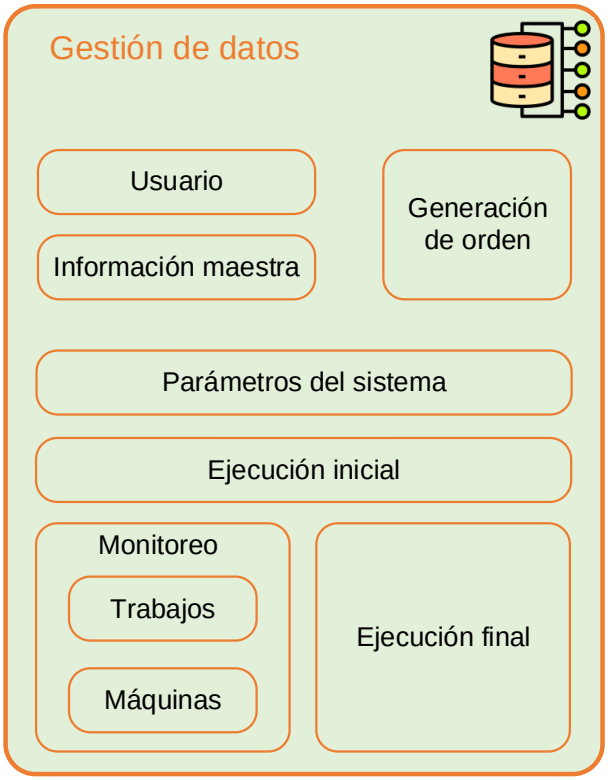

Figura 14. Esquema general del subsistema de gestión de datos.

Módulo de información maestra

Este módulo contiene información del conjunto de posibles trabajos J a ser producidos en el piso de manufactura. La información de cada trabajo j se compone de:

(1) La secuencia de operaciones, I_j, que requiere para ser procesado.

(2) El conjunto de máquinas capaces de realizar la secuencia de operaciones, $R_{ij} \in M$.

(3) El tiempo de procesamiento de cada operación i en cada máquina m, p_{ijm}.

Módulo de usuario

Este módulo toma la información ingresada por el usuario, en el subsistema de interfaz de usuario, relacionada con la cantidad de trabajos (*jobs)* que se van a producir y su correspondiente tiempo de entrega (*due date*).

Módulo de generación de orden

Este módulo genera la orden de producción que será utilizada para generar la programación inicial por el algoritmo genético en el subsistema de gestión de modelos predictivos. Tiene como entradas (1) el módulo de usuario y (2) el módulo de información maestra.

Módulo de parámetros del sistema

Este módulo toma la información del diseño (*layout*) del piso de manufactura (*shop-floor*). Como se indicó en el Capítulo 2, el diseño del piso se considera dado por el grafo $G = (S, V)$ (Figura 6). Este módulo contiene información del conjunto de máquinas $M \subseteq V$ disponibles para ejecutar operaciones, de los segmentos S que configuran el sistema de transporte en la célula de manufactura y de los vértices de punto de giro $PG \subseteq V$ que hacen parte del sistema a con sus respectivas posibles direcciones de transporte o de giro. A partir de la información anterior, este módulo calcula la distancia entre máquinas y consecuentemente, del tiempo de transporte de un trabajo entre máquinas, tt_{m1m2}. Se considera también, la información del número máximo de trabajos, MJ, que pueden ser procesados al tiempo en el piso de manufactura.

Este submódulo es una entrada para el submódulo de algoritmo genético, en el subsistema de gestión de modelos predictivos, ya que indica restricciones adicionales, de existir, que deben ser consideradas durante la ejecución del algoritmo.

Módulo de ejecución inicial

Este módulo tiene como función comunicar y enviar la información de la programación inicial al piso de manufactura para su ejecución (fase *offline*). Este módulo toma la información del submódulo de programación inicial, del subsistema de gestión de modelos predictivos, y codifica las decisiones de producción, asociadas al problema FJSSP (de ruteo y de secuenciación), a enviar a cada uno de los trabajos a ser procesados en el piso de manufactura.

Adicionalmente, en este módulo se deciden las rutas de transporte entre máquinas M, a partir de la información del módulo de parámetros del sistema. El criterio de decisión de transporte es: la ruta más corta entre dos máquinas. Para esto, se consideran todas las posibles rutas que permiten llegar a una máquina M_{i+1} desde una máquina M_i, configuradas por vértices y segmentos, y se selecciona la más corta de las posibles. Lo anterior implica que, las rutas que siguen los productos J para alcanzar las diferentes máquinas M son únicas y fijas durante la ejecución de la programación en el piso de manufactura.

Este módulo envía a los trabajos en el piso de manufactura la siguiente información:

(1) Secuencia de entrada al sistema. Cada trabajo j conoce la posición en fila que le corresponde para entrar a ser procesado en el piso de manufactura.

(2) Ruta de procesamiento. Cada trabajo *j* conoce a que conjunto de máquinas, $M_{ij} \in M$, dirigirse para procesar cada una de sus operaciones I_j una vez entra al sistema.

(3) Ruta de transporte. Cada trabajo *j* conoce la ruta que debe seguir en la célula de manufactura para alcanzar la siguiente máquina *m + 1* desde la máquina *m*. La ruta de transporte implica que el producto *j* conoce sobre que segmentos *S* transportarse y la dirección que debe toma en los vértices de punto de giro *PG.*

(4) Tiempo estimado de inicio de operación. Cada trabajo *j* conoce el tiempo de inicio estimado de la operación *i* en la máquina *m*.

Módulo de monitoreo

Este módulo se encarga de monitorear, de manera continua y a nivel local, los aspectos físicos y de ejecución de la producción en el piso de manufactura (fase *online*) con el fin de detectar e identificar las perturbaciones en el sistema relacionadas con los recursos o con los trabajos (característica 2).

En cuanto al monitoreo de las máquinas, se consideran los siguientes atributos:

(1) Estado. Se monitorean cuatro estados asociados.

(a) Máquina vacía. La máquina *m* no está procesando trabajos y tampoco está esperando uno. Este estado tiene implícito el funcionamiento correcto de la máquina.

(b) Máquina trabajando. La máquina *m* está procesando un trabajo *j*.

(c) Máquina en espera. La máquina *m* aún no tiene el trabajo *j* en procesamiento pero está esperando su llegada. El trabajo *j* ya abandono el sistema de transporte principal para dirigirse a la máquina.

(d) Máquina con fallas. La máquina *m* no tiene permitida la entrada de productos mientras está fallando. Detección e identificación de perturbación asociada al recurso.

(2) Operaciones en capacidad de realizar. La máquina *m* tiene definido un conjunto de operaciones iniciales que está en capacidad de procesar. Se monitora esta información por dos razones. La primera, para informar a los trabajos *j* las operaciones *i* que puede realizar *m*, el caso que el trabajo *j* deba cambiar la ruta de procesamiento inicial y seleccionar una nueva máquina, $R_{ij} \in M$. El trabajo *j* estaría en capacidad de saber cuál máquina *m* le sirve y cuál no. La segunda razón, para conocer si la máquina *m* puede realizar las mismas operaciones *i* durante la ejecución de la programación. Se puede dar el caso que, la máquina sea mejorada y se añada una operación *i* al conjunto de operaciones iniciales o que, por el contrario sufra una falla parcial y una operación *i* sea removida del conjunto de operaciones iniciales durante la ejecución. Detección e identificación de perturbación asociada al recurso.

(3) Trabajo en operación. La máquina *m* conoce que trabajo *j* entra a procesado.

(4) Operación de trabajo. La máquina *m* conoce que operación *i* se va a realizar sobre el trabajo *j* que tiene para procesado.

(5) Tiempo de entrada de trabajo. La máquina *m* conoce en qué momento empieza el procesamiento de la operación *i* a realizar sobre el trabajo *j*.

(6) Tiempo de procesamiento de operación. La máquina *m* conoce el tiempo de procesamiento de la operación *i*, p_{ijm}, establecido en el módulo de información maestra. Esta información permite controlar el tiempo de realización real de la operación y por ende, identificar si hay diferencias en los tiempos establecidos y reales de ejecución. Estas diferencias se pueden dar por mejora de operación de la máquina o por degradación de esta. Detección e identificación de perturbación asociada al recurso.

(7) Ubicación. La máquina *m* tiene asociadas unas coordenadas *(x,y)* definidas por los parámetros del sistema y verificadas por este módulo. Esta información es una entrada para el subsistema de gestión de modelos reactivos.

En cuanto al monitoreo de los trabajos, se consideran los siguientes atributos:

(1) Estado. Se monitorean siete estados asociados.

   (a) Set-up. El trabajo *j* esta fuera del sistema. En línea de espera para entrar.

   (b) En máquina. El trabajo *j* está siendo procesado en una máquina *m.*

   (c) En movimiento. El trabajo *j* se encuentra en el sistema de transporte principal del piso de manufactura.

   (d) En tránsito. El trabajo *j* sale del sistema de transporte principal del piso de manufactura y entra en intercambiadores de sección o en el sistema de transporte directo a máquinas.

   (e) En espera. El trabajo *j* está en capacidad de identificar si hay trabajos *j'* cerca del él. Si este es el caso, el trabajo *j* se detiene hasta que el trabajo *j'* este a una distancia mínima de seguridad con el fin de evitar colisiones de trabajos. Esta distancia es definida como radio de colisión y es un parámetro contenido en el módulo de parámetros del sistema.

   (f) Terminado. El trabajo *j* tiene realizadas todas las operaciones $I_{j.}$ El trabajo está listo para dejar el piso de manufactura pero aún está dentro de este.

   (g) Fuera. El trabajo *j* esta fuera del piso de manufactura una vez ha sido terminado.

(2) Secuencia de operaciones. Esta información es determinada por el módulo de información maestra. Se monitorea esta información para conocer que operaciones *i* del trabajo *j* ya han sido realzadas.

(3) Ruteo de procesamiento. Esta información es determinada por el algoritmo genético y enviada al piso de manufactura a través del submódulo de ejecución inicial. Se monitorea esta información para determinar si el trabajo *j* siguió la ruta predeterminada o cambio de ruta en alguna máquina *m,* tal como se indicó en el monitoreo de máquinas: (2) Operaciones en capacidad de realizar.

(4) Tiempo estimado de inicio de operación. Esta información es determinada por el algoritmo genético y enviada al piso de manufactura a través del submódulo de ejecución inicial. Se monitorea esta información para determinar si hay retraso parcial (L_{ij}) y tardanza parcial (T_{ij}) en las operaciones O_{ij} durante su procesamiento.

En este trabajo se define el retraso parcial, L_{ij}, como, $L_{ij} = tr_{ij} - te_{ij}$, siendo tr_{ij} el tiempo real de inicio de la operación O_{ij} y te_{ij} el tiempo estimado de inicio de la operación O_{ij}. Se define la tardanza parcial, T_{ij}, como $T_{ij} = max\ (L_{ij}, 0)$.

Sin embargo, los indicadores anteriores son mediciones discretas, ya que solo se realizan al inicio de cada operación, por ende, no identifican retrasos y tardanzas inmediatas en la ejecución de las operaciones $O_{ij.}$ Para solucionar la situación descrita, y considerando que en el piso de manufactura se realiza el monitoreo continuamente, se define la tardanza del trabajo, L_j, como $L_j = te - te_{ij}$, siendo *te* el tiempo de ejecución en el piso de manufactura. En el momento en que un trabajo entra a una máquina $te = tr_{ij}$.

Esta información es una entrada para el subsistema de gestión de modelos reactivos, ya que la política de reprogramación determina si, a partir de los indicadores medidos, es necesaria una actualización de la programación (*scheduling update*). Detección e identificación de perturbación asociada al trabajo.

(5) Máquina actual. El trabajo *j* conoce en que máquina *m* está siendo procesado. Esta información es análoga con el monitoreo de máquinas: (3) Trabajo en operación.

(6) Operación actual. El trabajo *j* conoce que operación *i* se está realizando sobre sí. Esta información es análoga con el monitoreo de máquinas: (4) Operación de trabajo.

(7) Siguiente operación. El trabajo *j* conoce operación $i + 1$ siguiente, de acuerdo con su secuencia de operaciones, una vez termine la operación actual *i*. Esta información es una entrada para el subsistema de gestión de modelos reactivos, ya que el método de reprogramación determina si, la actualización de la programación puede dejar este trabajo *j* en la máquina actual *m.* Para esto, la máquina *m* debe estar en capacidad de procesar la operación $i + 1$.

(8) Siguiente máquina. El trabajo *j* conoce la siguiente máquina que lo procesará, $m + 1$, una vez termine la operación actual *i* en la máquina actual *m*.

Esta información es una entrada para el subsistema de gestión de modelos reactivos. En el caso de que se decida cambiar la ruta de procesamiento, este atributo actúa en conjunto con los atributos de monitoreo de máquinas: (2) Operaciones en capacidad

de realizar y con los atributos de monitoreo de trabajos: (3) Ruteo de procesamiento y (7) Siguiente operación.

(9) Ubicación. El trabajo *j* conoce su coordenada asociada *(x,y)*. Esta información es una entrada para el subsistema de gestión de modelos reactivos.

La ejecución (fase *online*) considera el uso de simulación basada en agentes para implementar la reprogramación en caso de perturbaciones. El modelo basado en agentes considera dos tipos de agentes: trabajos y maquinas. Los atributos correspondientes a cada agente son los listados anteriormente.

Módulo de ejecución final

Este módulo toma la información de ejecución en el piso de manufactura tras su finalización. Contiene la información final de la máquina *m* en la que se procesó cada operación *i* del trabajo *j* y el tiempo real de inicio de ejecución de operaciones, tr_{ij}. Así mismo, este módulo se encarga de guardar la información de las medidas de desempeño establecidas tras la ejecución de la programación, L_j, T_j y C_{max}.

La información de este módulo es la misma que el usuario visualiza en tiempo real a través del módulo de interfaz *online*, en el subsistema de interfaz de usuario.

Las interacciones del subsistema de gestión de datos con los otros subsistemas del sistema de soporte de decisiones, anteriormente descritas, se muestran en la Figura 15.

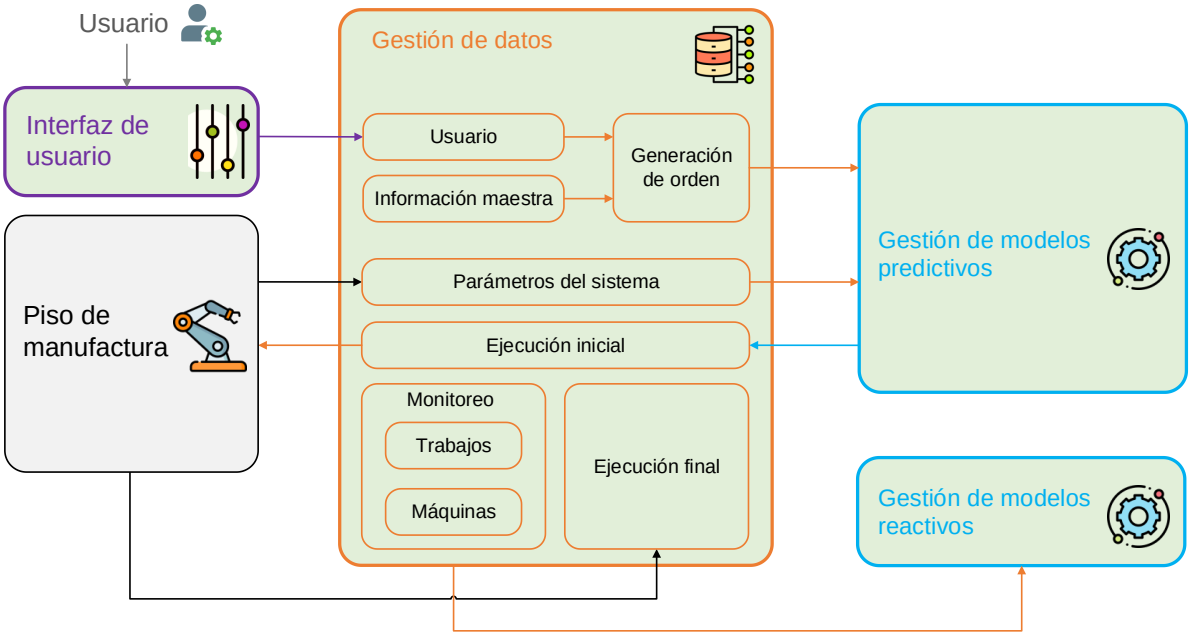

Figura 15. Esquema detallado del subsistema de gestión de datos.

Subsistema de gestión de modelos predictivos

El subsistema de gestión de modelos predictivos se compone de tres módulos, cada uno responsable de una tarea específica, para generar las decisiones predictivas de la programación (característica 7). Este módulo genera la programación de operaciones inicial a ser ejecutada en el piso de manufactura a partir de la información obtenida en el subsistema de gestión de datos. Esta programación tiene como objetivo minimizar la medida de desempeño predeterminada, *makespan* (característica 6).

La Figura 16 presenta el esquema general del subsistema de gestión de modelos predictivos. Esta sección describe el funcionamiento general de cada módulo y su relación con los otros subsistemas.

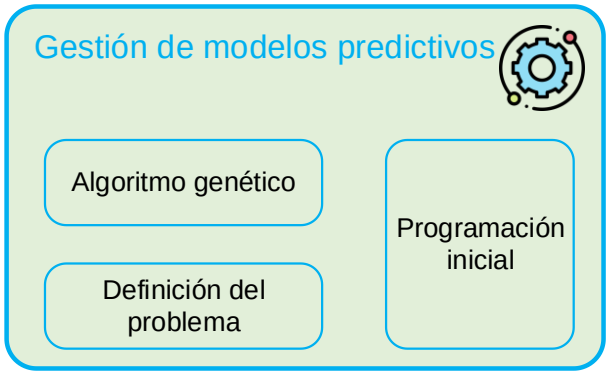

Figura 16. Esquema general del subsistema de gestión de modelos predictivos.

Módulo de definición del problema

En este módulo se define el *flexible job shop scheduling problem* (FJSSP) en términos de la literatura de investigación de operaciones, particularmente en términos de programación lineal entera mixta. En este módulo se asume un comportamiento ideal en la ejecución, es decir, las perturbaciones no son consideradas, puesto que están son consideradas en el subsistema de gestión de modelos reactivos.

El problema descrito en este trabajo de investigación considera la definición clásica del FJSSP, presentado en la primera sección del Capítulo 2 y dos restricciones adicionales, no presentadas en la teoría clásica, a saber: tiempo de transporte entre máquinas y capacidad máxima de procesamiento en el piso de manufactura.

Se muestra a continuación el modelo de programación lineal entera mixta (MILP: *Mixed Integer Linear Program*) de dicho problema.

A. Notación de conjuntos e índices

La notación de conjuntos e índices utiliza la misma nomenclatura utilizada en la sección anterior “Subsistema de gestión de datos”.

J	Conjunto de trabajos, $J = \{1, 2, \ldots, j\}$
M	Conjunto de máquinas, $M = \{1, 2, \ldots, m\}$
I_j	Conjunto de operaciones del trabajo j, $I_j = \{1, 2, \ldots, \lvert I_j \rvert\}$, $j \in J$
O_{ij}	Operación i del trabajo j
R_{ij}	Conjunto de máquinas capaces de realizar la operación O_{ij}, $R_{ij} \in M$

B. Notación de parámetros

p_{ijm}	Tiempo de procesamiento de cada operación O_{ij} en máquina m
tt_{m1m2}	Tiempo de transporte entre máquina m_1 y m_2
MJ	Número máximo de trabajos en la planta
β	Número muy grande
ε	Número muy pequeño

C. Notación de variables

t_{ij}	Tiempo de finalización de la operación O_{ij}
μ_{ijm}	Variable binaria $\begin{cases} 1 \text{ si la operación } O_{ij} \text{ es realizada en la máquina } m \\ 0 \text{ en caso contrario} \end{cases}$
b_{ijkl}	Variable binaria $\begin{cases} 1 \text{ si la operación } O_{ij} \text{ es realizada antes de la operación } O_{kl} \\ 0 \text{ en caso contrario} \end{cases}$
tr_{ijm1m2}	Variable binaria $\begin{cases} 1 \text{ si el trabajo } j \text{ es transportado a la máquina } m_2 \text{ después de realizar la } O_{ij} \\ \text{en máquina } m_1 \\ 0 \text{ en caso contrario} \end{cases}$
z_{lj}	Variable binaria $\begin{cases} 1 \text{ si el trabajo } l \text{ y } j \text{ están al mismo tiempo en el piso de manufactura} \\ 0 \text{ en caso contrario} \end{cases}$

D. Función objetivo

Para evaluar el rendimiento del sistema se ha seleccionado como medida el *makespan*, definido como el tiempo en el cual la última operación del último trabajo en el sistema es finalizada. De acuerdo, con las notaciones expresadas anteriormente, el *makespan* se calcula como se muestra a continuación.

$$C_{max}=max_{\forall i\epsilon I_j \forall j\,\epsilon\, J} t_{ij}$$

Se puede considerar también que, el tiempo de flujo en el sistema del trabajo *j* como C_j. Siendo C_j el tiempo de finalización de la última operación del trabajo *j*. En este caso, el tiempo en el sistema solo considera el tiempo de procesamiento de cada operación en las máquinas que las realizan. Por consiguiente, el *makespan* también se puede expresar como sigue.

$$C_{max}=max_{\forall j\epsilon J} C_j$$

E. Restricciones

Restricciones disyuntivas. Una máquina puede procesar una sola operación al tiempo, y una operación es realizada solo por una máquina.

$$t_{ij} + p_{klm}\mu_{klm} + \beta b_{ijkl} \leq t_{kl} + \beta \qquad \forall i, k\epsilon I,\ \forall j, l\epsilon J,\ \forall m\epsilon R_{ij} \qquad (1)$$

$$b_{ijkl} + b_{klij} \leq 1 \qquad \forall i\epsilon I_j,\ k\epsilon I_l,\ \forall j, l\epsilon J \qquad (2)$$

$$\sum_{m\epsilon R_{ij}} \mu_{ijm} = 1 \qquad \forall i\epsilon I_j,\ \forall j\epsilon J \qquad (3)$$

Restricciones de precedencia. Estas restricciones aseguran la secuencia de producción de los trabajos. El tiempo de finalización de la operación siguiente considera el tiempo de finalización de la operación anterior.

$$t_{(i+1)j} \geq t_{ij} + p_{(i+1)jm} + \sum_{m1,m2\,\epsilon\, M} tt_{m1m2} tr_{ijm1m2} \qquad \forall i\epsilon I_j,\ \forall j\epsilon J,\ \forall m_1,m_2\epsilon R_{ij} \qquad (4)$$

$$\sum_{\substack{m1,m2\epsilon\, M \\ m1\neq m2}} tr_{ijm1m2} \leq 1 \qquad \forall i\epsilon I_j,\ \forall j\epsilon J \qquad (5)$$

Restricciones de transporte. Si operaciones consecutivas se realizan en máquinas diferentes, hay un tiempo de transporte entre estas.

$$\mu_{ijm1} + \mu_{(i+1)jm2} - 1 \leq tr_{ijm1m2} \qquad \begin{array}{l}\forall i\epsilon I_j,\ \forall j\epsilon J \\ \forall m_1,m_2\epsilon R_{ij},\ m_1 \neq m_2\end{array} \qquad (6)$$

$$\mu_{ijm1} + \mu_{(i+1)jm2} \geq (1+\varepsilon)\, tr_{ijm1m2} \qquad \begin{array}{l}\forall i\epsilon I_j,\ \forall j\epsilon J \\ \forall m_1,m_2\epsilon R_{ij},\ m_1 \neq m_2\end{array} \qquad (7)$$

Restricción de límite de trabajos en el sistema. Se limita el número de trabajos simultáneos en el sistema por *MJ*. Para este conjunto de restricciones O_{0j} define la primera operación del trabajo *j* y O_{uj} defina la última operación del trabajo *j*.

$$\sum_{\substack{l \in J \\ l \neq j}} z_{lj} \leq MJ - 1 \qquad \forall j \in J \qquad (8)$$

$$Z_{jl} \geq b_{0luj} + b_{0j0l} - 1 \qquad \forall j, l \in J \qquad (9)$$

$$Z_{jl} \geq b_{0l0j} + b_{ujul} - 1 \qquad \forall j, l \in J \qquad (10)$$

$$Z_{jl} \leq 1 - b_{0l0j} + b_{ujul} \qquad \forall j, l \in J \qquad (11)$$

Restricciones lógicas.

$$t_{ij} \geq p_{ijm} \qquad \forall i \in I_j,\ \forall j \in J \qquad (12)$$

$$\mu_{ijm} \in \{0,1\} \qquad \forall i \in I_j,\ \forall j \in J,\ \forall m \in R_{ij} \qquad (13)$$

$$b_{ijkl} \in \{0,1\} \qquad \forall i \in I_j,\ \forall j \in J,\ \forall k \in I_l,\ \forall l \in J \qquad (14)$$

$$tr_{ijm1m2} \in \{0,1\} \qquad \forall i \in I_j,\ \forall j \in J,\ \forall m_1, m_2 \in R_{ij} \qquad (15)$$

$$z_{lj} \in \{0,1\} \qquad \forall j, l \in J \qquad (16)$$

Este módulo toma la información existente, obtenida del subsistema de gestión de datos, y la traslada a términos matemáticos para ser utilizada por el algoritmo genético. Define de manera matemática la medida de desempeño *makespan* e introduce dos criterios adicionales en la asignación de operaciones en el algoritmo genético, tiempo de transporte entre máquinas y capacidad máxima de procesamiento. Esta última implica que, si el sistema de manufactura no puede ingresar un nuevo producto para procesar, dado que su capacidad esta al máximo, el algoritmo genético asigna operaciones *i* únicamente de los trabajos *j* presentes en el sistema.

En el sistema de soporte de decisiones propuesto, por defecto, se consideran todas las restricciones presentadas en esta subsección. Si el usuario desea no considerar alguna de estas, debe interactuar con el DSS por medio de lenguaje de programación. Finalmente se aclara que, el modelo presentado no se resuelve de forma exacta dada la naturaleza del problema. Es decir, se considera un problema complejo de optimización combinatoria y *NP-hard* en términos de complejidad computacional.

La siguiente subsección presenta la técnica con la que se resuelve el problema enunciado.

Módulo de algoritmo genético

Este módulo ejecuta el algoritmo genético para la generación de la programación inicial tomando como entradas, los módulos del subsistema de gestión de datos, (1) generación de la orden, (2) parámetros del sistema y (3) el módulo de decisión del problema de este subsistema.

A continuación, se describe en detalle el algoritmo genético propuesto para resolver el problema mencionado.

Primero, se describe un ejemplo, que será usado a lo largo de toda la subsección, para guiar el desarrollo del algoritmo. Se considera un FJS *2 x 3*, es decir, un FJS con dos trabajos $J = \{J_1, J_2\}$ y tres máquinas $M = \{M_1, M_2, M_3\}$. Se define además, $J_1 = \{O_{11}, O_{21}, O_{31}\}$ y $J_2 = \{O_{12}, O_{22}, O_{32}\}$. La Tabla 6 y la Tabla 7 muestran los tiempos de procesamiento y de transporte respectivamente en unidades de tiempo.

Tabla 6. Ejemplo de FJSSP – tiempo de procesamiento.

p_{ij}		M_1	M_2	M_3
	O_{11}		4	5
J_1	O_{21}	5		3
	O_{31}	4	5	
	O_{12}			5
J_2	O_{22}	5	3	
	O_{32}		4	3

Tabla 7. Ejemplo de FJSSP – tiempo de transporte entre máquinas

t_{rm1m2}	M_1	M_2	M_3
M_1	0	2	2
M_2	1	0	1
M_3	2	1	0

A. Representación del cromosoma

La representación del cromosoma tiene dos componentes (dos sub-cadenas): (1) selección de máquina *(MS*) y (2) operador de desempate (*TB*). La representación propuesta no requiere de mecanismo de reparación en la sub-cadena de selección de máquina, ya que cada gen contiene solo valores válidos de máquina. En la sub-cadena de operador de desempate tampoco se realiza reparación, si se da el caso que, haya dos operaciones listas para ser asignadas con el mismo valor

de operador de desempate se escoge alguna de manera aleatoria uniforme. La Tabla 8 muestra una posible representación del cromosoma.

Para la sub-cadena de selección de máquina, se usa un subconjunto de máquinas $R_{ij} \subseteq M$, en el que cada valor corresponde a una máquina capaz de realizar la operación O_{ij}. Por ejemplo, la operación O_{11} puede ser realizada en el subconjunto de máquinas $R_{11} = \{M_2, M_3\}$ y la operación O_{32} tiene el subconjunto $R_{32} = \{M_1, M_2\}$. Una máquina es seleccionada del subconjunto de manera aleatoria bajo una distribución uniforme.

Tabla 8. Representación de un cromosoma factible.

O_{11}	O_{21}	O_{31}	O_{12}	O_{22}	O_{32}
Sub-cadena de selección de máquinas					
M_2	M_3	M_2	M_3	M_1	M_2
Sub-cadena de operador de desempate					
3	*6*	*1*	*5*	*2*	*4*

El proceso descrito anteriormente, soluciona el subproblema de ruteo descrito en el Capítulo 2.

Para la sub-cadena de operador de desempate, se usa un conjunto de valores enteros entre *1* y la cantidad total de operaciones a ser realizadas. Por ejemplo, de acuerdo con la información presentada en la Tabla 6 el conjunto fue definido como $TB = \{1,2,3,4,5,6\}$. La codificación para cada cromosoma se da por la reorganización de manera aleatoria de los valores dentro del conjunto, es decir, cada valor (gen) toma una nueva posición en el conjunto. La Tabla 8 presenta un posible conjunto de codificación dado por $TB_1 = \{3,6,1,5,2,4\}$.

De modo que el gen correspondiente a la operación O_{11} tiene un valor *MS* a M_2 y valor *TB* de *3*.

B. Población inicial

La población inicial (*N*) es generada de acuerdo con la codificación descrita en la subsección anterior. En este algoritmo se seleccionaron las máquinas a realizar cada operación de manera aleatoria uniforme. De la misma manera, se reorganizo aleatoriamente el conjunto de valores *TB*.

C. Función de evaluación

Como se mencionó en la sección anterior, el objetivo de la función de evaluación es la minimización de *makespan*. Para ello, se determina la secuencia de las operaciones en las

máquinas. La secuencia es factible si se respetan las restricciones de precedencia entre las operaciones del mismo trabajo, es decir, la operación O_{i+1j} no puede ser procesada antes que la operación O_{ij}. El proceso de secuenciación se describe a continuación y es repetido para cada uno de los cromosomas que hacen parte de la población inicial y de las generaciones generadas.

En el proceso de secuenciación primero se identifican las operaciones *i* que están listas para ser asignadas a las máquinas. Las operaciones listas son las operaciones iniciales ($i = 1$) de cada trabajo, O_{1j}, y las operaciones cuyas operaciones precedentes ya han sido realizadas. En general, habrá al menos dos operaciones listas para ser asignadas (con excepción del momento en el que solo queda un trabajo por asignar), por lo que en este algoritmo define la secuencia por la siguiente regla: se asigna primero la operación con el valor *TB* asociado menor. Segundo, se asigna la operación seleccionada a la máquina correspondiente, valor contenido en la sub-cadena *MS*. Cuando se asigna la operación a la máquina se evalúa si esta está ocupada o no: en caso de que la máquina este ocupada, el tiempo de inicio de operación asignada es inmediatamente después del término de la operación previamente realizada; en caso de que la máquina no esté ocupada, el tiempo de inicio de operación asignada es el tiempo de la llegada a la máquina.

Así mismo, se considera el tiempo de transporte entra máquinas: si la operación asignada es inicial, es decir, O_{1j}, no se considera tiempo de transporte; si la operación asignada no es inicial, se considera el tiempo de transporte entre la máquina en la que se realizó la operación anterior y la máquina correspondiente. Este proceso se realiza hasta que todas las operaciones son asignadas. De manera análoga, durante cada una de las asignaciones de operaciones, se evalúa si la cantidad de trabajos máxima en el sistema es respetada o no. Se considera que un trabajo entra al sistema una vez su operación inicial ($i = 1$) empieza a ser procesada; se considera que un trabajo deja el sistema una vez todas sus operaciones *i* son realizadas. Si el sistema de manufactura está en su capacidad máxima, se espera hasta que un trabajo sea finalizado para dar ingreso al siguiente en la fila, de existir trabajos en espera.

El proceso de secuenciación descrito anteriormente, soluciona el subproblema de secuenciación presentado en el Capítulo 2.

Una vez todas las operaciones son asignadas, se evalúa el tiempo de finalización del último trabajo que deja el sistema, *makespan*. Este valor, es el valor de la función asociado al cromosoma evaluado.

Por ejemplo, en la representación del cromosoma de la Tabla 8, la operación O_{11} y O_{12} están listas para ser asignadas. Dado que el valor del operador de desempate de la operación O_{11} es menor, esta operación es asignada primero, en este caso, a la máquina 2 (asignación 1). Tras la asignación de la operación O_{11}, la operación O_{21} está lista para ser asignada, la operación O_{12} sigue

en la lista. Aplicando el mismo criterio, se asigna la operación O_{12}, en este caso a la máquina M3. El ciclo completo de asignación es mostrado en la Tabla 9.

Tabla 9. Ciclo de asignación de operaciones.

Asignación 1		
Operación	**Máquina**	**Desempate**
O11	*M2*	*3*
O12	*M3*	*5*
Asignación 2		
O12	*M3*	*5*
O21	*M3*	*6*
Asignación 3		
O22	*M1*	*2*
O21	*M3*	*6*
Asignación 4		
O32	*M2*	*4*
O21	*M3*	*6*
Asignación 5		
O21	*M3*	*6*
Asignación 6		
O31	*M2*	*1*

El diagrama de Gantt correspondiente al cromosoma explicado es mostrado en la Figura 17. La secuenciación del trabajo $J = 2$ permite ver la consideración del tiempo de transporte entre máquinas. En este caso, la operación O_{22}, que es realizada en la máquina M_1, inicia dos unidades de tiempo después de la finalización de la operación O_{12}, que es realizada en la máquina M3. Esto es, el tiempo de transporte entre las dos máquinas es de dos unidades como indicado en la Tabla 7. El valor de *makespan* del cromosoma explicado es 23 unidades de tiempo.

