Programación de horarios flexibles con muchos objetivos para el personal de seguridad y vigilancia basado en GRASP y NSGA-II

Daniel Fernando Gómez Ortiz

Rene Jalvin Narváez

Director: PhD. Carlos Alberto Cobos Lozada

Universidad del Cauca

Facultad de Ingeniería Electrónica y Telecomunicaciones

Departamento de Sistemas

Programa Ingeniería de Sistemas

Grupo de I+D en Tecnologías de la Información (GTI)

Línea de Investigación: Sistemas Inteligentes

Popayán, noviembre de 2022

Programación de horarios flexibles con muchos objetivos para el personal de seguridad y vigilancia basado en GRASP y NSGA-II

Daniel Fernando Gómez Ortiz

Rene Jalvin Narváez

Trabajo de grado presentado a la Facultad de Ingeniería Electrónica y Telecomunicaciones de la Universidad del Cauca para obtener el título de

Ingeniero de Sistemas

Director: PhD. Carlos Alberto Cobos Lozada

Universidad del Cauca

Facultad de Ingeniería Electrónica y Telecomunicaciones

Departamento de Sistemas

Programa Ingeniería de Sistemas

Grupo de I+D en Tecnologías de la Información (GTI)

Línea de Investigación: Sistemas Inteligentes

Popayán, noviembre de 2022

TABLA DE CONTENIDO

LISTA DE TABLAS

LISTA DE FIGURAS

LISTA DE ALGORITMOS

RESUMEN

Las empresas de seguridad y vigilancia como muchas otras entidades que manejan gran cantidad de personal (talento humano), como por ejemplo los hospitales, son entidades que enfrentan retos importantes en la asignación de las diferentes actividades y requerimientos que deben realizar sus empleados. Normalmente una empresa de seguridad y vigilancia debe asignar su personal de vigilancia en los sitios y horarios de trabajo según las necesidades de sus clientes, asignaciones que normalmente varían con el paso del tiempo. Adaptarse a estas necesidades cuando se tiene poco personal y pocos clientes es una tarea relativamente sencilla, pero cuando la empresa tiene gran cantidad de empleados y clientes el problema se hace muy complejo de manejar. Tener que contratar nuevo personal y o revisar maneras de cubrir los horarios de vigilancia con el personal actual no es una tarea sencilla, ya que resulta en horarios de trabajo poco eficientes y extensos para sus empleados, o por el contrario falta de uso y desaprovechamiento de los empleados.

Teniendo en cuenta lo anterior y que las metaheurísticas son una alternativa viable para la solución de estos problemas complejos, en el presente trabajo se propuso adaptar dos metaheurísticas para la programación de horarios flexibles con muchos objetivos para el personal de seguridad y vigilancia. Las metaheurísticas usadas fueron GRASP con una adaptación Multiobjetivo y NSGA-II. Estas metaheurísticas se encargaron de generar diferentes soluciones a la asignación del personal, permitiendo así que las empresas puedan escoger las soluciones que más le convenga. También, se construyó un aplicativo web para que cualquier persona interesada pueda ingresar los datos del problema y obtener los resultados de la programación de sus recursos usando estos algoritmos. Los resultados de las metaheurísticas se evaluaron con problemas (datos) sintéticos y con datos reales para verificar que tan eficiente es una con respecto a la otra y contra la programación proporcionada por una empresa de seguridad y vigilancia. Finalmente, los resultados obtenidos muestran que los métodos propuestos generan soluciones muy útiles y que aún existe potencial de mejora en especial en contextos donde la programación de los turnos de trabajo no es flexible sino estática.

DEDICATORIA

A nuestros padres por todo el apoyo, amor y dedicación que han tenido con nosotros, todo lo que hemos sido, somos y seremos es por ellos y para ellos.

AGRADECIMIENTOS

A través de estas líneas queremos expresar nuestro más sincero agradecimiento a todas las personas que con su soporte científico y humano han colaborado en la realización de este trabajo de investigación. Queremos agradecer en primer lugar a la Universidad del Cauca que ha hecho posible la realización del trabajo presentado en esta memoria de tesis. Muy especialmente agradecidos con nuestros tutor y director de tesis el Dr. Carlos Alberto Cobos, por la acertada orientación, el soporte y discusión crítica que nos permitió un buen aprovechamiento en el trabajo realizado, y que esta tesis llegara a buen término.

Finalmente, agradecemos a nuestra familia por su comprensión, y constante apoyo en el proceso y en toda nuestra vida, esta tesis va dedicada a ellos quienes nos brindaron la oportunidad de estudiar y convertirnos en los profesionales que somos.

CAPÍTULO 1

1 INTRODUCCIÓN

1.1 PLANTEAMIENTO DEL PROBLEMA

El problema de programación de personal es un problema combinatorial NP Hard [1] que se presenta en diferentes sectores, como por ejemplo en la programación de guardias de seguridad y vigilancia, la programación de personal médico y de enfermería en hospitales y clínicas, la programación del personal en tierra de los aeropuertos, entre otros. En todos estos casos se busca la asignación óptima y eficiente del personal de trabajo teniendo en cuenta las diferentes necesidades y problemas que se presentan en cada sector. La solución del problema de programación de personal tiene diversos objetivos, entre ellos: la reducción de costos, el balanceo de la carga de trabajo del personal involucrado, la satisfacción del cliente y las necesidades individuales del personal.

Los problemas de asignación han sido originalmente divididos entre estáticos y dinámicos. La programación estática tiene una estructura que no cambia en el tiempo, por ejemplo, una programación de vuelos para un aeropuerto que se mantiene igual durante un mediano o largo periodo de tiempo. Por otro lado, la programación dinámica a menudo tiene una variación en su estructura como lo puede ser la programación de citas. Otras clasificaciones usadas para describir el problema de programación de personal se hacen según las características del personal (jornada de trabajo, habilidades o rangos), el tipo de decisión del negocio (basado en tareas, basada en grupos, secuencia de turnos o basada en tiempo) y según la flexibilidad en los turnos o las restricciones [2].

Diferentes modelos, algoritmos o técnicas han sido usados para resolver estos problemas, siendo los más comunes los métodos exactos, las metaheurísticas y las técnicas de simulación. Entre estos, el método de programación lineal es donde se han realizado más investigaciones [2], [3], siendo el más utilizado y aplicado al problema de asignación de personal en enfermería. Otro escenario donde el problema reviste gran importancia es el de la programación de personal de seguridad y vigilancia, pero a la fecha son pocos los estudios que se han realizado en este sector. Este problema es de mucho impacto para las empresas de seguridad y vigilancia ya que los costos operativos aumentan, la rotación de personal aumenta y la calidad del servicio de su personal disminuye cuando la programación definida implica turnos extensos, la asignación del trabajo en sitios lejanos a sus hogares, turnos con muchas horas extras y con poco tiempo de descanso entre ellos, entre otras posibles situaciones. Estas programaciones

inadecuadas y usadas a lo largo del tiempo le generan al personal de seguridad y vigilancia demasiado estrés que los lleva incluso a tomar la decisión de renunciar.

Por las razones anteriormente mencionadas, las empresas de seguridad y vigilancia están cada vez más interesadas en encontrar nuevas maneras que ayuden a programar las actividades de su personal en los diferentes sitios que se deben vigilar, y de esta forma mejorar las condiciones laborales de sus empleados. Sin embargo, realizar esta tarea de forma manual requiere mucho tiempo y esfuerzo, y los resultados obtenidos de esta manera terminan siendo costosas debido a la obtención de programaciones medianamente optimizadas. Es así como diversas empresas de seguridad y vigilancia del país y del Cauca han estado buscando nuevas estrategias para generar automáticamente la programación de su personal de seguridad y vigilancia, priorizando la disminución de sus costos y un mayor confort para sus empleados.

Una forma muy usada recientemente para resolver estos inconvenientes es con el uso de algoritmos de optimización probabilística también conocidos como metaheurísticas, los cuales se diseñan para encontrar la mejor solución posible en el marco de ciertas limitaciones de tiempo y recursos computacionales. Este problema en concreto se puede resolver desde un enfoque de optimización mono objetivo [1][2][3][4][5][6][7][8] o desde un enfoque de optimización multiobjetivo [9][10][11][12].

Este último enfoque (multiobjetivo) permite definir objetivos más aplicables a problemas reales como, por ejemplo: minimizar la cantidad de guardias, minimizar la carga de trabajo, minimizar costos, maximizar preferencias en los horarios, entre muchos otros. Un ejemplo claro de esto se presenta en [9] donde se usó la heurística de transporte para optimizar 3 objetivos en la programación de guardias de seguridad y vigilancia en una empresa de Israel, o en [1] donde se propuso un algoritmo genético (GA) para optimizar 5 objetivos en el personal de enfermería logrando superar el proceso de programación manual, o en [12] donde aplicaron tres metaheurísticas multiobjetivo (el algoritmo Keshtel multiobjetivo - MOKA, el algoritmo genético de clasificación no dominado - NSGA-II y la búsqueda Tabú multiobjetivo - MOTS) en la programación de turnos en enfermería, encontrando soluciones óptimas en un tiempo razonable siendo MOKA el algoritmo que mostró mejores resultados, o en [10] donde se trabajó la programación de personal de enfermería para un hospital de japón, proponiendo el algoritmo PPD-NSGA-II basado en NSGA-II para optimizar 12 objetivos con excelentes resultados.

Por otra parte, el procedimiento de búsqueda adaptativa aleatoria codiciosa (Greedy Randomized Adaptive Search Procedure, GRASP) [7] es una técnica de optimización de estado simple mono objetivo que tiene un enfoque constructivo y de mejoras iterativas, lo que le permite evolucionar dentro del espacio factible de soluciones reduciendo drásticamente el tiempo de búsqueda en el espacio de soluciones total, ya que evita aquellas que no son factibles. De GRASP se destaca que puede ser fácilmente adaptable a diversos problemas obteniendo soluciones muy competitivas y que se puede convertir en un algoritmo multiobjetivo con cierta facilidad. Lo anterior porque el uso de operadores de mutación dentro del espacio factible de soluciones es fácil de modificar en GRASP, y la inclusión de este

algoritmo en un envoltorio que realice la gestión de una población de soluciones, el manejo, y comparación de estas con múltiples o muchos objetivos no modifica en gran medida los conceptos básicos que hacen a GRASP un algoritmo competitivo en el estado del arte. Por otro lado, NSGA-II es uno de los algoritmos de optimización multiobjetivo más conocidos y utilizados en la solución de diversos problemas, no solo por la buena calidad de las soluciones que reporta sino por su baja complejidad computacional y su diseño relativamente sencillo. En el contexto del problema con el que se ocupa este proyecto, los operadores de selección, cruce, mutación y reemplazo tuvieron que ser adaptados para no realizar una búsqueda a ciegas en el espacio de soluciones completo, sino para que se generara soluciones factibles en cada uno de los intentos por generar descendencia de las poblaciones actuales.

GRASP y NSGA-II han reportado buenos resultados en la programación de personal médico y de enfermería, que es un problema que tiene importantes similitudes con la programación de guardias de seguridad y vigilancia [13][7][10], por tal razón y teniendo en cuenta las características de GRASP Y NSGA-II así como las del problema de programación de personal, y dado que solo se puede deducir de forma experimental la mejor metaheurística para resolver un tipo de problema de optimización específico [14], se consideró apropiado realizar una adaptación de GRASP convirtiéndolo en una propuesta de múltiples o muchos objetivos y NSGA-II para resolver el problema de programación de personal para una empresa de seguridad y vigilancia.

Teniendo en cuenta lo anterior, en este trabajo se planteó la siguiente pregunta de investigación: ¿Cómo se puede adaptar el algoritmo GRASP y NSGA-II para resolver el problema de programación de guardias de seguridad y vigilancia en múltiples sitios con un enfoque de muchos objetivos (4 o más)?

1.2 APORTES DEL PROYECTO

Desde la perspectiva de investigación, los aportes de este trabajo de grado se centraron en la generación de nuevo conocimiento obtenido a partir de la evaluación del comportamiento de nuevos algoritmos basados en GRASP y NSGA-II que se adaptaron a problema multiobjetivo de programación de personal de seguridad y vigilancia. A la fecha no se ha reportado este enfoque de investigación con los objetivos que se trataron en este proyecto en ninguna de las bases de datos o índices bibliográficas revisadas, a saber: IEEE, Springer, Scopus, Web of Science y ScienceDirect.

En cuanto a la innovación, esta investigación propuso la adaptación, implementación y uso de GRASP (una propuesta para muchos objetivos) y NSGA-II junto con el uso de algunos operadores para la construcción de los turnos en el proceso de optimización local de las soluciones y de esta manera obtener mejores soluciones al problema real que se trabajó. Esta propuesta se implementó en Python como lenguaje de programación y el código fuente queda disponible en https://github.com/renejal/assignment-of-truns a la comunidad académica y científica para facilitar su uso por parte de empresas o en futuras investigaciones.

La programación de personal de seguridad y vigilancia en esta investigación buscó optimizar simultáneamente cuatro objetivos que son considerados de gran importancia en el sector, a saber: la cantidad de turnos no asignados, la cantidad de guardias de seguridad y vigilancia requeridos, la carga laboral del personal de seguridad y vigilancia, y la distancia entre el hogar del guardia y el puesto de trabajo. Esta propuesta permite con esto a una empresa la selección según su necesidad (frente de Pareto) de la solución más adecuada a sus propias necesidades.

1.3 OBJETIVOS

A continuación, se presentan los objetivos como fueron aprobados por el Consejo de Facultad de la Facultad de Ingeniería Electrónica y Telecomunicaciones al inicio del proyecto. Se modificó parcialmente el cuarto objetivo específico para quitar el nombre de la empresa que facilitó los datos debido a temas privacidad, seguridad y protección de dichos datos.

1.3.1 OBJETIVO GENERAL

Proponer dos algoritmos, uno basado en GRASP y otro en NSGA-II, para abordar el problema de programación del personal de seguridad y vigilancia para una empresa de seguridad con horarios flexibles, minimizando simultáneamente cuatro objetivos (la cantidad de turnos no asignados, la cantidad de guardias de seguridad requeridos, las horas extras del personal de seguridad, y la distancia entre el hogar del guardia de seguridad y el puesto de trabajo).

1.3.2 OBJETIVOS ESPECÍFICOS

- Modelar un algoritmo para la construcción de una solución inicial factible y aleatorizada que cumpla con las restricciones del problema de programación de personal de seguridad y defina los valores de los objetivos que se busca optimizar en la presente investigación.
- Adaptar la metaheurísticas GRASP a un enfoque de muchos objetivos para que en el proceso de mejora aplique operadores de mutación que mantengan la solución en el espacio factible y durante el proceso iterativo logre la optimización de los cuatro objetivos propuestos.
- Adaptar la metaheurísticas NSGA-II para que en el proceso de cruce y mutación genere descendencia en el espacio factible de solución y durante el proceso iterativo logren la optimización de los cuatro objetivos propuestos.
- Realizar un análisis comparativo entre los resultados obtenidos por las metaheurísticas propuestas basadas en GRASP y NSGA-II y los resultados proporcionados por la empresa basado en las métricas hipervolumen y distancia generacional invertida.