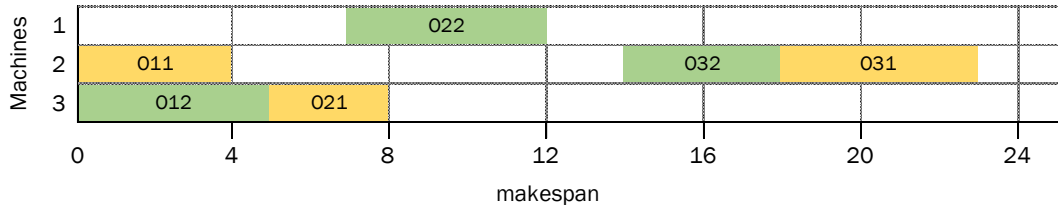

Figura 17. Diagrama de Gantt de un cromosoma factible del algoritmo genético.

La evolución genética preferirá los cromosomas con valores *makespan* bajos, dado que este es el objetivo de la función de evaluación. La siguiente subsección describe la selección de cromosomas para las siguientes generaciones.

D. Selección

La fase de selección se encarga de escoger los cromosomas para la reproducción. En este algoritmo todos los cromosomas de la población inicial y de cada generación son seleccionados para ser parte de la piscina de reproducción. No obstante, se define un criterio de ranking lineal para organizar los cromosomas.

Los cromosomas de la población inicial se organizan de menor a mayor, de acuerdo con su valor obtenido en la función de evaluación. El mejor individuo (el de menor *makespan*) obtiene el ranking *1* y el peor obtiene el ranking *N*. Luego, se selecciona un par de cromosomas de manera aleatoria uniforme y sin remplazo (un cromosoma solo puede ser seleccionado una vez) para la reproducción. La probabilidad de selección de cada cromosoma es la misma dado que se defina una distribución uniforme. Esta selección permite que se escojan individuos con diferentes características brindado mayor variedad en la reproducción. Es decir, se asegura que, por ejemplo, un cromosoma con bajo *makespan* (buen individuo) se seleccione para reproducción con un cromosoma con alto *makespan* (mal individuo) o que dos "buenos individuos" se seleccionen para entrecruzamiento.

Este proceso es repetido hasta seleccionar el último par de cromosomas de la población inicial.

E. Entrecruzamiento / diversificación de la búsqueda

El objetivo de esta fase es mejorar las soluciones por medio del intercambio de información entre los cromosomas seleccionados para reproducción. La operación de entrecruzamiento se realiza en dos pasos. Primero, el par de cromosomas seleccionados en la fase anterior son retomados (padres). Segundo, se realiza el cruce a cada par de cromosomas para obtener dos nuevos individuos (hijos). Se selecciona un operador de entrecruzamiento de múltiples puntos (*c*) y se realiza en ambas sub-cadenas del cromosoma (*MS* y *TB*).

Por ejemplo, la Figura 18 muestra dos posibles cromosomas a partir de la información presentada en la Tabla 6. Los dos cromosomas seleccionados para ser padres son operados con dos puntos de entrecruzamiento, indicados por las flechas, y se intercambia la información de ambas cadenas. El cromosoma 3 (hijo) tiene la información del cromosoma 1 (padre) hasta el primer punto de cruce, luego toma la información del cromosoma 2 (madre) y tras el segundo punto de cruce, toma la información del cromosoma 1.

F. Mutación / profundización de la búsqueda

La mutación introduce cierta variabilidad adicional en la población para mejorar la diversidad y evitar la convergencia a los óptimos locales (H. Chen, Ihlow, & Lehmann, 1999).

La operación de mutación seleccionada en este algoritmo solo afecta la sub-cadena de selección de máquina. Esta operación funciona de la siguiente manera: se asocia un valor aleatorio (uniforme) de probabilidad a cada cromosoma, si este valor es menor a la tasa de mutación (*mr*) seleccionada, se realiza la mutación a dicho cromosoma. Se selecciona un gen del cromosoma a mutar de manera aleatoria uniforme. Después, se selecciona aleatoriamente y uniforme una máquina del subconjunto de máquinas $R_{ij} \subseteq M$, y se reasigna al gen anteriormente seleccionado. Este proceso es repetido para cada cromosoma de cada nueva generación obtenida.

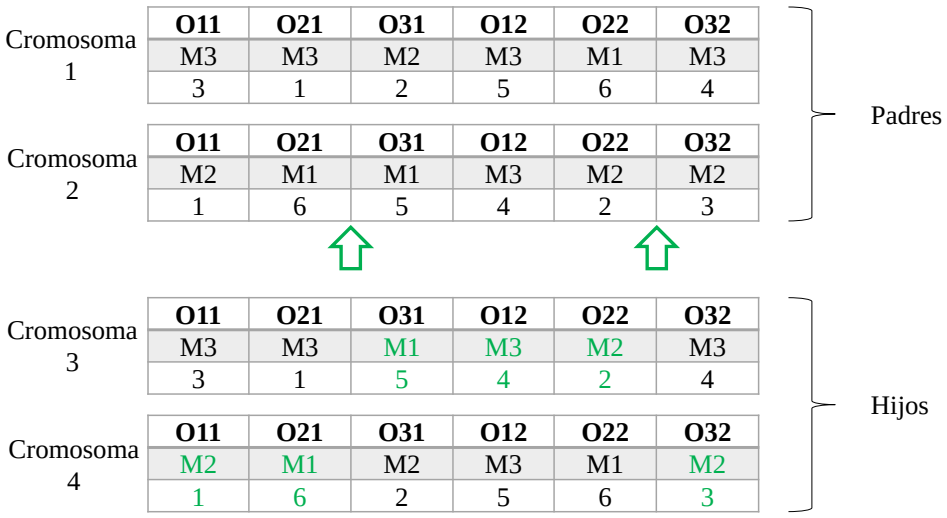

Figura 18. Operación de entrecruzamiento del algoritmo genético.

Por ejemplo, la Figura 19 muestra un cromosoma sobre el que se realiza la operación de mutación. El gen de la operación O_{21} es seleccionado. Antes de la operación de mutación tenía un valor *MS* de M_3. Tras la mutación se asocia un valor *MS* de M_1.

Se debe notar que tras la operación de mutación es posible que valor *MS* sea el mismo, pues la máquina inicial hace parte del subconjunto $R_{ij} \subseteq M$.

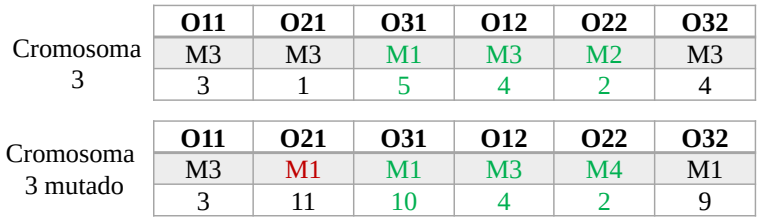

Figura 19. Operación de mutación del algoritmo genético.

G. Remplazo

Después de la producción de nuevos cromosomas (hijos) es necesario decidir qué cromosomas sobreviven en la nueva generación. Se implementó el siguiente procedimiento en este

algoritmo: umbral de aceptación. Se define un punto de corte (*thv*) en la lista generada en la operación de selección (piscina de padres – población inicial). Este punto de corte es un valor porcentual sobre la longitud total de la lista (*N*). Se selecciona el valor de *makespan* en el punto de corte como el umbral de aceptación. Los cromosomas de la piscina de reproducción (hijos) con valor de *makespan* menor al valor de aceptación pasan a la siguiente generación. Para mantener la población total de cromosomas, *N*, pasan a la siguiente generación los cromosomas padres con mejor makespan, hasta completar *N* individuos.

La Figura 20 muestra de manera general la operación de remplazo.

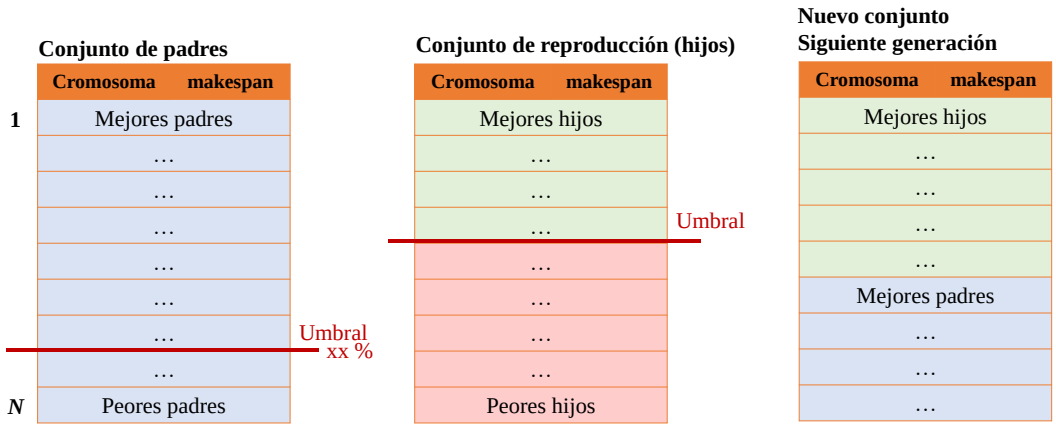

Figura 20. Operación de remplazo del algoritmo genético.

A continuación se resumen los parámetros del algoritmo genético.

- ✓ Tamaño de población (*N*): 1000
- ✓ Número de generaciones (*ng*): 500
- ✓ Puntos de entrecruzamiento (*c*): 5
- ✓ Probabilidad de mutación (*mr*): 25%
- ✓ Umbral de aceptación (*thv*): 90%
- ✓ Criterio de parada: el algoritmo se detiene si la proporción de la diferencia del valor promedio de *makespan* de la población y el mejor valor de *makespan* con el mejor valor de *makespan* es menor al 5%.

$$\frac{\frac{\sum_i^N makespan_i}{N} - \min(makespan_i)}{\min(makespan_i)} < 0{,}05$$

La parametrización del algoritmo genético fue realizada por medio de un diseño de experimentos y se probó con las instancias definidas por Brandimarte (1993). Una descripción detallada se presenta en el apéndice A.

La Figura 21 muestra el diagrama de flujo del algoritmo propuesto en la presente subsección.

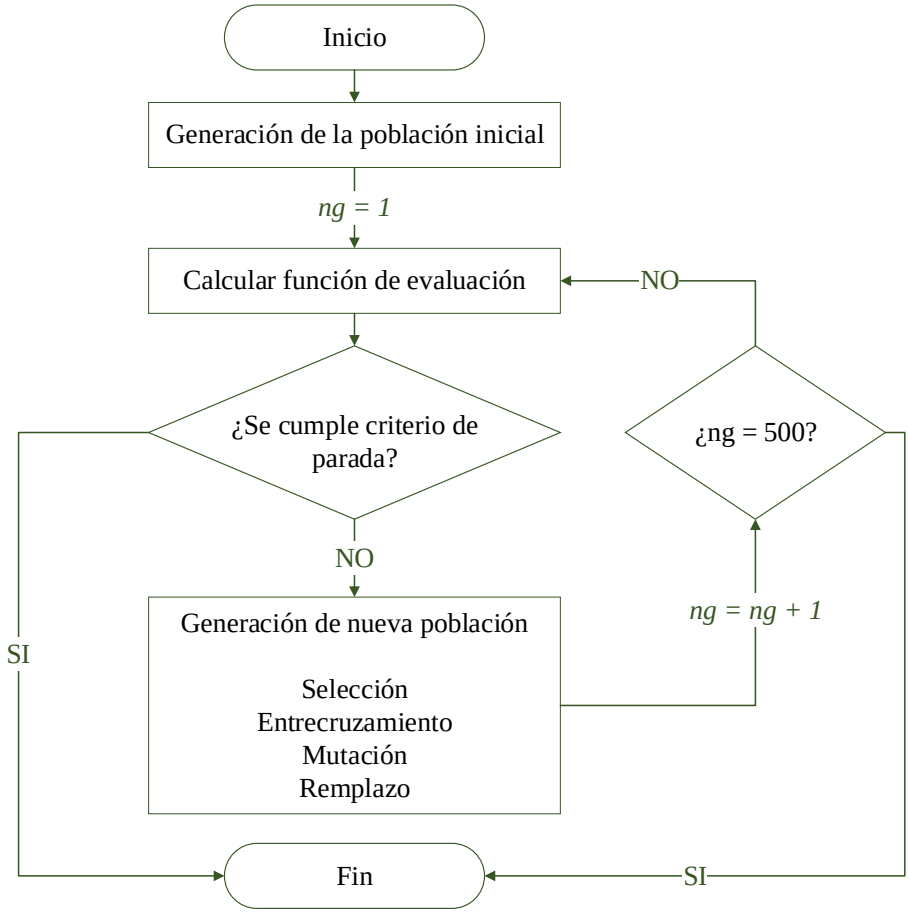

Figura 21. Diagrama de flujo del algoritmo genético.

Módulo de programación inicial

Este submódulo toma de la información de salida del módulo de algoritmo genético y la traduce en información legible tanto para el subsistema de gestión de datos -módulo de ejecución inicial- como para el subsistema de interfaz de usuario -módulo de interfaz *offline*-.

En el módulo de ejecución inicial, se comunican las decisiones (1) secuencia de entrada al sistema, (2) ruta de transporte y (3) tiempo estimado de inicio de operación. Lo anterior, fue indicado y explicado en el módulo de ejecución inicial en la sección anterior.

En el módulo de interfaz *offlin*e se permite interactuar al usuario con las siguientes salidas: (1) información de producción, (2) diagrama de Gantt y (3) rendimiento del algoritmo. Lo anterior, se explicará a mayor detalle en la sección “Subsistema de interfaz de usuario”.

Las interacciones del subsistema de gestión de modelos predictivos con los otros subsistemas del sistema de soporte de decisiones, anteriormente descritas, se muestran en la Figura 22.

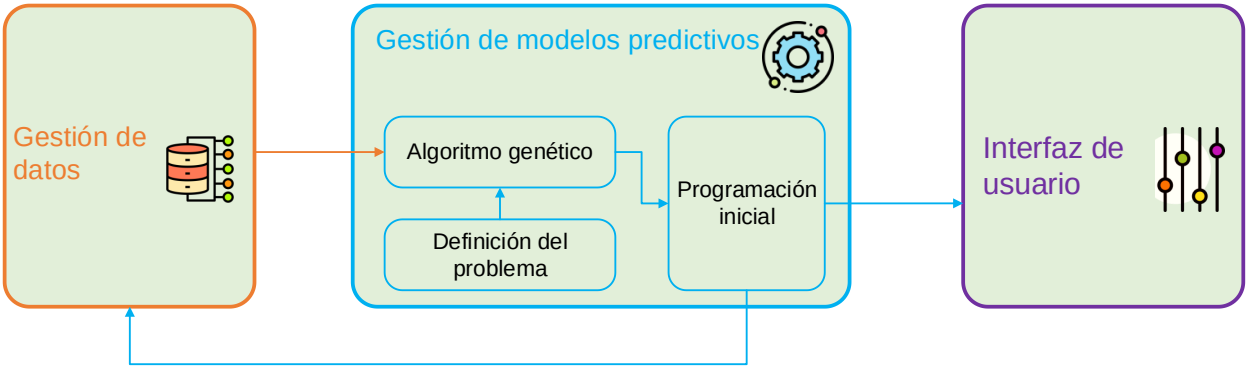

Figura 22. Esquema detallado del subsistema de gestión de modelos predictivos.

Subsistema de gestión de modelos reactivos

El subsistema de gestión de modelos reactivos se compone de dos módulos, cada uno responsable de una tarea específica, para generar las decisiones reactivas (característica 7) tras la ocurrencia de perturbaciones (característica 5) durante la ejecución de la programación de operaciones en el piso de manufactura.

En esta sección se detalla la política de reprogramación establecida, la cual determina el inicio del proceso de reprogramación. Se detallan también los métodos utilizados para la generación de las nuevas decisiones de programación, las cuales tienen como objetivo minimizar dos medidas de desempeño, retraso (*lateness*) y tardanza (*tardiness*) (característica 6).

La Figura 23 presenta el esquema general del subsistema de gestión de modelos reactivos.

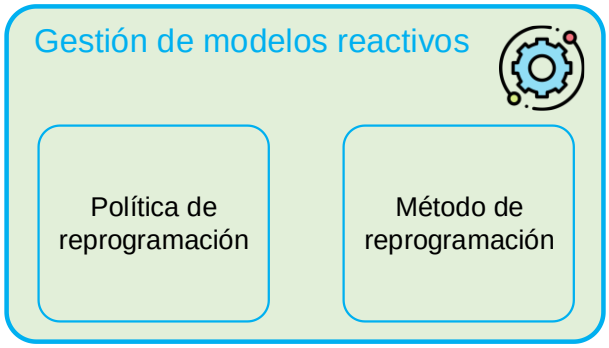

Figura 23. Esquema general del subsistema de gestión de modelos reactivos.

Módulo de política de reprogramación

Este módulo tiene como función principal definir el inicio del proceso de reprogramación. Para ello, se define el objetivo de la reprogramación y después, se define el periodo de reacción. En este módulo también se presentan los aspectos de la política definida. Esté módulo responde a la pregunta (1) Cuándo: debe ser conocido el momento para ejecutar una reprogramación, planteada en la última sección del Capítulo 2.

Dado que el objetivo del DSS es ejecutar una reprogramación parcial de operaciones y considerando las definiciones presentadas en el Capítulo 2, se debe seleccionar una medida de desempeño diferente a la usada en la fase *offline*, es decir, una medida diferente al *makespan*. Las medidas de desempeño sustitutas seleccionadas para la reprogramación son el retraso (*lateness*) y la tardanza (*tardiness*). Estas medidas permiten identificar si hay una degradación del valor esperado de la medida de desempeño durante la ejecución de la programación en el piso de manufactura, en consecuencia, permiten identificar una perturbación en tiempo real. La degradación mencionada se mide sobre las medidas de desempeño seleccionadas en este módulo, porque una medición sobre el valor esperado del *makespan* solo es posible hacerla una vez la ejecución de la programación ha terminado. Esto conllevaría a no tener capacidad de reactividad en tiempo real, pues solo se identificaría una perturbación hasta finalizada la ejecución de la programación y en ese punto ya no es posible cambiar alguna decisión de la programación.

El objetivo de la reprogramación es la minimización del retraso total y medio y de la tardanza total y media tras la ocurrencia de perturbaciones en el piso de manufactura. En este trabajo se investigación, las medidas indicadas se definen como se muestra en la Tabla 10.

El módulo de monitoreo, del subsistema de gestión de datos, calcula la degradación de las medidas en tiempo real, denominadas como retraso parcial y tardanza parcial, L_{ij} y T_{ij}, respectivamente. Cuando se realiza la última operación i de cada trabajo j, $i = |I_j|$, se tiene el caso particular, $L_{|Ij|j} = L_j$ y $T_{|Ij|j} = T_j$.

Tabla 10. Medidas de desempeño sustitutas para la reprogramación de operaciones.

Medida	**Notación**	**Formula**
Retraso total	L	$\sum_{j=1}^{n} L_j$
Retraso medio	$\bar{L}$	$\frac{\sum_{j=1}^{n} L_J}{n}$
Tardanza total	T	$\sum_{j=1}^{n} T_j$
Tardanza media	$\bar{T}$	$\frac{\sum_{j=1}^{n} T_J}{n}$

En general, los valores L_{ij} y T_{ij} son iguales a cero mientras no se presenten perturbaciones en el sistema de manufactura, esto es, las operaciones *i* del conjunto de trabajos *J* se realizan de acuerdo con los tiempos estimados de inicio de operación puesto que no hay eventos aleatorios no planeados que interfieran en el comportamiento ideal de ejecución. En particular, el valor L_{ij} puede ser negativo, lo que indica que se tuvo un inicio de operación *i* antes de lo esperado; y puede ser positivo, lo que significa que se tuvo un inicio de operación *i* después de lo esperado. El valor T_{ij}, por definición, no toma valores negativos; y valores mayores a cero suponen de la misma manera, un inicio de operación *i* después de lo esperado.

Dado que el objetivo de la reprogramación es minimizar el valor de las medidas de desempeño, en esta tesis se considera que un periodo de reacción más corto se alinea dicho objetivo. Por lo anterior, se define que, si durante la ejecución en el piso de manufactura, en el instante $t = te$ (tiempo de ejecución en el piso de manufactura), un trabajo *j* con conjunto de operaciones I_j y valores asociados L_{ij} y T_{ij}, tiene valor asociado $L_{ij} > 0$ antes de la realización de la siguiente operación $i \epsilon I_j$, es considerado por el módulo de método de reprogramación para generar una nueva decisión de programación (característica 2). En consecuencia, un trabajo *j* identifica una perturbación en el sistema cuando su valor $L_{ij} > 0$.

A continuación, se presentan los aspectos de la política de reprogramación, a partir de lo enunciado anteriormente.

(1) Punto de reprogramación. Dado que el monitoreo se realiza de manera continua y que durante cualquier momento de la ejecución un trabajo *j* puede tener un valor asociado $L_{ij} > 0$, no hay un punto de reprogramación definido. Una decisión de reprogramación puede ser ejecutada en cualquier momento *t* durante el periodo de ejecución (tiempo de ejecución) *te*.

(2) Periodo de reacción. En la política definida, se considera la perturbación que identifica el trabajo *j*, es decir $L_{ij} > 0$, para la toma reactiva de decisiones. Lo anterior

implica que, aunque dentro del sistema de manufactura se identifique una perturbación (fallo de máquina, falta de material), no se ejecutará el módulo de reprogramación en función de esta, sino de la identificación que le da el trabajo *j* a dicha perturbación, cuando su valor de retraso es mayor a cero. (Respuesta a la pregunta (1) Cuándo).

El periodo de reacción, para el trabajo, es inmediato.

(3) Periodo congelado. Se consideran las operaciones I_j de los trabajos *J* que están en ejecución o que tienen inicio inminente (en camino a ser procesadas por una máquina).

(4) Periodo de reprogramación. Continuo. No hay un punto de reprogramación definido.

(5) Objetivo de reprogramación. Minimización del retraso total y medio y de la tardanza total y media.

En la siguiente subsección se describen las técnicas de reprogramación propuestos para la generación de decisiones de reprogramación.

Módulo de método de reprogramación

Este módulo tiene como función principal definir las técnicas usadas para la toma de decisiones reactivas en la ejecución de la programación en el piso de manufactura. Para ello, se definen primero las posibles decisiones que tomar y sus alternativas y después se indica el alcance del DSS propuesto.

Esté módulo responde a las preguntas (2) Qué: la reprogramación debe reasignar nuevas decisiones a máquinas y trabajos para cumplir el objetivos u objetivos establecidos y (3) Cómo: se deben establecer las técnicas que ejecuten la reprogramación que contribuyan al alcance del objetivo o medida de desempeño establecida, planteadas en la última sección del Capítulo 2.

En la Tabla 11 se resumen las decisiones, que en este trabajo de investigación se han considerado tomar, para un entorno de manufactura *flexible job shop*.

El DSS propuesto genera decisiones de reprogramación en la decisión siguiente máquina.

Cuando se identifica un trabajo con valor asociado $L_{ij} > 0$, la decisión de reprogramación asigna un nuevo valor al atributo siguiente máquina y por ende hay un cambio en la ruta de procesamiento. La decisión de ruta de transporte también es afectada, dado que el producto debe dirigirse a una nueva máquina $m + 1$, esto implica que tanto los segmentos *S* definidos para transportarse como las direcciones establecidas en los vértices de punto de giro *PG* cambian. Sin embargo, la decisión de ruta de transporte no es tomada por este módulo, ya que, como se indicó

en el módulo de ejecución inicial, las rutas para alcanzar las diferentes máquinas del conjunto *M* son únicas y fijas, lo que significa que están previamente establecidas.

Tabla 11. Decisiones asociadas al problema *flexible job shop*.

Decisión	**Definición**
Secuencia de entrada al sistema	Cada trabajo *j* conoce la posición en fila que le corresponde para entrar a ser procesado en el piso de manufactura.
Ruta de procesamiento	Cada trabajo *j* conoce a que conjunto de máquinas, $M_{ij} \in M$, dirigirse para procesar su conjunto de operaciones I_j una vez entra al sistema.
Siguiente máquina	Cada trabajo *j* conoce la siguiente máquina que lo procesará, $m + 1$, una vez termine la operación actual *i* en la máquina actual *m.*
Ruta de transporte	Cada trabajo *j* conoce la ruta que debe seguir en el piso de manufactura para alcanzar la siguiente máquina $m + 1$ desde la máquina *m.* La ruta de transporte implica que el producto *j* conoce sobre que segmentos *S* transportarse y la dirección que debe toma en los vértices de punto de giro *PG*.

A continuación se describe en detalle las técnicas heurísticas propuestas para la reprogramación.

Primera máquina disponible

La técnica heurística primera máquina disponible (FAM: *First Available Machine*) asigna una nueva máquina al conjunto de trabajos $j_t \in J$ cuyo valor asociado L_{ij} es mayor a cero (condición de reprogramación). Bajo esta técnica, la asignación se realiza en los vértices de punto de giro *PG* (punto físico de reprogramación).

Para la asignación de una nueva máquina (decisión de reprogramación) al conjunto de trabajos $j_t \in J$ que cumple con la condición de reprogramación, se utiliza la información de atributos mostrada en la Tabla 12 que se obtiene a través del módulo de monitoreo.

Tabla 12. Información necesaria para la técnica FAM.

Información obtenida del módulo de monitoreo	
De las máquinas.	De los trabajos.
(1) Estado.	(7) Siguiente operación.
(2) Operaciones en capacidad de realizar.	(8) Siguiente máquina.
(7) Ubicación.	(9) Ubicación.

La técnica heurística FAM se ejecuta individualmente sobre cada uno de los trabajos *j* que cumplan la condición de reprogramación y se encuentren en los puntos físicos de reprogramación. La descripción en detalle de la técnica se presenta a continuación mostrando el proceso que sigue cada uno de los trabajos que cumplen con las condiciones anteriormente mencionadas.

El proceso de reprogramación, conducido por la técnica FAM, inicia identificando si un trabajo *j* tiene un valor asociado $L_{ij} > 0$. Si el trabajo *j* no tiene retraso parcial positivo, no se considera para ejecutar una decisión de reprogramación sobre él. En caso, contrario, si el trabajo *j* tiene retraso parcial positivo, se considera para asignar una nueva máquina.

Una vez identificados los trabajos que cumplen la condición de reprogramación, se procede a identificar la categoría del nodo (vértice) sobre el cual se encuentra el trabajo *j*, ya que, como se indicó antes, el punto físico de reprogramación es un vértice de giro *PG*. Los vértices de punto de giro *PG* se pueden clasificar en dos categorías: la primera, puntos de giro, cuyas direcciones de giro asociadas, al menos una, puedan llevar al trabajo *j* directamente a una estación de trabajo (una máquina); estos se denotan por $EM \subseteq PG$. La segunda, puntos de giro, cuyas direcciones de giro asociadas, todas, dejen al trabajo *j* dentro del conjunto de segmentos *S* que configuran el sistema de transporte principal en la célula de manufactura; estos se denotan por $TN \subseteq PG$. Tomando como ejemplo la Figura 6, el vértice 1 (*v1*) es un vértice *EM*, pues sus direcciones de giro conducen a un trabajo *j* directamente a la máquina *M1*. El vértice 7 (*v7*) es un vértice *TN* pues sus direcciones de giro no conducen a un trabajo *j* directamente a una máquina, es un nodo transitorio en el sistema de transporte. El vértice 3 (*v3*) también se considera *EM*, pues una de sus direcciones de giro lleva a un trabajo *j* a la máquina *M4*.

Si el trabajo *j* se encuentra en un nodo *EM* se evalúa si la máquina *m*, a la cual dirige ese nodo, está en capacidad de procesar la operación *i* que el trabajo *j* esta por empezar. Esta información se obtiene directamente del parámetro del trabajo (7) siguiente operación y se evalúa con el parámetro de máquina (2) operaciones en capacidad de realizar. Si la máquina *m* no está en capacidad de procesar la operación *i = (7) siguiente operación*, el proceso se da por terminado, es decir, no se ejecuta ninguna decisión de reprogramación. En caso contrario, se evalúa si la máquina *m* está vacía, información obtenida del parámetro de maquina (1) estado. Si la máquina *m* está vacía, se ejecuta la decisión de reprogramación, es decir, se asigna una nueva máquina al trabajo *j*. La técnica FAM asigna el valor de la máquina evaluada *m* al parámetro del trabajo (8) siguiente máquina. Si la máquina *m* no está vacía, se realiza una última evaluación, a través del parámetro del trabajo (8) siguiente máquina. Se evalúa cuantos de los otros trabajos $j', j' \neq j$, tienen a la máquina *m* en el parámetro (8) siguiente máquina, *m = siguiente máquina*, es decir, se contabilizan los trabajos *j'* que tienen como siguiente máquina, la misma máquina *m* a la cual puede llegar el trabajo *j* en el vértice de punto de giro *EM*. Si hay más de un trabajo *j'* con destino siguiente máquina *m*, no se ejecuta ninguna decisión de reprogramación sobre el trabajo *j*, lo que significa que, el trabajo *j* sigue con la misma maquina m_i con la que inició el proceso de reprogramación. Si

hay un trabajo o menos con destino siguiente máquina *m*, se ejecuta la decisión de reprogramación de la técnica FAM. En este trabajo de investigación se fija un valor de máquinas *j'* con parámetro siguiente maquina *m* en uno, como se explicó en la evaluación anterior, de modo que, el trabajo *j* encuentre como máximo un trabajo por delante al dirigirse a la máquina *m*. Lo anterior pretende no añadir más trabajos a la cola de una maquina *m*, que los determinados inicialmente por el algoritmo genético.

Si el trabajo *j* se encuentra en un vértice de punto de giro *TN*, se evalúa el conjunto de máquinas más cercanas, *Mf*, desde la ubicación del trabajo *j*. Lo anterior se logra evaluando los parámetros del trabajo y de máquina (9) ubicación y (7) ubicación respectivamente. Para la técnica FAM se define $|Mf| = 2$ y parámetro del trabajo *j siguiente máquina* $\notin Mf$. Esto quiere decir que, se consideran solo las dos máquinas más cercanas desde el nodo *TN* y que, la siguiente maquina (m_s) del trabajo *j* no se considera dentro del conjunto *Mf*. Esto último, porque al considerar la máquina que el trabajo ya tiene determinada en su ruta, m_s, se podrir seleccionar $m = m_s$, lo que no sería en la práctica ninguna reprogramación, pues se asignaría la misma máquina que se tenía inicialmente programada. Una vez determinado el conjunto *Mf*, se selecciona la primera máquina $mf \in Mf$, y se asigna $m = mf$. Seleccionada la maquina *m*, se realiza la misma evaluación que en los nodos *EM*, se evalúa si la máquina *m*, está en capacidad de procesar la operación *i* que el trabajo *j* este por empezar. Si la máquina *m* está en capacidad de procesar la operación *i = (7) siguiente operación* se sigue con el mismo procedimiento descrito para los nodos *EM*. En caso contrario, se evalúan la siguiente máquina *mf* del conjunto *Mf*. Este proceso se repite hasta que una maquina $mf = m$ puede realizar la operación requerida o se hayan evaluado todas las máquinas del conjunto *Mf*. Si esto último es el caso, no se ejecuta ninguna decisión de reprogramación y se da por terminado el proceso.

En general, este proceso se realiza repetidamente si se cumplen la condición de reprogramación y el trabajo *j* se encuentra en el punto físico de reprogramación. Además, una decisión de reprogramación tomada en un vértice, tiene una influencia sobre una siguiente decisión tomada en otro vértice, por ende, se consideran decisiones interdependientes (característica 4).

La Figura 24 muestra el diagrama de flujo, del procedimiento anteriormente descrito, de la técnica FAM.

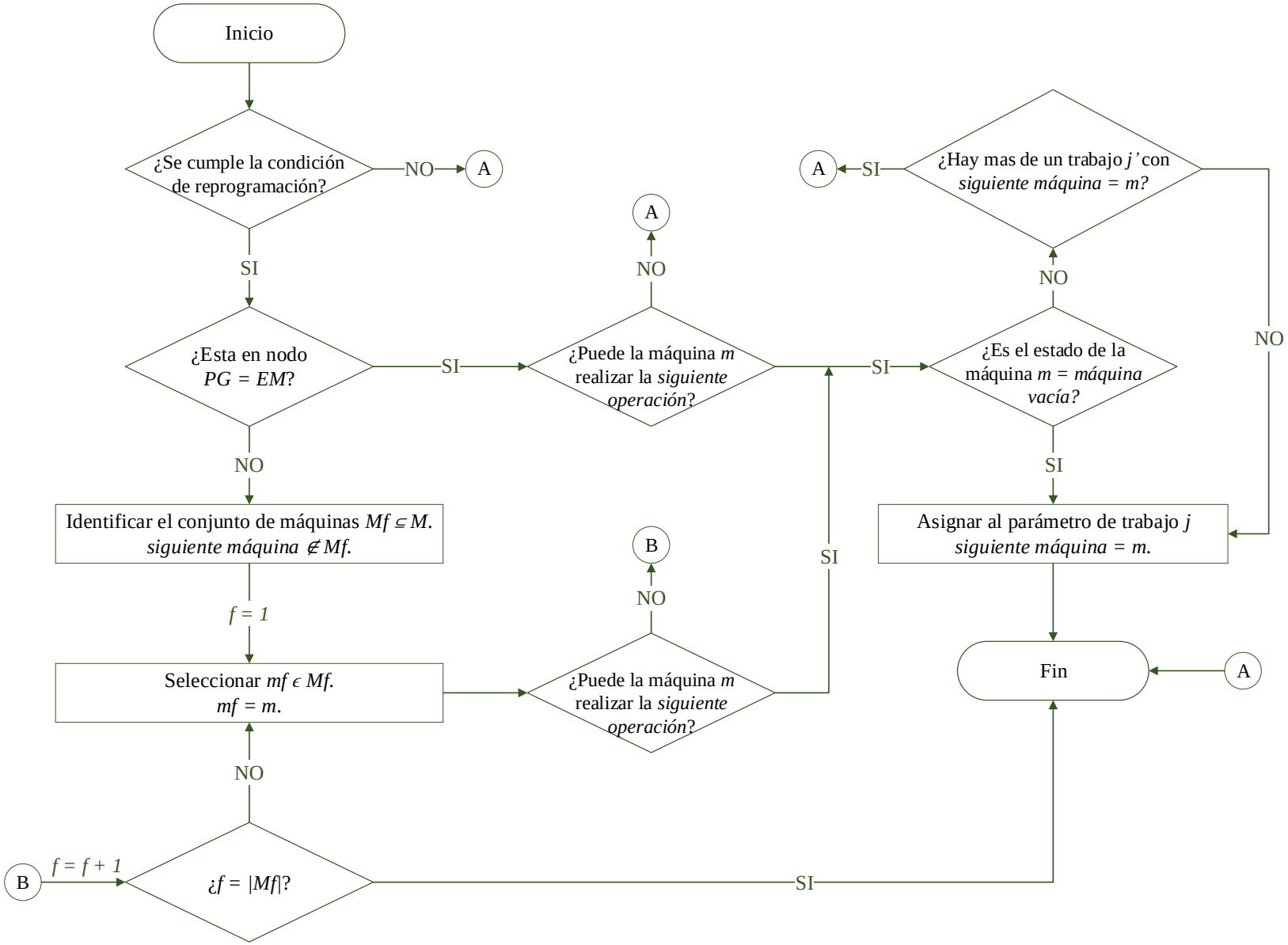

Figura 24. Diagrama de flujo de la técnica FAM.