1.4 RESULTADOS OBTENIDOS

A continuación, se resumen los resultados principales del presente trabajo de grado:

1. **Monografía de trabajo de grado**: Se refiere al presente documento en el cual se presenta la motivación del problema, el enfoque planteado para la realización

del proyecto y el estado del arte en el área de la programación de personal. Luego, se muestra la solución inicial propuesta para la generación de horarios flexibles con muchos objetivos para el personal de seguridad y vigilancia haciendo uso en las metaheurísticas GRASP y NSGA-II. Posteriormente se muestra el funcionamiento y la adaptación realizada a las metaheurísticas GRASP y NSGA-II para poder ser usadas en el problema. Luego se explica las métricas de evaluación utilizadas para evaluar las metaheurísticas, y la optimización de los parámetros de las mismas. Seguido se explica el aplicativo web construido para que cualquier persona interesada pueda ingresar los datos del problema y obtener los resultados de la programación de sus recursos usando estos algoritmos. Por último, se muestran los resultados y comparaciones obtenidas por cada una de las metaheurísticas en datos sintéticos y reales, las conclusiones del trabajo y el trabajo futuro que el grupo de investigación espera desarrollar en el corto plazo.

2. **Solución inicial y metaheurísticas propuestas**: Se refiere a los pasos y requisitos propuestos que se siguieron para generar una solución factible, la cual se convierte en la base para hacer uso de las metaheurísticas adaptadas para resolver este problema.
3. **Aplicación web**: Aplicativo que permite realizar el registro del dataset o problema y uso de los algoritmos para la generación de horarios flexibles con muchos objetivos para el personal de seguridad y vigilancia, de la cual se destacan los siguientes productos principales, a saber:

    - **Código fuente**: Hace referencia al código fuente con el que se desarrollaron los componentes de la aplicación, entre los cuales se incluyen JavaScript para el desarrollo del Front-end con Vue.js y Python para el desarrollo del Back-end como servicio API Rest con Django, además el uso de la tecnología Docker para fácil despliegue en cualquier sistema operativo. El código fuente está disponible en https://github.com/renejal/assignment-of-truns.
    - **Documentación del código de la aplicación**: Hace referencia a la documentación realizada sobre el código y los componentes de la aplicación desarrollada.

4. **Artículo**: Un artículo con los resultados del trabajo de investigación elaborado en formato IEEE que se espera enviar a evaluación a una revista indexada internacional.

1.5 ESTRUCTURA DE LA MONOGRAFÍA

A continuación, se describe de manera general el contenido y organización de la presente monografía:

CAPITULO 1: INTRODUCCIÓN: Hace referencia al presente capítulo que introduce el tema de investigación, presenta la pregunta de investigación que originó el trabajo, los aportes realizados con el desarrollo del trabajo de grado, los objetivos

(general y específicos) definidos para el proyecto, un breve resumen de los resultados obtenidos y finalmente la organización de la monografía.

CAPITULO 2: CONTEXTO TEÓRICO Y ESTADO DEL ARTE: En este capítulo primero se define formalmente el problema y luego se presentan los trabajos más recientes en el área de la programación de personal en especial el de personal de seguridad y vigilancia, pero también otros como el de programación de persona médico y de enfermería.

CAPITULO 3: SOLUCIÓN INICIAL: En este capítulo se presenta el proceso para la construcción de una solución inicial al problema de programación de horarios flexibles con muchos objetivos para el personal de seguridad y vigilancia, cuya solución es factible y luego es usado por las adaptaciones de las metaheurísticas NSGA-II y GRASP para resolver el problema.

CAPITULO 4: NSGA-II: En este capítulo se presenta en forma detallada la adaptación de la metaheurística NSGA-II (Non-dominated Sorting Genetic Algorithm II) con sus respectivos procesos de selección y mutación para resolver el problema de programación de horarios flexibles con muchos objetivos para el personal de seguridad y vigilancia.

CAPITULO 5: GRASP MULTIOBJETIVO: En este capítulo se presenta en forma detallada la adaptación de la metaheurística GRASP (Greedy Randomized Adaptive Search Procedure) adaptada a la solución del problema de programación de horarios flexibles con muchos objetivos para el personal de seguridad y vigilancia.

CAPITULO 6: METRICAS Y OPTIMIZACIONES: En este capítulo se presenta los algoritmos utilizados para analizar los resultados de las metaheurísticas y el proceso realizado para el afinamiento de los parámetros de los algoritmos propuestos.

CAPITULO 7: APLICATIVO WEB: En este capítulo se presenta el aplicativo realizado para la definición del problema y la definición de los horarios para dicho problema usando los algoritmos propuestos.

CAPITULO 8: EXPERIMENTACIÓN Y RESULTADOS: En este capítulo se presentan los resultados obtenidos y las comparaciones realizadas con los algoritmos propuestos.

CAPITULO 9: CONCLUSIONES Y TRABAJOS FUTUROS: En este capítulo se presentan las conclusiones obtenidas al finalizar el trabajo de grado y una lista de ideas que el grupo de investigación espera realizar como trabajo futuro.

CAPITULO 10: BIBLIOGRAFÍA: Este último capítulo contiene las referencias bibliográficas de los artículos y libros consultados para la realización del proyecto.

CAPÍTULO 2

2 CONTEXTO TEÓRICO Y ESTADO DEL ARTE

A continuación, se presenta la definición formal del problema de programación de personal en el contexto específico de guardias de seguridad y vigilancia, que es el que se espera resolver, y el estado del arte de los métodos utilizados para la solución de las variantes del problema de programación de personal más parecidos al problema que atañe a esta investigación.

2.1 DEFINICIÓN DEL PROBLEMA Y NOTACIÓN:

En esta sección se presenta inicialmente la notación y luego el enunciado del problema de programación de guardias de seguridad y vigilancia con franjas de trabajo variable que se busca resolver.

$v = Indice\ del\ vigilante\ donde\ V\ es\ el\ total\ de\ vigilantes.$

$s = Indice\ del\ sitio\ a\ vigilar\ donde\ S\ es\ el\ total\ de\ sitios\ a\ vigilar.$

$w = Número\ de\ semanas\ a\ programar\ donde\ W\ es\ el\ total\ de\ semanas.$

$p = Indice\ del\ periodo$[1] $donde\ P\ es\ el\ total\ de\ periodos\ (168 * W)\ o\ (24 * 7 * W).$

t $= Indice\ del\ turno\ donde\ T_s\ es\ el\ total\ de\ turnos\ del\ sitio$

$V_{t,s} = Número\ de\ vigilantes\ requeridos$ en el turno t para el $sitio\ a\ vigilar\ s.$

$P^{max} = Duración\ máxima\ de\ periodos\ que\ un\ vigilante\ debe\ trabajar\ de\ seguido\ en\ un\ turno.$

$P^{min} = Duración\ mínima\ de\ periodos\ que\ un\ vigilante\ debe\ trabajar\ de\ seguido\ en\ un\ turno.$

$P^{rest} = Duración\ mínima\ de\ periodos\ que\ un\ vigilante\ debe\ descansar\ entre\ turnos.$

$e^{max} = Máximo\ número\ de\ horas\ extras\ que\ se\ puede\ trabajar\ en\ la\ semana\ por\ vigilante.e.g.\ 12\ horas.$

$n^{max} = Máximo\ número\ de\ periodos\ que\ se\ puede\ trabajar\ en\ la\ semana\ por\ vigilante.e.g.\ 56\ periodos.$

$V_{p,s} = Número\ de\ vigilantes\ requeridos\ en\ el\ periodo\ p\ para\ el\ sitio\ a\ vigilar\ s.$

[1] Representa la hora exacta de un día del mes. e.g. el t igual a 32 representa las 2.p.m. del segundo día de la semana.

$C_{v,s} = Distancia\ en\ Km\ del\ lugar\ de\ vivienda\ del\ vigilante\ v\ al\ sitio\ a\ vigilar\ s.$

$n^{ideal} = Número\ ideal\ de\ periodos\ que\ debe\ trabajar\ en\ la\ semana\ un\ vigilante.\ e.g.\ 48\ periodos$

$v^{min} = Cantidad\ de\ vigilantes\ a\ la\ que\ se\ desea\ optimizar.$

$n_{v,w} = Número\ de\ periodos\ trabajadas\ en\ la\ semana\ w\ por\ el\ vigilante$ v.

$e_{v,w} = Número\ de\ horas\ extras\ trabajadas\ en\ la\ semana\ w\ por\ el\ vigilante\ v.$

En este problema se busca definir un plan o programa de trabajo para V vigilantes en S sitios (de lunes a domingo) teniendo en cada día diferentes jornadas posibles, un número de vigilantes diferentes en cada sitio, y un total de P periodos, donde cada periodo p corresponde a 1 hora. Para modelar el problema se utilizó la variable de decisión $y_{p,s,v} = 1$, para indicar que el vigilante v se asigna al sitio a vigilar s en el periodo p, y es igual a 0 (cero) si no se asigna, y un conjunto de variables derivadas que permiten controlar las asignaciones de los vigilantes, a saber:

$$y_{p,s,v}^{shift} = 1,$$

$$Si\ el\ vigilante\ v\ comienza\ un\ turno\ en\ el\ sitio\ a\ vigilar\ s, en\ el\ periodo\ p, 0\ si\ no.$$

$$y_{p,v}^{rest} = 1, Si\ el\ vigilante\ v\ comienza\ a\ descansar\ en\ el\ periodo\ p, 0\ si\ no.$$

$$y_v = 1, Si\ el\ vigilante\ v\ fue\ asignado\ a\ algún\ periodo, 0\ si\ no.$$

El problema está sujeto a las siguientes restricciones:

$$y_{p,s,v}^{shift} = Y_{p,s,v} * \left(1 - Y_{\mathrm{p-1,s,v}}\right) \forall \mathrm{p} \in \mathrm{P}, \mathrm{s} \in \mathrm{S}, \mathrm{v} \in \mathrm{V}\ donde\ Y_{\mathrm{0,s,v}} = 0 \quad (1)$$

$$y_{p,v}^{rest} = Y_{p-1,\mathrm{s,v}} * \left(1 - Y_{p,\mathrm{s,v}}\right) \forall \mathrm{p} \in \mathrm{P}, \mathrm{s} \in \mathrm{S}, \mathrm{v} \in \mathrm{V}\ donde\ Y_{\mathrm{0,s,v}} = 0 \quad (2)$$

$$\sum_{v=1}^{V} Y_{p,s,v} - V_{p,\mathrm{S}} \leq 0 \ \forall \mathrm{p} \in \mathrm{P}, \mathrm{s} \in \mathrm{S} \quad (3)$$

$$Y_{p,s,v} \leq V_{p,\mathrm{s}} \forall \mathrm{p} \in \mathrm{P}, \mathrm{s} \in \mathrm{s}, \mathrm{v} \in \mathrm{V} \quad (4)$$

$$\sum_{\mathrm{s}=1}^{S} Y_{p,\mathrm{s,v}} <= 1 \ \forall \mathrm{p} \in \mathrm{P}, \mathrm{v} \in \mathrm{V} \quad (5)$$

$$\sum_{P=\mathrm{p}}^{\min\left(\mathrm{p}+P^{min}-1,|P|\right)} Y_{P,\mathrm{s,v}} \geq \min\left(P^{min}, |P| - \mathrm{p} + 1\right) y_{p,\mathrm{s,v}}^{shift} \ \forall \mathrm{p} \in \mathrm{P}, \mathrm{s} \in \mathrm{S}, \mathrm{v} \in \mathrm{V} \quad (6)$$

$$\sum_{\mathrm{P=p}}^{\mathrm{p}+P^{max}} (1 - Y_{p,\mathrm{s},\mathrm{v}}) \geq y_{p,\mathrm{s},\mathrm{v}}^{shift} \; \forall \mathrm{s} \in \mathrm{S}, \mathrm{v} \in \mathrm{V}, \mathrm{p} \in \{1, \dots, |P| - P^{max}\} \quad (7)$$

$$\sum_{\mathrm{P=p}}^{\min(\mathrm{p}+P^{rest}-1,|P|)} \left(1 - Y_{P,\mathrm{s},\mathrm{v}}\right) \geq \min(P^{rest}, |P| - \mathrm{p} + 1)\, y_{p,\mathrm{v}}^{rest} \; \forall \mathrm{p} \in \mathrm{P}, \mathrm{v} \in \mathrm{V} \quad (8)$$

$$\sum_{\mathrm{p}=1+(168*(a-1))}^{168*a} \sum_{s=1}^{S} Y_{p,\mathrm{s},\mathrm{v}} - n^{max} \leq e^{max} \; \forall \mathrm{v} \in \mathrm{V}, a \in \{1,2, \dots, W\} \quad (9)$$

La restricción (1) define el inicio de un turno para un periodo p. La restricción (2) define el inicio de la hora de descanso para un guardia. La restricción (3) asegura que no se exceda la cantidad de guardias de seguridad por periodo. La restricción (4) se asegura que el periodo exista en la programación del sitio a vigilar. En la restricción (5) se verifica que un guardia no pueda estar en dos sitios en el mismo periodo. Las restricciones (6) y (7) aseguran que un turno debe durar una cantidad mínima y máxima de períodos consecutivos. Mientras que la restricción (8) asegura que cada personal de seguridad y vigilancia deba cumplir con el número de horas mínimo de descanso entre cada turno. En la restricción (9) se verifica que ningún personal de seguridad y vigilancia debe exceder el número máximo de horas extras trabajadas.

El problema de programación de guardias de seguridad y vigilancia que se trabajó en esta investigación buscó la optimización (minimización) de cuatro objetivos: cantidad de turnos faltantes, distancia entre lugar de residencia del guardia y sus sitios a vigilar, cantidad de guardias requeridos y cantidad de horas extra trabajadas. Es decir:

$$Minimizar\ f = \{f_1, f_2, f_3, f_4\}$$

Donde f_1 corresponde al primer objetivo a minimizar (la cantidad de turnos no asignados) y se expresa formalmente con la Eq. (12).

$$Min \sum_{\mathrm{p}=1}^{P} \sum_{\mathrm{s}=1}^{S} \left[V_{p,\mathrm{s}} - \sum_{\mathrm{v}=1}^{V} Y_{p,\mathrm{s},\mathrm{v}} \right] \quad (12)$$

f_2 corresponde al segundo objetivo a minimizar (distancia recorrida por los guardias de seguridad y vigilancia desde sus hogares a sus puestos de trabajo) y se expresa formalmente con la Eq. (13).

$$Min \sum_{S=1}^{S} \sum_{\mathrm{v}=1}^{V} \left(C_{v,s} * \sum_{\mathrm{p}=1}^{P} y_{p,\mathrm{v},s}^{shift} \right) \quad (13)$$

f_3 corresponde al tercer objetivo a minimizar (cantidad de guardias de seguridad requeridos para resolver el problema) y se expresa formalmente con la Eq. (14).

$$Min \sum_{v=1}^{V} \left(y_v - v^{min} + \sum_{w=1}^{W} n^{ideal} - n_{v,w} \right) \quad (14)$$

f_4 corresponde al último objetivo a minimizar (número de horas extras trabajadas por el personal de seguridad y vigilancia en la semana) y se expresa formalmente con la Eq. (15).

$$Min \sum_{w=1}^{W} \sum_{v=1}^{V} e_{v,w} \quad (15)$$

2.2 ESTADO DEL ARTE

En la literatura el problema de asignación de personal ha sido un tema bastante estudiado, por lo que se han reportado el uso de diferentes técnicas y métodos para su solución en diferentes campos de aplicación como la enfermería, aviación, seguridad, universidades, entre otros. Estas múltiples alternativas han dificultado su clasificación o catalogación. En esta investigación se han organizado en tres categorías según su relevancia e importancia para el problema que se busca resolver: 1) En el contexto de la programación de guardias de seguridad y vigilancia, 2) En el contexto de la programación de personal médico y de enfermería y 3) En otros contextos de aplicación.

- **En el contexto de la programación de guardias de seguridad y vigilancia.**

El primer artículo relevante encontrado fue publicado en 1986 [9] donde se investigó el problema para una empresa de Israel. En este modelo el personal de seguridad y vigilancia debe ser asignado en turnos de trabajo y descanso de 12 horas. Además de su trabajo un guardia debe pasar por cuatro tipos de entrenamientos durante cada mes y estos son impartidos en determinadas fechas. El plan de programación se enfocó en cumplir tres objetivos principales, satisfacer el número de personal necesario para cada turno, satisfacer las solicitudes de vacaciones del personal y en la medida de lo posible satisfacer las necesidades de entrenamiento periódicos; manteniendo el respectivo orden de prioridad de los objetivos. Como resultado, se desarrolló un algoritmo heurístico intuitivo basado en el algoritmo de transporte [15]. La solución final no es necesariamente óptima y a veces ni siquiera cubre todas las necesidades de los turnos de los trabajadores o las necesidades de formación o las solicitudes de vacaciones, incluso la solución óptima podría no cumplir con todos los requisitos anteriores. Sin embargo, a pesar de lo anterior, la eficiencia del algoritmo heurístico redujo los costos en casi un 50%.

En 2019 [16] se presentó un problema de programación de personal en seguridad aeroportuaria, que consta de tres fases, programación de días libres, programación de turnos y asignación del personal. Se utilizó una estrategia Greedy y una asignación global que proporcionan una solución inicial, y que se mejora mediante un algoritmo iterativo de destrucción/construcción. Dos algoritmos, un método

codicioso para asignación de turnos (Gshifts) y un método de asignación global (AFshifts) fueron implementados en java bajo estadísticas reales y se comprobó que para este modelo la complejidad del tiempo de ejecución de AFshifts es mayor que Gshifts.

En 2018 [17] se planteó una solución basada en búsqueda tabú con el objetivo de garantizar el poder defensivo y equilibrar la carga de trabajo. En este caso, se tienen en cuenta tres turnos (mañana, tarde, noche) cada uno con un número determinado de personal de seguridad y vigilancia dependiendo de los requerimientos de cada empresa que solicita el servicio. Los resultados computacionales muestran solidez del enfoque de búsqueda Tabú, la solución se probó con datos reales de una empresa de seguridad y vigilancia, donde se demuestra que el modelo puede hacer frente a casos reales y es muy adaptable a casos similares.

En el 2020 [18] se investigó la programación de personal de seguridad y vigilancia para múltiples puertas de seguridad. La propuesta buscó resolver dos objetivos: la carga de trabajo y la asignación del personal de seguridad y vigilancia en diferentes turnos. El primer objetivo se centró en determinar los requisitos del personal y los horarios de trabajo de los empleados para las puertas a vigilar. El segundo objetivo buscó asignar guardias a horarios de trabajo específicos para minimizar el costo laboral anual (salarios, horas extras, primas de turnos, asignaciones y beneficios). Para la solución se utilizó Programación Entera en un ciclo de 28 días que garantizan que todas las puertas cuentan con personal uniforme 24/7. El modelo considera varios tipos de empleados, múltiples turnos y ubicaciones de trabajo, produciendo un horario de trabajo óptimo que redujo el tamaño de la fuerza laboral en un 23% y el costo laboral en un 26%.

- **En el contexto de la programación de personal médico y de enfermería.**

En 2010 [19] se desarrolló un enfoque de modelado que requiere que los turnos se generen implícitamente en vez de tener un número predeterminado de turnos y horas de inicio. En esta investigación los turnos pueden comenzar en cualquier día y periodo del cronograma logrando así la programación flexible en los turnos de los médicos. El problema se formula como un programa de enteros mixtos y la planificación se divide en periodos de 1 hora que cubren todo el calendario. El problema además toma en cuenta los días libres, descansos y un servicio de llamadas que es necesario para brindar cobertura fuera del horario de trabajo del hospital, normalmente noches y fines de semana. El problema tiene como objetivo reducir los costos relacionados a las horas fuera del horario laboral, las horas extras y el uso de médicos externos al hospital. El estudio fue realizado en un hospital de Alemania y se tomaron 336 periodos, 16 enfermeros y 2 semanas de programación. Se obtuvieron resultados de alta calidad en un tiempo razonable en comparación con la creación manual, sin embargo, el problema tiende a ser más complejo y difícil de resolver de manera óptima para instancias grandes.

En 2010 [20] se combinaron estrategias aleatorias Greedy y heurísticas para el problema de programación de turnos del personal médico con el principal objetivo de equilibrar las cargas de trabajo. El procedimiento propuesto permitió encontrar soluciones adecuadas cuando se cuenta con el suficiente número de empleados

disponible para cada tarea. Este programa se aplicó con éxito en el Hospital General Universitario de Alicante para planificar el calendario del año 2009 con 28 empleados y dos o tres turnos entre semana.

En 2015 [7] se implementó una propuesta para el problema de programación del personal de enfermería tomando como dirección un enfoque híbrido. El objetivo de esta investigación fue estudiar el comportamiento de una combinación entre algoritmos genéticos y GRASP con el fin de obtener mejoras significativas en la solución del problema. Este estudio se basó en datos reales y en un modelo de programación multiobjetivo con variables binarias, mientras que la función objetivo está representada por un vector de 7 restricciones blandas. Para el estudio se usaron 6 diferentes turnos en 10 generaciones. Los resultados muestran que el método propuesto ahorra un tiempo considerable en comparación con el algoritmo de optimización de colonias de abejas (BCO) y la optimización de colonias de hormigas multiobjetivo (MOACO), y sus resultados son tan buenos como los dos métodos comparados en cantidades pequeñas de datos y con un poco de menor calidad con grandes cantidades de datos.

En 2018 [21] se investigó el problema de programación de personal de enfermería para un hospital estándar en Japón usando los 3 turnos básicos (mañana, tarde, noche), 23 enfermeros y 28 días de programación. Este problema fue solucionado con un enfoque multiobjetivo usando el algoritmo NSGA-II, optimizando así 12 objetivos. Dado que la optimización de NSGA-II es deficiente cuando se quiere optimizar un problema de 4 o más funciones objetivas se utilizó una técnica basada en la dominancia parcial de Pareto, proponiendo una técnica denominada PPD NSGA-II. Con esta técnica el ordenamiento de la parte no dominada se ejecuta usando un subconjunto aleatoriamente seleccionado de todos los objetivos evitando que el jefe encargado tenga que realizar la lista de selección manualmente. Para realizar la comparación entre el algoritmo propuesto y NSGA-II se usó el valor Norm y el valor máximo de propagación como medios de medición, dando como resultado que el método propuesto es algo inferior en la diversidad de la población comparado contra NSGA-II, sin embargo, fue sumamente eficaz en la convergencia de la población al conjunto de soluciones óptimas de Pareto.

En 2019 [22] se investigó el problema de programación de personal médico y de enfermería para un gran hospital que está dividido en varias clínicas. Cada clínica tiene su propio personal de enfermería a diferencia de los médicos que deben ofrecer sus servicios en las diferentes clínicas del hospital. Esta investigación tomó en cuenta la demanda, la disponibilidad del hospital y la carga de trabajo del personal, teniendo como propósito minimizar la insatisfacción, el costo y la desviación de frecuencia de trabajo de los médicos en diferentes clínicas. Para resolver el problema se propuso un algoritmo híbrido que incorpora el algoritmo seno-coseno (SCA) y la búsqueda de vecindario variable (VNS) teniendo como base el algoritmo de iteración Hungarian. Esta propuesta fue comparada con los algoritmos originales SCA, VNS, optimización por enjambre de partículas (PSO), GA y recocido simulado (SA), teniendo en cuenta 22 instancias con diferentes cantidades de médicos y clínicas. Los resultados fueron analizados usando la desviación porcentual relativa mostrando que el rendimiento del algoritmo propuesto

fue el mejor, logrando casi encontrar la mejor solución en cada instancia entre los 6 algoritmos, ampliando además esta diferencia a medida que el número de médicos aumentaba.

También en 2019 [11] se realizó un estudio con un enfoque multiobjetivo con ayuda de un algoritmo genético optimizando 5 objetivos enfocadas en la sobrecarga de trabajo y la salud del personal de enfermería. Esta investigación tomó 3 turnos (mañana, tarde y noche), 31 días y 12 enfermeros. Los resultados obtenidos muestran que el fitness en el proceso manual es mayor (mejor). Sin embargo, el tiempo requerido en el proceso manual es extremadamente largo comparado con la ejecución del algoritmo propuesto, además de no tener ninguna violación de restricciones, por lo que se concluyó que el proceso de programación de enfermeras utilizando el algoritmo genético multiobjetivo propuesto puede manejar correctamente los problemas de programación en hospitales.

Además, en 2019 [23] se propuso un enfoque de extensiones en turnos variables para abordar el problema de imprevistos de horas extras no planeadas en los médicos. Este enfoque permite tener diferentes horas de trabajo e inició en los turnos. Para esto se formuló una programación de enteros mixtos, una heurística basada en la descomposición de generación de columnas y se evaluó con 168 periodos y 17 médicos en un hospital de Alemania. Los resultados muestran que este enfoque reduce las horas extras asignadas en más del 80%, sin embargo, como el modelo es NP completo no es capaz de resolver el problema de manera óptima dentro de límites de tiempo realistas.

Debido al aumento del envejecimiento de la población, en 2017 [24] se investigó sobre el problema de programación de personal de enfermería encargado de cuidar ancianos, teniendo en cuenta las extensas jornadas (24/7) y planteando una solución en dos etapas. En la primera etapa, los turnos se diseñan de acuerdo con la variación del número de enfermeras necesarias en el día. La disposición de cada turno y el número de enfermeras necesarias se logra determinar mediante el uso de algoritmos genéticos. En la segunda etapa se determinan los turnos de acuerdo con la jerarquía de las enfermeras mediante unas reglas establecidas para el problema. Las horas de inicio y finalización de un turno pueden variar y deben estar entre las 4 y 8 horas diarias. La función objetivo busca minimizar el costo total de recursos del personal, teniendo en cuenta los diferentes niveles y habilidades de las enfermeras. Los resultados muestran que cuando cambia la demanda en cada período, la solución óptima y el total del costo del personal se ve afectado negativamente en cierta medida.

También en 2020 [12] se utilizaron tres metaheurísticas multiobjetivo llamadas, MOKA, NSGA-II y MOTS incluyendo en éstas la categoría, preferencia y compatibilidad entre el personal de enfermería. En esta investigación se optimizaron tres objetivos: minimización del costo de personal, minimización de la incompatibilidad entre los estilos de toma de decisiones de los enfermeros asignados a los mismos turnos y la maximización de la satisfacción general de los enfermeros. Las metaheurísticas propuestas demostraron afinidad para encontrar soluciones óptimas en un tiempo razonable siendo MOKA el algoritmo que mostró superioridad. Posteriormente MOKA fue empleado en un gran hospital en Teheran

(Irán), mejorando así la satisfacción laboral y los errores de las enfermeras al aplicar la programación propuesta.

En 2021 [6] se propuso un algoritmo para la solución al problema de asignación de médicos con horarios flexibles con ayuda de una heurística basada en la descomposición de generación de columnas para encontrar buenas soluciones en un tiempo de ejecución razonable. En este estudio se tuvo como único objetivo la minimización del costo del salario total del personal. Para lograr la flexibilidad, el estudio determinó una longitud máxima y mínima en los turnos, en este caso siendo de 7-12 horas y periodos de 1 hora. La disponibilidad de los turnos está predeterminada por una matriz que sirve de entrada para el modelo matemático, esta matriz se encarga de cubrir todos los posibles turnos con diferentes horas de inicio y de duración. Para evaluar el rendimiento, la solución fue evaluada con información de la vida real en un hospital de Alemania, tomando como parámetros máximo 2 días de descanso, diferentes descansos y una programación de 1 a 6 semanas. Los resultados muestran que el enfoque funciona bastante bien, sin embargo, debido a la complejidad del modelo no es posible resolver el problema a su valor óptimo en la mayoría de las pruebas.

También en 2021 [13] se propuso un estudio para la asignación de médicos para las salas de emergencia usando como base un calendario anual real que incluyera festivos, teniendo como objetivos principales la demanda, la ergonomía, equidad y carga de trabajo. En la investigación se formuló una representación matemática del mundo real con un algoritmo híbrido que combinó programación lineal (Linear Programming, LP) y GRASP. Se utilizó LP para construir un modelo general del cubrimiento del problema, el cual fue usado como una guía en la fase de construcción de GRASP con el fin de obtener soluciones completas de programación para finalmente ser mejoradas por una aplicación iterativa del algoritmo de búsqueda del descenso de vecindario variable (VNDS) y por la optimización del flujo de red (NFO). Para el análisis del problema se utilizaron 42 médicos y 19 turnos con diferentes horas de trabajo dando como resultado una superioridad en el enfoque propuesto sobre el método de LP en instancias de diferente tamaño y dificultad inspirados en un caso real. Sin embargo, no se pudo lograr una solución efectiva debido a la dificultad del problema para las instancias grandes que se querían tratar.

- **En otros contextos de aplicación.**

En 2020 [25] se investigó el problema de la asignación de cajeros y supervisores en las estaciones de servicio de Kuwait National Petroleum Corporation, con el fin de asignar los empleados en 86 estaciones distribuidas por todo Kuwait con la típica jornada de tres turnos de 8 horas 24/7. El problema se modeló como programación de enteros mixtos. Debido a la complejidad del problema se propone un enfoque de dos etapas, donde la primera etapa asigna empleados a las estaciones y la segunda etapa específica turnos y días libres para cada empleado. Las funciones objetivo están relacionadas a las preferencias de los empleados, equilibro de trabajo y número de empleados utilizados. Los resultados computacionales relacionados a la solución de los modelos se hicieron a través de IBM ILOG CPLEX Optimizer (CPLEX) y heurísticas especializadas. Sin embargo, los casos prácticos del

problema no pueden resolverse directamente a través del paquete CPLEX debido a la abrumadora cantidad de variables y restricciones en el modelo formulado, por lo tanto, se optó por el procedimiento heurístico que demostró mejores resultados de manera más sencilla.

En 2020 [26] se planteó un modelo de programación de enteros mixtos para asignar el personal de tierra para la industria aeroportuaria, teniendo como objetivo la minimización del personal de cada periodo, reduciendo así los costos de la mano de obra. El estudio asume 10 turnos con diferentes horas tanto de trabajo como de inicio y el personal se divide en diferentes trabajos como zona de equipaje, zona de administración, zona de tiquetes, entre otras. El algoritmo fue evaluado con datos de la vida real para un horario de 28 días y 35 trabajadores dando como resultado que este modelo es útil para la planificación del personal en la aerolínea y que el enfoque propuesto es superior al enfoque tradicional de dos etapas adoptados por las aerolíneas.