Permanecer en la máquina actual

La técnica heurística permanecer en la máquina actual (SCM: *Stay in Current Machine*) asigna una nueva máquina al conjunto de trabajos $j_t \epsilon J$ cuyo valor asociado L_{ij} es mayor a cero (condición de reprogramación). Bajo esta técnica, la asignación se realiza en los vértices de máquinas *M* (punto físico de reprogramación).

Para la asignación de una nueva máquina (decisión de reprogramación) al conjunto de trabajos $j_t \epsilon J$ que cumple con la condición de reprogramación, se utiliza la información de atributos mostrada en la Figura 13 que se obtiene a través del módulo de monitoreo.

Tabla 13. Información necesaria para la técnica SCM.

Información obtenida del módulo de monitoreo	
De las máquinas.	De los trabajos.
(1) Estado.	(7) Siguiente operación.
(2) Operaciones en capacidad de realizar.	(8) Siguiente máquina.

La técnica heurística SCM se ejecuta individualmente sobre cada uno de los trabajos *j* que cumplan la condición de reprogramación y se encuentren en los puntos físicos de reprogramación. La descripción en detalle de la técnica se presenta a continuación mostrando el proceso que sigue cada uno de los trabajos que cumplen con las condiciones anteriormente mencionadas.

El proceso de reprogramación, conducido por la técnica SCM, inicia identificando si un trabajo *j* tiene un valor asociado *Lij > 0*. Si el trabajo *j* no tiene retraso parcial positivo, no se considera para ejecutar una decisión de reprogramación sobre él. En caso, contrario, si el trabajo *j* tiene retraso parcial positivo, se considera para asignar una nueva máquina.

Una vez identificados los trabajos que cumplen la condición de reprogramación, se procede a identificar si el nodo sobre el cual se encuentra el trabajo *j* es un vértice *M*, ya que, como se indicó antes, este es el punto físico de reprogramación. Si el trabajo *j* se encuentra en un nodo *M*, se evalúa si la máquina *m*, en la cual esta, está en capacidad de procesar la operación *i* que el trabajo *j* esta por empezar. Esta información se obtiene directamente del parámetro del trabajo (7) siguiente operación y se evalúa con el parámetro de máquina (2) operaciones en capacidad de realizar. Si la máquina *m* no está en capacidad de procesar la operación *i = (7) siguiente operación*, el proceso se da por terminado, es decir, no se ejecuta ninguna decisión de reprogramación. En caso contrario, se ejecuta la decisión de reprogramación, es decir, se asigna una nueva máquina al trabajo *j*. La técnica SCM asigna el valor de la máquina evaluada *m* al parámetro del trabajo (8) siguiente máquina. Adicionalmente, cambia el estado de la máquina *m* a máquina trabajando. Esta

última asignación se hace sobre la máquina *m* y es necesaria para que el sistema de manufactura tenga conocimiento de que la máquina *m* sigue procesando un trabajo *j*.

Si el trabajo *j* no se encuentra en un nodo *M* se termina el proceso de reprogramación.

Al igual que con la técnica FAM, la técnica SCM realiza decisiones interdependientes y que son repetitivas (característica 4).

La Figura 25 muestra el diagrama de flujo, del procedimiento anteriormente descrito, de la técnica SCM.

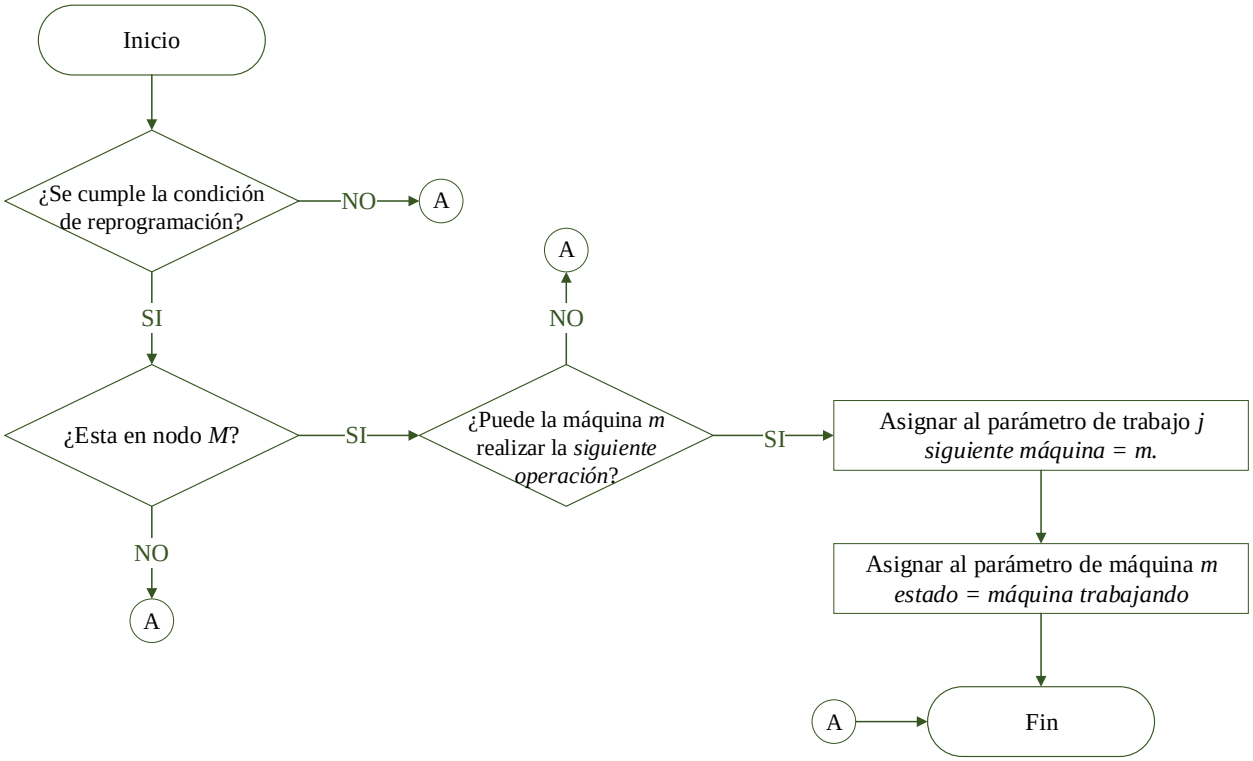

Figura 25. Diagrama de flujo de la técnica SCM.

En la siguiente sección se indica que el usuario puede seleccionar la ejecución de las dos técnicas, FAM y SCM, al mismo tiempo para reaccionar a una perturbación en el sistema de manufactura. Esto es posible ya que ambas técnicas tienen puntos físicos de reprogramación diferentes, por ende, no se interpone una técnica sobre la otra.

Las interacciones del subsistema de gestión de modelos reactivos con los otros subsistemas del sistema de soporte de decisiones, anteriormente descritas, se muestran en la Figura 26.

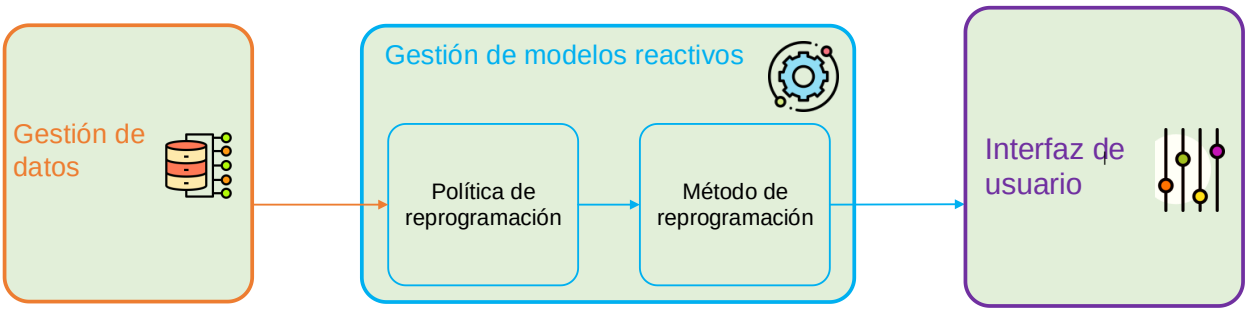

Figura 26. Esquema detallado del subsistema de gestión de modelos reactivos.

Subsistema de interfaz de usuario

El subsistema de interfaz de usuario permite la interacción y comunicación ente el usuario con el sistema de soporte de decisiones y el piso de manufactura. Esta sección describe el funcionamiento general del subsistema y su relación con los otros subsistemas. La Figura 27 presenta el esquema general del subsistema de gestión de datos.

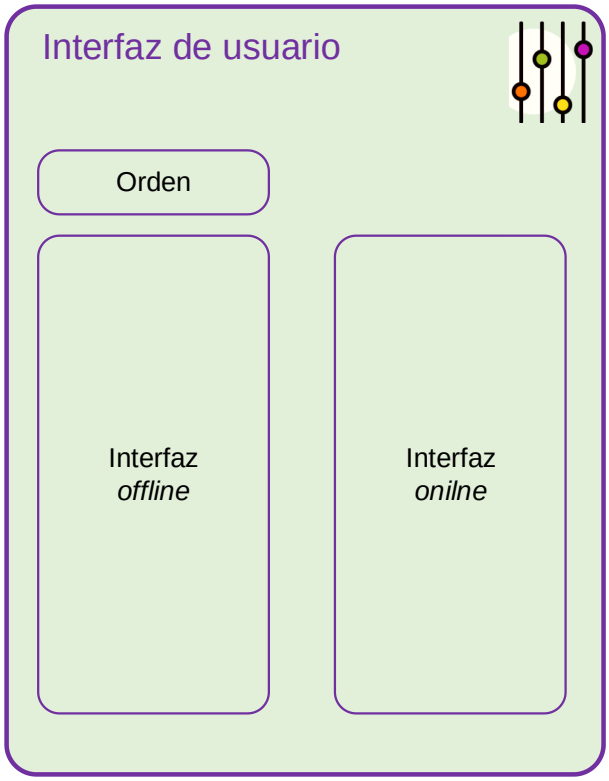

Figura 27. Esquema general del subsistema de interfaz de usuario.

La interfaz del usuario, propuesta en este trabajo de investigación, se realizó en el software de simulación basada en agentes NetLogo (Wilensky & Evanston, 1999). El componente principal

de interacción en la interfaz es el botón, el cual es un objeto estándar. También se usan otros objetos como el control deslizante y el menú desplegable. Además, la interacción también se puede realizar por medio de lenguaje de programación. La Figura 28 muestra la interfaz de usuario principal propuesta, la cual mantiene la misma organización del esquema general.

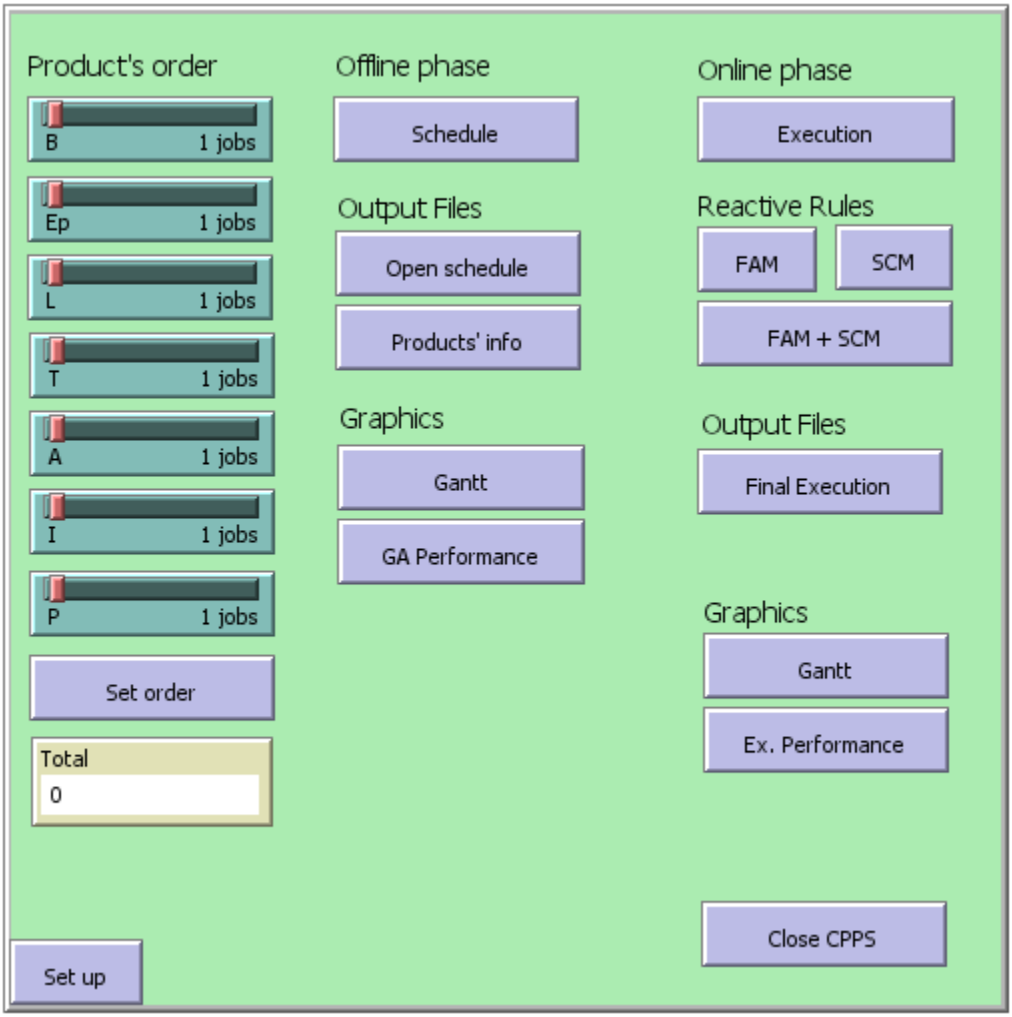

Figura 28. Interfaz de usuario principal.

Se describen a continuación cada uno de los módulos de este subsistema indicado las entradas (*inputs*) y salidas (*outputs*) esperadas.

Módulo de orden

Este módulo permite al usuario seleccionar la cantidad de productos (trabajos) que se quieren realizar en el piso de manufactura. Como se muestra en la Figura 28, el objeto estándar para interactuar con el DSS y seleccionar la cantidad de productos es el control deslizante.

Una vez se define la cantidad de productos de la orden (*input*), se procede a generar formalmente la orden por medio del botón (objeto estándar) '*Set order'*. Esto genera dos salidas (*outputs*). La primera no es visible para el usuario y es enviada directamente al módulo generación de orden en el subsistema de gestión de datos. La segunda salida es visible para el usuario y confirma la cantidad total de productos envidos para la generación de la programación inicial. Esta salida es visible en pantalla (*PC display*) por medio del objeto estándar, monitor, con la etiqueta '*Total*'.

La interfaz de usuario mostrada en la Figura 28 permite seleccionar la cantidad de productos a realizar, *B, E, L, T, A, I* y *P*, que se definirán en el caso experimental mostrado en el Capítulo 4. En general, el módulo de orden se programa de acuerdo con el conjunto de posibles trabajos *J* realizados en el piso de manufactura.

En el presente documento se presenta la parte funcional del DSS, por ende, la cantidad de productos a realizar es ingresada manualmente por el usuario. En el caso de que el DSS propuesto se integrara como una extensión de un sistema MES, siguiendo los lineamientos del modelo ISA-95 (mencionado en el Capítulo 2), se tendrían interfaces consistentes para el intercambio de datos entre el sistema ERP y el DSS. Por lo anterior, la cantidad de productos a realizar se obtendría directamente de la programación maestra de producción (MPS).

Módulo de interfaz *offline*

Este módulo permite al usuario interactuar con el subsistema de gestión de modelos predictivos. Como se muestra en la Figura 28, el objeto estándar para interactuar con el DSS es el botón.

Una vez generada la orden, en el subsistema de gestión de datos, el algoritmo genético tiene la información necesaria para encontrar la programación inicial. El usuario da la orden (*input*) de ejecutar el GA por medio del botón '*Schedule*'. El software NetLogo cuenta con un centro de comandos '*Command Center*', el cual, en este trabajo de investigación, también se usó como objeto de interacción de salidas (*output*) con el usuario, *PC Display*.

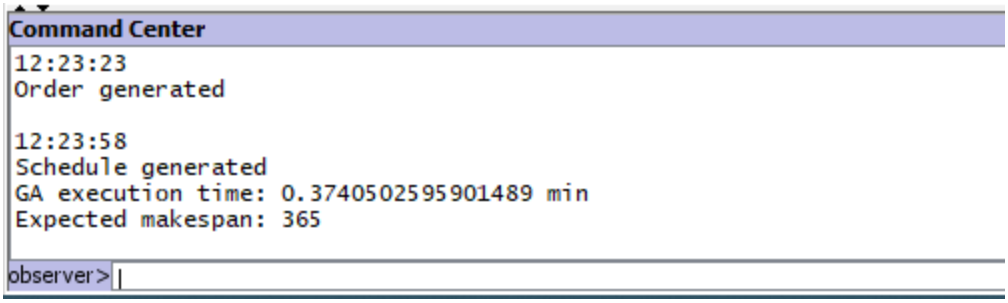

Figura 29. Centro de comandos de NetLogo.

Cuando el algoritmo genético ha generado las decisiones de la programación inicial, se le informa al usuario como se muestra en la Figura 29. Se muestra el tiempo de ejecución del algoritmo y el valor esperado de la medida de desempeño, *makespan*.

El módulo de interfaz *offline* cuenta con dos tipos de salidas adicionales, una de texto y una gráfica, denotadas como '*Output Files*' y '*Graphics*' en la Figura 28 respectivamente.

La primera salida de texto es una hoja de cálculo de Microsoft Excel, la cual contiene la información de las decisiones de programación, generadas por el algoritmo genético, para cada trabajo *j* y sus respectivas operaciones I_j. La Figura 30 muestra el esquema general de la salida de texto mencionada y la información asociada para tres tipos de trabajo *B*, *E* y *L*. En general, se muestra la etiqueta de trabajo (asignada en la generación de la orden), la etiqueta de operaciones y el tipo de trabajo, es decir, columnas Job-ID, Job-Op-ID y Typ, respectivamente. Se muestra también la información de la máquina *m* que procesará una operación $i \in I_j$ en la columna Mach-ID. Finalmente, se muestra la información de tiempo estimado de inicio y finalización de operación en las columnas t_srt y t_end respectivamente; también se muestra la duración de la operación en la columna Op. time.

Por ejemplo, al trabajo tipo *B*, se le asignó la etiqueta de trabajo *J1*. Sus operaciones se denotan como *J1-1* y *J1-10* (inicial y final en este caso). La operación *J1-1* se realizará en la maquina *M1* y se tiene un tiempo estimado de inicio de operación en *te = 30*. Dicha operación tiene una duración teórica de 10.

El usuario puede acceder a la hoja de cálculo mencionada por medio del botón '*Open Schedule*'

Job-ID	Job-Op-ID	Mach-ID	Typ	Op. time	t_str	t_end
J1	J1-1	M1	B	10	30	40
J1	J1-10	M1	B	10	263	273
J2	J2-1	M1	E	10	50	60
J2	J2-2	M2	E	20	111	131
J3	J3-1	M1	L	10	0	10

Figura 30. Esquema general de salida de texto – programación.

La segunda salida de texto es una hoja de cálculo de Microsoft Excel, la cual contiene la información general de los trabajos *j* a ser procesados en el piso de manufactura. La Figura 31 muestra el esquema general de la salida mencionada y la información asociada para tres tipos de trabajo *B*, *E* y *L*. En general, se muestra el tipo de trabajo, la etiqueta de trabajo (análoga a la salida de texto anterior), y el identificador del trabajo en el piso de manufactura, es decir, columnas typ, ga_id y who, respectivamente. Se muestra también la información de la secuencia de entrada al

piso de manufactura, columna sequence. Finalmente, la información inicial de los parámetros de trabajo (1) estado y (9) ubicación, columna status y xcor y ycor, para definir la coordenada (*x.y*) de ubicación.

Por ejemplo, al trabajo tipo *B*, se le asignó la etiqueta de trabajo *J1* (*ga_id = 1*, en esta salida) y su identificador en el piso de manufactura es 50. El trabajo *B* debe esperar a la entrada de tres trabajos *j'* previos, es decir, su secuencia de entrada es la cuarta posición. El estado de los trabajos *j* es (a) Set-Up, como indicado en el módulo de monitoreo, el trabajo *j* esta fuera del sistema. En línea de espera para entrar.

El usuario puede acceder a la hoja de cálculo mencionada por medio del botón '*Products' info*'.

who	xcor	ycor	status	typ	ga_id	sequence
50	-2	22	setup	B	1	4
51	-2	22	setup	E	2	6
52	-2	22	setup	L	3	1

Figura 31. Esquema general de salida de texto – información de trabajo.

Se cuentan con dos salidas graficas. La primera es el diagrama de Gantt de la programación inicial, a la cual se accede por medio del botón '*Gantt*'. La segunda, de carácter académico, corresponde a la gráfica de desempeño del algoritmo genético, a la cual se puede acceder por medio del botón '*GA Performance*'.

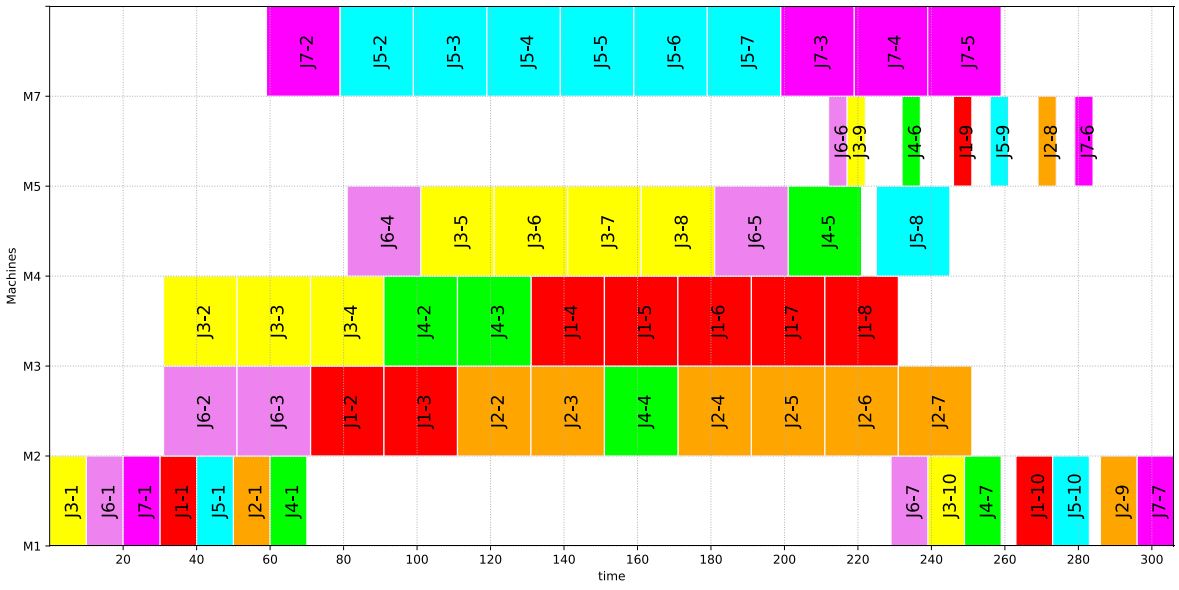

Figura 32. Esquema general de salida grafica – diagrama de Gantt.

El esquema del diagrama de Gantt propuesto se muestra en la Figura 32. Sobre el eje *x* se muestra el tiempo de ejecución y sobre el eje *y* las máquinas del piso de manufactura. En el diagrama se muestran las operaciones que realiza cada una de las maquinas con su respectiva etiqueta de trabajo y de operación. Al comparar la información de esta salida con las salidas de texto, se puede identificar fácilmente cada uno de los productos indicados en la orden de producción.

La Figura 33 muestra el esquema de la gráfica de desempeño del GA: En el eje *x* se muestran las generaciones del algoritmo y en el eje *y* la medida de desempeño, *makespan*. Se muestran dos escenarios: el mejor resultado de todos los cromosomas y el promedio de todos los cromosomas.

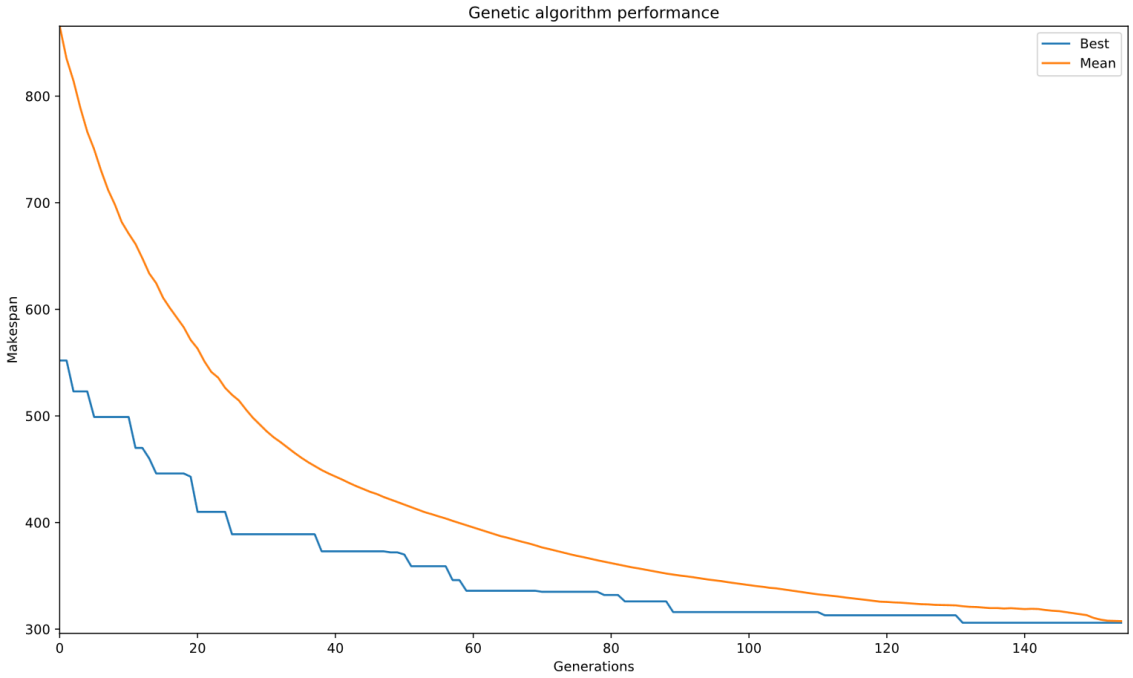

Figura 33. Esquema general de salida gráfica – desempeño de GA.

Módulo de interfaz *online*

Este módulo permite al usuario interactuar con el subsistema de gestión de modelos reactivos. Como se muestra en la Figura 28, el objeto estándar para interactuar con el DSS es el botón.

Una vez generada la programación inicial, en el subsistema de gestión de modelos predictivos, se tiene la información necesaria para ejecutar la programación en el piso de manufactura, acción que el usuario puede realizar por medio del botón '*Execution*'. Este botón dirige al usuario a la interfaz secundaria de usuario, la cual es una representación ciber-física del sistema de manufactura.

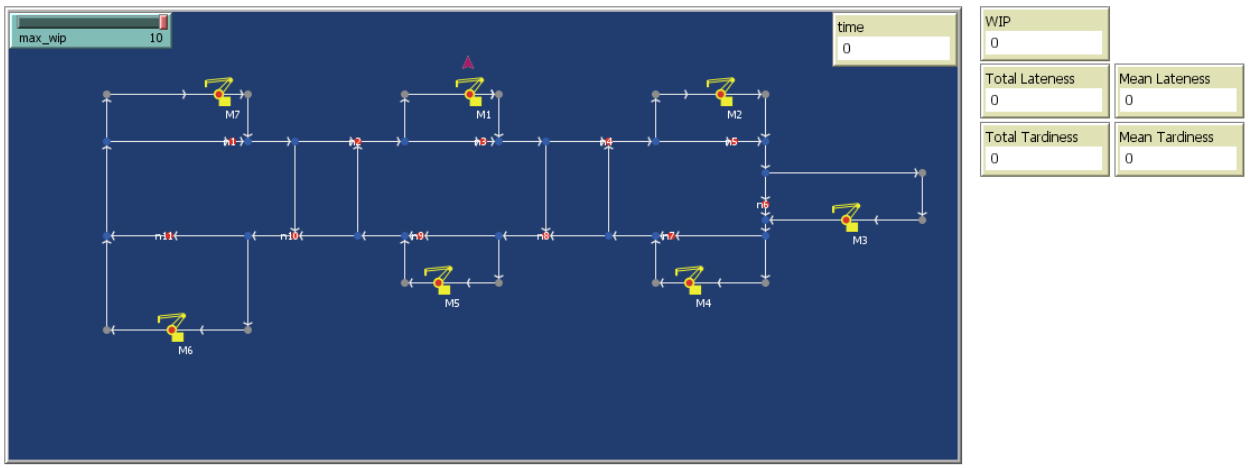

Figura 34. Interfaz de usuario secundaria.

La interfaz de usuario secundaria mostrada en la Figura 34, es realizada también por medio del software NetLogo. Esta representa el sistema de manufactura que se definirá en el caso experimental desarrollado en el Capítulo 4. En general, se incluye la representación de grafo de la topología de la célula de manufactura flexible. Además, la interfaz permite al usuario ver el tiempo de ejecución en el piso de manufactura (*te*) por medio del monitor '*time*' y las medidas de desempeño de retraso y tardanza totales y media por medio de los monitores '*Total Lateness*', '*Mean Lateness*', '*Total Tardiness*' y '*Mean Tardiness*' respectivamente. Finalmente, el usuario conoce fácilmente la cantidad de trabajos en el sistema a través del monitor '*WIP*'.

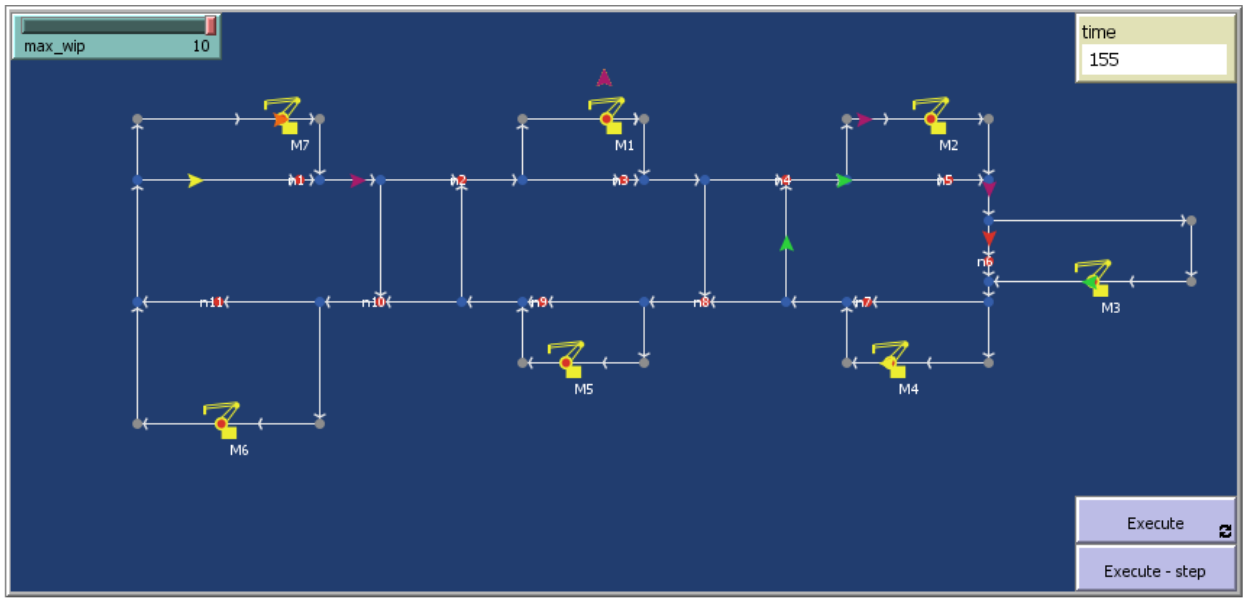

Figura 35. Interfaz de usuario secundaria - ejecución.

La Figura 35 muestra la interfaz secundaria durante la ejecución de una orden de producción. Se tiene un tiempo de ejecución *te = 155*, 10 productos dentro del sistema de

manufactura y otros en fila de espera, representados sobre la máquina *M1*. Los productos se representan como triángulos y las máquinas como brazos robóticos (color amarillo).

Adicionalmente, la interfaz secundaria permite al usuario acceder a la información asociada a productos y máquinas por medio del objeto estándar, menú desplegable. La Figura 36 muestra la representación de la información de monitoreo visible para el usuario, a la izquierda los atributos del trabajo y a la derecha los atributos de la máquina. Los indicadores de retraso y tardanza parcial del trabajo *j* don denotados por *lti* y *tdi*, respectivamente. La Tabla 14 muestra la correspondencia entre los atributos enunciados en el módulo de monitoreo y los elementos visibles para el usuario.

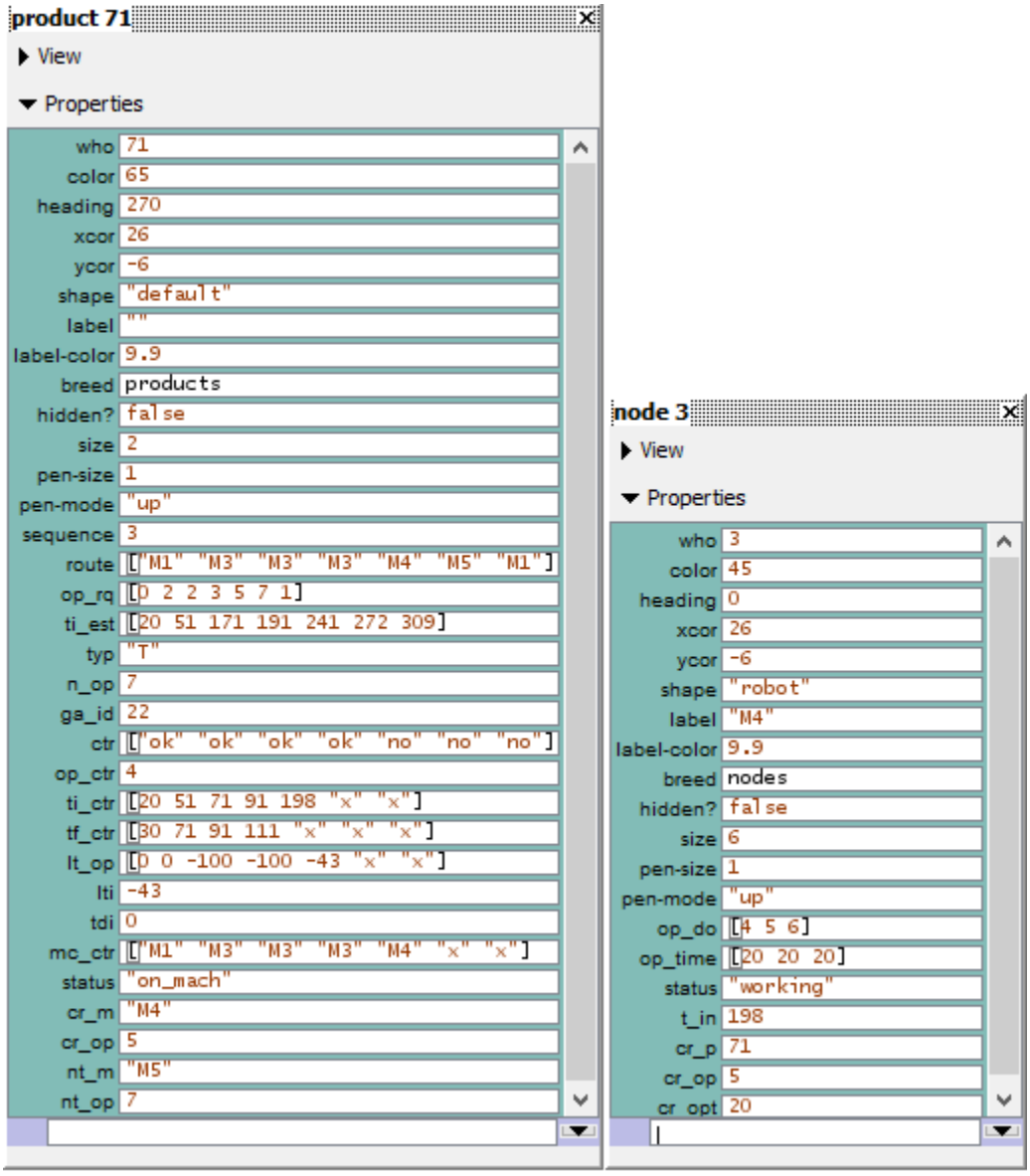

Figura 36. Interfaz de usuario secundaría – información de monitoreo.

Por ejemplo, en la Figura 36 se está procesando un trabajo tipo *T* (*typ*), con etiqueta de trabajo *J22* (*ga_id*) e identificador en el piso de manufactura *71* (*who*). Se encuentra en máquina *M4* (*cr_m*) y se está ejecutando la operación *5* (*cr_op*). Adicionalmente, el usuario puede ver que el retaso parcial del trabajo *J22* en ese momento es de *-43* (*lti*). De manera análoga, se ve que la maquina *M4* (*label*) está procesando el trabajo *J22* (identificador en piso de manufactura *71* – *c_rp*) y la misma operación *5*. Esta interfaz permite ver que el trabajo entro a la máquina en el tiempo de ejecución *te = 198* (*t_in*).

Tabla 14. Correspondencia de atributos en el DSS.

Atributo módulo de monitoreo trabajo	**Interfaz**	**Atributo módulo de monitoreo máquina**	**Interfaz**
Estado	*status*	Estado	*status*
Secuencia de operaciones	*op_rq*	Operaciones en capacidad de realizar	*op_do*
Ruteo de procesamiento	*route*	Trabajo en operación	*cr_p*
Tiempo estimado de inicio de operación	*ti_ctr*	Operación de trabajo	*cr_op*
Máquina actual	*cr_m*	Tiempo de entrada de trabajo	*t_in*
Operación actual	*cr_op*	Tiempo de procesamiento de operación	*cr_opt*
Siguiente operación	*nt_op*	Ubicación	*(xcor, ycor)*
Siguiente máquina	*nt_m*		
Ubicación	*(xcor, ycor)*		

Cuando se produce una perturbación en el sistema de manufactura, esta se informa al usuario por medio del centro de comandos de NetLogo en la interfaz de usuario secundaria. Por defecto, en este trabajo de investigación, la heurística de reactividad activa para generar las decisiones de reprogramación es la FAM+SCM, es decir, se tienen activas las dos técnicas explicadas en la sección anterior. La alarma mostrada en el centro de comandos le permite al usuario conocer que la técnica activa es la indicada anteriormente.

El módulo de interfaz *online* cuenta con un centro de decisiones y los mismos dos tipos de salidas del módulo de interfaz offline.

El centro de decisiones, denotado como *Reactive Rules*, permite interactuar con el DSS por medio de botones. Cada botón tiene la etiqueta de la técnica heurística que se ejecutaría si es seleccionado. El usuario puede cambiar la técnica definida por defecto y que está siendo ejecutada en el piso de manufactura al dar la orden por medio del botón. Si se realiza un cambio de técnica ejecutada, el centro de comando informa del cambio al usuario y muestra la técnica activa en el piso de manufactura. Si el usuario selecciona dos veces el mismo botón se desactiva la técnica en ejecución y por ende, no se ejecutaría ninguna hasta la siguiente activación a realizar por el usuario.

En general, el usuario puede usar su conocimiento sobre el sistema para ejecutar una regla diferente a la definida por defecto o no ejecutar ninguna, ya que las técnicas provistas por el DSS son de complemento y soporte para la toma de decisiones (característica 10). Además, el centro de decisiones permite al usuario la elección de una respuesta tras la ocurrencia de una perturbación (característica 2), lo que permite modificar la programación inicial por medio de las decisiones reactivas (característica 9).

El módulo de interfaz *online* cuenta con dos tipos de salidas adicionales, una de texto y una gráfica, denotadas como '*Output Files*' y '*Graphics*' en la Figura 28 respectivamente.

La salida de texto es una hoja de cálculo de Microsoft Excel, la cual contiene la información final de ejecución en el piso de manufactura. Esta hoja mantiene la misma estructura mostrada en la Figura 30. El usuario puede acceder a la hoja de cálculo mencionada por medio del botón '*Final Execution*'.

Se cuentan con dos salidas graficas. La primera es el diagrama de Gantt de la ejecución final a la cual se accede por medio del botón '*Gantt*'. Si se tomaron decisiones reactivas durante la ejecución tras la ocurrencia de una perturbación, tanto la salida de texto como el diagrama deben reflejarlas. Este diagrama mantiene la estructura mostrada en la Figura 32.

La segunda, de carácter académico, corresponde a la gráfica de progreso de la medida de desempeño en la ejecución, a la cual se puede acceder por medio del botón '*Ex. Performance*'. Esta gráfica permite evaluar la robustez de la reprogramación, ya que muestra la degradación de la medida de desempeño sustituta durante la ejecución de la producción en el piso de manufactura. Además, mantiene la estructura desarrollada en la Figura 11, en la cual se muestra sobre el eje *x* el tiempo de ejecución y sobre el eje *y* el valor de la medida de desempeño.

La Figura 37 muestra el esquema de a gráfica de robustez de la reprogramación. La medida de desempeño mostrada es la tardanza media, la cual, en este caso, tiene un valor esperado de *0* antes de la ocurrencia de la perturbación. La perturbación afecta la ejecución de la programación de operaciones inicial y tras la ejecución de las decisiones de reprogramación, el valor de tardanza media se estabiliza en *35,86* en *te = 337*.

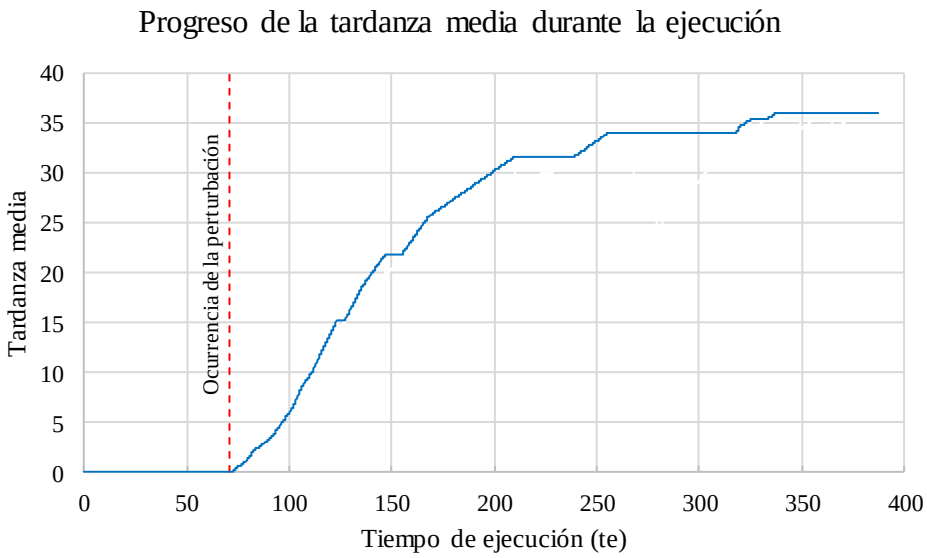

Figura 37. Esquema general de salida gráfica – robustez de la reprogramación.

La interacción del subsistema de interfaz de usuario con los otros subsistemas del DSS se muestran en la

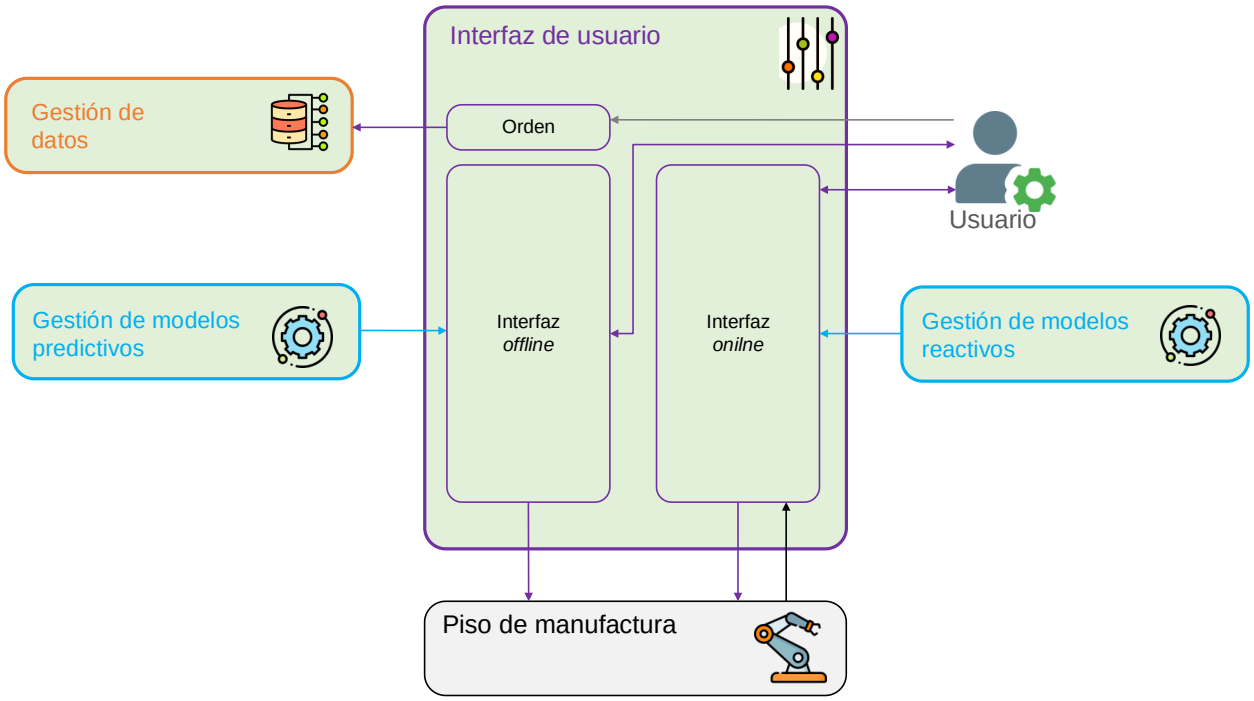

Figura 38. Esquema detallado del subsistema de interfaz de usuario.

El esquema detallado del sistema de soporte de decisiones propuesto es mostrado en la Figura 39. La aplicación detallada del DSS propuesto es mostrada en el Apéndice B.

En el siguiente capítulo se realizan estudios experimentales de la aplicación y desempeño del DSS y los modelos propuestos, respectivamente.

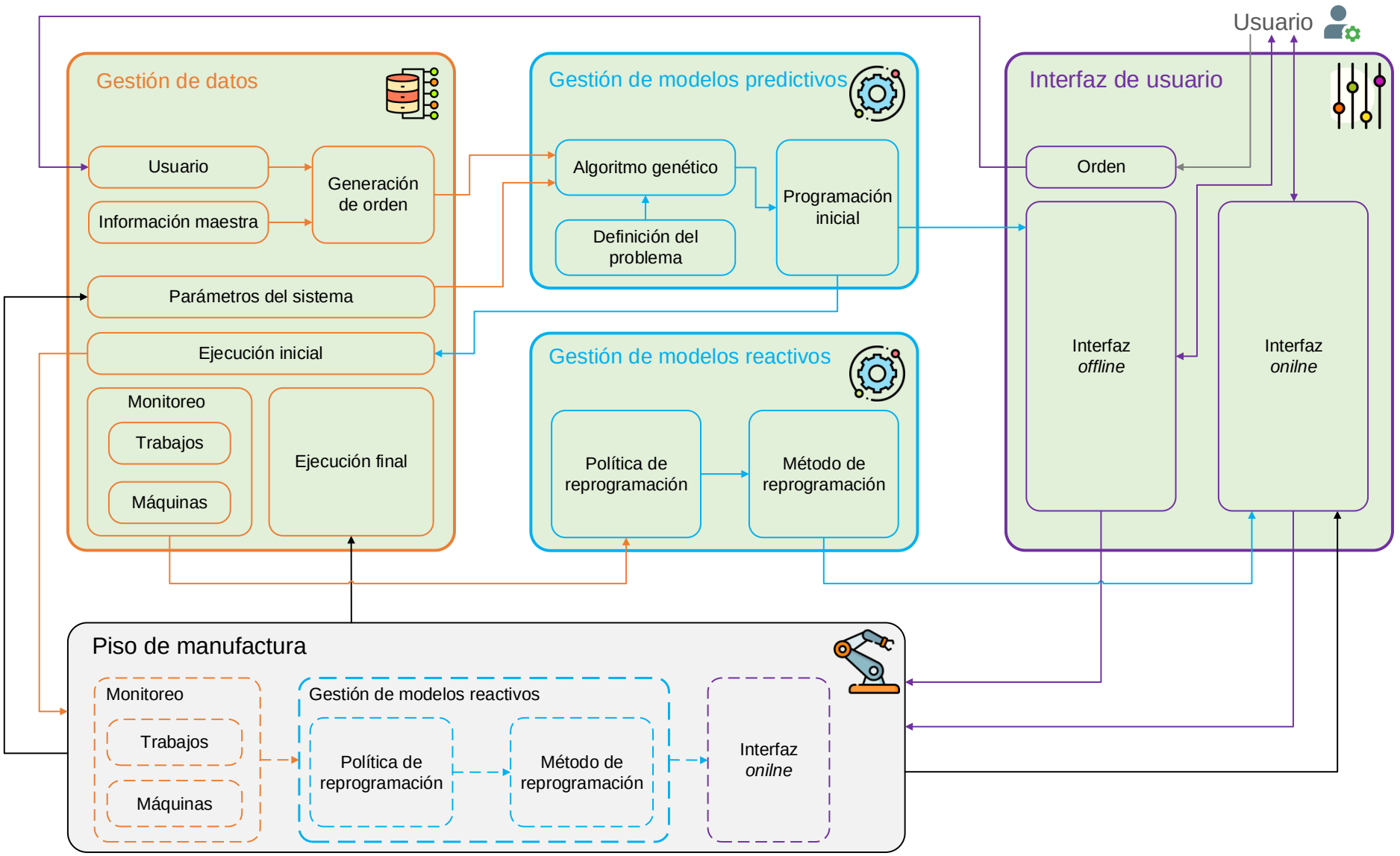

Figura 39. Esquema detallado del sistema de soporte de decisiones propuesto.

Capítulo 4
Sistema de soporte de decisiones – AIP
Un caso de estudio experimental

Introducción

En este capítulo se implementa y validan los beneficios del sistema de soporte de decisiones propuesto realizando experimentos en un escenario simulado de un entorno de manufactura flexible real ubicado en Valenciennes, Francia.

La siguiente sección describe los detalles de la célula de manufactura AIP. La segunda sección presenta el protocolo experimental que se lleva a cabo en la implementación funcional del DSS. El primer experimento (Experimento A) corresponde a la ejecución de las decisiones de programación iniciales, generadas por el DSS, en el piso de manufactura. En el segundo experimento (Experimento B) se genera una perturbación en el sistema y no se activa el módulo de reactividad del DSS. Los experimentos C, D y E demuestran la aplicabilidad del módulo reactivo del DSS y refuerzan el hecho el enfoque predictivo-reactivo permite disminuir la degradación de las medidas de desempeño. En la tercera sección se muestra el detalle de la implementación del DSS con una instancia de prueba, se muestran los resultados obtenidos por la instancia definida siguiendo el protocolo experimental. Finalmente, se presentan los resultados de los experimentos realizados a cada las instancias presentadas.

Descripción del sistema de manufactura

Esta sección presenta, primero, la descripción general del sistema de manufactura. Segundo, presenta en detalle las especificaciones del sistema AIP, las cuales se dividen en diseño del sistema y especificaciones de producción. El diseño del sistema incluye la descripción de las máquinas y del sistema de manejo de material. Las especificaciones de producción incluyen, productos, componentes y el proceso de manufactura. La descripción presentada en esta sección es esta basada en el benchmarking presentado por (Trentesaux et al., 2013) y en el trabajo de doctorado presentado por (Jiménez, 2017).

El sistema de manufactura flexible al que se hace referencia en este capítulo es una célula de ensamblaje ubicada en la *Université de Valenciennes et du Hainaut-Cambrésis* (Francia), llamada AIP-PRIMECA Valenciennes (*Atelier Inter-établissement de Productique* (*Pôle de Ressources Informatique pour la Mécanique*)). El sistema AIP (abreviatura de AIP-PRIMECA) se compone de un conjunto de elementos industriales, que incluyen robots, sensores, actuadores y un sistema de manejo, que se colocan juntos para procesar un conjunto de trabajos definidos, como

se muestra en la Figura 40. Desde la perspectiva de la investigación de operaciones, este sistema puede ser visto como un *flexible job shop*, llevando a la formulación del *flexible job shop scheduling problem* (Trentesaux et al., 2013).

Esta célula es un ejemplo de un sistema de manufactura flexible que presenta los siguientes tipos de flexibilidad (Jiménez, 2017):

(1) Flexibilidad de la máquina. Se pueden realizar varias operaciones sin un cambio en el alistamiento de máquina.

(2) Flexibilidad de enrutamiento. Existen diferentes rutas de transferencia entre al menos dos máquinas diferentes.

(3) Flexibilidad de proceso. Se puede fabricar un conjunto de trabajos sin cambios importantes de alistamiento.

(4) Flexibilidad de secuencia de máquina. Se pueden secuenciar máquinas alternativas para realizar una operación.

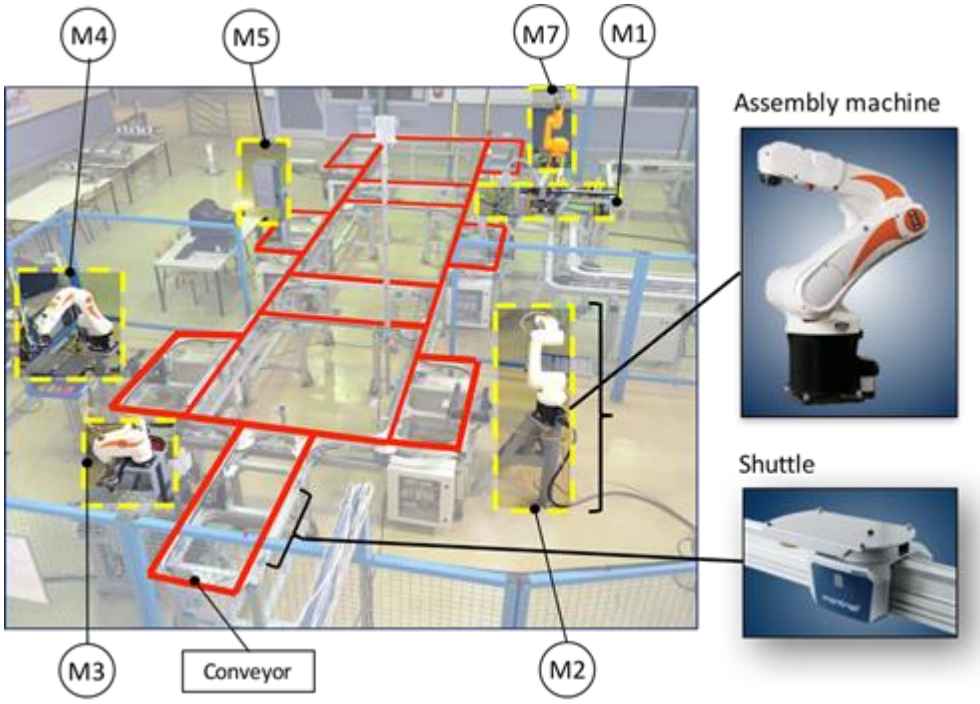

Figura 40. AIP-PRIMECA: Sistema de manufactura flexible ubicado en la Universidad de Valenciennes (Jiménez, 2017).

Diseño de la célula AIP

La célula de fabricación AIP está compuesta por siete máquinas conectadas a través de un sistema unidireccional y flexible de transporte. Esta célula puede procesar trabajos específicos, que son transferidos en lanzadores (*shuttles*) autopropulsados a través del sistema de transporte.

Máquinas

Hay siete máquinas (*machines*) en la célula AIP. Estos son responsables de ejecutar las operaciones de manufactura del sistema de producción. La célula AIP presenta flexibilidad parcial porque un subconjunto de máquinas puede realizar las mismas operaciones, y algunas operaciones pueden ser solo realizadas solo por una máquina. Cada máquina tiene un área de almacenamiento de entrada que sirve como posición de suministro de la máquina. Las máquinas disponibles en la célula son:

(1) Una unidad de máquina de carga y descarga (*M1*).
(2) Tres robots de ensamblaje KUKA (*M2*, *M3* y *M4*).
(3) Una unidad de inspección automática (*M5*).
(4) Una unidad de inspección manual (*M6*).
(5) Un robot de ensamblaje redundante STÄUBLI (*M7*).

Sistema de manejo de materiales

El sistema responsable de manejar los trabajos dentro de la celda AIP es un monocarril. Este sistema de transporte (*transportation system*), específicamente un sistema Montech, lleva los lanzadores a través de la célula y los guía a sus destinos mediante rotación en puntos de transferencia (*transfer gate*) distribuidos dentro del sistema. El sistema de transporte contiene 29 nodos: 7 nodos de máquinas (*M1* a *M7*) y 22 nodos de puntos de transferencia. Adicionalmente, la célula AIP contiene un área de almacenamiento de lanzadores (*shuttle storage area*) para la carga de trabajos a la célula, un área de almacenamiento de entrada (*input storage area*) como área de regulación para cada máquina y los lanzadores que son transportados a través del monocarril.

El diseño de célula AIP es mostrado en la Figura 41.

Especificaciones de producción de la célula AIP

Productos o trabajos

Siete tipos de productos se ofrecen a los clientes. Cada uno de estos productos es correspondiente con un trabajo a procesar en la célula AIP y son distinguidos con las letras *B*, *E*, *L*, *T*, *A*, *I*, y *P*. Los trabajos con realizados a partir del ensamble de un conjunto de componentes, los cuales siguen un conjunto de operaciones ejecutadas por las máquinas de la célula. La Figura 42 muestra los trabajos que pueden ser procesados en la célula AIP.

Componentes

Seis componentes son necesarios para el procesamiento de los trabajos distinguidos como *Plate*, *Axis component*, *I component*, *L component*, *r component* y *screw component*. Los componentes son material de entrada para la célula AIP y se asume que hay disponibilidad infinita

de estos componentes. En la Figura 43 se muestran los componentes y se presenta el ejemplo de la lista de materiales (BOM: *Bill of Materials*) del trabajo *T*.

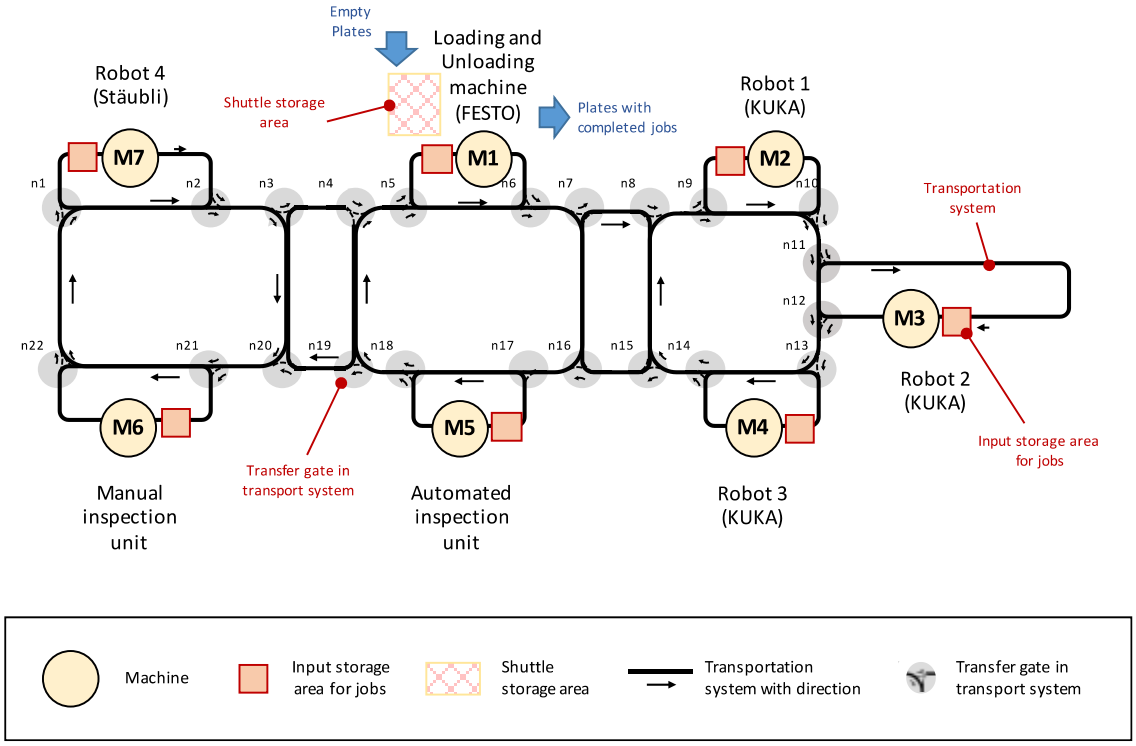

Figura 41. Diseño del sistema de manufactura flexible AIP-PRIMECA (Jiménez, 2017).

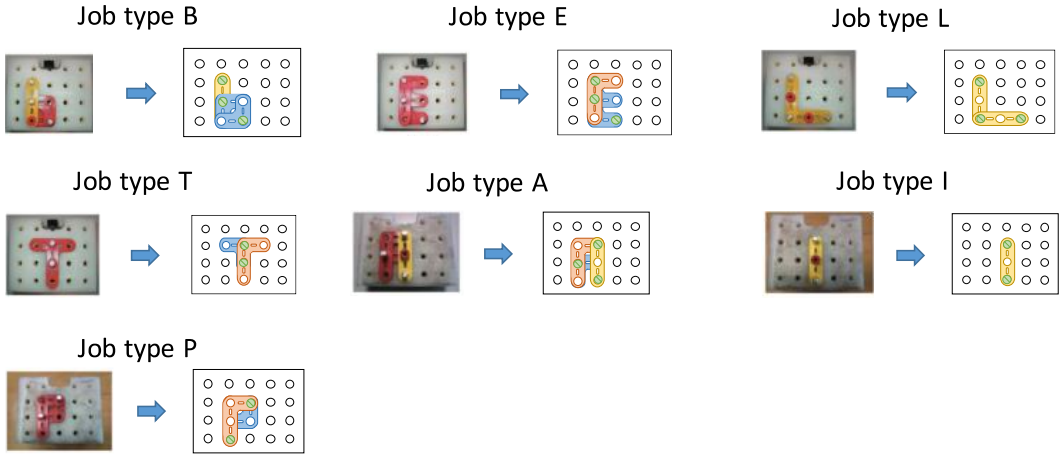

Figura 42. Tipos de trabajos procesados en la célula AIP (Jiménsez, 2017).

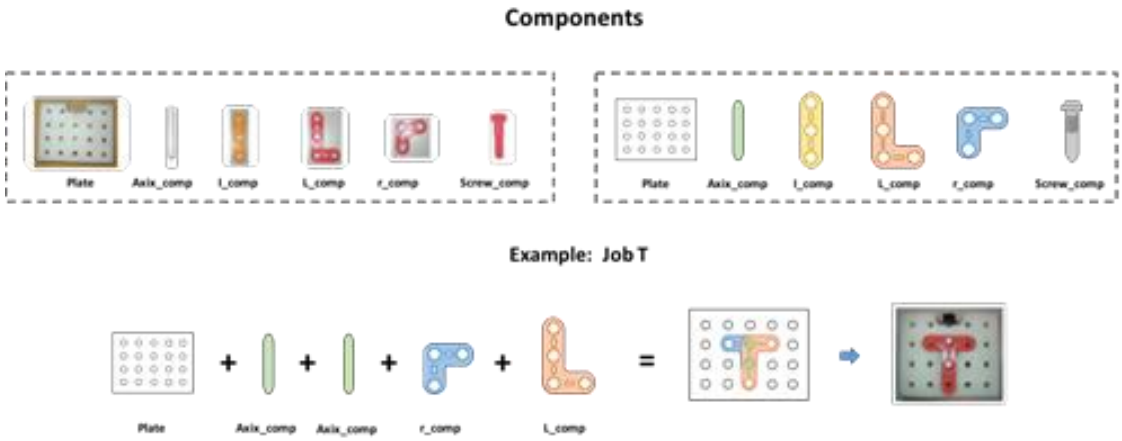

Figura 43. Componentes de los trabajos en la célula AIP (Jiménez, 2017).

Operaciones

Ocho operaciones de manufactura pueden ser procesadas en la célula AIP distinguidas como *Plate loading*, *Axis mounting*, *r_component mounting*, *I_component mounting*, *L_component mounting*, *Screw mounting*, *Inspection* y *Plate unloading*. Por ejemplo, la operación *r_component mounting* significa que el componente *r component* se ha montado en la placa (*Plate*). Cada una de las operaciones puede ser procesada por al menos una máquina en la célula AIP. La Tabla 15 muestra las operaciones de manufactura que cada máquina puede realizar proporcionando los tiempos de procesamiento correspondientes.

La máquina 6 (*M6*) no se considera en este trabajo de investigación, dado que, no se considerará la realización de inspección manual.

Secuencia de producción

Sobre cada trabajo se ejecuta una secuencia de producción para su terminación. En general, sobre los trabajos se ejecuta la siguiente secuencia: carga en la placa (*Plate loading*), ensamblaje de un conjunto de componentes, inspección sencilla (*Inspection*) y descarga de la placa (*Plate unloading*). La Tabla 16 presenta las operaciones y la secuencia de producción de cada uno de los trabajos a partir de la columna ID indicada en la Tabla 15.

Además, si dos operaciones consecutivas no se realizan en la misma máquina, se debe considerar el tiempo de transporte entre estas. La Tabla 17 muestra los tiempos de transporte entre máquinas, la cual fue realizada tras analizar los tiempos de transporte entre nodos adyacentes presentados en la Tabla 3 de (Trentesaux et al., 2013). Se tienen en cuenta las relaciones posibles entre máquinas, es decir, las relaciones indicadas a partir de la secuencia de producción y de las máquinas que pueden realizar dichas operaciones. Por ejemplo, se sabe que un trabajo después de ser procesado en la máquina 1 (*M1*), se dirige a la maquina 2 (*M2*), a la máquina 3 (*M3*) o a la

máquina 7 (*M7*) como posibles relaciones. Un trabajo después de ser procesado en la máquina 5 (*M5*) siempre se dirige a la máquina 1 (*M1*).