En 2022 [27] se propuso un modelo de programación de enteros mixtos para abordar el problema de programación de personal que surgió a partir de la pandemia por covid-19. Se desarrolló un modelo con un solo objetivo y restricciones suaves. El objetivo principal fue asignar empleados a los turnos, los cuales son variados en su hora de inicio y finalización con el propósito de mejorar el trabajo en casa y que en las oficinas no sobrepasen su capacidad prudente de distanciamiento social, de tal manera que todas las actividades se realicen de manera adecuada y se maximicen las preferencias de los empleados. Los resultados muestran que el modelo encuentra soluciones óptimas en un tiempo de cómputo corto incluso para grandes instancias.

Esta página ha sido dejada intencionalmente en blanco.

CAPÍTULO 3

3 SOLUCIÓN INICIAL

3.1 ESTRATEGIA PROPUESTA

Como se presentó en el anterior capítulo, existen diferentes maneras y estrategias para abordar un problema de asignación de personal. A la fecha no se puede soportar una afirmación en relación con cuál tipo de algoritmo es mejor que otro, ya que esto depende completamente de los requerimientos que se estén abordando. En la literatura revisada no se encontró un problema que fuera exactamente igual al otro, aunque existan semejanzas en el problema general entre las diferentes investigaciones, como es el caso del tema de asignación de enfermeras, estos varían en sus objetivos a optimizar y además en cada uno de ellos no se utilizan los mismos algoritmos, metaheurísticas y estrategias.

Teniendo en cuenta lo anterior, para el problema objeto de este trabajo de grado se planteó una solución inicial factible[2], que fue la base del problema de asignación de personal de seguridad y vigilancia. Al tener construida esta solución, se procedió a mejorarla progresivamente con ayuda de diferentes operadores locales, los cuales se incluyeron en las metaheurísticas GRASP multiobjetivo y NSGA-II.

3.2 SOLUCIÓN INICIAL PROPUESTA

Dada la complejidad del problema y la inmensidad del espacio de búsqueda[3], la primera característica que se consideró clave para esta solución inicial fue que cumpliera con todas las restricciones duras, y con la mayoría sino todas las restricciones blandas, es decir que fuera factible (viable). Para su construcción, se incluyeron unos pasos que ayudaron a que la solución tuviera cierto grado de "inteligencia", evitando generar soluciones "absurdas". El proceso se realiza teniendo en cuenta tres (3) conceptos principales: La ***generación de los turnos*** a ser cubiertos por los vigilantes, la ***representación de la solución***; y la ***asignación de los vigilantes*** en los turnos generados.

[2] Una solución factible o viable es aquella que cumple con las restricciones del problema.

[3] Espacio de búsqueda se refiere al dominio de la función a ser optimizada. En el caso de problemas discretos se refiere al conjunto de todas las posibles soluciones candidatas a un problema.

3.2.1 GENERACIÓN DE TURNOS

Para iniciar, se define un turno a uno o un conjunto de periodos consecutivos que un vigilante debe de cubrir, es decir, el rango de periodos entre la hora del día que el vigilante empieza y termina de trabajar. Un ejemplo sería la unión de los periodos *30,31,32,33,34,35* que generaría un turno de 6 horas, empezando a las 6 a.m. del martes, y finalizando a las 12 p.m. del mismo día. Para la generación de los turnos se debe tener en cuenta que los horarios de vigilancia (horas del día de la semana que requieren vigilancia) proporcionados por la empresa de seguridad y vigilancia pueden ser flexibles (no sólo turnos fijos), es decir, que estos se pueden ir creando según las necesidades. Teniendo en cuenta esto, se estableció que un turno puede tener como mínimo 1 periodo y máximo 12 periodos consecutivos. Por lo general tener turnos de 1 a 5 periodos consecutivos no es un caso que se suele presentar, pero con horarios de vigilancia flexibles son una posibilidad. Que un turno se genere de esta manera depende totalmente de la empresa y los horarios que decidan manejar.

Para las empresas de seguridad y vigilancia el proceso de generación de los turnos es indiferente, la única información que se requiere conocer de las empresas es el horario de vigilancia que se debe cubrir en cada uno de los sitios, y la cantidad de vigilantes necesarios en esos horarios. En la **Tabla 1** se muestran los diferentes turnos que se generan según el número de periodos consecutivos en el horario de vigilancia.

Tabla 1. *Posibles turnos por generar.*

Número de periodos consecutivos	Cantidad de turnos generados	Horas por vigilar de cada turno	Número de periodos consecutivos	Cantidad de turnos generados	Horas por vigilar de cada turno
1	1	1	13	2	6, 7
2	1	2	14	2	7, 7
3	1	3	15	2	8, 7
4	1	4	16	2	8, 8
5	1	5	17	2	9, 8
6	1	6	18	2	9, 9
7	1	7	19	2	10, 9
8	1	8	20	2	10, 10
9	1	9	21	3	7, 7, 7
10	1	10	22	3	8, 7, 7
11	1	11	23	3	8, 8, 7
12	1	12	24	3	8, 8, 8

Los horarios en las empresas de seguridad y vigilancia en general no se diseñan para vigilar uno o varios sitios durante un solo día a la semana, sino que por el contrario se deben vigilar diariamente las 24 horas del día durante los 7 días de la semana (24/7). Por esta razón se definió **24** horas como el máximo número de

periodos consecutivos para generar los turnos, si hay más periodos consecutivos después de este número, se consideran como otra secuencia de generación de turnos. Según el horario de vigilancia presentado por las empresas de seguridad, se pueden llegar a presentar diversos casos para generar los turnos; estos se explican a continuación.

Caso 1 - Horario de vigilancia de 24 horas seguidas: La **Figura 1** muestra una empresa que requiere 2 guardias en un sitio durante todo el lunes (24 horas), el martes no requiere ningún vigilante, y el miércoles vuelve a requerir 2 vigilantes durante 24 horas. La **Tabla 2** muestra la solución a este caso, donde el horario de vigilancia se divide en 6 turnos, cada uno de 8 periodos. Fíjese que donde no se solicitan vigilantes no se generan turnos (martes).

Hora del día	Lunes	Martes	Miércoles
12:00 a.m.	2	0	2
1:00 a.m.	2	0	2
2:00 a.m.	2	0	2
3:00 a.m.	2	0	2
4:00 a.m.	2	0	2
5:00 a.m.	2	0	2
6:00 a.m.	2	0	2
7:00 a.m.	2	0	2
8:00 a.m.	2	0	2
9:00 a.m.	2	0	2
10:00 a.m.	2	0	2
11:00 a.m.	2	0	2
12:00 p.m.	2	0	2
13:00 p.m.	2	0	2
14:00 p.m.	2	0	2
15:00 p.m.	2	0	2
16:00 p.m.	2	0	2
17:00 p.m.	2	0	2
18:00 p.m.	2	0	2
19:00 p.m.	2	0	2
20:00 p.m.	2	0	2
21:00 p.m.	2	0	2
22:00 p.m.	2	0	2
23:00 p.m.	2	0	2

***Figura 1**. Representación del horario de vigilancia de 24 horas por día.*

***Tabla 2.** Generación de turnos para un horario de 24 horas seguidas.*

Identificador del turno	Periodo inicial	Periodo final	Vigilantes requeridos
1	0	7	2
2	8	15	2
3	16	23	2
4	48	55	2
5	56	63	2
6	64	71	2

Caso 2 - Horario de vigilancia dividido: La **Figura 2** muestra un ejemplo de un horario de vigilancia dividido, para este caso, los turnos se definen según los periodos que son consecutivos, un turno creado de esta forma se finaliza cuando se encuentra un 0 en el horario de vigilancia, en este caso en particular se genera un turno de 6 periodos (de 1:00 a.m. a 6:00 a.m.), un turno de 12 periodos (de 12:00 p.m. a 23:00 p.m.) el día lunes, y el martes 3 turnos de 7 periodos (1:00 a.m. a 21:00 p.m.) (ver **Tabla 3**).

Hora del día	Lunes	Martes	Miércoles
12:00 a.m.	0	0	0
1:00 a.m.	2	2	0
2:00 a.m.	2	2	0
3:00 a.m.	2	2	0
4:00 a.m.	2	2	0
5:00 a.m.	2	2	0
6:00 a.m.	2	2	0
7:00 a.m.	0	2	0
8:00 a.m.	0	2	0
9:00 a.m.	0	2	0
10:00 a.m.	0	2	0
11:00 a.m.	0	2	0
12:00 p.m.	2	2	0
13:00 p.m.	2	2	0
14:00 p.m.	2	2	0
15:00 p.m.	2	2	0
16:00 p.m.	2	2	0
17:00 p.m.	2	2	0
18:00 p.m.	2	2	0
19:00 p.m.	2	2	0
20:00 p.m.	2	2	0
21:00 p.m.	2	2	0
22:00 p.m.	2	0	0
23:00 p.m.	2	0	0

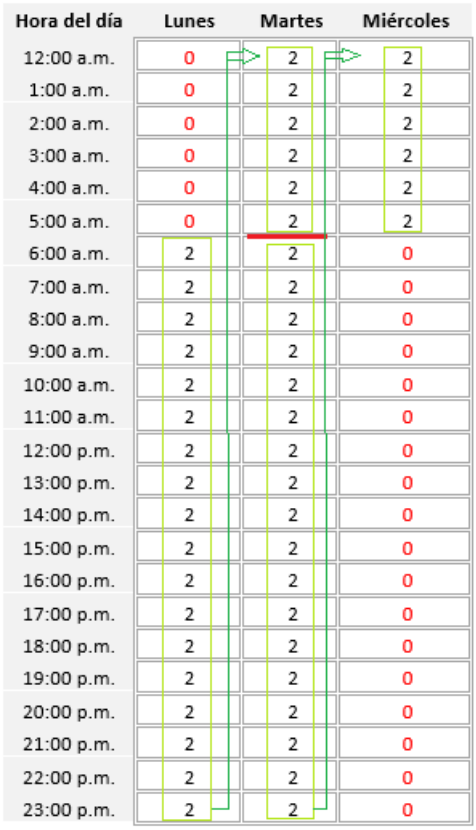
***Figura 2**. Representación del horario de vigilancia divido en los días.*

***Tabla 3.** Generación de turnos para un horario divido en los días.*

Identificador del turno	Periodo inicial	Periodo final	Vigilantes requeridos
1	1	6	2
2	12	23	2
3	25	31	2
4	32	38	2
5	39	45	2

Caso 3 – Horario de vigilancia continuo: La **Figura 3** muestra un ejemplo de un horario de vigilancia de tres días consecutivos, que no necesita de personal de vigilancia (sin requerimientos de personal) en los bordes (al inicio y al fin). En este caso, al finalizar el lunes (23:00 p.m.) se presenta la continuidad de la secuencia en el horario con respecto al martes (12:00 a.m.), fíjese que el martes se debe cubrir

todo el día, pero es preciso recordar que el máximo número de periodos consecutivos puede ser de 24, por lo que a partir de las 6:00 a.m. del día martes se empieza a contar como una nueva secuencia en la consecutividad de periodos, generando 3 turnos de 8 periodos cada uno. Esto mismo se repite para el martes y miércoles (ver **Tabla 4**).



Hora del día	Lunes	Martes	Miércoles
12:00 a.m.	0	2	2
1:00 a.m.	0	2	2
2:00 a.m.	0	2	2
3:00 a.m.	0	2	2
4:00 a.m.	0	2	2
5:00 a.m.	0	2	2
6:00 a.m.	2	2	0
7:00 a.m.	2	2	0
8:00 a.m.	2	2	0
9:00 a.m.	2	2	0
10:00 a.m.	2	2	0
11:00 a.m.	2	2	0
12:00 p.m.	2	2	0
13:00 p.m.	2	2	0
14:00 p.m.	2	2	0
15:00 p.m.	2	2	0
16:00 p.m.	2	2	0
17:00 p.m.	2	2	0
18:00 p.m.	2	2	0
19:00 p.m.	2	2	0
20:00 p.m.	2	2	0
21:00 p.m.	2	2	0
22:00 p.m.	2	2	0
23:00 p.m.	2	2	0

***Figura 3**. Representación del horario de vigilancia continuo.*

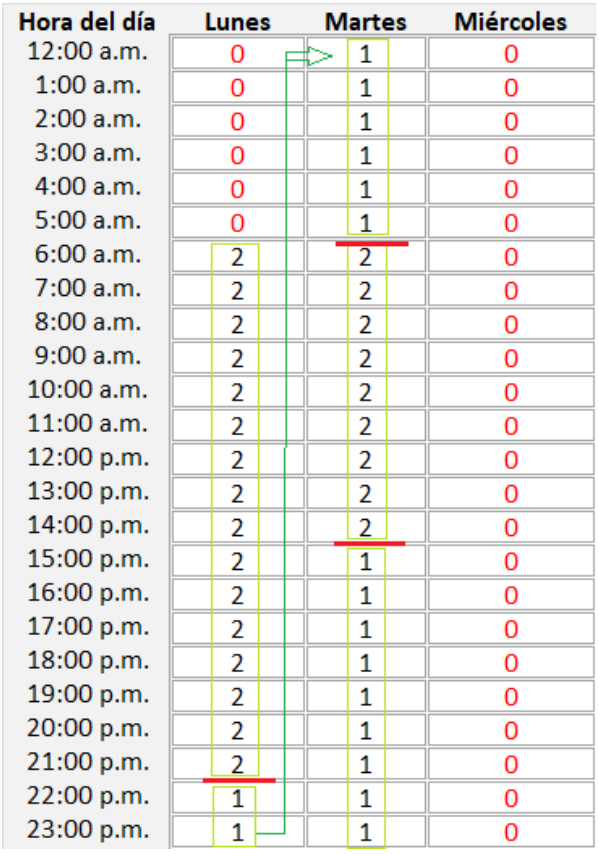
***Tabla 4.** Generación de turnos para un horario continuo.*

Identificador del turno	Periodo inicial	Periodo final	Vigilantes requeridos
1	6	13	2
2	14	21	2
3	22	29	2
4	30	37	2
5	38	45	2
6	46	53	2

Caso 4 – Horario con cantidad de vigilantes diferentes:

Para este caso, la generación de los turnos se define según se encuentra un cambio de vigilantes requeridos en la consecutividad de los periodos, la **Figura 4** muestra un ejemplo del horario con cantidad diferente de vigilantes a asignar en los periodos. Fíjese que desde las 6:00 a.m. a las 21:00 p.m. se requiere de dos vigilantes, y a partir de las 22:00 p.m. empieza a requerir de uno solo, cuando se encuentra ese

cambio, se generan los turnos según la cantidad de periodos consecutivos que se hayan tenido hasta el momento, y después se empieza una nueva secuencia. En este caso se generan 2 turnos, que requieren de 2 vigilantes y de 8 periodos cada uno, luego se genera 1 turno de 8 periodos, pero que requiere de 1 vigilante (22:00 p.m. – 5:00 a.m. le aplica el caso de horario de vigilancia continuo). Lo mismo sucedería para los demás horarios (ver **Tabla 5**).