Tabla 15. Tiempo en segundos (s) de procesamiento de las operaciones de manufactura (adaptado de Trentesaux et al., 2013)

Operaciones	**ID**	**M1**	**M2**	**M3**	**M4**	**M5**	**M6**	**M7**
Plate loading	0	10						
Plate unloading	1	10						
Axis mounting	2		20	20				20
r_component mounting	3		20	20				20
I_component mounting	4			20	20			20
L_component mounting	5		20		20			20
Screw mounting	6			20	20			
Inspection	7					5		

Tabla 16. Secuencia de producción de cada trabajo (adaptado de Trentesaux et al., 2013)

B	**E**	**L**	**T**	**A**	**I**	**P**
0	0	0	0	0	0	0
2	2	2	2	2	2	2
2	2	2	2	2	2	2
2	2	2	3	2	4	3
3	3	4	5	3	6	5
3	3	4	7	5	7	7
4	5	6	1	4	1	1
6	7	6		6		
7	1	7		7		
1		1		1		

Tabla 17. Tiempo de transporte entra máquinas.

	M1	**M2**	**M3**	**M4**	**M5**	**M6**	**M7**
M1	0	11	21	-	-	-	29
M2	-	0	13	10	18	-	35
M3	-	16	0	7	15	-	32
M4	-	12	22	0	11	-	28
M5	12	-	-	-	0	-	-
M6	-	-	-	-	-	0	-
M7	-	19	29	26	20	-	0

Protocolo experimental

Esta sección presenta el protocolo experimental para evaluar el DSS propuesto. La intención de realizar estos experimentos es evaluar el desempeño del DSS, medir el resultado sobre la medida inicial *makespan* y evaluar la capacidad de reactividad del sistema con las medidas de retraso y tardanza.

Experimentos

Los experimentos presentados usan un modelo de simulación en NetLogo de la célula AIP presentada. Este modelo está integrado con Python, software en el cual se programó y desarrollo la técnica predictiva, algoritmo genético. Las técnicas reactivas presentadas se programaron en el software NetLogo.

(1) Experimento A. No se consideran perturbaciones. Se ejecutan las decisiones de programación generadas por el algoritmo genético.

Dado que el algoritmo genético no considera que en el sistema de manufactura real es posible que una máquina *M* este ocupada cuando un trabajo *j* se dirige a esta (*M*), se notará una diferencia entre el valor de *makespan* teórico, obtenido por el GA, y el valor de *makespan* real, simulado en NetLogo. El evento anterior lleva a una recirculación del trabajo *j* en la célula de manufactura hasta que encuentra la máquina *M* destino vacía para su operación. La recirculación es una causa de la degradación en las medidas de desempeño analizadas y es abordada también por las técnicas reactivas.

El algoritmo genético programado, es un modelo genérico que sigue la formulación del modelo presentado en la sección “Módulo de definición del problema”, por ende, no tiene en cuenta el diseño particular del piso de manufactura definido por le grafo *G = (S, V)*.

Los resultados de este experimento son tomados como línea base de comparación con los resultados de los experimentos siguientes.

(2) Experimento B. Considera perturbaciones. Se ejecutan las decisiones de programación generadas por el algoritmo genético. El subsistema de gestión de modelos reactivos aún no es ejecutado.

(3) Experimento C. Considera perturbaciones. Se ejecutan las decisiones de programación generadas por el algoritmo genético. El subsistema de gestión de modelos reactivos ejecuta la técnica FAM para actualizar la programación.

(4) Experimento D. Considera perturbaciones. Se ejecutan las decisiones de programación generadas por el algoritmo genético. El subsistema de gestión de modelos reactivos ejecuta la técnica SCM para actualizar la programación.

(5) Experimento E. Considera perturbaciones. Se ejecutan las decisiones de programación generadas por el algoritmo genético. El subsistema de gestión de modelos reactivos ejecuta la técnica FAM+SCM para actualizar la programación.

Instancias

El siguiente conjunto de ordenes fue definido para evaluar el DSS. Las siguientes consideraciones fueron tenidas en cuenta sobre las restricciones para todas las órdenes.

(1) El número máximo de trabajos *j* en el sistema de manufactura se definió en 10, *MJ = 10*.

(2) No se consideran colas en el sistema. Si un trabajo *j* que se dirige a la máquina *m* la encuentra ocupada, permanece moviéndose dentro del sistema de transporte principal hasta que la maquina *m* este vacía o hasta que se le asigne una maquina *m’* de acuerdo con las técnicas de reprogramación.

(3) Se consideran los tiempos de transporte entre máquinas mostrados en la Tabla 17.

La Tabla 18 presenta el conjunto de instancias propuesto a partir de la descripción del sistema de manufactura.

El índice de complejidad es un valor teórico, propuesto en este trabajo de investigación, que pretende caracterizar la dificultad de resolución de la instancia a partir de (1) la dificultad de

procesamiento de cada trabajo *j* y (2) del tiempo estimado de transporte entre máquinas de cada trabajo *j*. La dificultad de procesamiento considera (1) la disponibilidad de las máquinas $m \epsilon M$ que pueden realizar el conjunto de operaciones I_j de cada trabajo *j* y (2) la cantidad de operaciones |*Ij*| de cada trabajo *j*. El tiempo estimado de transporte entre máquinas es un cálculo ponderado del tiempo que requiere un trabajo para desplazarse entre las máquinas en capacidad de realizar sus operaciones.

Tabla 18. Instancias de experimentos.

Instancia	**Número total de productos**	**Orden - Productos**							**Índice de complejidad**
		B	**E**	**L**	**T**	**A**	**I**	**P**	
mrj_101	10	1	2	1	2	1	1	2	9,16
mrj_102	10	1	0	2	2	1	2	2	8,63
mrj_103	10	1	3	0	2	1	0	3	9,30
mrj_151	15	3	2	1	3	3	3	0	16,03
mrj_152	15	2	3	1	4	2	2	1	14,48
mrj_153	15	4	2	2	4	1	2	0	15,24
mrj_201	20	4	5	5	2	1	1	2	21,37
mrj_202	20	4	3	1	3	3	3	3	20,22
mrj_203	20	1	3	5	3	3	3	2	19,74
mrj_251	25	5	5	0	3	3	6	3	24,13
mrj_252	25	2	5	3	8	3	4	0	23,21
mrj_253	25	5	4	3	3	5	4	1	27,92
mrj_301	30	5	8	7	4	2	3	1	31,57
mrj_302	30	4	1	2	6	8	4	5	31,14
mrj_303	30	6	4	8	4	1	0	7	30,03

Perturbación

La perturbación propuesta está basada en la lista de escenarios dinámicos presentada en (Trentesaux et al., 2013).

Descripción

En un momento dado, una de las máquinas redundantes tendrá una falla durante un periodo de tiempo determinado. Una máquina se considera redundante si las operaciones que realiza pueden ser procesada por una o más máquinas de la célula. Por ende, una operación puede ser reasignada (reprogramada) a otra máquina aprovechando la naturaleza de los sistemas *job shop* flexibles.

Comportamiento esperado

Este escenario simula que las máquinas tienen una confiabilidad limitada. Se evalúa la capacidad del sistema de soporte de decisiones para administrar una falla de maquina redundante.

Parámetros

Máquina: robot de ensamblaje redundante STÄUBLI (*M7*).

Tiempo de inicio: justo después de la salida del primer trabajo *j* que llega a *M7*.

Duración: *15 x número total de trabajos* (s).

Implementación del DSS al sistema de manufactura AIP

En esta sección se implementa el DSS siguiendo los pasos descritos en la última sección del Capítulo 3 “Aplicación del sistema de soporte de decisiones”. Se indica el detalle de la implementación del DSS y se muestran los resultados obtenidos para cada uno de los escenarios planteados para la instancia mrj_151.

El propósito general del sistema de soporte de decisiones es guiar la ejecución de operaciones de una orden de producción en un entorno de manufactura *flexible job shop* minimizando el *makespan* y respondiendo a posibles perturbaciones. Por ende, el DSS genera las siguientes decisiones: (1) secuencia de entrada al sistema de manufactura AIP. (2) ruta de procesamiento de cada trabajo para completar sus operaciones y (3) ruta de transporte que guía los trabajos a través del monocarril de la célula AIP.

La ruta de procesamiento es la decisión que el subsistema de reactividad afecta, dado que bajo el sistema de manufactura flexible, *flexible job shop*, se aseguran máquinas redundantes. Es decir, una operación puede ser procesada por más de una máquina y por ende, una operación puede ser reasignada (reprogramada) a otra máquina.

Implementación técnica del caso de estudio

Los modelos realizados en Python y NetLogo fueron ejecutados en un computador de mesa DELL Optiplex 7060 Intel Core i7-8700 CPU@ 3,2 GHz con 16 GB de memoria RAM.

Generación de la programación

El usuario ingresa la cantidad de productos a producir. Por lo definido en la Tabla 18, en la instancia mrj_151 se van a producir tres trabajos tipo *B*, dos trabajos tipo *E*, un trabajo tipo *L*, tres trabajos tipo *T*, tres trabajos tipo *A*, tres trabajos tipo *I* y ningún trabajo tipo *P*.

Una vez generada la orden y ejecutado el algoritmo genético, se tienen las siguientes salidas. El m*akespan* de la instancia es *566* s. Adicionalmente, el tiempo de ejecución del GA fue 10,87 min.

La salida de texto con la información de los productos se muestra en la Figura 44. Se comprueba la orden ingresada en por el usuario en la columna typ. Además, la columna sequence muestra el orden de entrada de los trabajos al piso de manufactura. La etiqueta del trabajo es mostrada en la columna ga-id y es usada para identificar con facilidad cada trabajo *j* en el diagrama de Gantt mostrado en la Figura 45.

who	xcor	ycor	status	typ	ga_id	sequence
50	-2	22	setup	B	1	6
51	-2	22	setup	B	2	4
52	-2	22	setup	B	3	10
53	-2	22	setup	E	4	2
54	-2	22	setup	E	5	5
55	-2	22	setup	L	6	8
56	-2	22	setup	T	7	9
57	-2	22	setup	T	8	13
58	-2	22	setup	T	9	14
59	-2	22	setup	A	10	11
60	-2	22	setup	A	11	3
61	-2	22	setup	A	12	7
62	-2	22	setup	I	13	1
63	-2	22	setup	I	14	12
64	-2	22	setup	I	15	15

Figura 44. Información de trabajos – mrj_151.

El diagrama de Gantt permite corroborar la relación entre las salidas que permite el DSS. Por ejemplo, se ve que el trabajo con etiqueta 13 (*J13*) es el primero en entrar al sistema de manufactura seguido por los trabajos *J4* y *J11*.

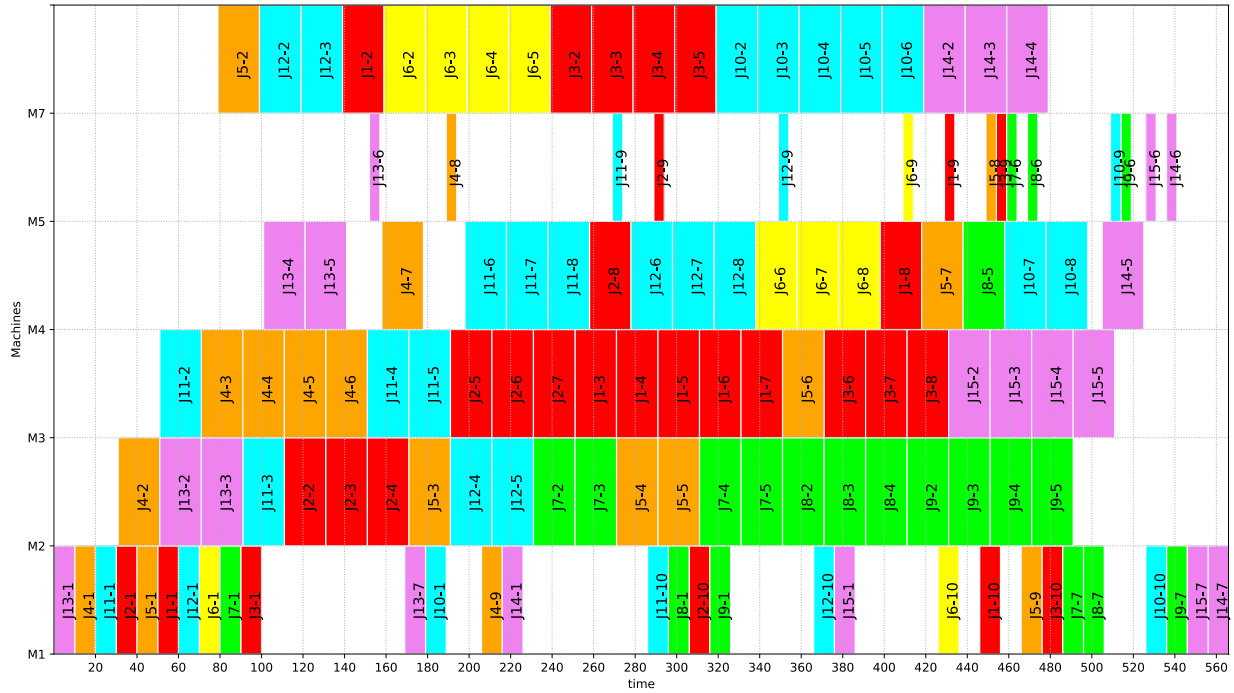

Figura 45. Diagrama de Gantt mrj_151 - GA.

El diagrama permite también validar dos de las restricciones indicadas en la subsección “Instancias”. Restricción (1) – número máximo de trabajos permitidos en el sistema. Se permite la entrada de trabajos hasta el trabajo *J3* (trabajo número diez en el sistema). Una vez el trabajo 13, *J13*, abandona el sistema (operación *J13-7* en la máquina *M1 – te = 179*) se permite el ingreso del trabajo *J10*, para establecer de nuevo el numeró de trabajos en el sistema en diez. En el tiempo de ejecución *te = 216* se evidencia el mismo proceso. El trabajo 4, *J4*, abandona el sistema (operación *J14-9* en la máquina *M1*) y se permite el ingreso al trabajo *J14*. Lo anterior, también se evidencia en la Figura 44, la secuencia de entrada los trabajos *J10* y *J14* es 11 y 12 respectivamente.

Restricción (3) – tiempo de transporte entre máquinas. La operación 2 del trabajo 4, *J4-2*, empieza, en la máquina *M2*, *11* s después de dejar la máquina *M1*. Este es el tiempo de transporte indicado en la Tabla 17. Lo mismo ocurre con la operación 2 del trabajo 11, *J11-2*, la cual empieza, en la máquina *M3*, 21 s después de dejar la máquina *M1*.

Relación con el piso de manufactura

En la simulación realizada en NetLogo se modelan los cinco experimentos anteriormente descritos.

Experimento A

Makespan (C_{max}) de la instancia: *778,3* s. Lo anterior verifica lo estipulado en la sección “Instancias” y explicado en el Experimento A: (2) No se consideran las colas del sistema. Esto es,

el GA no considera las colas que pueden existir en el sistema de manufactura dada su configuración, por ende, el valor de *makespan* cambia, aun cuando se ejecutan las decisiones generadas por el GA.

Se tienen los siguientes resultados de las medidas de desempeño sustitutas de la instancia.

(1) Retraso total (L): *1064,30* s

(2) Retraso medio ($\bar{L}$): *70,95* s.

(3) Tardanza total (T): *1483,50* s.

(4) Tardanza media ($\bar{T}$): *95,90* s.

Los anteriores son los valores de referencia para comparar con los resultados de los otros experimentos.

La Figura 46 muestra el diagrama de Gantt de la ejecución en el piso de manufactura de la instancia mrj_151. Las decisiones de programación generadas por el GA se mantienen, pero se evidencia el efecto de no considerar las colas en la generación de la programación. Es decir, los tiempos de inicio de algunas operaciones se ven afectados al encontrar las máquinas de destino ocupadas. Por lo anterior, los valores de las medidas de desempeño sustitutas también aumentan, los cuales tienen un valor esperado de cero.

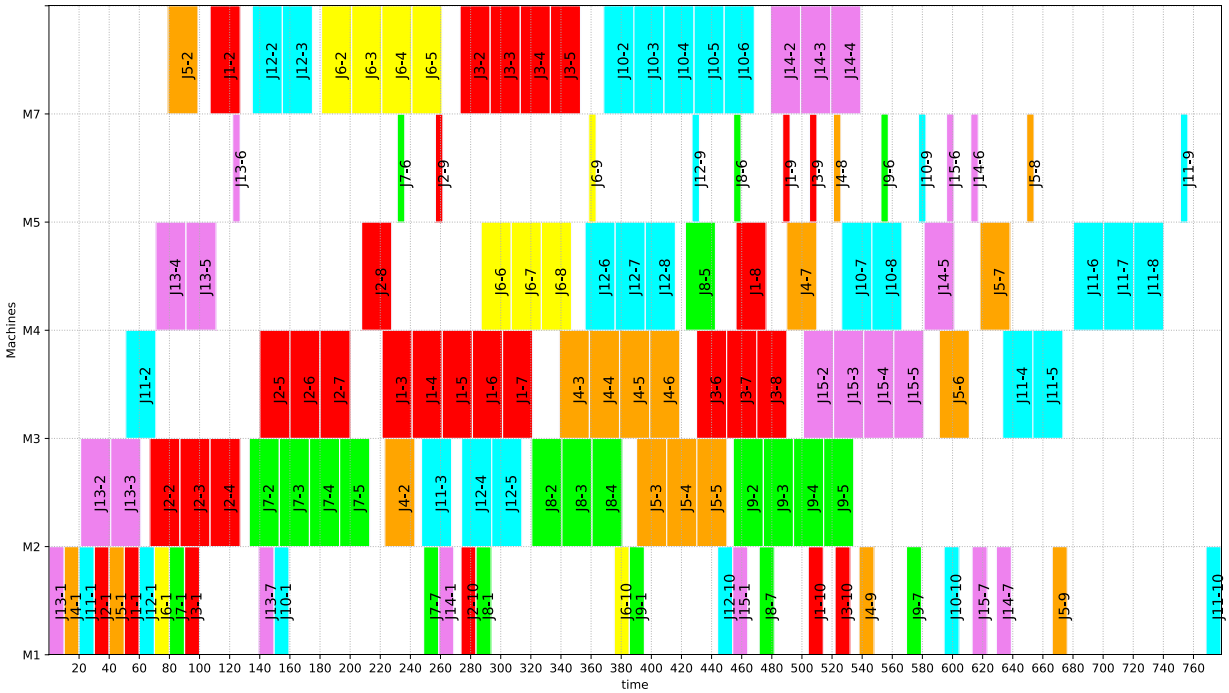

Figura 46. Diagrama de Gantt mrj_151 – Ejecución en *shop floor*.

Experimento B

Se tienen los siguientes parámetros de la perturbación.

(1) Máquina: robot de ensamblaje redundante STÄUBLI (M7).

(2) Tiempo de inicio: justo después de la salida del primer trabajo j que llega a M7, *te = 99* s.

(3) Duración: *15 x número total de trabajos* (s). *15 x 15 = 225* s.

La Tabla 19 muestra los resultados de las medidas de desempeño analizadas. La columna mkp hace referencia al valor de *makespan*; tlt y mlt hacen referencia al valor de retraso total y medio respectivamente y ttd y mtd hacen referencia al valor de tardanza media y total respectivamente.

Esta tabla muestra adicionalmente, la variación porcentual de las medidas con respecto al Experimento A (columna A%). La variación porcentual, tanto para las medidas totales como medias, de retraso y de la tardanza es la misma, por ende solo se muestra en una columna. Finalmente, se evidencia la degradación (aumento de valor) de las medidas de desempeño dada la perturbación generada en el sistema.

Tabla 19. Resultados mrj_151 – Experimento B.

				Experimento B				
Instancia	**mkp**	**A%**	**tlt**	**mlt**	**A%**	**ttd**	**mtd**	**A%**
mrj_151	921,0	+18,3%	1855,8	123,7	+74,4%	2220,6	148,0	+54,4%

La Figura 47 muestra el diagrama de Gantt de la ejecución con perturbación en el piso de manufactura de la instancia mrj_151. Las decisiones de programación generadas por el GA se mantienen, ya que no se han activados las técnicas de reprogramación. Se ve el efecto que tiene la perturbación sobre el tiempo de inicio de las operaciones, tanto en la máquina *M7* como en las operaciones siguientes. Por ejemplo, la operación *J1-2*, que antes de la perturbación sería realizada en la máquina *M7* en *te = 107* s, es realizada en *te = 593* s tras la perturbación. La operación *J6-2* se realizaba antes de la perturbación en la máquina *M7* en *te = 181* s, se realiza tras la perturbación en *te = 526,2* s.

Una vez la máquina está disponible de nuevo, en el tiempo *te = 324* s, los productos cuya ruta de procesamiento incluyera la máquina *M7*, pueden ingresar a ejecutar sus operaciones. El primer producto en entrar de nuevo a la máquina *M7*, es el producto 10 a ejecutar su operación 2, *J10-2* en el tiempo *te = 332* s.

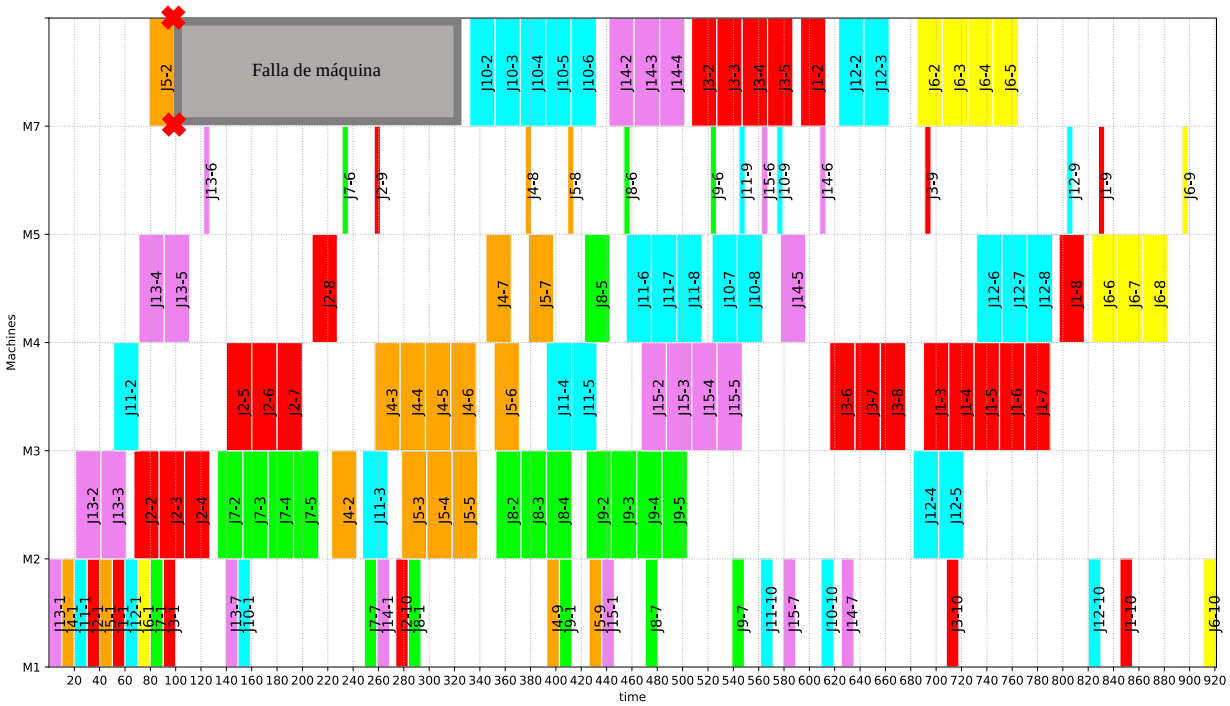

Figura 47. Diagrama de Gantt mrj_151 – Ejecución con perturbación en *shop floor*.

Actualización de la programación

El usuario ejecuta las técnicas reactivas que brinda el DSS.

Experimento C

Se ejecuta la técnica reactiva FAM. Las decisiones de reprogramación inicialmente generadas por el GA son actualizadas a través de la técnica FAM, la cual permite reprogramar una nueva máquina *m*, del conjunto *M* de máquinas disponibles en el piso de manufactura, a la operación *i* del trabajo *j* cuando se cumple la condición de reprogramación.

La Tabla 20 muestra los resultados de las medidas de desempeño analizadas. Se muestra adicionalmente, dos variaciones porcentuales. La primera, es la variación porcentual de las medidas con respecto al Experimento B (columna B%). Esta columna muestra cual fue el efecto de la técnica FAM, sobre las medidas de desempeño, al actualizar la programación tras la ocurrencia de la perturbación. La segunda, es la variación porcentual de las medidas con respecto al Experimento A (columna A%). Esta columna muestra la variación final de las medidas de desempeño con respecto a los valores esperados en la ejecución ideal en el piso de manufactura. La variación porcentual, tanto para las medidas totales como medias, del retraso y de la tardanza es la misma, por ende solo se muestra en una columna.

Tabla 20. Resultados mrj_151 – Experimento C.

	Experimento C										
Instancia	**mkp**	**B%**	**A%**	**tlt**	**mlt**	**B%**	**A%**	**ttd**	**mtd**	**B%**	**A%**
mrj_151	838,1	-9,0%	+7,7%	1209,9	80,7	-34,8%	+13,7%	1505,1	100,3	-32,2%	+4,6%

La Figura 48 muestra el diagrama de Gantt de la ejecución de la técnica FAM tras la ocurrencia de la perturbación en el piso de manufactura de la instancia mrj_151.

Por ejemplo, el trabajo *J3,* que tenía definidas sus operaciones *J3-2*, *J3-3* y *J3-4* en la maquina *M7* (generación de la programación realizada por el GA) con tiempos estimados de inicio de ejecución, antes de la perturbación, *273* s, *293* s y *313* s respectivamente, como se muestra en la Figura 46. Este trabajo seleccionó muevas máquinas para la realización de dichas operaciones (actualización de la programación realizada por la técnica FAM). Las operaciones *J3-2* y *J3-3* se realizaron en la máquina *M2,* en el tiempo de ejecución *447,2* s y *599,1* s respectivamente. La operación *J3-4* se realizó en la máquina *M3,* en el tiempo de ejecución *632,1* s.

No solo los trabajos, cuya ruta de procesamiento fue afectada directamente por la perturbación cambian de ruta. Por ejemplo, el trabajo *J5*, cuyas operaciones *J5-3, J5-4* y *J5-5* tenían definida la máquina *M2* para su procesamiento con tiempos de inicio de ejecución, antes de la perturbación, *390,2* s, *410,2* s y *430,2* s, fueron reasignadas a las máquinas *M3, M7* y *M3* respectivamente. Sus correspondientes tiempos de ejecución fueron *336,2* s, *508,2* s y *558,2* s. En el caso de las operaciones *J5-4* y *J5-5* se evidencia el efecto de la perturbación al retrasar el inicio estimado de la operación.

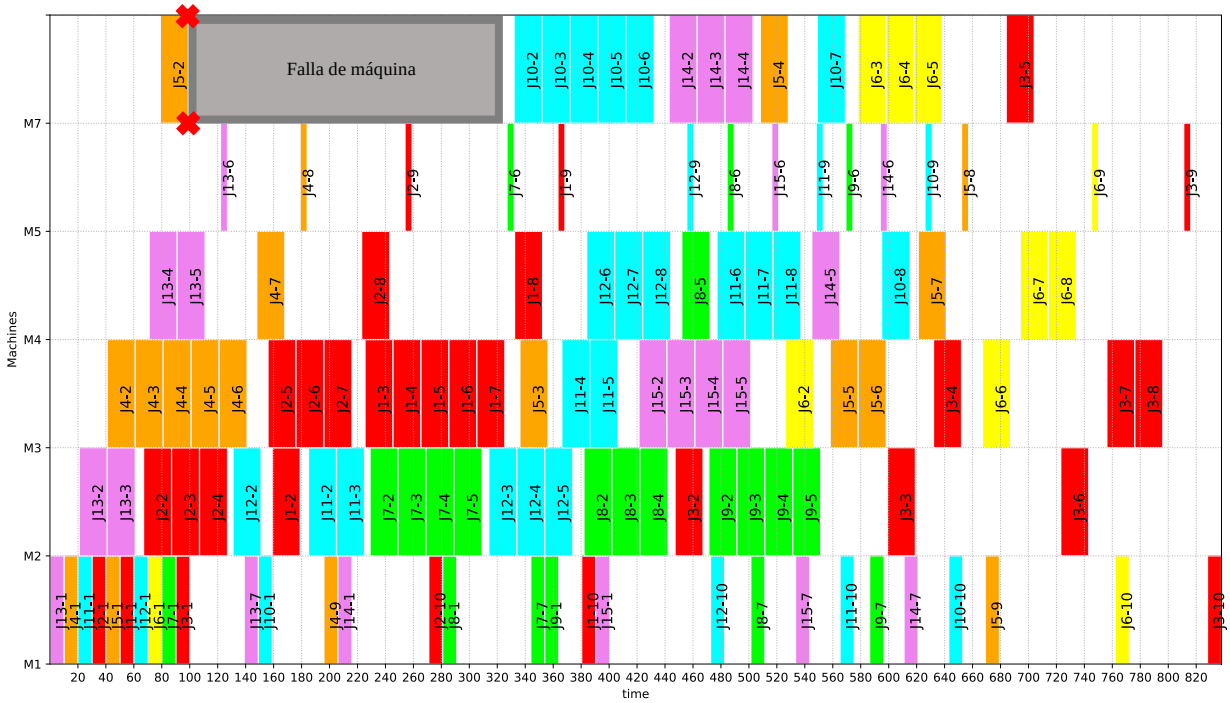

Figura 48. Diagrama de Gantt mrj_151 – Ejecución con perturbación y técnica FAM en *shop floor*.

Experimento D

Se ejecuta la técnica reactiva SCM. Las decisiones de reprogramación inicialmente generadas por el GA son actualizadas a través de la técnica SCM, la cual difiere de la técnica FAM, ya que asigna la máquina *m*, en la que se encuentra el trabajo *j*, si esta está en capacidad de realizar la siguiente operación *i* cuando se cumple la condición de reprogramación.

La Tabla 21 muestra los resultados de las medidas de desempeño analizadas. Se muestra adicionalmente, dos variaciones porcentuales. La primera, es la variación porcentual con respecto al Experimento B (columna B%) y la segunda, es la variación porcentual con respecto al Experimento A (columna A%). Esta tabla mantiene la misma estructura lógica de la Tabla 20.

Tabla 21. Resultados mrj_151 – Experimento D.

					Experimento D						
Instancia	**mkp**	**B%**	**A%**	**tlt**	**mlt**	**B%**	**A%**	**ttd**	**mtd**	**B%**	**A%**
mrj_151	1123,0	+21,9%	+44,3%	2680,7	178,7	+44,4%	+151,9%	3008,7	200,6	+35,5%	+109,2%

La Figura 49 muestra el diagrama de Gantt de la ejecución de la técnica SCM tras la ocurrencia de la perturbación en el piso de manufactura de la instancia mrj_151.

Por ejemplo, el trabajo *J12*, tenía definidas sus operaciones *J12-4*, *J12*-5 en la máquina *M2* y sus operaciones *J12-6* y *J12-7* en la máquina *M4* (generación de la programación realizada por el GA) con tiempos estimados de inicio de ejecución, antes de la perturbación, *274* s, *294* s, *356* s y *376* s respectivamente, como se muestra en la Figura 46. Tras la actualización de la programación realizada por la técnica FAM, el trabajo *J12*, permaneció en la máquina *M7* para realizar las operaciones antes mencionadas. Es decir, el trabajo *J12* finalizo la operación *J12-3* en la máquina *M7* y debía desplazarse a la maquina *M2*. Sin embargo, al evaluar y cumplir la condición de reprogramación y la máquina *M7* estar en capacidad de realizar las siguientes operaciones, se reprogramó su ruta de procesamiento inicial. Las operaciones mencionadas se realizaron en la máquina *M7* en el tiempo de ejecución *699* s, *719* s, *739* s y *759* s, respectivamente.

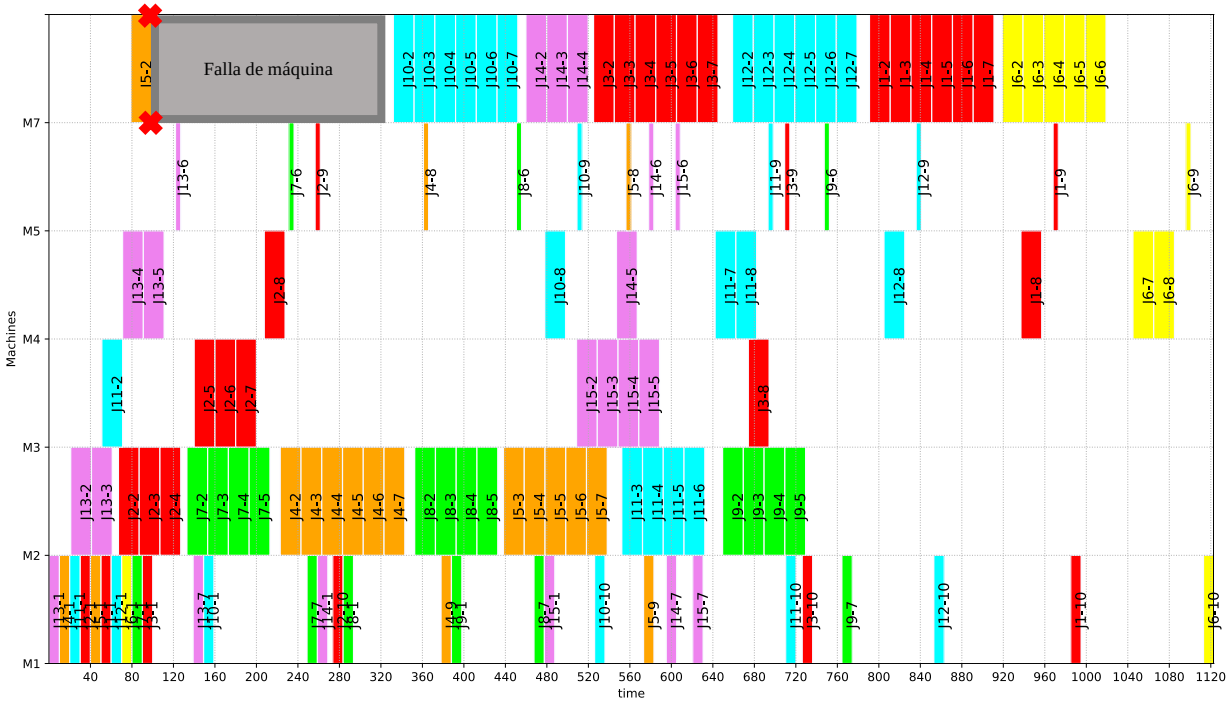

Figura 49. Diagrama de Gantt mrj_151 – Ejecución con perturbación y técnica SCM en *shop floor*.

No solo los trabajos, cuya ruta de procesamiento fue afectada directamente por la perturbación cambian de ruta. Por ejemplo, el trabajo *J4*, cuyas operaciones *J4-3*, *J4-4*, *J4-5* y *J4-6* tenían definida la máquina *M3* para su procesamiento con tiempos de inicio de ejecución, antes de la perturbación, *338,9* s, *358,9* s, *378,9* s y *398,9* s, fueron reasignadas a las máquinas *M2*. La operación *7*, *J4-7*, también fue reprogramada a la máquina *M2*, aun cuando inicialmente estaba programada en la máquina *M4*, con tiempo de inicio antes de la perturbación *489,9* s. Los tiempos de inicio de las operaciones mencionadas, en la maquina *M2*, en este experimento fueron *243* s, *263* s, *283* s, *303* s y *323* s, respectivamente. En este caso se adelantaron las operaciones con el fin de minimizar la degradación de las medidas de desempeño sustitutas.

Experimento E

Se ejecuta la técnica reactiva FAM+SCM. Las decisiones de reprogramación inicialmente generadas por el GA son actualizadas a través de la técnica FAM+SCM cuando se cumple la condición de reprogramación.

La Tabla 22 muestra los resultados de las medidas de desempeño analizadas. Se muestra adicionalmente, dos variaciones porcentuales. La primera, es la variación porcentual con respecto al Experimento B (columna B%) y la segunda, es la variación porcentual con respecto al Experimento A (columna A%). Esta tabla mantiene la misma estructura lógica de la Tabla 20.

Tabla 22. Resultados mrj_151 – Experimento E.

	Experimento E										
Instancia	**mkp**	**B%**	**A%**	**tlt**	**mlt**	**B%**	**A%**	**ttd**	**mtd**	**B%**	**A%**
mrj_151	727,2	-21,0%	-6,6%	919,6	61,3	-50,4%	-13,6%	1037,2	69,1	-53,3%	-27,9%

La Figura 50 muestra el diagrama de Gantt de la ejecución de la técnica FAM+SCM tras la ocurrencia de la perturbación en el piso de manufactura de la instancia mrj_151.

Por ejemplo, el trabajo *J14*, tenía definidas sus operaciones *J14-2*, *J14-3* y *J14-4* en la máquina *M7* y sus operación *J14-5* en la máquina *M4* (generación de la programación realizada por el GA) con tiempos estimados de inicio de ejecución, antes de la perturbación, *479,1* s, *499,1* s, *519,1* s y *581,1* s respectivamente, como se muestra en la Figura 46. Tras la actualización de la programación realizada por la técnica FAM+SCM, el trabajo *J14*, reprogramó la operación *J14-2* en la máquina *M3* (efecto de la técnica FAM) a iniciar en el tiempo de ejecución *553 s*. Estando en la máquina *M3*, permaneció en esta, dado que estaba en capacidad de realizar las siguientes operaciones, para procesar las operaciones *J14-3*, *J14-4* y *J14-5* (efecto de la técnica SCM). Las operaciones mencionadas se realizaron en la máquina *M3* en el tiempo de ejecución *573* s, *593* s, y *613* s, respectivamente.

No solo los trabajos, cuya ruta de procesamiento fue afectada directamente por la perturbación cambian de ruta. Por ejemplo, el trabajo *J11*, reasigna su operación *J11-2* a *J11-6* en la máquina *M2*. La Tabla 23 muestra el resumen de la reprogramación realizada por la técnica FAM+SCM sobre el trabajo *J11*.

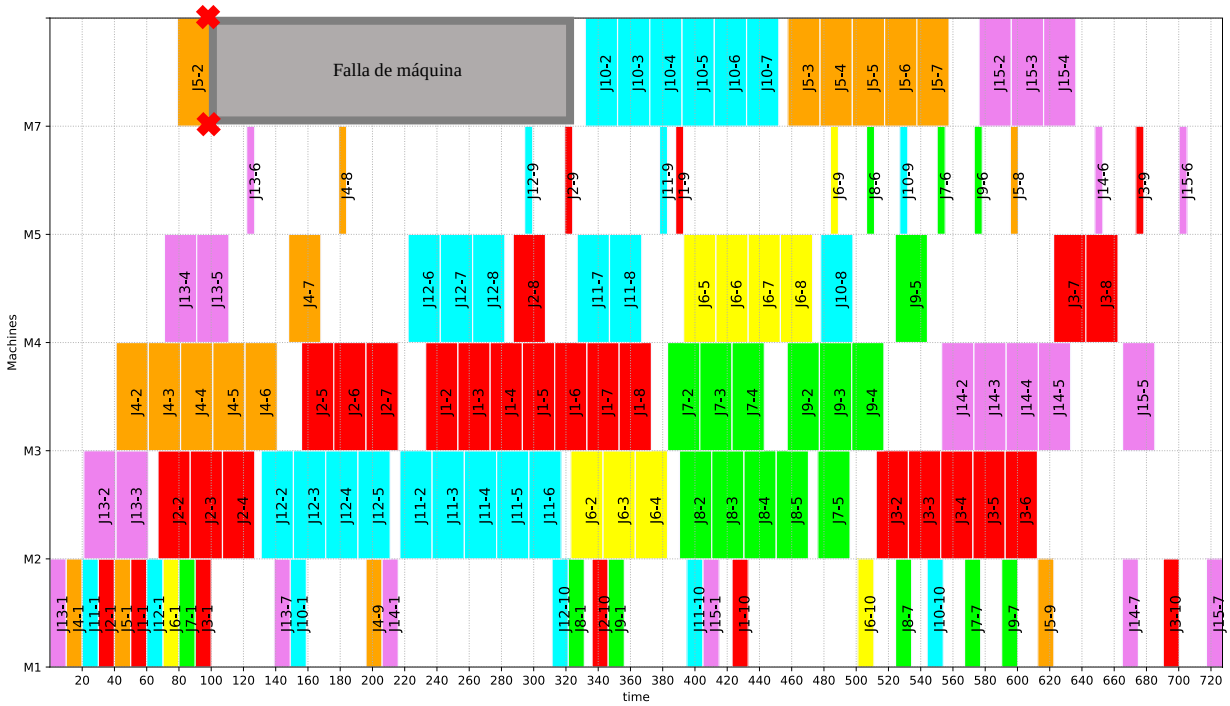

Figura 50. Diagrama de Gantt mrj_151 – Ejecución con perturbación y técnica FAM+SCM en *shop floor*.

Tabla 23. Actualización de la programación: mrj_151 – Experimento E.

Operación	Generación de la programación		Actualización de la programación	
	Máquina	**Tiempo de inicio (s)**	**Máquina**	**Tiempo de inicio (s)**
J11-2	*M3*	*51*	*M2*	*217*
J11-3	*M2*	*247,4*	*M2*	*237*
J11-4	*M3*	*633,3*	*M2*	*257*
J11-5	*M3*	*653,3*	*M2*	*277*
J11-6	*M4*	*680,3*	*M2*	*297*

Adicionalmente, el usuario puede ver la gráfica de progreso de las medidas de desempeño durante le ejecución.

De acuerdo con la condición de reprogramación definida en el subsistema de modelos reactivos, la cual indica que los trabajos con valor asociado $L_{ij} > 0$ son considerados para reprogramación, el interés de la gráfica de progreso recae sobre la medida de tardanza (*tardiness*). Esta se da porque la tardanza es la medida la que representa, como definido el Capítulo 2, la parte positiva del retraso (*lateness*) (Ver Figura 5).

La Figura 51 muestra el progreso de las medidas de retraso y tardanza durante la ejecución, sin perturbación, en el piso de manufactura de la instancia mrj_151. Se debe tener en cuenta que en el piso de manufactura se suman, de manera continua – en tiempo real, los valores asociados a las medidas de desempeño (L_{ij} y T_{ij}) de cada trabajo *j*, como se definió en la Tabla 10. Por lo anterior, el valor de la tardanza empieza a aumentar, aun cuando el valor de retraso permanece negativo. Sin embargo, se cumple que, el valor de la tardanza *T* solo toma valores mayores a cero.

El tiempo de ejecución final corresponde a lo reportado en el Experimento A, *makespan* = *778,3* s y valores finales de las medidas de desempeño retaso medio *70,95* s y tardanza media *95,90* s.

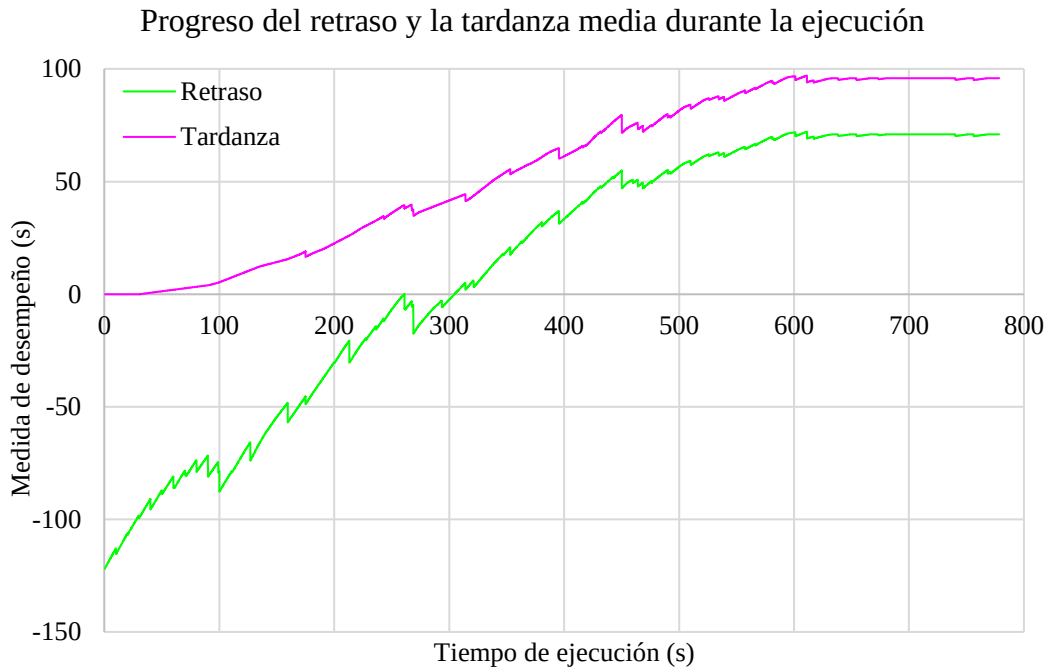

Figura 51. Progreso de las medidas de desempeño sustitutas – mrj_151.

Se muestra a continuación la degradación de la medida tardanza media ($\bar{T}$), tras la realización de los experimentos con la instancia mrj_151.

✓ Experimento A y B

La Figura 52 muestra el progreso de la tardanza media en la ejecución de la instancia mrj_151. En el Experimento A se degrada la medida de tardanza, porque las decisiones generadas por el algoritmo genético no consideraron las posibles colas en el sistema. Es decir, la degradación mostrada no es causada por la perturbación. El valor final de la tardanza media es *95,90* s y el tiempo de ejecución *makespan* fue *778,3* s. En el caso del Experimento B, en el cual perturbación ocurre en el tiempo de ejecución *te = 99* s, la línea tiene el mismo comportamiento que en el Experimento A hasta *te = 135,1* s. Después de este tiempo, se nota un aumento de la tardanza

media con mayor pendiente en dicho experimento. Esto implica que, aunque la perturbación ocurrió en *te = 99* s, el efecto real, por la medición realizada, fue percibido hasta *te = 135,1* s. La tardanza media sigue en aumento, en el Experimento B, hasta alcanzar un valor final de *148* s, representado un aumento del 54,4 % con respecto al Experimento A. El tiempo de ejecución, también se degradada, y termina en de *921* s, lo que significa un aumento de 18,3% con respecto al Experimento A.

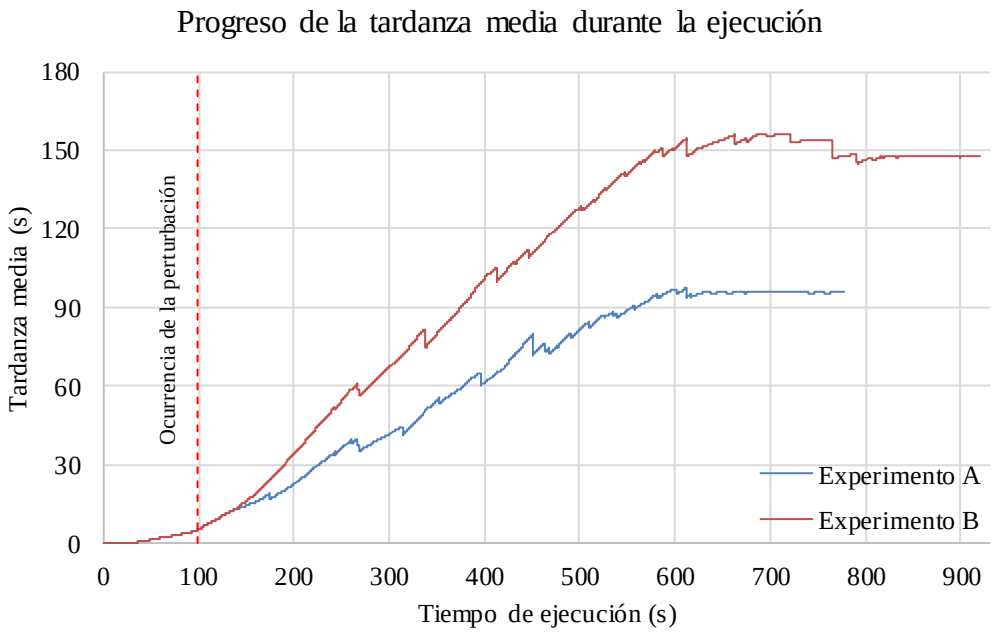

Figura 52. Progreso de la tardanza media – Experimento A y B.

✓ Experimento C, D y E

La Figura 53 compara el Experimento B (perturbación sin técnicas reactivas), con los experimentos en lo que una técnica reactiva esta activa. El comportamiento de las líneas, correspondientes a las técnicas de reactividad, sigue el mismo comportamiento descrito en la Figura 11. Es decir, el comportamiento de la tardanza media se mantiene estable hasta la ocurrencia de una perturbación; tras la perturbación, la técnica toma un periodo de cálculo para la reprogramación, periodo en el cual la degradación sigue aumentando; y tras la ejecución de las nuevas decisiones de programación en el piso, la pendiente de la degradación disminuye hasta alcanzar un valor estable.

En el caso de la instancia mrj_151, la técnica que mejor mitigó el efecto de la perturbación en la ejecución fue la FAM+SCM (Experimento E); disminuyó el valor de tardanza media en un 53,3%. La técnica FAM (Experimento C) también mostró buen desempeño y disminuyó el valor de tardanza media un 32,3%. La técnica SCM no fue eficiente e incluso aumentó el valor de la degradación un 35,5%.

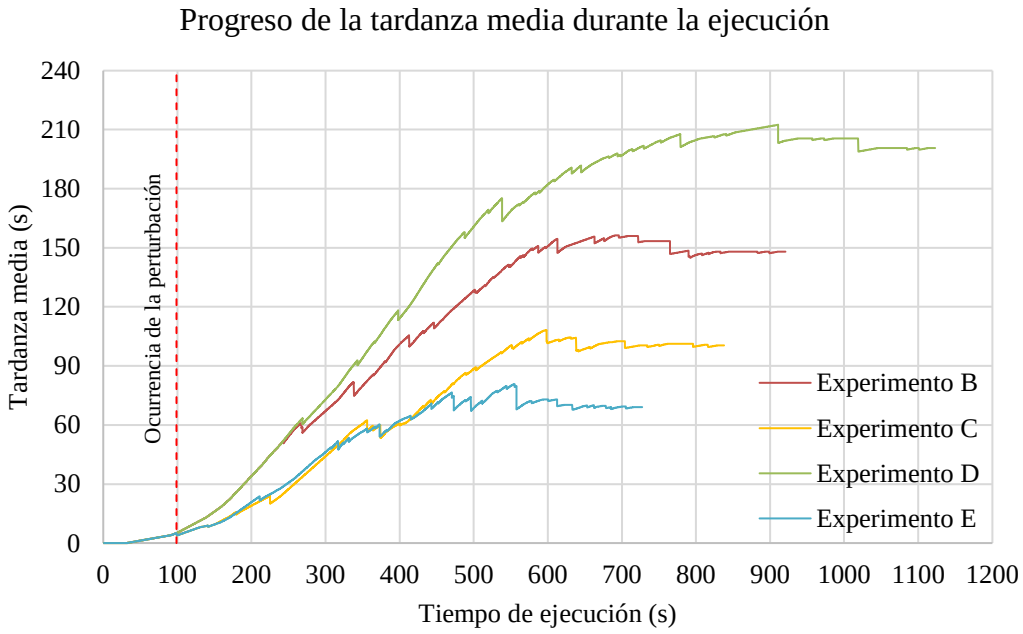

Figura 53. Progreso de la tardanza media – Experimento B, C, D y E.

El tiempo de ejecución final, *makespan* se comportó de manera similar a la tardanza media tras la activación de las técnicas definidas por el DSS. La técnica FAM+SCM logro una disminución del 21% y la técnica FAM de 9%. Por el contrario, la técnica SCM aumento el valor de *makespan* un 21,9%.

Las Figura 54 y Figura 55 muestran el resumen de resultados de las medidas de desempeño, *makespan*, retraso y tardanza de los experimentos realizados con la instancia mrj_151.

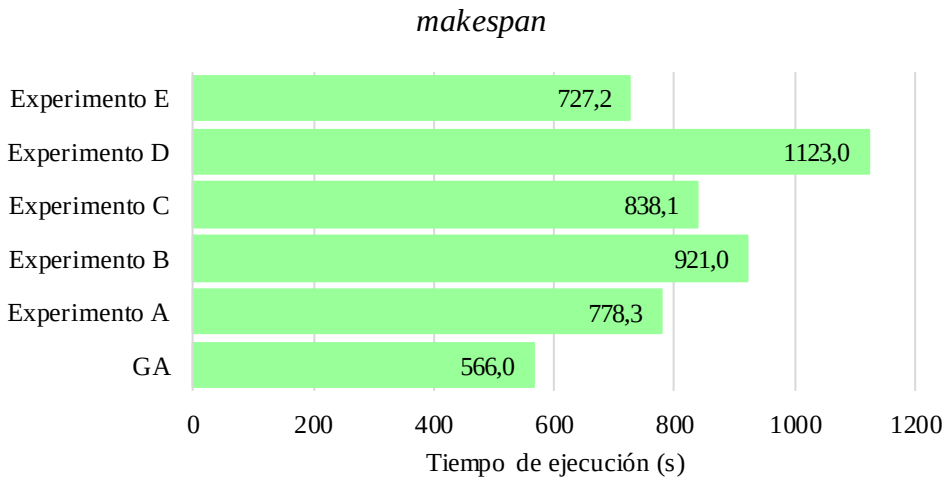

Figura 54. Resultados de experimentos – *makespan*.

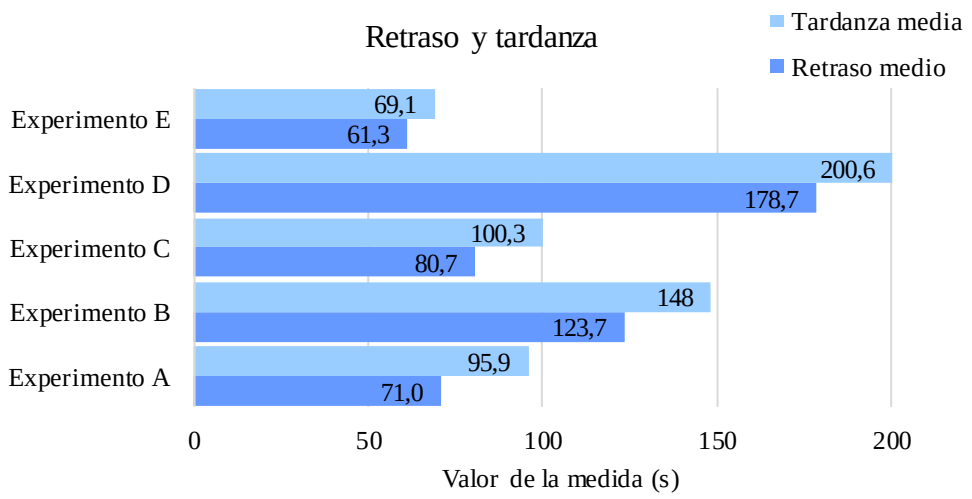

Figura 55. Resultados de experimentos – retraso medio y tardanza media.

En el caso de esta instancia, la técnica FAM+SCM (Experimento E) reportó mejores medidas de desempeño, tanto de *makespan* como de tardanza y retraso medias, en comparación con el escenario sin ocurrencia de perturbación (Experimento A). Se aclara que el objetivo de las técnicas de reactividad, en este trabajo de investigación, no es optimizar las medidas de desempeño con respecto al escenario base (Experimento A), sino por el contrario reducir la degradación de las medidas de desempeño sustitutas tras la ocurrencia de una perturbación (Experimento B).

Por lo anterior, la técnica mencionada representa un caso particular en la instancia mrj_151.

Resultados

En esta sección se presentan los resultados de los experimentos realizados a cada las instancias presentadas en la Tabla 18.

La Tabla 24 presenta el valor de *makespan* obtenido por la técnica predictiva del DSS, algoritmo genético. Además, presenta los resultados del Experimento A, es decir, los valores finales de las medidas de desempeño objetivo del DSS. Estos resultados refuerzan que, la simplificación de las restricciones del problema es una causa de la brecha entre la teoría y la práctica en programación de operaciones. Por ende, si la técnica predictiva no incorpora (o no está en capacidad de incorporar) todas las particularidades conocidas del piso de manufactura, las medidas de desempeño sufrirán una degradación, en la ejecución real, con respecto a lo obtenido de manera teórica.

En este caso particular, el algoritmo genético no incorporó en su totalidad el diseño del piso de manufactura, lo que causó la alteración por colas.

Tabla 24. Resultados generales del Experimento A.

	GA	Experimento A				
Instancia	**mkp**	**mkp**	**tlt**	**mlt**	**ttd**	**mtd**
mrj_101	376	507,4	343,4	34,3	493,4	49,3
mrj_102	373	444,0	343,7	34,4	343,7	34,4
mrj_103	386	517,0	342,4	34,2	597,8	59,8
mrj_151	566	778,3	1064,3	71,0	1438,5	95,9
mrj_152	529	704,6	797,8	53,2	1071,6	71,4
mrj_153	543	682,0	940,0	62,7	999,4	66,6
mrj_201	760	984,1	2331,2	116,6	2469,0	123,5
mrj_202	683	825,8	1188,8	59,4	1399,3	70,0
mrj_203	716	950,5	1677,1	83,9	1865,7	93,3
mrj_251	845	1071,5	1135,8	45,4	1432,8	57,3
mrj_252	810	1062,6	3090,4	123,6	3256,4	130,3
mrj_253	912	1183,0	2262,2	90,5	2849,8	114,0
mrj_301	1099	1352,6	3496,9	116,6	4167,2	138,9
mrj_302	1002	1240,9	3820,4	127,3	4209,9	140,3
mrj_303	1074	1298,6	2482,9	82,8	2893,2	96,4

La Tabla 25 resume los resultados del Experimento B. Estos resultados refuerzan que, las técnicas tradicionales de programación no enfrentan de manera adecuada las complejidades de los entornos de manufactura reales. Por ende, no están en capacidad de asegurar la optimalidad de medidas de desempeño definidas, en sistemas dinámicos que están sujetos a la ocurrencia de eventos aleatorios no planeados.

Se refuerza también que, una función objetivo simple es una causa de la brecha entre la teoría y la práctica en programación de operaciones. En este caso, en la ejecución real y tras la perturbación, las medidas de retraso y tardanza (parámetros que no fueron optimizados en el GA) sufren una degradación porcentual mayor que la que sufre la medida de *makespan* (único parámetro optimizado por el GA). El retraso y la tardanza se degradan en promedio un +138,1% y +112,7%, respectivamente, en comparación con +25,6% (*makespan*). Es decir, las medidas no incorporadas en modelos teóricos tienden a mostrar una degradación mayor en entornos de ejecución real.

A pesar de que los modelos tradicionales permiten la incorporación de funciones multiobjetivo, la elección de las diferentes medidas de desempeño debe ser un proceso riguroso (representación adecuada del sistema), pues un mayor número de objetivos implica una mayor dificultad en términos de solución computacional.

Tabla 25. Resultados generales del Experimento B.

				Experimento B				
Instancia	**mkp**	**A%**	**tlt**	**mlt**	**A%**	**ttd**	**mtd**	**A%**
mrj_101	582,2	+14,7%	597,6	59,8	+74,0%	747,6	74,8	+51,5%
mrj_102	570,0	+28,4%	716,9	71,7	+108,6%	716,9	71,7	+108,6%
mrj_103	651,0	+25,9%	852,0	85,2	+148,8%	1090,8	109,1	+82,5%
mrj_151	921,0	+18,3%	1855,8	123,7	+74,4%	2220,6	148,0	+54,4%
mrj_152	864,0	+22,6%	1934,7	129,0	+142,5%	2045,5	136,4	+90,9%
mrj_153	854,6	+25,3%	1372,0	91,5	+46,0%	1573,6	104,9	+57,5%
mrj_201	1188,6	+20,8%	4439,5	222,0	+90,4%	4645,8	232,3	+88,2%
mrj_202	1113,2	+34,8%	2629,1	131,5	+121,2%	2776,5	138,8	+98,4%
mrj_203	1098,0	+15,5%	3763,8	188,2	+124,4%	3885,0	194,2	+108,2%
mrj_251	1418,3	+32,4%	4828,3	193,1	+325,1%	5267,8	210,7	+267,7%
mrj_252	1332,6	+25,4%	6245,4	249,8	+102,1%	6452,6	258,1	+98,2%
mrj_253	1566,2	+32,4%	6556,4	262,3	+189,8%	7028,4	281,1	+146,6%
mrj_301	1672,9	+23,7%	8185,0	272,8	+134,1%	8244,9	274,8	+97,9%
mrj_302	1643,0	+32,4%	8812,0	293,7	+130,7%	9136,7	304,6	+117,0%
mrj_303	1702,0	+31,1%	8941,7	298,1	+260,1%	9363,1	312,1	+223,6%
Promedio		+25,6%			+138,1%			+112,7%

La Tabla 26, Tabla 27 y Tabla 28 resumen los resultados de los experimentos C, D y E. Estos resultados refuerzan que, la estrategia de programación dinámica predictiva-reactiva, es un enfoque valido para reducir la brecha entre la teoría y la practica en programación de operaciones. Lo anterior se justifica en que, la actualización de la programación de operaciones durante su ejecución, por medio de técnicas reactivas, reduce la degradación de medidas de desempeño que caracterizan el sistema de manufactura. Es decir, las medidas de desempeño reales (practica) están más cerca del valor inicial (teoría) tras la actualización, que en comparación con escenarios en los que no se ejecuta ninguna estrategia (Experimento B).

La medida de desempeño con mejor resultado (menor degradación), fue el *makespan*, ya que como se indicó antes, esta medida fue la única que se consideró en la programación predictiva. Sin embargo, las técnicas reactivas también lograron reducir la degradación de las medidas de retraso y tardanza.

La técnica FAM tuvo un desempeño positivo. Es decir, al realizar la actualización de operaciones por medio de esta técnica, el valor promedio de las medidas de desempeño disminuyó con respecto al Experimento B (no hay ejecución de técnicas reactivas). En promedio el *makespan* disminuyó 9,4%, el retraso 10,6% y la tardanza 10,0%. Con respecto al escenario de ejecución

ideal (Experimento A), la técnica FAM mostró una degradación promedio de +13,7% para la medida de *makespan*, +110,5% de retraso y +87,8% de tardanza.

En este experimento se logró una reducción de *makespan* en todas las instancias. Mientras que, en algunas instancias se aumentó la degradación de retaso y tardanza con respecto al Experimento B, como la instancia mrj102, mrj_152 o la mrj_202.

Tabla 26. Resultados generales del Experimento C.

	Experimento C										
Instancia	**mkp**	**B%**	**A%**	**tlt**	**mlt**	**B%**	**A%**	**ttd**	**mtd**	**B%**	**A%**
mrj_101	543,0	-6,7%	+7,0%	507,5	50,8	-15,1%	+47,8%	727,2	72,7	-2,7%	+47,4%
mrj_102	548,1	-3,8%	+23,4%	742,5	74,2	+3,6%	+116,0%	742,5	74,2	+3,6%	+116,0%
mrj_103	597,2	-8,3%	+15,5%	902,0	90,2	+5,9%	+163,4%	1160,9	116,1	+6,4%	+94,2%
mrj_151	838,1	-9,0%	+7,7%	1209,9	80,7	-34,8%	+13,7%	1505,1	100,3	-32,2%	+4,6%
mrj_152	832,7	-3,6%	+18,2%	2285,5	152,4	+18,1%	+186,5%	2424,3	161,6	+18,5%	+126,2%
mrj_153	790,2	-7,5%	+15,9%	1450,0	96,7	+5,7%	+54,3%	1661,9	110,8	+5,6%	+66,3%
mrj_201	1089,3	-8,4%	+10,7%	3261,5	163,1	-26,5%	+39,9%	3329,5	166,5	-28,3%	+34,9%
mrj_202	1018,6	-8,5%	+23,3%	2846,9	142,3	+8,3%	+139,5%	3037,4	151,9	+9,4%	+117,1%
mrj_203	1032,5	-6,0%	+8,6%	3170,8	158,5	-15,8%	+89,1%	3216,1	160,8	-17,2%	+72,4%
mrj_251	1188,9	-16,2%	+11,0%	4355,7	174,2	-9,8%	+283,5%	4455,7	178,2	-15,4%	+211,0%
mrj_252	1192,5	-10,5%	+12,2%	5201,2	208,0	-16,7%	+68,3%	5435,9	217,4	-15,8%	+66,9%
mrj_253	1243,7	-20,6%	+5,1%	4093,0	163,7	-37,6%	+80,9%	4406,1	176,2	-37,3%	+54,6%
mrj_301	1523,1	-9,0%	+12,6%	7706,4	256,9	-5,8%	+120,4%	7727,3	257,6	-6,3%	+85,4%
mrj_302	1499,9	-8,7%	+20,9%	8564,9	285,5	-2,8%	+124,2%	8942,5	298,1	-2,1%	+112,4%
mrj_303	1468,6	-13,7%	+13,1%	5719,9	190,7	-36,0%	+130,4%	6015,0	200,5	-35,8%	+107,9%
Promedio		-9,4%	+13,7%			-10,6%	+110,5%			-10,0%	+87,8%

La técnica SCM tuvo un desempeño negativo. Es decir, tras la actualización de las decisiones de programación conducidas por esta técnica, el valor promedio de las medidas de desempeño aumentó con respecto al Experimento B (en el cual no se ejecuta técnica reactiva tras la perturbación). En promedio el *makespan* aumentó 8,0%, el retraso 5,1% y la tardanza 4,2%.

A pesar de lo anterior, no se puede descartar la utilización de esta técnica. Por ejemplo, instancias como la mrj_101, mrj_201 o mrj_302, obtuvieron degradaciones de las medidas de desempeño menores a lo obtenido en el Experimento B.

Tabla 27. Resultados generales del Experimento D.

Experimento D											
Instancia	**mkp**	**B%**	**A%**	**tlt**	**mlt**	**B%**	**A%**	**ttd**	**mtd**	**B%**	**A%**
mrj_101	565,0	-3,0%	+11,4%	560,8	56,1	-6,2%	+63,3%	710,8	71,1	-4,9%	+44,1%
mrj_102	662,0	+16,1%	+49,1%	601,8	60,2	-16,1%	+75,1%	601,8	60,2	-16,1%	+75,1%
mrj_103	700,0	+7,5%	+35,4%	803,8	80,4	-5,7%	+134,8%	1055,6	105,6	-3,2%	+76,6%
mrj_151	1123,0	+21,9%	+44,3%	2680,7	178,7	+44,4%	+151,9%	3008,7	200,6	+35,5%	+109,2%
mrj_152	1183,2	+36,9%	+67,9%	3239,9	216,0	+67,5%	+306,1%	3350,7	223,4	+63,8%	+212,7%
mrj_153	816,2	-4,5%	+19,7%	1389,5	92,6	+1,3%	+47,8%	1558,5	103,9	-1,0%	+55,9%
mrj_201	1074,7	-9,6%	+9,2%	3549,6	177,5	-20,0%	+52,3%	3782,1	189,1	-18,6%	+53,2%
mrj_202	1097,0	-1,5%	+32,8%	2779,4	139,0	+5,7%	+133,8%	2841,6	142,1	+2,3%	+103,1%
mrj_203	1145,2	+4,3%	+20,5%	2919,8	146,0	-22,4%	+74,1%	3200,0	160,0	-17,6%	+71,5%
mrj_251	1542,4	+8,7%	+43,9%	6260,4	250,4	+29,7%	+451,2%	6472,5	258,9	+22,9%	+351,7%
mrj_252	1747,0	+31,1%	+64,4%	7954,7	318,2	+27,4%	+157,4%	8104,4	324,2	+25,6%	+148,9%
mrj_253	1826,1	+16,6%	+54,4%	7007,7	280,3	+6,9%	+209,8%	7341,7	293,7	+4,5%	+157,6%
mrj_301	1735,2	+3,7%	+28,3%	7002,6	233,4	-14,4%	+100,3%	7102,0	236,7	-13,9%	+70,4%
mrj_302	1549,6	-5,7%	+24,9%	7947,3	264,9	-9,8%	+108,0%	8416,7	280,6	-7,9%	+99,9%
mrj_303	1642,4	-3,5%	+26,5%	7912,5	263,8	-11,5%	+218,7%	8515,9	283,9	-9,0%	+194,3%
Promedio		+8,0%	+35,5%			+5,1%	+152,3%			+4,2%	+121,6%

En general, la técnica reactiva que mejor desempeño tuvo fue la SCM+FAM. Bajo esta técnica, tras la implementación del DSS a todas las instancias, la medida de *makespan* tuvo en promedio tras la ocurrencia de la perturbación, una degradación de +5,9% con respecto a las medidas del Experimento A, en el cual ninguna perturbación ocurrió. La técnica FAM+SCM disminuyó el efecto de la perturbación sobre la medida de *makespan* en los escenarios propuestos, ya que, se pasó de una degradación promedio de +25,6% (Experimento B, en el cual no se ejecuta ninguna actualización de la programación) a una degradación de +5,9% (Experimento E). Finalmente, bajo la técnica SCM+FAM, la reducción promedio de la degradación de *makespan* con respecto al Experimento B fue de 15,6%.