Hora del día	Lunes	Martes	Miércoles
12:00 a.m.	0	1	0
1:00 a.m.	0	1	0
2:00 a.m.	0	1	0
3:00 a.m.	0	1	0
4:00 a.m.	0	1	0
5:00 a.m.	0	1	0
6:00 a.m.	2	2	0
7:00 a.m.	2	2	0
8:00 a.m.	2	2	0
9:00 a.m.	2	2	0
10:00 a.m.	2	2	0
11:00 a.m.	2	2	0
12:00 p.m.	2	2	0
13:00 p.m.	2	2	0
14:00 p.m.	2	2	0
15:00 p.m.	2	1	0
16:00 p.m.	2	1	0
17:00 p.m.	2	1	0
18:00 p.m.	2	1	0
19:00 p.m.	2	1	0
20:00 p.m.	2	1	0
21:00 p.m.	2	1	0
22:00 p.m.	1	1	0
23:00 p.m.	1	1	0

Figura 4. *Representación del horario con vigilantes diferentes.*

Tabla 5. *Generación de turnos para un horario con vigilantes diferentes.*

Identificador del turno	Periodo inicial	Periodo final	Vigilantes requeridos
1	6	13	2
2	14	21	2
3	22	29	1
4	30	38	2
5	39	47	1

Dependiendo del horario proporcionada por la empresa de seguridad y vigilancia, se podría aplicar uno de los casos o incluso todos, estos mismos casos aplican sí el horario continúa en la siguiente semana, sí en la siguiente semana hay que cubrir la primera hora del día (12:00 a.m. del lunes), se verifica que en el horario de la semana anterior se cubrió la ultima hora de la semana (23:00 p.m. del domingo).

De esta manera se crean los diferentes turnos con respecto a cada sitio y sus necesidades específicas.

3.2.2 REPRESENTACIÓN DE LA SOLUCIÓN

Dada la complejidad del problema, representar la solución como un individuo de un array binario o una matriz de n x n, no se constituida en una forma viable o natural de realizar, gestionar, y mucho menos de implementar. Usar este tipo de representación conduce a hacer demasiados cambios en diferentes partes de la solución, produciendo incrementos en los tiempos de las operaciones y resultados que no son los más adecuados. Por lo anterior fue vital encontrar una representación que ayudara a visualizar y modificar los datos de una manera más natural, esto se logró representando la solución como un conjunto de componentes interconectados o relacionados. En la **Figura 5** se muestra la representación de la solución, cada componente (fila de la figura) representa el conjunto de turnos que se debe manejar para un sitio específico (s_1, s_2 …), la unión de estos componentes produce como resultado la solución final.

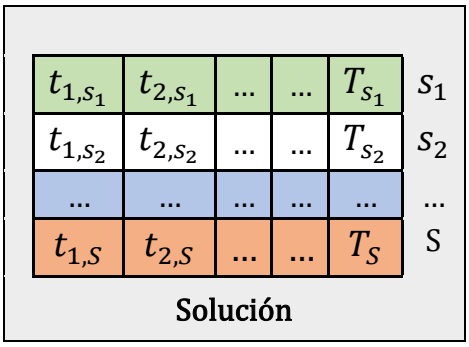


***Figura 5**. Representación de una solución.*

Un turno (por ejemplo t_{1,s_1}) está compuesto por un identificador (ID), el periodo de inicio $\boldsymbol{p_{inicial}}$, el periodo en el que termina $\boldsymbol{p_{final}}$, la cantidad de vigilantes requeridos $\boldsymbol{V_{t,s}}$, y los vigilantes v asignados (ver **Figura 6**). Cuando inicialmente se crea una solución no hay vigilantes asignados ($\boldsymbol{VA_{t,s}}$), pero se espera que al final del proceso queden asignados todos los vigilantes necesarios en el turno.

ID	$p_{inicial}$	p_{final}	Numero de vigilantes requeridos ($V_{t,s}$)	v_1	v_2	...	$VA_{t,s}$

***Figura 6**. Representación de un turno.*

Representar la solución de esta manera ayudó a agilizar el proceso de creación (inicialización) y de evolución (cruce, mutación y optimización local) en las metaheurísticas. Manejar periodos independientes hubiera hecho más complejas las operaciones, e incrementaba los tiempos, dado que se deben hacer más operaciones, por ejemplo, para validar las restricciones blandas y duras del problema. Con esta representación de antemano se sabe que los periodos existen

en el sitio, y que estos no exceden la cantidad mínima y máxima de períodos consecutivos.

3.2.3 ASIGNACIÓN DE VIGILANTES

Teniendo en cuenta el nivel de complejidad del problema, en especial la interrelación de las diferentes variables no era factible que en el proceso de asignación de los guardias de seguridad se hiciera solamente de forma aleatoria, dado que no genera resultados usables por el cliente o que cumplieran con las restricciones duras del problema. Por esto se creó un proceso que aprovecha el conocimiento del problema, pero que al mismo tiempo mantiene el carácter probabilístico (basado en la aleatoriedad) del proceso de inicialización de una solución en el marco de los algoritmos metaheurísticos.

El primer paso consiste en transformar los datos proporcionados por la empresa y crear los guardias y los sitios con sus respectivos turnos, posteriormente se calculan los criterios que ayudan a generar conocimiento sobre el problema. Además, a medida que se van creando los turnos se calcula la cantidad de violaciones posibles que pueden llegar a suceder en cada sitio. Esta información es útil para realizar comparaciones y normalizar los resultados.

Las violaciones que se pueden presentar son: La ***máxima cantidad de periodos faltantes*** expresada formalmente con la Eq. (16), la ***máxima cantidad de vigilantes necesarios*** expresada formalmente con la Eq. (17), la ***máxima cantidad de horas extras*** expresada formalmente con la Eq. (18), y la ***máxima* distancia entre el hogar del vigilante y el puesto de trabajo** expresada formalmente con la Eq. (19).

$$mctf = \sum_{s=1}^{S} \sum_{t=1}^{T_s} V_{t,s} * \ (t_{t,s}.p_{final} \ - \ t_{t,s}.p_{inicial} + 1) \tag{16}$$

$$\text{mcvn} = \text{V} + \sum_{w=1}^{W} \sum_{v=1}^{V} n^{ideal} - 1 \tag{17}$$

$En\ el\ peor\ escenario\ un\ vigilante\ trabaja\ solo\ 1\ periodo\ cada\ semana$

$$mche = \sum_{w=1}^{W} \sum_{v=1}^{V} n^{max} - n^{ideal} \tag{18}$$

$$mdv = \sum_{v=1}^{V} \sum_{s=1}^{S} C_{v,s} \tag{19}$$

Con respecto a la creación de los cronogramas de cada uno de los sitios, se dio prioridad a los siguientes criterios de asignación en el orden expresado a continuación:

1. Crear el cronograma de los sitios que más vigilantes necesitan.

2. Asignar los vigilantes que por una u otra razón deben cubrir un sitio especifico (vigilantes específicos solicitados por los clientes).
3. Asignar en los turnos los vigilantes que ya están asignados en el mismo sitio.
4. Asignar en los turnos los vigilantes más cercanos al sitio.

El **Algoritmo 1** resume el proceso de creación de una solución. Es preciso recordar que la creación de una solución inicial es la unión de varios componentes (cronograma de cada sitio), por lo que el proceso se repite para cada uno de ellos. Primero se obtiene los turnos que deben ser vigilados en el sitio (línea 4), luego se obtienen los vigilantes por defecto que deben ser asignados en el sitio (línea 5). Este es uno de los criterios principales a la hora de empezar a crear un componente. Después se obtienen los vigilantes ordenados según la distancia al sitio que se está programando (línea 6), ya que esta distancia cambia con respecto a los otros sitios. Con esta información se procede a crear el componente (línea 7) que finalmente será añadido a la solución. Cuando se han creado todos los componentes la solución queda completa y cumple con las restricciones duras del problema.

Entradas:	SO: Sitios de la empresa ordenados de mayor a menor por el número de vigilantes requeridos TS: Turnos que se deben asignar para cada sitio LVD: Lista de los vigilantes por defecto para cada sitio LVO: Lista de los vigilantes ordenados por distancia a cada sitio
Salida:	Solución Inicial
1.	**Inicio**
2.	Sol = vacío
3.	**Para** el sitio s **hasta** completar todos los sitios en SO **haga:**
4.	T_s = Obtener los turnos a vigilar para el sitio s de TS.
5.	lvd_s = Obtener vigilantes por defecto para el sitio s de LVD.
6.	lvo_s = Obtener vigilantes ordenados por distancia para el sitio s de LVO
7.	C = Crear_componente (s, T_s, lvd_s, lvo_s) //Crea cronograma para el sitio s.
8.	Sol = Sol ∪ C // Se une el componente a la solución
9.	**Fin Para**
10.	**Retornar** Sol
11.	**Fin**

Algoritmo 1. *Inicialización de la solución.*

La creación de un componente es un proceso que requiere del carácter probabilístico, de lo contrario al ser determinístico siempre generaría la misma solución. Por otro lado, los criterios de asignación ayudan a que en el proceso tenga conocimiento del problema y no se obtengan resultados absurdos. La aleatoriedad se logra asignando vigilantes al azar en los turnos según los criterios anteriormente mencionados, y revolviendo los turnos que se deben vigilar, es decir, se mezclan los turnos de manera que en cada iteración no se inicie con los mismos turnos. Mezclar los turnos es fundamental, porque así se evita generar cronogramas muy parecidos una y otra vez.

En el **Algoritmo 2** se explica el proceso de creación de un componente y los pasos que se deben realizar. Primero se debe mencionar que normalmente en las empresas de seguridad y vigilancia cuando un guardia se asigna en el turno de

algún sitio, este se mantiene como prioridad para seguir siendo asignado en los demás turnos del sitio, debido a que conoce el entorno y puede ofrecer mayor seguridad. Por esto, tener una lista donde se pueda validar los vigilantes que ya se encuentran asignados en el sitio es de gran importancia. Como los vigilantes por defecto deben estar asignados en el sitio al que fueron requeridos, estos se agregan a la lista desde el principio para tenerlos como prioridad (línea 3), posteriormente se procede a revolver los turnos (línea 4), y empieza el ciclo que busca asignar los vigilantes en cada uno de los turnos (línea 5). Como un turno puede requerir de varios vigilantes, en el proceso se intenta asignar un vigilante en el turno esa cantidad de veces.

Entradas:	s: Sitio a vigilar T_s: Turnos por asignar en el sitio lvd_s: Vigilantes por defecto en el sitio lvo_s: Vigilantes ordenados por distancia en el sitio z: Cantidad de vigilantes a dividir de la lista de vigilantes ordenados.
Salida:	Componente: Todos los turnos asignados para el Sitio s a vigilar.

```
1.  Inicio
2.      C = vacío                       // Componente
3.      lva_s = lvd_s                   // Lista de vigilantes asignados en el sitio
4.      T_s = Revolver_turnos ()        // Devuelve los turnos aleatorizados
5.      Para el turno t hasta completar los turnos en T_s haga:
6.          Para todos los vigilantes requeridos en el turno t para el sitio s (V_{t,s}) haga:
7.              v = vacío               // Vigilante a asignar
8.              Si lva_s no está vacía entonces:
9.                  Mientras no se haya escogido todos los vigilantes de lva_s haga:
10.                     va = Escoger_vigilante_aleatoriamente_de_lista (lva_s)
11.                     v = Validar_restricciones (va, t) // Se valida si el vigilante cumple
                            con las restricciones en el turno
12.                     Si v puede ser asignado al turno t entonces:
13.                         Terminar mientras
14.                     Fin Si
15.                 Fin Mientras
16.             Fin Si
17.             Si v es vacío entonces:
18.                 llvo = dividir la lista lvo_s en listas de vigilantes con tamaño z
19.                 Para lista lvo hasta completar las listas en llvo haga:
20.                     Mientras no se haya escogido todos los vigilantes de lvo haga:
21.                         va = Escoger_vigilante_aleatoriamente_de_lista (lvo)
22.                         v = Validar_asignacion (va, s) //Se valida si el vigilante
                                                cumple con las restricciones en el turno
23.                         Si v puede ser asignado al turno t entonces:
24.                             Terminar Para
25.                         Fin Si
26.                     Fin Mientras
27.                 Fin Para
28.             Fin Si
29.             Si v es diferente a vacío entonces:
30.                 Asignar_vigilante (v, t) // Se asigna el vigilante en el turno t
```

31.	$lva_s = lva_s \, U \, v$ // Se une el vigilante a la lista de vigilantes asignados del sitio en caso de que no se encuentre
32.	**Fin Si**
33.	**Fin Para**
34.	C = C U t // Se agrega el turno al componente
35.	**Fin Para**
36.	**Retornar** C
37.	**Fin**

Algoritmo 2. *Creación de un componente.*

Para seleccionar un vigilante primero se verifica que existan vigilantes asignados en el sitio (línea 8), si los hay se procede a escoger un vigilante aleatoriamente de la lista (línea 10), y a verificar que pueda ser asignado en el turno actualmente escogido (línea 11). Sí el vigilante no puede ser asignado, se continúa escogiendo de la lista hasta que se hayan revisado todos los vigilantes asignados (línea 9), sí no se pudo encontrar ningún vigilante, entonces se procede a escoger vigilantes de la lista de vigilantes ordenados por distancia (línea 17). A diferencia de la lista de los vigilantes asignados, en la lista de vigilantes ordenados por distancia no se revisan todos los vigilantes, sino que se divide en listas más pequeñas para evitar escoger vigilantes muy lejanos (línea 18). Estas listas son revisadas en orden (línea 19), y para cada una de ellas se repite el proceso de escoger un vigilante aleatoriamente (línea 21), y el de validar que pueda ser asignado (línea 22), en caso de que el vigilante tampoco pueda ser asignado, se sigue revisando con los demás vigilantes de la lista actual (línea 20).

Es preciso aclarar que un vigilante solamente puede ser asignado en un turno cuando cumple con todas las restricciones duras mencionadas en el **capítulo 2**, sí el vigilante puede ser asignado en cualquiera de los casos en que se valida las restricciones (línea 12, 23), entonces se deja de buscar otros vigilantes y se procede a asignar el vigilante al turno (línea 30) y a la lista de vigilantes asignados en el sitio (línea 31) en caso de que no se encuentre. Finalmente, el turno se asigna al componente (línea 34) y se repite todo el proceso con el próximo turno.

Esta página ha sido dejada intencionalmente en blanco.

CAPÍTULO 4

4 NSGA-II MULTIOBJETIVO

4.1 NOTACIÓN

En esta sección se presenta inicialmente la notación y la explicación general del algoritmo NSGA-II en el problema de programación de guardias de seguridad y vigilancia con horarios flexibles. Manteniendo las definiciones de NSGA-II, en esta adaptación se interpreta un sitio a vigilar (un componente completo) como un gen y una solución completa corresponde a un padre o hijo de una población.

Para este capítulo se debe tener en cuenta las siguientes definiciones:

$f = Indice\ de\ un\ padre\ (solucion)\ donde\ F\ es\ el\ total\ de\ padres\ (Poblacion).$

$s = indice\ del\ Sitio\ donde\ S_f\ es\ el\ total\ de\ sitos\ del\ padre\ f.$

$t = Indice\ del\ turno\ donde\ T_{s,f}\ es\ el\ total\ de\ turnos\ asignados\ a\ un\ sitio\ s\ del\ padre\ f.$

$v = Indice\ del\ vigilante\ donde\ V_{t,s,f}\ es\ el\ total\ de\ vigilantes\ asignados\ al\ turno\ t\ en\ el\ sitio\ s\ del\ padre\ f$

$p = indice\ del\ periodo\ donde\ P_{t,s,f}\ es\ el\ total\ de\ periodos\ en\ el\ turno\ t\ en\ el\ sitio\ s\ del\ padre\ f$

$V_{t,s} = Número\ de\ vigilantes\ requeridos\ en\ el\ turno\ t\ para\ el\ sitio\ a\ vigilar\ s.$

4.2 ALGORITMO NSGA-II

NSGA-II (Elitist Non-Dominated Sorting Genetic Algorithm) fue propuesto por Kalyanmoy Deb y sus estudiantes en el año 2000 [28], la cual se basa en la teoría de la evolución, tomando conceptos como la reproducción y supervivencia, donde el más apto permanece y los demás son eliminados. Esta metaheurística parte de una población inicial $\mathbf{F}$ de $\mathbf{f}$ padres, cada una creada de forma aleatoria en una búsqueda a ciegas, o incluyendo conocimiento del problema, de modo que se generen soluciones coherente (factible), tal como se explicó en el **Capítulo 3.2.2.** En el proceso evolutivo, esta población inicial $\mathbf{F}$ se usa para generar una población de hijos $\mathbf{H}$ usando mecanismo de selección, cruce y mutación. Los padres y los hijos se unen en una población $\mathbf{D}$ organizados en rankings o Frentes de Pareto (ver **Capitulo 4.5**), con el fin de seleccionar las mejores $\mathbf{N}$ soluciones de $\mathbf{D}$. Las soluciones seleccionadas iniciarán la nueva población $\mathbf{F}$ que se usará en la siguiente evolución. La selección de las mejores soluciones se hace basado en el número del Frente de Pareto en el que se encuentra, y en la distancia de Crowding de las

soluciones que están en el mismo frente [29][10]. El **Algoritmo 3** resume los pasos adaptados de NSGA-II para resolver el problema de programación de personal de seguridad y vigilancia.

Entradas: N: Tamaño de la población, O: Lista de M Objetivos a optimizar // Se asume minimización de todos ellos	
Salida: Frente de Pareto encontrado	
1.	**Inicio**
2.	Cree población F de N soluciones y evalúe sus O objetivos (Algoritmo 1)
3.	**Mientras** no se cumpla el criterio de parada **haga:**
4.	Cree una población H de N hijos usando operadores de selección, cruce y mutación.
5.	D = F U H // Agregue la población F y H en D
6.	Ordene la población D basado en el Frente de Pareto
7.	Inicialice F vacío y Rango = 1
8.	**Mientras** la población F cuente con menos de N padres **haga:**
9.	Agregue en F soluciones de D presentes en el Frente con el número de Rango ordenados por Distancia de Crowding (de mayor a menor distancia) sin superar las N soluciones
10.	Rango = Rango + 1
11.	**Fin Mientras**
12	**Fin Mientras**
13.	**Retornar** Frente de Pareto (Rango 1) de F
	Fin

Algoritmo 3*. NSGA-II. Tomado de* [30]

En la línea 2 se inicializa la población inicial teniendo en cuenta algunos criterios que permiten crear una solución factible, por ejemplo, que se asigne los vigilantes en turnos de 8 horas, que los vigilantes se asignen en orden de distancia al sitio, que se asignen los vigilantes solicitados explícitamente por los clientes de la empresa (vigilantes por defecto), entre otras. Este proceso de inicialización se explicó previamente en el **Capítulo 3**, y se usa para evitar partir de una población que cuenta con padres no factibles, logrando con esto reducir los tiempos de ejecución.

En la línea 4 se crea una población de $\mathbf{N}$ hijos. Para esto se hace un ciclo interno, donde se seleccionan dos soluciones padres, con el fin de generar dos soluciones hijas, usando mecanismos de **selección, cruce y mutación** explicados en la **sección 4.3** y en la **sección 4.4**. En la línea 5 y 6 se hace la unión de la población de los padres con la población de los hijos, y se ordenan basados en el Frente de Pareto que se explica en la **sección 4.5**. Para hacer el proceso de ordenamiento, primero se debe definir una forma de comparar dos soluciones, y saber si una solución domina a otra (tiene un mejor valor en por lo menos un objetivo y es igual en los otros), o si entre ellas no se puede establecer relación de dominio [29]. En la línea 8 y 9 se une a la nueva población $\mathbf{F}$ las mejores soluciones presentes en $\mathbf{D}$, según los rangos ordenados por la Distancia de Crowding. Finalmente, en la línea 13 se retorna la mejor evolución y frente encontrado.

4.3 PROCESO DE SELECCIÓN

El proceso de selección está basado en las leyes de la selección natural y la genética donde se buscan a los mejores padres o los más adaptados que puedan aportar mejores genes (sitios) a sus descendientes. Para explicar mejor este proceso, en la **Figura 7** se presenta una población inicial ordenada de forma descendente (minimización) por calidad (esta se define de acuerdo con el rango del Frente de Pareto y la distancia de Crowding de las soluciones en el mismo rango). Para el proceso de selección se toma una parte de los padres de la población inicial, y pasan a formar una lista denominada como lista restringida, la cual contiene los padres para generar los hijos necesarios para la próxima evolución. De la lista se selecciona de forma aleatoria al primer padre; para el ejemplo en este caso se seleccionada a $\boldsymbol{f_3}$ (en color verde).

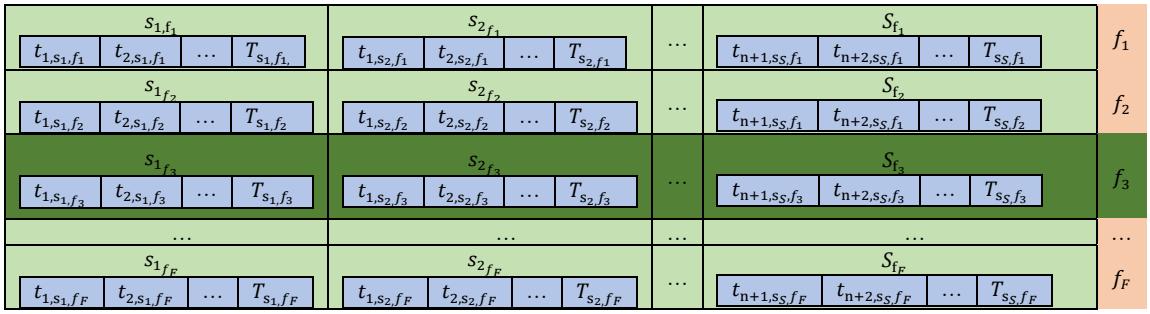


Figura 7. *Población inicial (NSGA-II).*

Del padre $\boldsymbol{f_3}$ se toma de forma aleatoria un gen (sitio) $\boldsymbol{s}$ defectuoso, se define a los genes defectuosos como los genes que menos adaptados están al padre, siendo el fitness el factor por el cual se determina qué tan defectuoso es el gen. Para el ejemplo se dice que el gen defectuoso de $\boldsymbol{f_3}$ es el gen $\boldsymbol{s_{3f_3}}$ (ver **Figura 8**).

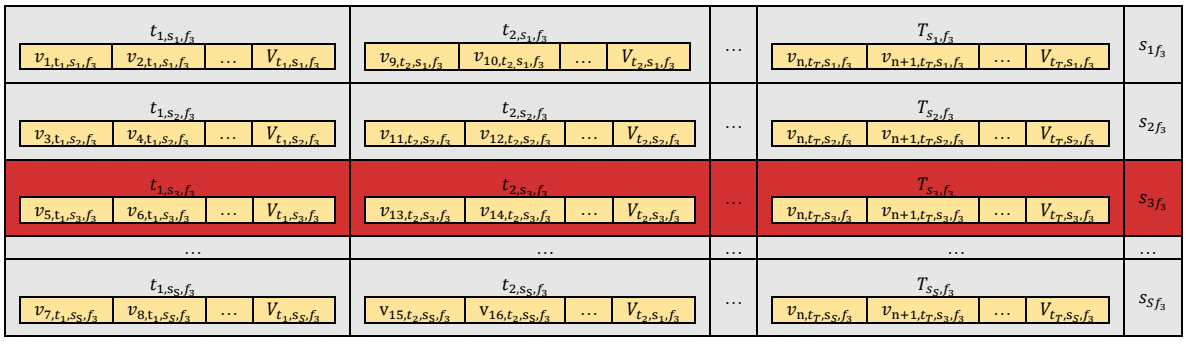


Figura 8*. Individuo $\boldsymbol{f_3}$ en detalle (proceso de selección NSGA-II).*

Seguidamente, se busca de la población un padre diferente con un gen similar a $\boldsymbol{s_{3f_3}}$, de tal manera que se aporte un mejor fitness al padre $\boldsymbol{f_3}$. Los genes similares son aquellos genes donde el periodo de inicio y de finalización de cada turno son iguales, y además deben tener la misma cantidad de vigilantes $\boldsymbol{V_{t,s}}$ requeridos en el turno. Al igual que los padres, los genes similares se almacenan en una lista ordenada de forma descendente, donde los genes más similares están de primero.

Para el ejemplo diremos que el gen s_1 del padre f_1 es uno de los mejores adaptados y más similar al gen s_{3f_3} (ver **Figura 9**).

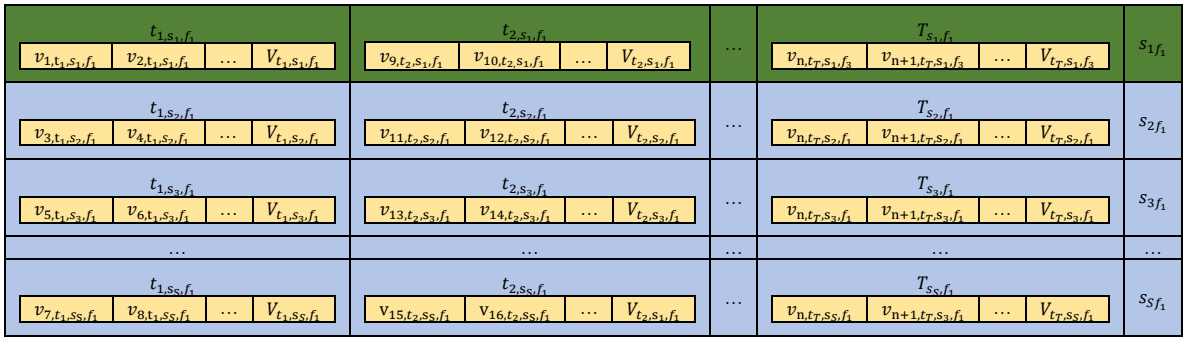


***Figura 9**. Individuo f_1 en detalle (proceso de selección NSGA-II).*

Por consiguiente, los padres seleccionados para realizar el cruce son f_3 y f_1, entre los genes s_{3f_3} y s_{1f_1}. En el **Algoritmo 4** se muestra el proceso de selección. De la línea 2 a la 11 se recorre toda la población, y por cada iteración se seleccionan dos padres, se obtiene el primer padre f_1 (línea 3) con el fin de modificar el gen defectuoso que contenga (línea 4). A partir de este gen se busca el segundo padre f_2 con un gen parecido que este mas optimizado entre la población (línea 5). Este mismo proceso (buscar en el primer padre un gen mejor adaptado para cruzar) se aplica al segundo padre f_2(línea 6 - 7). Finalmente, a cada uno de los padres seleccionados se les aplica el cruce (el proceso de cruce se explica en el **Capítulo 4.4**), generando un hijo en cada llamado (línea 8 - 10)

Entradas: F: Población actual // copia de la población original ordenada y restringida	
Salida: LH: Lista de hijos	
1.	**Inicio**
2.	**Mientras** no se hallan recorrido todos los padres de la población **haga:**
3.	F1 = obtener_padre (F) // obtiene el mejor padre con respecto al fitness que se quiere optimizar de una lista restringida.
4.	GDF1 = obtener_gen_defectuoso(F1) // Obtiene el gen defectuoso del padre uno de una lista restringida.
5.	GMF2, F2 = obtener_mejor_gen (GDF1, F) // Obtiene un mejor gen de la población, es decir el más parecido con el gen defectuoso GDF1 y su padre.
6.	GDF2 = obtener_gen_defectuoso (F2)
7.	GMF1 = obtener_gen_mejor_adaptado (F1)
8.	H1 = F1.cruce (GDF1, GMF2) // Genera el primer hijo de los padres F1 y F2 donde se traslada el GDF1 por el GMF2 teniendo como base F1.
9.	H2 = F2.cruce (GDF2, GMF1) // Genera el segundo hijo de los padres F1 y F2 donde se cruzan los genes GDF2 y GMF1 teniendo como base F2.
10.	LH.Agregar (H1, H2) // Agregar nuevos hijos a la lista
11.	**Fin Mientras**
12.	**Retorna** LH
13	**Fin**

***Algoritmo 4**. Proceso de selección y generación de nuevos descendientes.*

4.4 PROCESOS DE CRUCE

Buscando generar soluciones que permitan optimizar los cuatro objetivos del problema planteado en esta investigación, se implementaron tres cruces, a saber: cruce de turnos completo, cruce de turnos parcial y cruce de vigilantes, los cuales se explican a continuación.

4.4.1 CRUCE DE TURNOS COMPLETO

Después de haber hecho el proceso de selección, y partiendo de la línea 8 y 9 del **Algoritmo 4,** se procede a hacer el cruce entre los dos padres previamente encontrados, intercambiando los turnos del gen defectuoso por los del gen bueno.

La finalidad de este cruce es reasignar los turnos del gen defectuoso con el gen mejor adaptado, y su principal objetivo es mejorar la cantidad asignada de vigilantes para el sitio.

En el **Algoritmo 5** se crea una lista de los vigilantes más compatibles para intercambiar los turnos, en caso tal, que el gen mejor adaptado tenga más vigilantes asignados que en el gen defectuoso, se asigna un nuevo vigilante de la lista de vigilantes disponibles si es posible. En caso contrario, de que el gen defectuoso tenga más vigilantes asignados que el gen mejor adaptado, se desasignan los vigilantes; con esto, se garantiza que el número de vigilantes asignados en el gen mejor adapto sea igual que en el gen defectuoso.

Entradas: GD: Gen defectuoso, // Es equivalente a un sitio en el problema GM: Gen mejor adaptado, // Es equivalente a un sitio en el problema	
Salida:	
1.	**Inicio**
2.	**Mientras** no se hayan recorrido todos los vigilantes de GM **haga:**
3.	VGM = Obtener_vigilante (GM) // Obtiene vigilante de GM
4.	VC = GD.Obtener_vigilante_compatible (VGM) // Obtiene vigilante compatible con VGM
5.	**Si** se encontró el vigilante compatible, es decir, VC **no es nulo entonces:**
6.	GD.Cruce (VC, VGM) // Cruza los turnos de los vigilantes
7.	**Sino**
8.	GD.Asignar_nuevo_vigilante () // Asigna nuevo vigilante de la lista de vigilantes disponibles
9.	**Fin Mientras**
10.	**Si** hay vigilantes sobrantes en el gen defectuoso GD **entonces:**
11.	GD.Desasignar_vigilantes ()
12.	**Fin Si**
13.	**Fin**

***Algoritmo 5**. Cruce de sitios (NSGA-II).*

En la **Figura 10** se presenta la lista de vigilantes compatibles para la asignación de turnos, como por ejemplo el vigilante $\boldsymbol{v_3}$ del sitio $\boldsymbol{s_{3f_3}}$, que es compatible con el vigilante $\boldsymbol{v_3}$ del sitio $\boldsymbol{s_{1f_1}}$.