Las medidas de retraso y tardanza siguen el mismo comportamiento que el *makespan*. La mejor técnica para reducir la degradación de estas medidas es la FAM+SCM. Con relación al Experimento A, estas medidas sufrieron una degradación promedio de +54,6% y +40,0%, respectivamente. En comparación con el Experimento B, en el cual sucede la perturbación, pero no se activa ninguna técnica reactiva, se pasó de una degradación promedio de +138,1% a +54,6% (medida de retraso) y se pasó de una degradación promedio de +112,7% a +40,0% (medida de tardanza).

Tabla 28. Resultados generales del Experimento E.

	Experimento E										
Instancia	**mkp**	**B%**	**A%**	**tlt**	**mlt**	**B%**	**A%**	**ttd**	**mtd**	**B%**	**A%**
mrj_101	531,6	-8,7%	+4,8%	447,8	44,8	-25,1%	+30,4%	627,6	62,8	-16,1%	+27,2%
mrj_102	493,2	-13,5%	+11,1%	565,1	56,5	-21,2%	+64,4%	565,1	56,5	-21,2%	+64,4%
mrj_103	536,4	-17,6%	+3,8%	523,9	52,4	-38,5%	+53,0%	797,8	79,8	-26,9%	+33,5%
mrj_151	727,2	-21,0%	-6,6%	919,6	61,3	-50,4%	-13,6%	1037,2	69,1	-53,3%	-27,9%
mrj_152	774,1	-10,4%	+9,9%	1567,8	104,5	-19,0%	+96,5%	1678,6	111,9	-17,9%	+56,6%
mrj_153	790,4	-7,5%	+15,9%	1381,0	92,1	+0,7%	+46,9%	1477,0	98,5	-6,1%	+47,8%
mrj_201	1068,8	-10,1%	+8,6%	2628,5	131,4	-40,8%	+12,8%	2804,7	140,2	-39,6%	+13,6%
mrj_202	919,5	-17,4%	+11,3%	2326,8	116,3	-11,5%	+95,7%	2445,4	122,3	-11,9%	+74,8%
mrj_203	930,4	-15,3%	-2,1%	1548,7	77,4	-58,9%	-7,7%	1765,5	88,3	-54,6%	-5,4%
mrj_251	1109,2	-21,8%	+3,5%	3588,9	143,6	-25,7%	+216,0%	3707,5	148,3	-29,6%	+158,8%
mrj_252	1112,1	-16,5%	+4,7%	3477,5	139,1	-44,3%	+12,5%	3556,6	142,3	-44,9%	+9,2%
mrj_253	1186,4	-24,2%	+0,3%	3358,2	134,3	-48,8%	+48,4%	3647,4	145,9	-48,1%	+28,0%
mrj_301	1462,2	-12,6%	+8,1%	4920,2	164,0	-39,9%	+40,7%	5055,6	168,5	-38,7%	+21,3%
mrj_302	1354,8	-17,5%	+9,2%	5708,0	190,3	-35,2%	+49,4%	5839,4	194,6	-36,1%	+38,7%
mrj_303	1368,7	-19,6%	+5,4%	4310,7	143,7	-51,8%	+73,6%	4621,0	154,0	-50,6%	+59,7%
Promedio		-15,6%	+5,9%			-34,0%	+54,6%			-33,0%	+40,0%

La Figura 56 y Figura 57 resumen los resultados presentados en las tablas anteriores. Estas muestran la degradación promedio de las medidas de desempeño con respecto al escenario de ejecución ideal en el que no ocurren perturbaciones (Experimento A). Se ve como la técnica FAM+SCM logra la mayor reducción de la degradación de las medidas de desempeño.

La Figura 58 y Figura 59 muestran el cambio de degradación promedio de las medidas de desempeño con respecto al escenario en el que ocurrió la perturbación pero no se realizó una actualización de la programación (Experimento B). Se reafirme el hecho que, la técnica FAM+SCM logra mejores resultados que las otras dos técnicas reactivas.

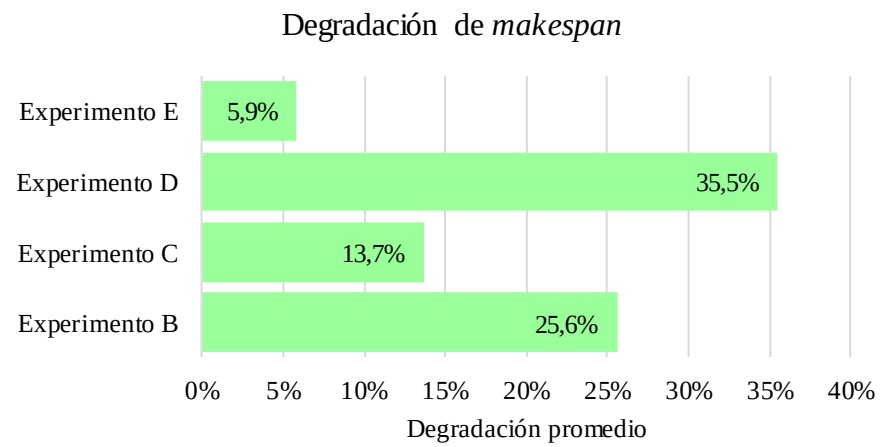

Figura 56. Resultado de los experimentos - degradación promedio de *makespan* (A%).

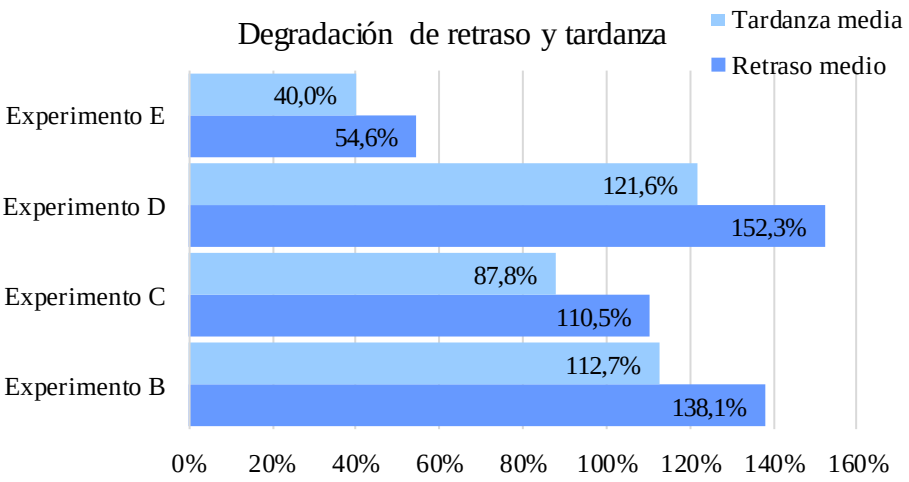

Figura 57. Resultado de los experimentos - degradación promedio de retraso y tardanza media (A%).

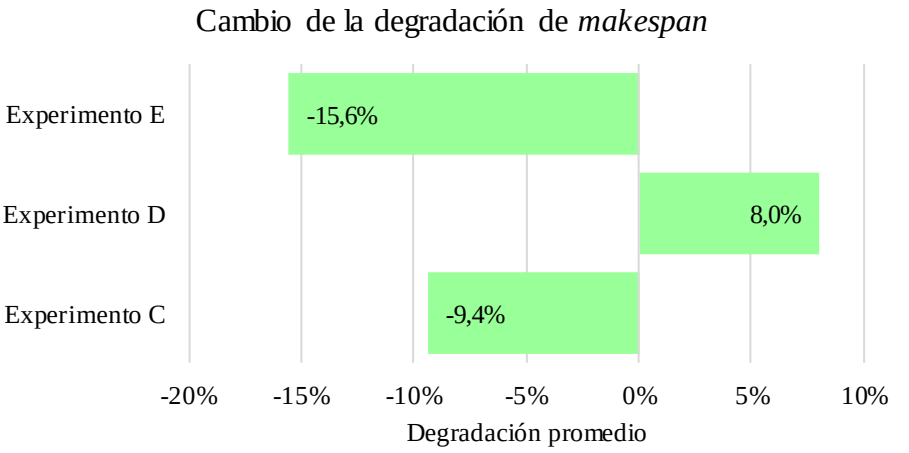

Figura 58. Resultado de los experimentos – cambio de la degradación promedio de *makespan* (B%).

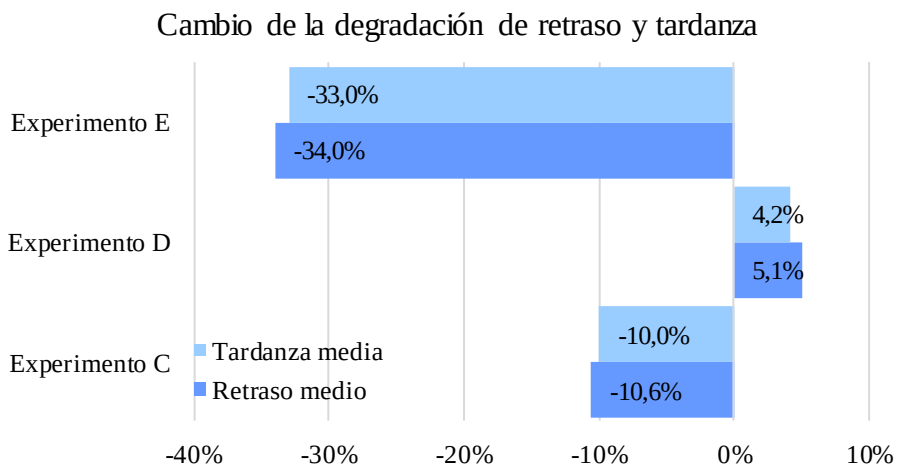

Figura 59. Resultado de los experimentos – cambio de la degradación promedio de retraso y tardanza media (B%).

La Figura 60 muestra el valor de *makespan* obtenido por cada una de las instancias, presentadas en la Tabla 18, de acuerdo con su índice de complejidad asociado. Se presentan los resultados de *makespan* del algoritmo genético, Experimento A y B.

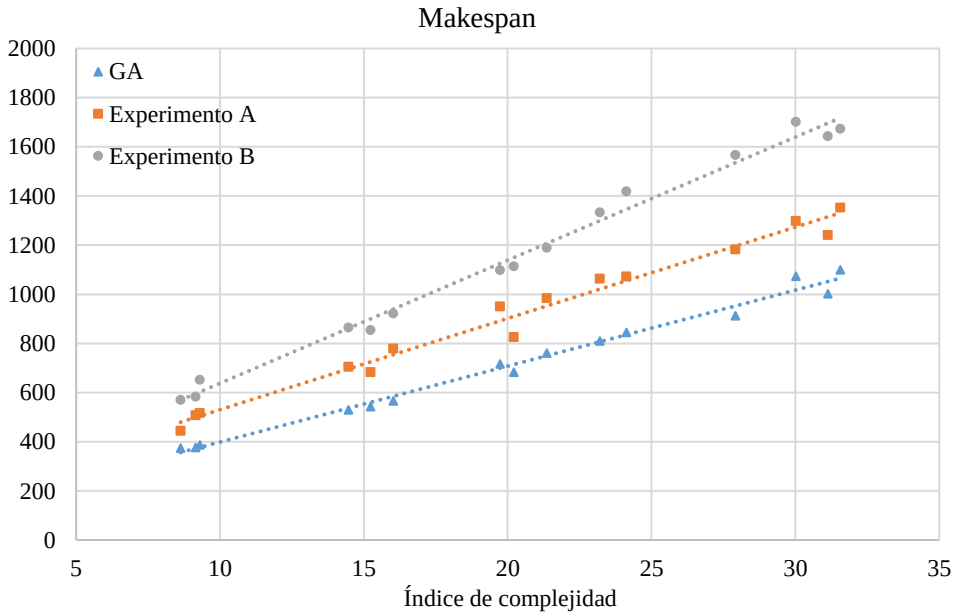

Figura 60. Resultado de los experimentos – *makespan* de cada instancia: GA, A y B.

La Figura 61 muestra el valor de *makespan* obtenido por cada una de las instancias, presentadas en la Tabla 18, de acuerdo con su índice de complejidad asociado. Se presentan los resultados de *makespan* de los experimentos C, D y E.

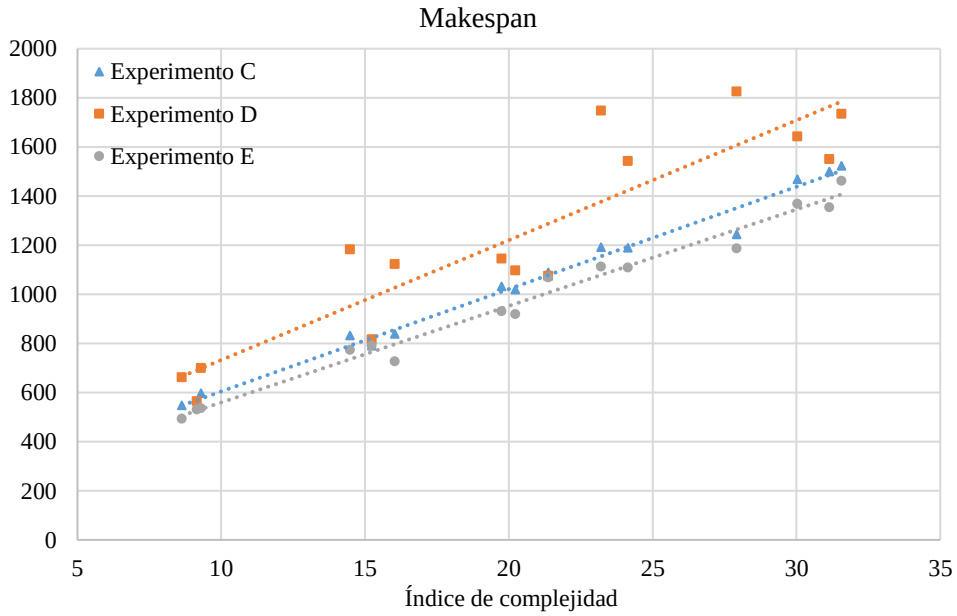

Figura 61. Resultado de los experimentos – *makespan* de cada instancia: C, D y E.

Las gráficas anteriores permiten identificar un comportamiento lineal del valor de *makespan* (variable dependiente) a partir del índice de complejidad de la instancia (variable independiente). Lo anterior implica que, el valor de *makespan* obtenido por cada una de las técnicas (heurísticas y metaheurísticas) propuestas en este trabajo, puede ser caracterizado por una ecuación lineal cuando se aplica al caso de estudio: célula de manufactura AIP.

Son de particular interés, para trabajos futuros que puedan surgir de esta investigación, los comportamientos de las rectas GA y Experimento A, ya que están representan casos genéricos de posibles ordenes de manufactura a procesar en la célula AIP. El comportamiento de estas rectas puede servir para validar los resultados obtenidos por otros investigadores. Las rectas de los experimentos B, C, D y E son casos particulares, que corresponden a la perturbación propuesta, por ende, solo tendrían validez ejecutado la misma perturbación.

En la Tabla 29 se presentan las ecuaciones correspondientes y el coeficiente de determinación asociado. Se considera *x* el índice de complejidad de la instancia y el valor estimado de *makespan*.

Tabla 29. Ecuaciones de los experimentos realizados - *makespan*

Experimento	**Ecuación de la recta**	**R^2**
GA	*y = 30,92x + 88,68*	0,9871
Experimento A	*y = 37,16x + 158,39*	0,9805
Experimento B	*y = 50,12x + 135,67*	0,9896

Tabla 29. Ecuaciones de los experimentos realizados - *makespan*

Experimento	**Ecuación de la recta**	**R^2**
Experimento C	*y = 41,66x + 187,95*	0,9874
Experimento D	*y = 48,83x + 243,64*	0,8306
Experimento E	*y = 39,30x + 166,03*	0,9821

Conclusiones y trabajo futuro

La programación de operaciones tradicional en entornos de manufactura *job shop* se concentra en programaciones centralizadas. Bajo la revolución industrial que representa la Industria 4.0, la programación debe tratar con sistemas de fabricación inteligente y distribuidos, respaldados por tecnologías de manufactura novedosas como sistemas ciber-físicos o gemelo digital. El enfoque de investigación en la programación de operaciones necesita cambiarse a un modelado y optimización de programaciones inteligente y distribuidas. Los sistemas de soporte de decisión en manufactura pueden ser integrados a los nuevos avances tecnológicos desarrollados en esta revolución y están en capacidad de definir los parámetros para una actualización de la programación, de manera distribuida, cambiando las decisiones de programación automáticamente y en tiempo real para minimizar la degradación del rendimiento global, causada por eventos inesperados.

El propósito de este trabajo de investigación es crear un sistema de soporte de decisiones de programación de operaciones que guie la ejecución operacional de la programación de operaciones inicial (*offline scheduling*) y responda a eventos no planeados en tiempo real (*online scheduling*). El objetivo principal era diseñar un sistema que redujera la brecha entre la teoría y la práctica en la ejecución de operaciones con el fin de reducir la degradación con respecto a la programación inicial en ambientes flexibles de manufactura. Para este fin, esta contribución definió dos contextos de la programación de operaciones, uno a nivel empresarial y otro a nivel investigativo. En el primer contexto, se mostró la ubicación de la programación de operaciones en los sistemas de manufactura tradicionales, se indicaron sistemas de soporte clásicos a nivel empresarial y se definió el marco de la toma de decisiones en presencia de perturbaciones. En el segundo contexto, se presentó la definición de problemas de programación *job shop*, la clasificación de este tipo de problemas, la caracterización de problemas de programación dinámicos y finalmente los métodos de solución para problemas de programación de operaciones. Desde este punto de partida, este trabajo a disertación se propuso hacer las siguientes contribuciones.

La primera contribución se centra en los resultados de la revisión de la literatura. Primero, se encontró que la programación dinámica es aún un tema en desarrollo y un enfoque que puede mejorar los resultados de la ejecución real de la programación de operaciones. Segundo, se encontró que en investigaciones previas con enfoque dinámico predictivo-reactivo, se tiende a usar el mismo mecanismo (técnica) de solución, tanto en la etapa *offline* como *online*, por lo que un sistema con mecanismos de solución diferentes es requerido. Tercero, se confirmó que hay una necesidad de llevar a la práctica investigaciones que contemplen la ocurrencia de eventos inesperados en tiempo real garantizando una baja degradación del desempeño del sistema.

Finalmente, la revisión de la literatura mostró la necesidad de responder las siguientes preguntas: (1) Cuándo: debe ser conocido el momento para ejecutar una reprogramación. (2) Qué: la reprogramación debe reasignar nuevas decisiones a máquinas y trabajos para cumplir el objetivos u objetivos establecidos. (3) Cómo: se deben establecer las técnicas que ejecuten la reprogramación que contribuyan al alcance del objetivo o medida de desempeño establecida.

La segunda contribución fue la definición de un marco general para el sistema de soporte de decisiones de producción. El DSS propuesto se compone de cuatro subsistemas con una función específica cada uno. En el subsistema de gestión de datos se propone la información a recolectar tanto de un sistema ERP como del piso de manufactura. De la misma manera, se establece el marco de monitoreo en el piso de manufactura presentando los atributos a considerar tanto de trabajos como de máquinas. En los subsistemas de modelos predictivos y reactivos se presentan las programaciones correspondientes, el usuario interactúa con estos únicamente a través de la interfaz. En el subsistema de interfaz de usuario se proponen las salidas, de texto y gráficas, a las que el usuario puede acceder. Esta interfaz propone una representación ciber-física del sistema de manufactura.

Por la definición de los modelos que componen el DSS, este puede ser ejecutado en cualquier entorno de manufactura flexible, pues el entorno *flexible job shop* es el caso general de los entornos de manufactura. Por ejemplo, un entorno *flexible flow shop* es un caso particular, en el cual todos los trabajos a procesar siguen la misma ruta de procesamiento.

La tercera contribución fue la programación de un algoritmo genético para solucionar el problema de programación de operaciones: *flexible job shop scheduling problem*, que propone una representación novedosa de la codificación del cromosoma por medio de dos sub-cadenas. En este algoritmo se definió como medida de desempeño el tiempo de finalización de la última operación en el sistema, *makespan* y el objetivo principal era la minimización es esta. Adicionalmente, el algoritmo incorpora dos restricciones no clásicas en la formulación del problema. La primera es el tiempo de transporte entre máquinas (existiendo diferentes rutas para llegar a una máquina). La segunda es la consideración de la capacidad máxima de trabajos en el sistema. La implementación del algoritmo con las instancias de prueba de Brandimarte mostró buenos resultados.

La cuarta contribución fue la programación de dos heurísticas para actualizar la programación de operaciones inicial tras la ocurrencia de una perturbación en el sistema de manufactura relacionada con la falla de una máquina. Estas técnicas están en capacidad de cambiar las decisiones, previamente establecidas, de los agentes del sistema: máquinas y productos. Dado que la implementación de estas técnicas se realiza de manera local, en el piso de manufactura, se puede decir, que los agentes están en capacidad de tomar decisiones autónomas de manera distribuida. Estas técnicas son las encargadas de la reducción de la degradación de las medidas de desempeño. La actualización de la programación por medio de las heurísticas propuestas está en

capacidad de considerar perturbaciones como aparición de ordenes urgentes *(rush order arrival*) o cancelación de ordenes (*order cancellation*). Por ende, el sistema propuesto contempla escenarios con demanda incierta.

La quinta contribución fue la definición de un módulo de la política de reprogramación, el cual ejecuta las técnicas anteriormente mencionadas cuando el proceso lo requiere. Este módulo, que responde a las preguntas (1) Cuándo, (2) Qué y (3) Cómo identificadas en la revisión de la literatura, esta soportado en el DSS con el módulo de monitoreo, el cual, de manera continua, monitorea el piso de manufactura y en el caso de una perturbación inicia el proceso de reconfiguración. El módulo de política de reprogramación propone el cumplimiento de una condición de reprogramación y define el punto físico de reprogramación.

El sistema de soporte de decisiones propuesto fue validado en un escenario simulado de un sistema de manufactura flexible real ubicado en Valenciennes (Francia), llamado AIP-PRIMECA Valenciennes. Se diseñaron cinco escenarios experimentales de simulación para evaluar el desempeño del DSS midiendo el resultado sobre la medida inicial *makespan* y valorando la capacidad de reactividad del sistema con las medidas de retraso y tardanza. Los cinco experimentos mostraron la viabilidad y la diversidad producida al tener diferentes modos de funcionamiento en el DSS.

El alcance del estudio fue limitado al desarrollo del marco general del sistema de soporte de decisiones. En este trabajo se presenta la parte teórica y funcional de sistema en lugar de su integración y experimentación en un sistema de manufactura real. Además, el DSS pretende integrar los métodos tradicionales de programación a sistemas inteligentes de manufactura en vez de realizar una nueva investigación teórica en esta área. Finalmente, la detección de perturbaciones el piso de manufactura y la correspondiente respuesta de reprogramación suponen utilizar las ventajas de las tecnologías ya existentes de la Industria 4.0.

De este trabajo de investigación pueden derivarse diferentes trabajos futuros. Estos están relacionados con la implementación del DSS propuesto, el marco de reprogramación predictiva-reactiva, el módulo predictivo, el módulo reactivo y pruebas experimentales con la célula AIP. En relación con la implementación del DSS propuesto, se establecen las bases funcionales de cada subsistema. Se propone seguir el estándar de manufactura ISA – 95 para desarrollar la interfaz que integre el sistema de planeación empresarial (ERP) y el sistema físico de manufactura al DSS.

En relación con la reprogramación predictiva-reactiva, cada técnica tiene un grado de gobernanza en la ejecución real de las operaciones. Es decir, algunas operaciones se ejecutan de acuerdo con lo definido por la técnica predictiva y algunas otras se ejecutan de acuerdo con lo actualizado por la técnica reactiva. Sin embargo, el nivel exacto de gobernanza de cada técnica es desconocido y esto puede ser de particular importancia para determinar sobre que técnica realizar

un mayor trabajo y hacerla más robusta. Por lo anterior, se propone crear una metodología que mida el grado de gobernanza de ambas técnicas en la ejecución real.

En relación con el módulo predictivo, se propone integrar un parámetro de entrada de la forma grafo *G = (S, V)*, de modo que el algoritmo genético este en capacidad de reconocer el diseño del piso de manufactura sobre el cual generara decisiones. Esto es de particular importancia si se quiere incluir como objetivo, la minimización del tiempo de transporte entre las rutas posibles de transporte entra máquinas. Además, permitiría calcular el tiempo de recirculación de un trabajo dentro del sistema de transporte, en caso de encontrar colas en máquinas. Por esto último, se propone integrar un parámetro de entrada que considere la longitud de la cola que una máquina puede tener.

En relación con el módulo reactivo, los aspectos asociados a la política tienen un impacto sobre la degradación obtenida por el DSS. En el subsistema de gestión de modelos reactivos, es posible que la política desencadene frecuentemente procesos de reprogramación sin dejar un período de estabilización (lo que conduce a un comportamiento nervioso). Para evaluar esto, se propone explorar otros parámetros de la política que logren limitar la frecuencia de reprogramación. Se propone evaluar otros métodos de reprogramación que lleven a valores diferentes de degradación de las medidas de desempeño; utilizar métodos propios de los agentes, como la negociación entre agentes.

En relación con las pruebas experimentales con la célula AIP se propone lo siguiente. Primero, ejecutar el DSS con otras perturbaciones para medir la reactividad del sistema. Para ello, se propone utilizar el trabajo de Trentesaux et al. (2013), el cual propone quince escenarios. Por ejemplo, el incremento de tiempo de procesamiento de todas las operaciones ejecutadas por una máquina o la no disponibilidad de una sección del sistema de transporte por mantenimiento. Segundo, ejecutar instancias adicionales para validar el comportamiento lineal del *makespan* de acuerdo con el índice de complejidad presentado en la Tabla 29. Este estudio puede ser ampliado a otros sistemas de manufactura para caracterizar el valor de las medidas de desempeño de acuerdo con una o varias variables independientes. Finalmente, se propone evaluar la reactividad del DSS al realizar experimentos con perturbaciones que toman lugar cerca al final de la ejecución de la programación.

Se propone evaluar las instancias propuestas en este trabajo de investigación por medio de simulación de eventos discretos con el fin de evaluar ventajas y desventajas de las técnicas de simulación. Los resultados presentados en este trabajo pueden guiar la simulación propuesta.

Bibliografía

Adibi, M. A., Zandieh, M., & Amiri, M. (2010). Multi-objective scheduling of dynamic job shop using variable neighborhood search. *Expert Systems with Applications*, *37*(1), 282–287. https://doi.org/10.1016/j.eswa.2009.05.001

Ahmadi, E., Zandieh, M., Farrokh, M., & Emami, S. M. (2016). A multi objective optimization approach for flexible job shop scheduling problem under random machine breakdown by evolutionary algorithms. *Computers and Operations Research*, *73*, 56–66. https://doi.org/10.1016/j.cor.2016.03.009

Akyol, D. E., & Bayhan, G. M. (2007). A review on evolution of production scheduling with neural networks. *Computers and Industrial Engineering*, *53*(1), 95–122. https://doi.org/10.1016/j.cie.2007.04.006

Alves, F., Varela, M. L. R., Rocha, A. M. A. C., Pereira, A. I., Barbosa, J., & Leitão, P. (2020). Hybrid system for simultaneous job shop scheduling and layout optimization based on multi-agents and genetic algorithm. *Advances in Intelligent Systems and Computing*, *923*, 387–397. https://doi.org/10.1007/978-3-030-14347-3_38

Amjad, M. K., Butt, S. I., Kousar, R., Ahmad, R., Agha, M. H., Faping, Z., … Asgher, U. (2018). Recent research trends in genetic algorithm based flexible job shop scheduling problems. *Mathematical Problems in Engineering*, *2018*, 1–32. https://doi.org/10.1155/2018/9270802

Arica, E., Strandhagen, J. O., & Hvolby, H. H. (2014). Designing a Decision Support System for Production Scheduling Task in Complex and Uncertain Manufacturing Environments. *IFIP Advances in Information and Communication Technology*, *438*(PART 1), 589–596. https://doi.org/10.1007/978-3-662-44739-0_72

Arisha, A., Young, P., & El Baradie, M. (2001). Job Shop Scheduling Problem: an Overview. *International Conference for Flexible Automation and Intelligent Manufacturing (FAIM 01)*, 682–693. Retrieved from https://arrow.tudublin.ie/buschmarcon

Bangsow, S. (2012). *Use cases of discrete event simulation: Appliance and research* (Vol. 9783642287). https://doi.org/10.1007/978-3-642-28777-0

Baykasoğlu, A., & Karaslan, F. S. (2017). Solving comprehensive dynamic job shop scheduling problem by using a GRASP-based approach. *International Journal of Production Research*, *55*(11), 3308–3325. https://doi.org/10.1080/00207543.2017.1306134

Bekkar, A., Guemri, O., Bekrar, A., Aissani, N., Beldjilali, B., & Trentesaux, D. (2016). An Iterative Greedy Insertion Technique for Flexible Job Shop Scheduling Problem. *IFAC-PapersOnLine*, *49*(12), 1956–1961. https://doi.org/10.1016/j.ifacol.2016.07.917

Berglund, M., & Karltun, J. (2007). Human, technological and organizational aspects influencing the production scheduling process. *International Journal of Production Economics*, *110*(1–2), 160–174. https://doi.org/10.1016/j.ijpe.2007.02.024

Bidot, J., Vidal, T., Laborie, P., & Beck, J. C. (2009). A theoretic and practical framework for scheduling in a stochastic environment. *Journal of Scheduling*, *12*(3), 315–344. https://doi.org/10.1007/s10951-008-0080-x

Brailsford, S. (2014). Discrete-event simulation is alive and kicking. *Journal of Simulation*, *8*(1), 1–8. https://doi.org/10.1057/jos.2013.13

Brandimarte, P. (1993). Routing and scheduling in a flexible job shop by tabu search. *Annals of Operations Research*, *41*(3), 157–183. https://doi.org/10.1007/BF02023073

Brooks, G. H., & White, C. R. (1965). An algorithm for finding optimal or near optimal solutions to the production scheduling problem. *Journal of Industrial Engineering*, *16*(1), 34–40.

Browne, J., Dubois, D., Rathmill, K., Sethi, S. P., & Stecke, K. E. (1984). Classification of flexible manufacturing systems. *The FMS Magazine*, *2*(2), 114–117.

Çaliş, B., & Bulkan, S. (2015). A research survey: review of AI solution strategies of job shop scheduling problem. *Journal of Intelligent Manufacturing*, *26*(5), 961–973. https://doi.org/10.1007/s10845-013-0837-8

Chaari, T., Chaabane, S., Aissani, N., & Trentesaux, D. (2014). Scheduling under uncertainty: Survey and research directions. *2014 International Conference on Advanced Logistics and Transport, ICALT 2014*, 229–234. https://doi.org/10.1109/ICAdLT.2014.6866316

Chaudhry, I. A., & Khan, A. A. (2016). A research survey: review of flexible job shop scheduling techniques. *International Transactions in Operational Research*, *23*(3), 551–591. https://doi.org/10.1111/itor.12199

Chen, H., Ihlow, J., & Lehmann, C. (1999). A genetic algorithm for flexible job-shop scheduling. *Proceedings 1999 IEEE International Conference on Robotics and Automation*, *2*(May), 1120–1125. https://doi.org/10.1109/ROBOT.1999.772512

Chen, K., & Ji, P. (2007). A mixed integer programming model for advanced planning and scheduling (APS). *European Journal of Operational Research*, *181*(1), 515–522. https://doi.org/10.1016/j.ejor.2006.06.018

Cowling, P., & Johansson, M. (2002). Using real time information for effective dynamic scheduling. *European Journal of Operational Research*, *139*(2), 230–244. https://doi.org/10.1016/S0377-2217(01)00355-1

De Snoo, C., Van Wezel, W., Wortmann, J. C., & Gaalman, G. J. C. (2011). Coordination activities of human planners during rescheduling: Case analysis and event handling procedure. *International Journal of Production Research*, *49*(7), 2101–2122. https://doi.org/10.1080/00207541003639626

De Ugarte, B. S., Artiba, A., & Pellerin, R. (2009). Manufacturing execution system - A literature review. *Production Planning and Control*, *20*(6), 525–539. https://doi.org/10.1080/09537280902938613

Elgendy, A. E., Hussein, M., & Elhakeem, A. (2017). Optimizing dynamic flexible job shop scheduling problem based on genetic algorithm. *International Journal of Current Engineering and Technology*, *77*(22), 2277–4106. Retrieved from http://inpressco.com/category/ijcet

Fazel Zarandi, M. H., Sadat Asl, A. A., Sotudian, S., & Castillo, O. (2018). A state of the art review of intelligent scheduling. *Artificial Intelligence Review*. https://doi.org/10.1007/s10462-018-9667-6

Felsberger, A., Oberegger, B., & Reiner, G. (2017). A review of decision support systems for manufacturing systems. *CEUR Workshop Proceedings*, *1793*. https://doi.org/10.5281/ZENODO.495120

Ferreirinha, L., Santos, A. S., Madureira, A. M., Varela, M. L. R., & Bastos, J. A. (2020). Decision support tool for dynamic scheduling. *Advances in Intelligent Systems and Computing*, *923*, 418–427. https://doi.org/10.1007/978-3-030-14347-3_41

French, S. (1982). *Sequencing and Scheduling: An Introduction to the Mathematics of the Job-Shop*. https://doi.org/10.2307/2581219

Fuchigami, H. Y., & Rangel, S. (2018). A survey of case studies in production scheduling: Analysis and perspectives. *Journal of Computational Science*, *25*, 425–436. https://doi.org/10.1016/j.jocs.2017.06.004

Garey, M. R., & Johnson, D. S. (1975). Complexity results for multiprocessor scheduling under

resource constraints. *SIAM Journal on Computing*, *4*(4), 397–411. https://doi.org/10.1137/0204035

Garey, M. R., & Johnson, D. S. (1979). Computers and Intractability: A Guide to the Theory of NP-Completeness. In *W.H. Freeman*. https://doi.org/10.1137/1024022

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning* (1st ed.). USA: Addison-Wesley Longman Publishing Co., Inc.

Gorry, G. A., & Scott Morton, M. S. (1971). *A framework for management information systems*.

Hefetz, N., & Adiri, I. (1982). An efficient optimal algorithm for the two-machines unit-time job shop schedule-length problem. *Mathematics of Operations Research*, *7*(3), 354–360. https://doi.org/10.1287/moor.7.3.354

Hermann, M., Pentek, T., & Otto, B. (2016). Design principles for industrie 4.0 scenarios. *Proceedings of the Annual Hawaii International Conference on System Sciences*, *2016-March*, 3928–3937. https://doi.org/10.1109/HICSS.2016.488

Ho, N. B., & Tay, J. C. (2004). GENACE : An efficient cultural algorithm solving the flexible job-shop problem. *Congress on Evolutionary Computation, 2004. CEC2004.*, 1759–1766. https://doi.org/10.1109/CEC.2004.1331108

Holland, J. H. (1975). *Adaptation in natural and artificial systems*. Ann Arbor, Mich, USA: University of Michigan Press.

Horng, S. C., Lin, S. S., & Yang, F. Y. (2012). Evolutionary algorithm for stochastic job shop scheduling with random processing time. *Expert Systems with Applications*, *39*(3), 3603–3610. https://doi.org/10.1016/j.eswa.2011.09.050

Hvolby, H. H., & Steger-Jensen, K. (2010). Technical and industrial issues of Advanced Planning and Scheduling (APS) systems. *Computers in Industry*, *61*(9), 845–851. https://doi.org/10.1016/j.compind.2010.07.009

Ivert, L. K. (2012). Shop floor characteristics influencing the use of advanced planning and scheduling systems. *Production Planning and Control*, *23*(6), 452–467. https://doi.org/10.1080/09537287.2011.564218

Jain, A. S., & Meeran, S. (1998). *A state-of-the-art review of job-shop scheduling techniques*. Retrieved from https://pdfs.semanticscholar.org/fae0/2d96dc5780627f5b7582f548395c03db9f83.pdf

Jia, H. Z., Nee, A. Y. C., Fuh, J. Y. H., & Zhang, Y. F. (2003). A modified genetic algorithm for distributed scheduling problems. *Journal of Intelligent Manufacturing*, *14*(3–4), 351–362. https://doi.org/10.1023/A:1024653810491

Jiménez, J.-F. (2017). *Dynamic and hybrid architecture for the optimal reconfiguration of control systems : application to manufacturing control* (Université de Valenciennes et du Hainaut-Cambresis). Retrieved from https://tel.archives-ouvertes.fr/tel-01668578

Johnson, S. M. (1954). Optimal two- and three-stage production schedules with setup times included. *Naval Research Logistics Quarterly*, *1*(1), 61–68. https://doi.org/10.1002/nav.3800010110

Juan, A. A., Barrios, B. B., Vallada, E., Riera, D., & Jorba, J. (2014). A simheuristic algorithm for solving the permutation flow shop problem with stochastic processing times. *Simulation Modelling Practice and Theory*, *46*, 101–117. https://doi.org/10.1016/j.simpat.2014.02.005

Kagermann, H. (2015). Change through digitization—value creation in the age of industry 4.0. In *Management of Permanent Change* (pp. 23–45). https://doi.org/10.1007/978-3-658-05014-6_2

Kalra, M., & Singh, S. (2015). A review of metaheuristic scheduling techniques in cloud

computing. *Egyptian Informatics Journal*, *16*(3), 275–295. https://doi.org/10.1016/j.eij.2015.07.001

Kasie, F. M., Bright, G., & Walker, A. (2017). Decision support systems in manufacturing: a survey and future trends. *Journal of Modelling in Management*, Vol. 12, pp. 432–454. https://doi.org/10.1108/JM2-02-2016-0015

Keen, P. G., & Scott Morton, M. S. (1978). Decision Support Systems: An Organizational Perspective. In *Science*. Addison-Wesley Pub. Co.

Koh, S. C. L., & Saad, S. M. (2003). A holistic approach to diagnose uncertainty in ERP-controlled manufacturing shop floor. *Production Planning and Control*, *14*(3), 273–289. https://doi.org/10.1080/0953728031000111487

Kück, M., Ehm, J., Freitag, M., Frazzon, E. M., & Pimentel, R. (2016). A Data-Driven Simulation-Based Optimisation Approach for Adaptive Scheduling and Control of Dynamic Manufacturing Systems. *Advanced Materials Research*, *1140*, 449–456. https://doi.org/10.4028/www.scientific.net/amr.1140.449

Kulkarni, K., & Venkateswaran, J. (2015). Hybrid approach using simulation-based optimisation for job shop scheduling problems. *Journal of Simulation*, *9*(4), 312–324. https://doi.org/10.1057/jos.2014.40

Kundakci, N., & Kulak, O. (2016). Hybrid genetic algorithms for minimizing makespan in dynamic job shop scheduling problem. *Computers and Industrial Engineering*, *96*, 31–51. https://doi.org/10.1016/j.cie.2016.03.011

Larsen, R., & Pranzo, M. (2019). A framework for dynamic rescheduling problems. *International Journal of Production Research*, *57*(1), 16–33. https://doi.org/10.1080/00207543.2018.1456700

Lee, J., & Lau, C. H. (2007). Corporate Modeling with DSS and ERP Systems. In *Wiley Encyclopedia of Electrical and Electronics Engineering*. https://doi.org/10.1002/047134608X.W3312

Leitão, P. (2009). Agent-based distributed manufacturing control: A state-of-the-art survey. *Engineering Applications of Artificial Intelligence*, *22*(7), 979–991. https://doi.org/10.1016/j.engappai.2008.09.005

Leusin, M., Frazzon, E., Uriona Maldonado, M., Kück, M., & Freitag, M. (2018). Solving the Job-Shop Scheduling Problem in the Industry 4.0 Era. *Technologies*, *6*(4), 107. https://doi.org/10.3390/technologies6040107

Lin, L., Hao, X. C., Gen, M., & Jo, J. B. (2012). Network modeling and evolutionary optimization for scheduling in manufacturing. *Journal of Intelligent Manufacturing*, *23*(6), 2237–2253. https://doi.org/10.1007/s10845-011-0569-6

Liu, Y., Wang, L., Wang, X. V., Xu, X., & Zhang, L. (2019). Scheduling in cloud manufacturing: state-of-the-art and research challenges. *International Journal of Production Research*, Vol. 57, pp. 4854–4879. https://doi.org/10.1080/00207543.2018.1449978

Lomnicki, Z. A. (1965). A "branch-and-bound" algorithm for the exact solution of the three-machine scheduling problem. *Journal of the Operational Research Society*, *16*(1), 89–100. https://doi.org/10.2307/3006687

Macal, C. M. (2016). Everything you need to know about agent-based modelling and simulation. *Journal of Simulation*, *10*(2), 144–156. https://doi.org/10.1057/jos.2016.7

MacCarthy, B. L., & Liu, J. (1993a). A new classification scheme for flexible manufacturing systems. *International Journal of Production Research*, *31*(2), 299–309. https://doi.org/10.1080/00207549308956726

MacCarthy, B. L., & Liu, J. (1993b). Addressing the gap in scheduling research: A review of optimization and heuristic methods in production scheduling. *International Journal of Production Research*, *31*(1), 59–79. https://doi.org/10.1080/00207549308956713

Macchi, M., Polenghi, A., Sottoriva, E., Fumagalli, L., & Negri, E. (2018). A novel scheduling framework: integrating genetic algorithms and discrete event simulation. *International Journal of Management and Decision Making*, *17*(1), 1. https://doi.org/10.1504/ijmdm.2018.10016046

Madureira, A., Ramos, C., & Silva, S. do C. (2003). Using genetic algorithms for dynamic scheduling. *14th Annual Production and Operations Management Society Conference (POMS 2003)*. Retrieved from http://www.pomsmeetings.org/ConfProceedings/001/Papers/PSC-12.1.pdf

Mahdavi, I., Shirazi, B., & Solimanpur, M. (2010). Development of a simulation-based decision support system for controlling stochastic flexible job shop manufacturing systems. *Simulation Modelling Practice and Theory*, *18*(6), 768–786. https://doi.org/10.1016/j.simpat.2010.01.015

Manetti, J. (2001). How technology is transforming manufacturing. *Production and Inventory Management Journal*, *42*(1), 54–64.

Manne, A. S. (1960). On the job-shop scheduling problem. *Operations Research*, *8*(2), 219–223. https://doi.org/10.1287/opre.8.2.219

Mastrolilli, M. (n.d.). Flexible job shop problem. *Http://Www.Idsia.Ch/~monaldo/%0Afjsp.Html*. Retrieved from http://www.idsia.ch/~monaldo/%0Afjsp.html

McMahon, G., & Florian, M. (1975). On scheduling with ready times and due dates to minimize maximum lateness. *Operations Research*, *23*(3), 475–482. https://doi.org/10.1287/opre.23.3.475

Mellor, P. (1966). A review of job shop scheduling. *Journal of the Operational Research Society*, *17*(2), 161. https://doi.org/10.2307/3007281

Mohan, J., Lanka, K., & Rao, A. N. (2019). A review of dynamic job shop scheduling techniques. *Procedia Manufacturing*, *30*, 34–39. https://doi.org/10.1016/j.promfg.2019.02.006

Monostori, L., Váncza, J., & Kumara, S. R. T. (2006). Agent-based systems for manufacturing. *CIRP Annals - Manufacturing Technology*, *55*(2), 697–720. https://doi.org/10.1016/j.cirp.2006.10.004

Ortíz, M. A., Betancourt, L. E., Negrete, K. P., De Felice, F., & Petrillo, A. (2018). Dispatching algorithm for production programming of flexible job-shop systems in the smart factory industry. *Annals of Operations Research*, *264*(1–2), 409–433. https://doi.org/10.1007/s10479-017-2678-x

Ouelhadj, D., & Petrovic, S. (2009). A survey of dynamic scheduling in manufacturing systems. *Journal of Scheduling*, *12*(4), 417–431. https://doi.org/10.1007/s10951-008-0090-8

Parthanadee, P., & Buddhakulsomsiri, J. (2010). Simulation modeling and analysis for production scheduling using real-time dispatching rules: A case study in canned fruit industry. *Computers and Electronics in Agriculture*, *70*(1), 245–255. https://doi.org/10.1016/j.compag.2009.11.002

Pezzella, F., Morganti, G., & Ciaschetti, G. (2008). A genetic algorithm for the flexible job-shop scheduling problem. *Computers and Operations Research*, *35*(10), 3202–3212. https://doi.org/10.1016/j.cor.2007.02.014

Pfeiffer, A., Kádár, B., & Monostori, L. (2007). Stability-oriented evaluation of rescheduling strategies, by using simulation. *Computers in Industry*, *58*(7), 630–643.

https://doi.org/10.1016/j.compind.2007.05.009

Pinedo, M. L. (2016). *Scheduling: theory, algorithms, and systems* (5th ed.). https://doi.org/10.1007/978-0-387-78935-4

Potts, C. N., & Van Wassenhove, L. N. (1985). A branch and bound algorithm for the total weighted tardiness problem. *Operations Research*, *33*(2), 363–377. https://doi.org/10.1287/opre.33.2.363

Rahmani, D., & Ramezanian, R. (2016). A stable reactive approach in dynamic flexible flow shop scheduling with unexpected disruptions: A case study. *Computers and Industrial Engineering*, *98*, 360–372. https://doi.org/10.1016/j.cie.2016.06.018

Rossit, D. A., Tohmé, F., & Frutos, M. (2019). Industry 4.0: Smart Scheduling. *International Journal of Production Research*, *57*(12), 3802–3813. https://doi.org/10.1080/00207543.2018.1504248

Sarin, S. C., Ahn, S., & Bishop, A. B. (1988). An improved branching scheme for the branch and bound procedure of scheduling n jobs on m parallel machines to minimize total weighted flowtime. *International Journal of Production Research*, *26*(7), 1183–1191. https://doi.org/10.1080/00207548808947934

Semini, M., Fauske, H., & Strandhagen, J. O. (2006). Applications of discrete-event simulation to support manufacturing logistics decision-making: A survey. *Proceedings - Winter Simulation Conference*, 1946–1953. https://doi.org/10.1109/WSC.2006.322979

Sharma, P., & Jain, A. (2014). Analysis of dispatching rules in a stochastic dynamic job shop manufacturing system with sequence-dependent setup times. *Frontiers of Mechanical Engineering*, *9*(4), 380–389. https://doi.org/10.1007/s11465-014-0315-9

Sharma, P., & Jain, A. (2016). New setup-oriented dispatching rules for a stochastic dynamic job shop manufacturing system with sequence-dependent setup times. *Concurrent Engineering Research and Applications*, *24*(1), 58–68. https://doi.org/10.1177/1063293X15599814

Shukla, C. S., & Chen, F. F. (1996). The state of the art in intelligent real-time FMS control: A comprehensive survey. *Journal of Intelligent Manufacturing*, *7*(6), 441–455. https://doi.org/10.1007/BF00122834

Siebers, P. O., MacAl, C. M., Garnett, J., Buxton, D., & Pidd, M. (2010, September). Discrete-event simulation is dead, long live agent-based simulation! *Journal of Simulation*, Vol. 4, pp. 204–210. https://doi.org/10.1057/jos.2010.14

Smith, J. S. (2003). Survey on the use of simulation for manufacturing system design and operation. *Journal of Manufacturing Systems*, *22*(2), 157–171. https://doi.org/10.1016/S0278-6125(03)90013-6

Sodhi, M. M. S. (2001). Applications and opportunities for operations research in internet-enabled supply chains and electronic marketplaces. *Interfaces*, *31*(2), 56–69. https://doi.org/10.1287/inte.31.2.56.10633

Steger-Jensen, K., Hvolby, H. H., Nielsen, P., & Nielsen, I. (2011). Advanced planning and scheduling technology. *Production Planning and Control*, *22*(8), 800–808. https://doi.org/10.1080/09537287.2010.543563

Sule, D. R. (2018). Production Planning and Industrial Scheduling: Examples, Case Studies and Applications. In *Production Planning and Industrial Scheduling*. https://doi.org/10.1201/9781420044218

Trentesaux, D., Pach, C., Bekrar, A., Sallez, Y., Berger, T., Bonte, T., … Barbosa, J. (2013). Benchmarking flexible job-shop scheduling and control systems. *Control Engineering Practice*, *21*(9), 1204–1225. https://doi.org/10.1016/j.conengprac.2013.05.004

Turban, E., Aronson, J. E., & Liang, T.-P. (2004). *Decision Support Systems and Intelligent Systems* (7th Editio). Upper Saddle River, NJ, USA: Prentice-Hall, Inc.

Turker, A. K., Aktepe, A., Inal, A. F., Ersoz, O. O., Das, G. S., & Birgoren, B. (2019). A decision support system for dynamic job-shop scheduling using real-time data with simulation. *Mathematics*, *7*(3), 278. https://doi.org/10.3390/math7030278

Varela, M. L. R., & Ribeiro, R. A. (2014). Distributed Manufacturing Scheduling Based on a Dynamic Multi-criteria Decision Model. In L. A. Zadeh, A. M. Abbasov, R. R. Yager, S. N. Shahbazova, & M. Z. Reformat (Eds.), *Recent Developments and New Directions in Soft Computing* (pp. 81–93). https://doi.org/10.1007/978-3-319-06323-2_6

Vieira, G. E., Herrmann, J. W., & Lin, E. (2003). Rescheduling manufacturing systems: A framework of strategies, policies, and methods. *Journal of Scheduling*, *6*(1), 39–62. https://doi.org/10.1023/A:1022235519958

Vollmann, T. E., Berry, W. L., Whybark, D. C., & Jacobs, F. R. (2004). *Manufacturing Planning & Control Systems for Supply Chain Management: The Definitive Guide for Professionals* (5th ed.). McGraw-Hill Higher Education.

Wagner, H. M. (1959). An integer linear programming model for machine scheduling. *Naval Research Logistics Quarterly*, *6*(2), 131–140. https://doi.org/10.1002/nav.3800060205

Wang, L., Luo, C., & Cai, J. (2017). A variable interval rescheduling strategy for dynamic flexible job shop scheduling problem by improved genetic algorithm. *Journal of Advanced Transportation*, *2017*, 1–12. https://doi.org/10.1155/2017/1527858

Wilensky, U., & Evanston, I. (1999). NetLogo. Center for connected learning and computer-based modeling. *Northwestern University. Evanston, IL*.

Xia, W., & Wu, Z. (2005). An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems. *Computers and Industrial Engineering*, *48*(2), 409–425. https://doi.org/10.1016/j.cie.2005.01.018

Xie, N., & Chen, N. (2018). Flexible job shop scheduling problem with interval grey processing time. *Applied Soft Computing Journal*, *70*, 513–524. https://doi.org/10.1016/j.asoc.2018.06.004

Xiong, H., Fan, H., Jiang, G., & Li, G. (2017). A simulation-based study of dispatching rules in a dynamic job shop scheduling problem with batch release and extended technical precedence constraints. *European Journal of Operational Research*, *257*(1), 13–24. https://doi.org/10.1016/j.ejor.2016.07.030

Yang, X., Zeng, Z., Wang, R., & Sun, X. (2016). Bi-objective flexible job-shop scheduling problem considering energy consumption under stochastic processing times. *PLoS ONE*, *11*(12), e0167427. https://doi.org/10.1371/journal.pone.0167427

Zahmani, M. H., Atmani, B., Bekrar, A., & Aissani, N. (2015). Multiple priority dispatching rules for the job shop scheduling problem. *3rd International Conference on Control, Engineering and Information Technology, CEIT 2015*. https://doi.org/10.1109/CEIT.2015.7232991

Zhang, J., Ding, G., Zou, Y., Qin, S., & Fu, J. (2017). Review of job shop scheduling research and its new perspectives under Industry 4.0. *Journal of Intelligent Manufacturing*, *30*(4), 1–22. https://doi.org/10.1007/s10845-017-1350-2

Zhang, S., & Wong, T. N. (2017). Flexible job-shop scheduling/rescheduling in dynamic environment: a hybrid MAS/ACO approach. *International Journal of Production Research*, *55*(11), 3173–3196. https://doi.org/10.1080/00207543.2016.1267414

Zhang, Y., Huang, G. Q., Sun, S., & Yang, T. (2014). Multi-agent based real-time production scheduling method for radio frequency identification enabled ubiquitous shopfloor

environment. *Computers and Industrial Engineering*, *76*(1), 89–97. https://doi.org/10.1016/j.cie.2014.07.011

Apéndice A

Estimación de los parámetros del algoritmo genético

Este apéndice presenta el diseño de experimento realizado sobre los parámetros del algoritmo genético para estudiar el efecto de estos sobre la variable de salida, *makespan*. Se realizó un diseño de experimentos factorial 3^k con $k = 3$ (tres factores).

Se definieron los siguientes factores:

(1) Puntos de entrecruzamiento (*c*)
(2) Probabilidad de mutación (*mr*)
(3) Umbral de aceptación (*thv*)

Para cada factor se definieron tres niveles, denominados como bajo, medio y alto. La Tabla 30 presenta los valores cuantitativos de los niveles para cada factor.

Tabla 30. Arreglo factorial.

Factor	Niveles		
	Bajo	Medio	Alto
Puntos de entrecruzamiento	2	5	10
Probabilidad de mutación	0,25	0,50	0,75
Umbral de aceptación	0,01	0,05	0,10

A partir de la información de la tabla anterior y considerando todas las posibilidades de combinación de los niveles de los factores, se forman 27 tratamientos. Para cada uno de los tratamientos se llevan a cabo diez (10) replicas.

En el presente diseño de experimentos, los tratamientos se realización usando la instancia ‘mk1’ propuesto por Brandimarte (1993) y disponible vía librería OR (Mastrolilli, n.d.).

La Tabla 31 muestra el análisis de varianza (ANOVA) realizado tras la realización completa del experimento. Este análisis permite saber si los efectos principales y de interacción son estadísticamente significativos. Con el fin de probar las hipótesis de los efectos activos, se define un valor de significancia $\alpha = 0,05$.

Tabla 31. Análisis de varianza.

Source	DF	Adj SS	Adj MS	F-Value	P-Value
Model	26	14128,1	543,39	323,82	0,000
Linear	6	13608,3	2268,04	1351,58	0,000
c_points	2	42,8	21,40	12,75	0,000
threshold	2	9552,7	4776,36	2846,34	0,000
mut_r	2	4012,7	2006,37	1195,64	0,000
2-Way Interactions	12	464,5	38,71	23,07	0,000
c_points*threshold	4	46,6	11,65	6,94	0,000
c_points*mut_r	4	13,7	3,41	2,03	0,088
threshold*mut_r	4	404,2	101,06	60,22	0,000
3-Way Interactions	8	55,3	6,91	4,12	0,000
c_points*threshold*mut_r	8	55,3	6,91	4,12	0,000
Error	513	860,9	1,68		
Total	539	14988,9			

Del ANOVA se concluye que los tres efectos individuales, c_points: puntos de entrecruzamiento (c), threshold: umbral de aceptación (thv) y mut_r: probabilidad de mutación (mr) están activos. Con respecto a los efectos de interacción, el único efecto no activo es el definido como c_points*mut_r. La significancia de la interacción detectada por el ANOVA se observa en el hecho de que las líneas de la Figura 62 y Figura 63 tienen pendientes diferentes.

En este diseño, se parte del hecho de que, la representación graficas de los efectos principales y de interacción son suficientes para elegir el mejor tratamiento del experimento. Como lo que interesa en este experimento es minimizar el valor de la variable de respuesta, *makespan*, se concluye lo siguiente.

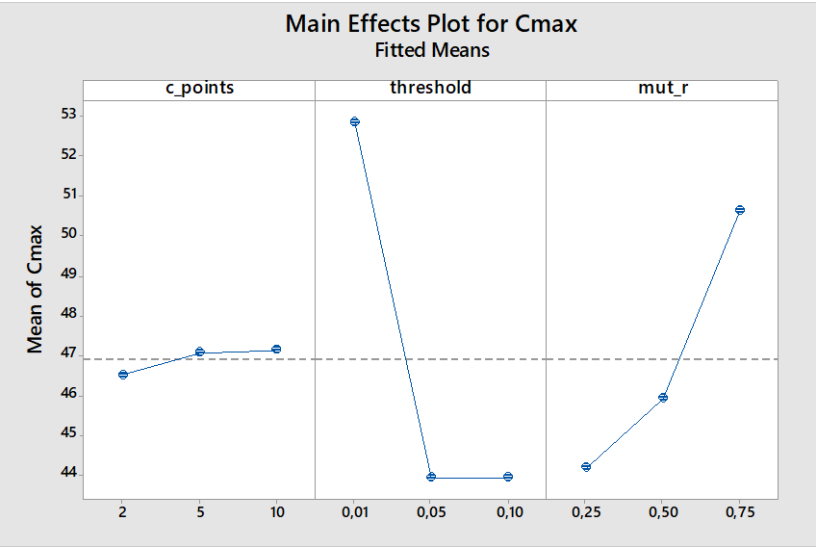

Figura 62. Efectos principales del experimento

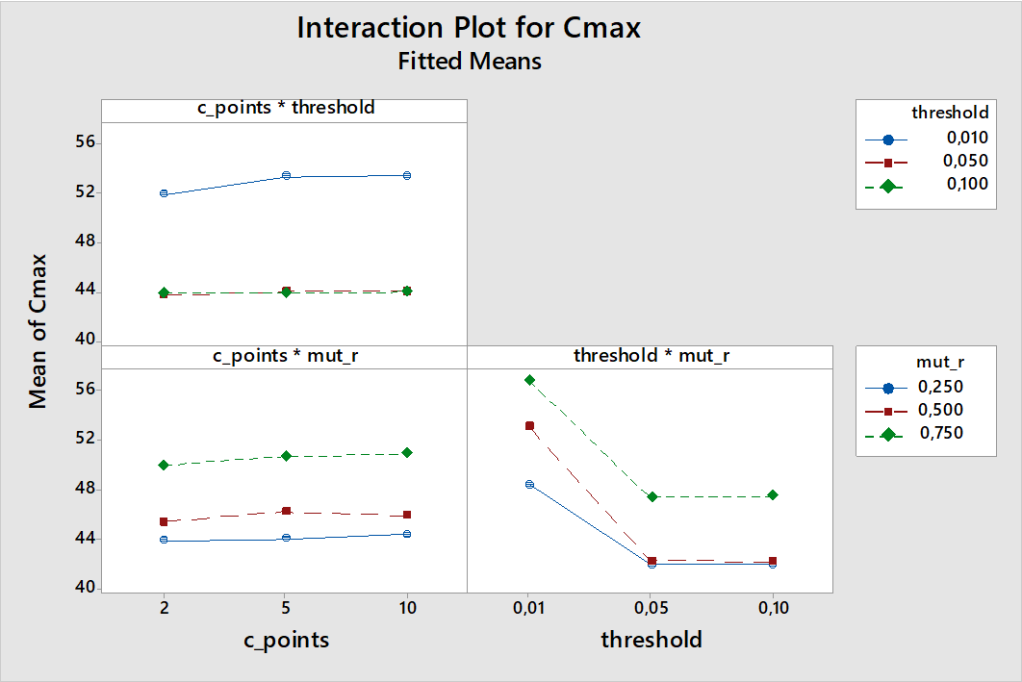

Figura 63. Efectos de interacción del experimento.

El efecto principal más importante es el threshold, seguido por el de mut_r (dado por los valores F del ANOVA). El efecto de interacción más importante es el de threshold*mut_r. Al analizar el efecto de interacción de threshold*mut_r se observa que los valores de makespan disminuyen con tasas de mutación media y baja (0,5 y 0,25) y con valores de umbral de aceptación medio y alto (0,05 y 0,1). Ya que las cuatro combinaciones de tratamientos mencionadas llevan al mismo valor estadístico de *makespan* se analizan los efectos individuales. La interacción individual de mut_r muestra que valores menores de la variable de salida se obtienen con mut_r = 0,25. En cuanto al valor de threshold se selecciona 0,10, aun cuando no hay diferencia estadísticamente significativa al seleccionar 0,05. Finalmente, el efecto individual de c_points muestra que el makespan se minimiza con c_points = 2.

Por lo tanto las condiciones de operación (parámetros del algoritmo) que convienes son:

(1) Puntos de entrecruzamiento (c) = 2
(2) Probabilidad de mutación (mr) = 0,25
(3) Umbral de aceptación (thv) = 0,10

La Tabla 32 compara el desempeño del algoritmo genético propuesto, tras la determinación de los parámetros, en 10 instancias FJSSP de Brandimarte (mk1 a mk10). La comparación se realiza con los algoritmos propuestos por Pezzella et al. (2008), H. Chen et al. (1999), Jia, Nee, Fuh, & Zhang (2003) y Ho & Tay (2004).

La primera columna indica la instancia, la segunda y la tercera columna muestran la cantidad de trabajos y las máquinas disponibles, respectivamente. La cuarta columna muestra el mejor resultado del algoritmo propuesto en este trabajo. Se realizaron diez corridas del algoritmo y se seleccionó la de mejor desempeño. Las columnas restantes muestran el mejor resultado reportado por los otros algoritmos y la desviación correspondiente con el algoritmo propuesto.

Tabla 32. Comparación con otros algoritmos – instancias de prueba.

Instancia	**J**	**M**	**GA**	**Pezzella**	**dev(%)**	**Chen**	**dev(%)**	**Jia**	**dev(%)**	**Ho**	**dev(%)**
mk1	10	6	40	40	0,0%	40	0,0%	40	0,0%	41	-2,4%
mk2	10	6	28	26	7,7%	29	-3,4%	28	0,0%	29	-3,4%
mk3	15	8	204	204	0,0%	204	0,0%	204	0,0%	204	0,0%
mk4	15	8	65	60	8,3%	63	3,2%	61	6,6%	67	-3,0%
mk5	15	4	176	173	1,7%	181	-2,8%	176	0,0%	176	0,0%
mk6	10	15	75	63	19,0%	60	25,0%	62	21,0%	68	10,3%
mk7	20	5	144	139	3,6%	148	-2,7%	145	-0,7%	148	-2,7%
mk8	20	10	523	523	0,0%	523	0,0%	523	0,0%	523	0,0%
mk9	20	10	345	311	10,9%	309	11,7%	310	11,3%	328	5,2%
mk10	20	15	277	212	30,7%	212	30,7%	216	28,2%	231	19,9%

Apéndice B

Aplicación del sistema de soporte de decisiones

La presente sección resume la información presentada en las secciones anteriores. En esta sección se describe el funcionamiento general del DSS, particularmente de la prueba experimental del concepto, como fue indicado en la sección "Alcances y limitaciones" del presente documento.

El propósito general del sistema de soporte de decisiones es guiar la ejecución de operaciones de una orden de producción en un entorno de manufactura flexible job shop minimizando el makespan y respondiendo a posibles perturbaciones. En el caso de ocurrencia de perturbaciones, el DSS debe minimizar la degradación sobre las medidas de retraso y tardanza a través de decisiones reactivas de reprogramación de operaciones (característica 1).

En general, la estructura de control de un sistema, se define por la organización de sus componentes, es decir, por el tipo de comunicación o relación entre ellos. Estas relaciones se pueden clasificar en dos tipos: relaciones jerárquicas y heterárquicas. Se produce una relación jerárquica cuando dos componentes tienen una relación maestro-esclavo, en la cual uno domina completamente las acciones del otro. Por el contrario, una relación heterárquica está presente cuando dos componentes tienen una relación cooperativa en la que colaboran y comparten la responsabilidad de ejecutar sus acciones y el logro de sus propios objetivos en conjunto (Jiménez, 2017).

Teniendo en cuanta el propósito del DSS propuesto, se considera que su estructura es hibrida o semi-heterárquica. Esto es, durante la primera fase de la estrategia predictiva-reactiva, generación de la programación, hay una relación jerárquica en la que las acciones de los trabajos (decisiones de programación), pues estas son establecidas completamente por el algoritmo genético (modelo predictivo). Por otra parte, durante la segunda fase de la estrategia predictiva-reactiva, actualización de la programación, las decisiones son tomadas por cada trabajo en conjunto considerando el objetivo a lograr y el estado de las máquinas y otros trabajos en el piso de manufactura.

La implementación y funcionamiento general del DSS se describe a continuación.

<u>Generación de la programación</u>

1. Ingreso de cantidad de productos a producir. El usuario, a través del subsistema de interfaz de usuario, ingresa la cantidad de productos a ser ejecutados en el piso de

manufactura. Esta información es tomada por el módulo ‘Orden’ y es enviada al módulo ‘Usuario’ en el subsistema de gestión de datos.

2. Generación de la orden. El módulo ‘Generación de orden’ toma la información de los módulos ‘Usuario’ e ‘Información maestra’ para la generación de la orden en el DSS. La orden se genera en un tipo de archivo legible por el módulo de ‘Algoritmo genético’ en el subsistema de gestión de modelos predictivos.
3. Ejecución de la técnica predictiva. El usuario ejecuta el algoritmo genético a través del subsistema de interfaz de usuario. Para la ejecución interna del algoritmo en el DSS, el módulo ‘Algoritmo genético’ toma las siguientes entradas. Módulo ‘Generación de orden’, el cual fue descrito en el numeral anterior. Módulo ‘Parámetros del sistema’, el cual contiene la información del diseño del piso de manufactura. Módulo ‘definición del problema’, el cual define la medida de desempeño, makespan y define matemáticamente los parámetros del sistema.

   En este trabajo de investigación, el algoritmo genético fue programado en Python 3.

Relación con el piso de manufactura

4. Ejecución en el piso de manufactura. El usuario, a través del subsistema de interfaz de usuario, inicia la ejecución de la producción en el piso de manufactura. Allí, se ejecutan las decisiones de la programación inicial encontradas y generadas por el módulo ‘Algoritmo genético’ y comunicadas por el módulo ‘Ejecución inicial’ al piso de manufactura.

   En este trabajo de investigación, el piso de manufactura es simulado en NetLogo. NetLogo es un software que simulación que permite realizar modelos basados en agentes para observar el comportamiento de sistemas y sus respuestas en tiempo real (Alves et al., 2020). Para implementar el modelo en NetLogo, las máquinas y los trabajos fueron modelados como agentes.

   (a) Agente de máquina. Estos agentes representan las máquinas de trabajo, las cuales realizan el conjunto de operaciones de manufactura para obtener los productos (trabajos). Son inmóviles, totalmente pasivos y están sujetos a interrupciones o fallas.
   (b) Agente de operación (trabajo). Estos agentes representan los trabajos. En general, un trabajo se define como un conjunto de operaciones, el cual se mueve a lo largo del piso de acuerdo con la programación y la ubicación de los agentes de máquinas.

El modelo basado en agentes desarrollado en NetLogo está conectado con Python, lo que permite el intercambio de información, particularmente de las decisiones de programación inicial. La conexión de Python a NetLogo también es posible, lo que permite una integración dinámica de estas plataformas.

Actualización de la programación

5. Monitoreo en el piso de manufactura. El módulo ‘Monitoreo’ se encarga de revisar de manera continua la ejecución de la orden en el piso de manufactura con el fin de detectar e identificar perturbaciones en el sistema. La información monitoreada es la entrada para el módulo ‘Política de reprogramación’.
6. Ejecución de la técnica reactiva. El módulo ‘Política de reprogramación’ define que la ejecución de la técnica reactiva se inicia para todos los trabajos cuyo valor asociado $Lij > 0$ (información obtenida a través de monitoreo). El módulo ‘Método de reprogramación’ genera nuevas decisiones sobre la programación inicial. Estas decisiones son locales, es decir, se toman durante la ejecución de la programación en el piso de manufactura.
7. Control del usuario. El usuario, a través del subsistema de interfaz de usuario, controla la ejecución de la producción en el piso de manufactura. La interfaz le permite al usuario conocer si una perturbación ocurrió y ejecutar el método de reprogramación que mejor se ajuste, según su criterio, en el piso.

El usuario hace seguimiento de la ejecución de producción interactuando con las diferentes salidas, de texto y gráficas, que provee el DSS.