En la **Figura 11** se hace una representación gráfica de un árbol genealógico de padres e hijos generados en el cruce de sitios. El primer nivel corresponde a la

población, en este caso con todos los posibles padres, el segundo nivel corresponde a los padres seleccionados, en este caso f_1 y f_3 que se encontraron en el proceso de selección, según lo explicado en la **sección 4.3**, y en el tercer nivel los genes defectuoso s_{3f_3} del padre f_1 (ver **Figura 12**) y el mejor gen adaptado s_{3f_3} del padre f_3 (ver **Figura 13**), en el cuarto nivel se muestra los turnos asignados, y los respectivos vigilantes en cada turno, como por ejemplo, para el sitio s_{3f_3} los vigilantes v_1, v_2 tiene asignado el turno t_1.

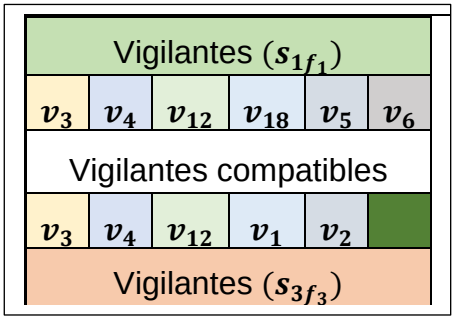


Figura 10. *Lista de vigilantes compatibles para asignación de turnos (NSGA-II).*

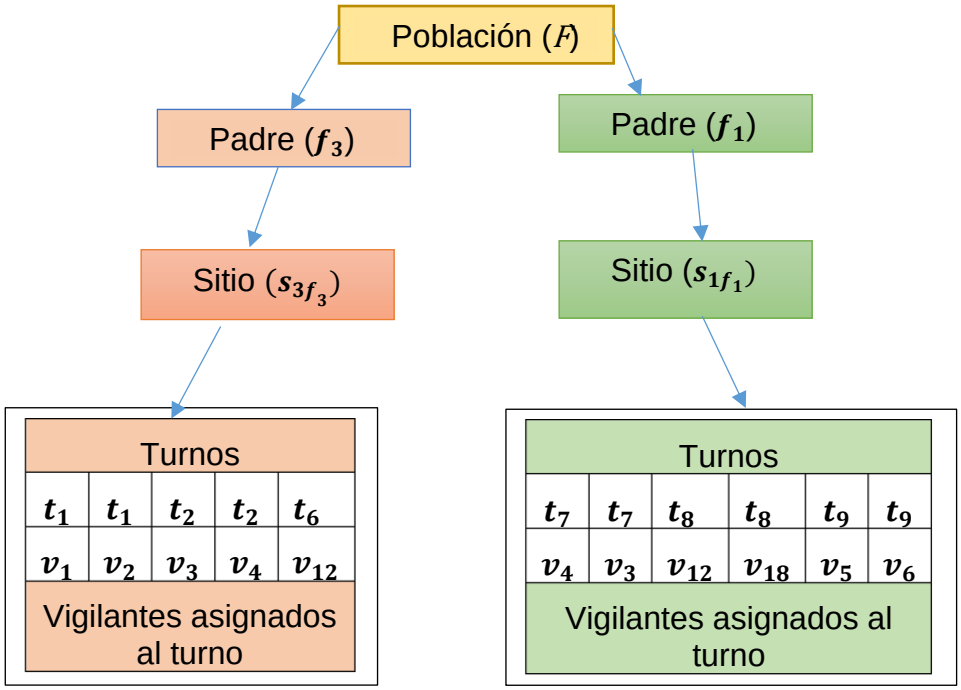


Figura 11. *Representación del árbol genealógico de un cruce (NSGA-II).*

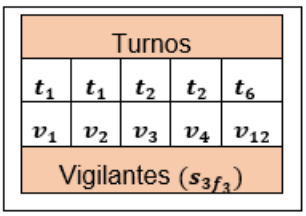


Figura 12. *Gen defectuoso del cruce de turnos (NSGA-II).*

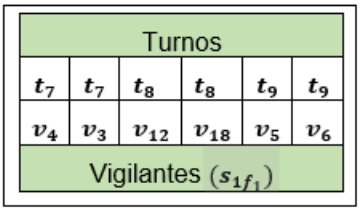
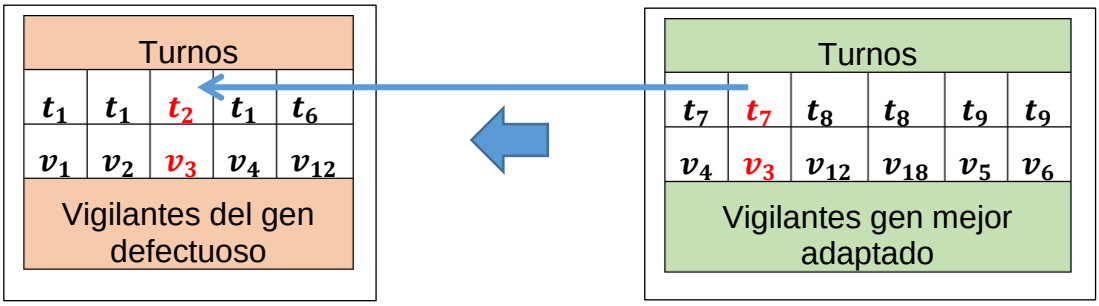


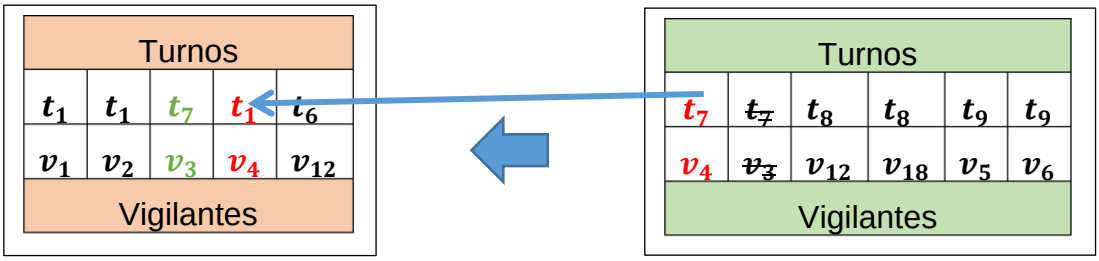
Figura 13. *Gen mejor adaptado del cruce de turnos (NSGA-II).*

A continuación, se describe paso a paso cómo se hace el cruce.

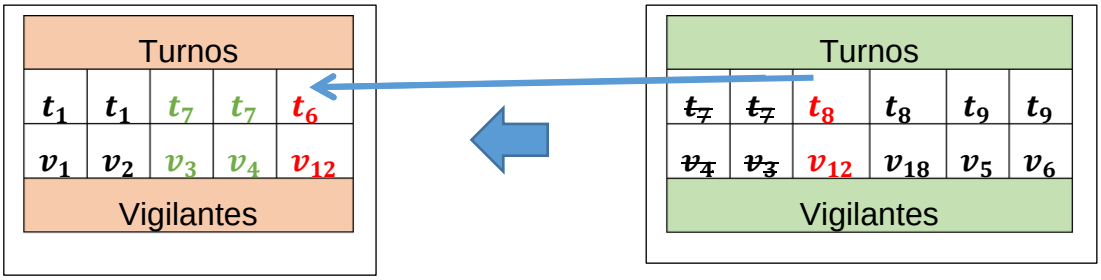
Paso 1: Para el cruce se debe tener en cuenta el orden de la lista de vigilantes compatibles **Figura 10.** Por ejemplo, al vigilante v_3 en rojo del gen defectuoso se le asigna el turno t_7 del vigilante v_3 del gen mejor adaptado.



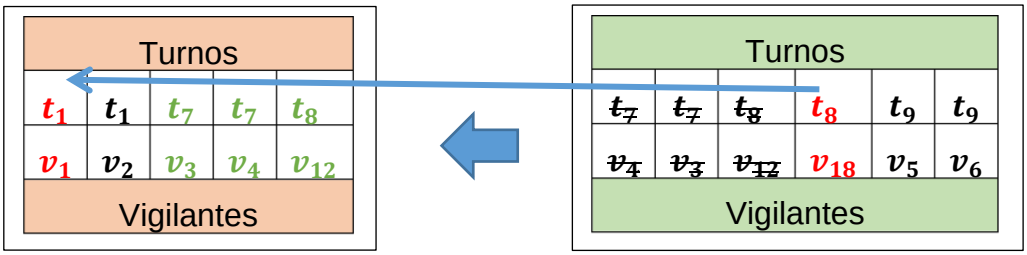
Paso 2: Según la lista de la **Figura 10,** el vigilante siguiente es v_4, al cual se le asigna el turno t_7.



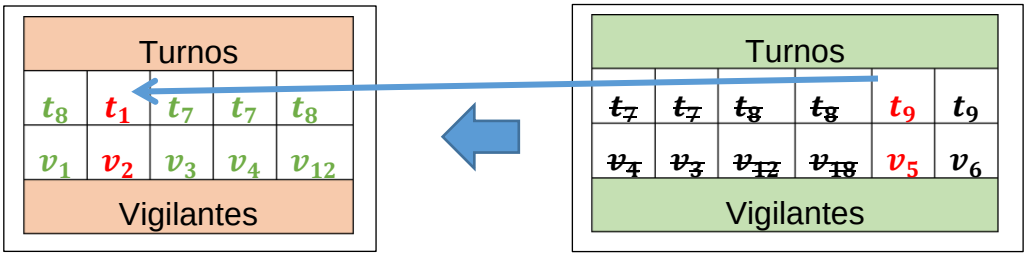
Paso 3: Según la lista de la **Figura 10,** el siguiente vigilante es v_{12}, al cual se le asigna el turno t_8.



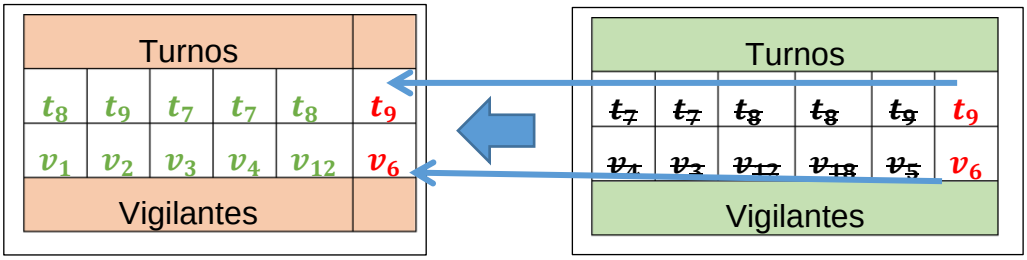
Paso 4: El siguiente vigilante es v_1, y se le asigna el turno t_8.



Paso 5: El siguiente vigilante es v_2, y se le asigna el turno t_9.



Paso 6: Como el ultimo vigilante v_6 no tiene pareja, se procede a asignarlo como un nuevo vigilante al sitio del gen defectuoso. Si es posible, se asigna el mismo vigilante, de lo contrario, se busca uno de la lista disponible de vigilantes.



A Continuación, se muestra en la **Figura 14** el resultado del cruce.

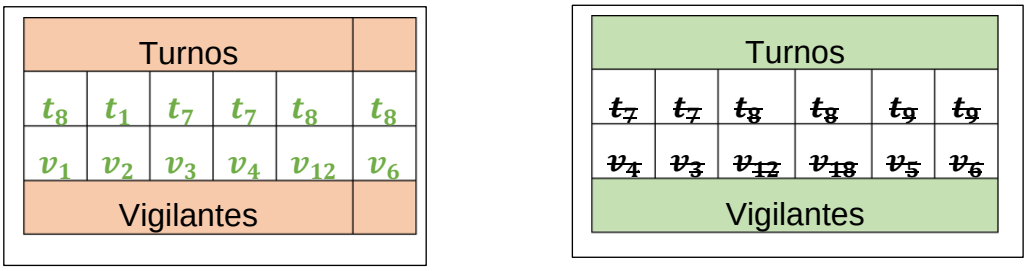


***Figura 14**. Cruce de turno entre vigilantes NSGA-II.*

4.4.2 CRUCE DE TURNOS PARCIAL

Teniendo en cuenta que la asignación de turnos en la solución inicial es a partir de tunos predefinidos (ver **Tabla 1)**, con este cruce se busca dar mayor variabilidad a la solución, con el principal objetivo de optimizar (minimizar) la cantidad de horas extras para los vigilantes.

Para la optimización, se seleccionan dos padres con respecto al fitness de minimización de horas extras de forma aleatoria de la lista restringida. Luego se toman dos sitios de forma aleatoria, un sitio por cada padre y por último se toma

dos turnos, un turno de cada sitio. Para este ejemplo, se continúa trabajando con los padres $\mathbf{f_1}$ y $\mathbf{f_3}$, y los sitos $\mathbf{s_{1,f_1}}$ y $\mathbf{s_{3,f_3}}$, de los cuales se selecciona el turno $\mathbf{t_{1,s_1}}$ del padre $\mathbf{f_1}$ (ver **Figura 15**) y el turno $\mathbf{t_{3,s_3}}$ del padre $\mathbf{f_3}$ (ver **Figura 16**).

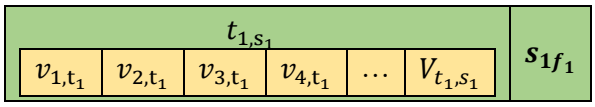


Figura 15. *Turno t_{1,s_1} de f_1 (Cruce parcial).*

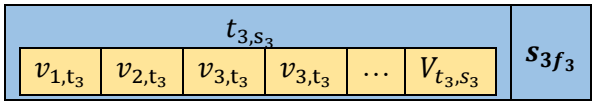


Figura 16. *Turno t_{3,s_3} de f_3 (Cruce parcial).*

Los turnos $\mathbf{t_{1,s_1}}$ y $\mathbf{t_{3,s_3}}$ se pueden representar como en la **Figura 17** y **Figura 18** respectivamente. Los periodos iniciales y finales de estos turnos son aleatorios, y fueron tomados como ejemplo solamente.

ID	$p_{inicial} = 20$	$p_{final} = 26$	V_{t_1,s_1}	v_{1,s_1}	v_{2,s_1}	…	VA_{t_1,s_1}

Figura 17. *Representación del turno t_{1,s_1} del padre f_1 (Cruce parcial).*

ID	$p_{inicial} = 18$	$p_{final} = 28$	V_{t_3,s_3}	v_{1,s_3}	v_{1,s_3}	…	VA_{t_1,s_3}

Figura 18. *Representación del turno t_{3,s_3} del padre f_3 (Cruce parcial).*

Para hacer el cruce de turnos, es necesario identificar cada periodo dentro de un turno de la siguiente forma $\boldsymbol{P_{t,s,f}}$, donde p representa el periodo en el turno t en el sitio s en el padre f. La **Figura 19** muestra la representación de los periodos para los turnos del ejemplo.

p_{20,t_1s_1,f_1}	p_{21,t_1s_1,f_1}	p_{22,t_1s_1,f_1}	p_{23,t_1s_1,f_1}	p_{24,t_1s_1,f_1}	p_{25,t_1s_1,f_1}	p_{26,t_1s_1,f_1}

P_{18,t_3s_3,f_3}	P_{19,t_3s_3,f_3}	P_{20,t_3s_3,f_3}	P_{21,t_3s_3,f_3}	…	P_{25,t_3s_3,f_3}	P_{26,t_3s_3,f_3}	P_{27,t_3s_3,f_3}	P_{28,t_3s_3,f_3}

Figura 19. *Representación periodos para turnos t_{1,s_1} y t_{3,s_3} (Cruce parcial).*

Pare dar mayor claridad, se agregan más turnos tanto al inicio como al final del turno $\mathbf{t_{1,s_1}}$ y del turno $\mathbf{t_{3,s_3}}$, y son representados de color más claro en la **Figura 20** y la **Figura 21**. En la **Figura 20** se tiene un turno $\mathbf{t_{1,s_1}}$ que va desde el periodo $p_{inicial} = p20$ al $p_{final} = p26$, y en la **Figura 21** el turno $\mathbf{t_{3,s_3}}$ va desde $p_{inicial} = p18$ y $p_{final} = p28$.

Con lo anterior se procede a realizar el cruce entre los sitios $\mathbf{s_{1,f_1}}$ y $\mathbf{s_{3,f_3}}$, por lo que el turno $\mathbf{t_{1,s_1}}$ se traslada en su totalidad al sitio $\mathbf{s_{3,f_3}}$, y seguidamente se hacen las reparaciones de los turnos anteriores y posteriores afectados por el cruce en el sitio $\mathbf{s_{3,f_3}}$.

t_0	…	p_{14}	p_{15}	p_{16}	p_{17}	p_{18}	p_{19}	p_{20}	p_{21}	p_{22}	p_{23}	p_{24}	p_{25}	p_{26}	p_{27}	p_{28}	p_{29}	p_{30}	p_{31}	…	T_{s,f_1}
		$\mathbf{t}_{(1-1),s_1}$						$\mathbf{t}_{1,s_1}$							$\mathbf{t}_{(1+1),s_1}$						

Figura 20. *Periodos del padre* f_1 *turno* t_{1,s_1} *(Cruce parcial).*

t_0	…	p_{14}	p_{15}	p_{16}	p_{17}	p_{18}	p_{19}	p_{20}	p_{21}	p_{22}	p_{23}	p_{24}	p_{26}	p_{27}	p_{28}	p_{29}	p_{30}	p_{31}	p_{32}	p_{33}	…	T_{s,f_3}
		$\mathbf{t}_{(3-1),s_3}$				$\mathbf{t}_{3,s_3}$										$\mathbf{t}_{(3+1),s_3}$						

Figura 21. *Periodos del padre* f_3 *turno* t_{3,s_3} *(Cruce parcial).*

En la **Figura 22** al finalizar el cruce se tiene que el turno $\mathbf{t}_{3,s_3}$ cambio su periodo inicial de $p18$ a $p20$, y su periodo final de $p28$ a $p26$, y para el turno inmediatamente anterior t_{3-1,s_3} cambio su periodo final de $p17$ a $p19$, y por último el turno t_{3+1,s_3} cambio su periodo inicial de $p29$ a $p27$.

Del cruce realizado se puede decir que los vigilantes que están trabajando en el turno 3 del sitio 3 es decir t_{3,s_3}, pasaron de trabajar 10 horas ($10p$) a trabajar 7 horas en ese turno, los vigilantes del turno anterior t_{3-1,s_3}, pasaron de trabajar 4 a 6 horas, y para el turno siguiente t_{3+1,s_3} pasaron de trabajar 5 horas a trabajar 7 horas.

t_0	…	p_{14}	p_{15}	p_{16}	p_{17}	p_{18}	p_{19}	p_{20}	p_{21}	p_{22}	p_{23}	p_{24}	p_{25}	p_{26}	p_{27}	p_{28}	p_{29}	p_{30}	p_{31}	p_{23}	p_{23}
	$\mathbf{t}_{(3-1),s_3}$							$\mathbf{t}_{3,s_3}$							$\mathbf{t}_{(3+1),s_3}$						

Figura 22. *Cruce de turnos parcial turnos* t_{1,s_1} *y* t_{3,s_3} *(Cruce parcial).*

En el proceso se tuvieron en cuenta 3 tipos posibles de combinaciones, para hacer el proceso más sencillo se hace las siguientes suposiciones:

Las figuras siguientes representan el total de periodos de un sitio s_a y s_b, donde ***a*** y ***b*** representan cualquier sitio dentro de los padres f_c y f_d, y donde **c** y ***d*** representan cualquier padre dentro de la población $\mathbf{F}$, tomados de forma aleatoria; para el ejemplo s_a y s_b contienen 24 periodos, los cuales están separados por colores.

Caso 1 (t_{a,s_a} *contiene a* t_{b,s_b})**:** Se quiere insertar el turno t_{a,s_a} (turno de color rojo) del sitio s_{a,f_a} (ver **Figura 23**) en el sitio s_{b,f_b}, de la **Figura 24** se observa que t_{b,s_b} y t_{b+1,s_b} tiene periodos en común con t_{a,s_a}. Como t_{a,s_a} y t_{b,s_b} comienzan en el mismo periodo, solo se debe ajustar el $p_{final} = p6$ a $p_{final} = p7$ del turno t_{b,s_b}, y del turno t_{b+1,s_b} solo se debe ajustar el $p_{Inicial} = p7$ a $p_{Inicial} = p8$, siendo así el cruce resultante el que se presenta en la **Figura 25**.

	t_{a,s_a}																					
s_{a,f_a}	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23

Figura 23. *Caso 1. Nuevo turno del sitio a (Cruce parcial).*

T	t_{b,s_b}							t_{b+1,s_b}										t_{b+2,s_b}						
s_{b,f_b}	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23

Figura 24. *Caso 1. Turno del sitio b (Cruce parcial).*

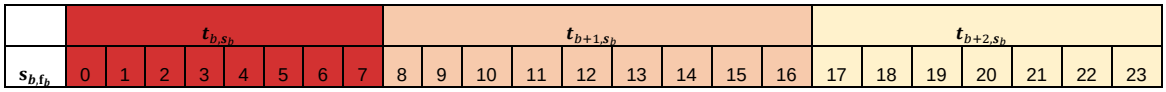


Figura 25. *Caso 1. Cruce turno parcial.*

Caso 2 ($\boldsymbol{t_{b,s_b}}$ *contiene a* $\boldsymbol{t_{a,s_a}}$): Se quiere insertar el turno $\boldsymbol{t_{a,s_a}}$ en el sitio $\boldsymbol{s_{b,f_b}}$. Se observa que $\boldsymbol{t_{a,s_a}}$ tiene periodos en común con $\boldsymbol{t_{b,s_b}}$, y como los dos turnos comienzan en el mismo periodo solo se debe ajustar del turno $\boldsymbol{t_{b,s_b}}$ el periodo $p_{final} = p_{10}$ a $p_{final} = p_5$, en ese proceso quedaron sin turnos los periodos p_6, p_7, p_8, p_9, p_{10} (huérfanos) del sitio $\boldsymbol{s_{b,f_b}}$, para solucionar este caso hay dos opciones; Opción 1, asignar los periodos huérfanos al turno siguiente si son menores a 4, opción 2, crear un nuevo turno con los periodos huérfanos si su cantidad de periodos es mayor o igual que 4. En este caso se toma la segunda opción dado que la cantidad de periodos huérfanos es 5. Por último, se asignan vigilantes en el nuevo turno si es posible. El cruce resultante se puede ver en la **Figura 26**.

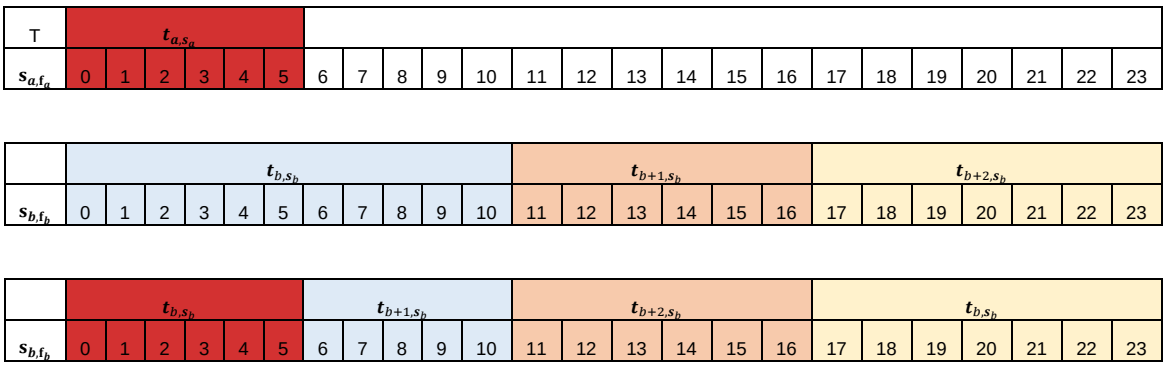
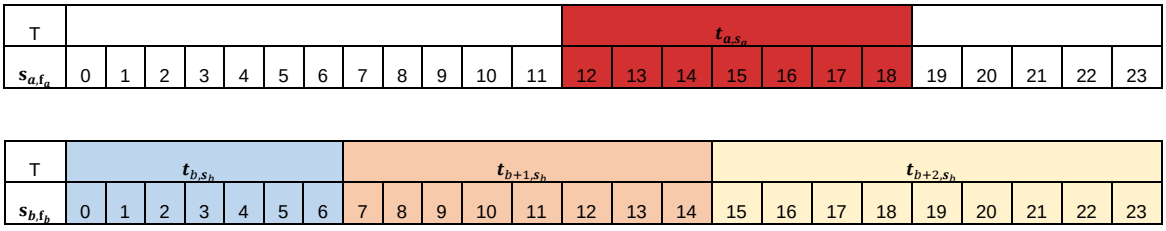


Figura 26. *Caso 2. Cruce de turno parcial.*

Caso 3 ($\boldsymbol{t_{a,s_a}}$ *esta contendio en mas de un turno*): Se quiere insertar el turno $\boldsymbol{t_{a,s_a}}$ en el sitio $\boldsymbol{s_{b,f_b}}$. Se observa que $\boldsymbol{t_{a,s_a}}$ tiene periodos en común con $\boldsymbol{t_{b+1,s_b}}$ y $\boldsymbol{t_{b+2,s_b}}$, para este caso se debe ajustar del turno $\boldsymbol{t_{b+1,s_b}}$ con $p_{final} = p11$ y se debe ajustar también el turno $\boldsymbol{t_{b+2,s_b}}$ que pasara a ser el turno $\boldsymbol{t_{b+3,s_b}}$ con su $p_{Inicial} = p19$. Finalmente se asigna el nuevo turno en la posición $\boldsymbol{t_{b+2,s_b}}$ con $p_{Inicial} = p12$ y $p_{Inicial} = p18$, de ser necesario se asignan nuevos vigilantes para el turno, siendo así el cruce resultante el que se muestra en la **Figura 27**.



	t_{b,s_b}							t_{b+1,s_b}					t_{b+2,s_b}							t_{b+3,s_b}				
s_{b,f_b}	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23

Figura 27. *Caso 3. Cruce de turno parcial.*

4.4.3 CRUCE DE VIGILANTES

Este cruce de vigilantes se realiza basado en la distancia del hogar del vigilante al sitio que se debe cubrir. Para explicar este cruce se recurre a un ejemplo. En este ejemplo cada vigilante tiene asociado una distancia a cada sitio de trabajo tal como se muestra en la **Tabla 6.**

Tabla 6. *Distancia Vigilante vs Sitio.*

Vigilante	s_1	s_2	s_3	s_4
v_1	50	20	30	60
v_2	10	20	30	20
v_3	20	10	50	60
v_4	60	40	20	15

Para la optimización de la distancia se hizo un intercambio de vigilantes teniendo como criterio el orden de los padres con respecto al fitness (objetivo) de distancia. Para cruzar el padre $f_1$con el padre f_2 se toman de forma aleatoria de la lista restringida.

En la **Figura 28** se tiene a f_1 y f_2 que corresponden a la soluciones padres y **c** como la solución hija. Los vigilantes $v_5, v_{11}, v_{9,}, v_{16}$ de f_1 se cruzan con f_2 para dar como resultado h_1. En el proceso de cruce se debe realizar la reparación de los genes dado que cuando se pasa un vigilante v_a a un padre f_b existirá una versión del vigilante v_a en el padre f_b , por este motivo se debe reasignar todos los vigilantes afectados en todos los sitios del padre f_b, es decir, que el cruce mantiene los turnos y varia los vigilantes, con esto se da por entendido que cuando un vigilante cambia de sitio existe la posibilidad de que se asigne a un puesto más cercano de trabajo.

En la **Figura 29** y **Figura 32** se muestra el proceso de reparación paso a paso. En la **Figura 29** se muestra en el primer paso donde el vigilante 3 en el padre 2 v_{3f_2} se desea cambiar por el vigilante v_{5f_1}, pero se tiene que este vigilante ya existe en f_2 por este motivo el v_{5f_2} se le debe desasignar sus turnos para evitar que cuando ingrese el nuevo vigilante v_{5f_1} no haya una sobreasignación de turnos para este vigilante. El siguiente paso es tomar el vigilante v_{3f_2} y asignarlo a los turnos del vigilante v_{5f_2}, por último, se pasa el nuevo vigilante v_{5f_1} al sitio s_2 del padre f_2 con los turnos que eran del vigilante v_{3f_2}.

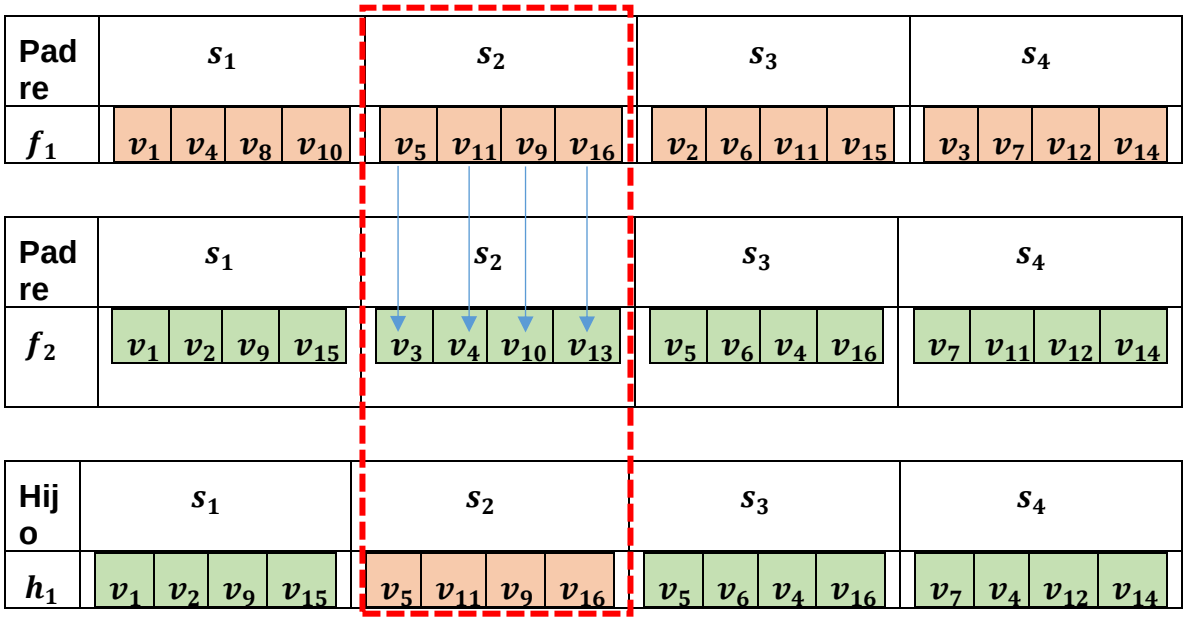


Figura 28. *Cruce de vigilantes.*

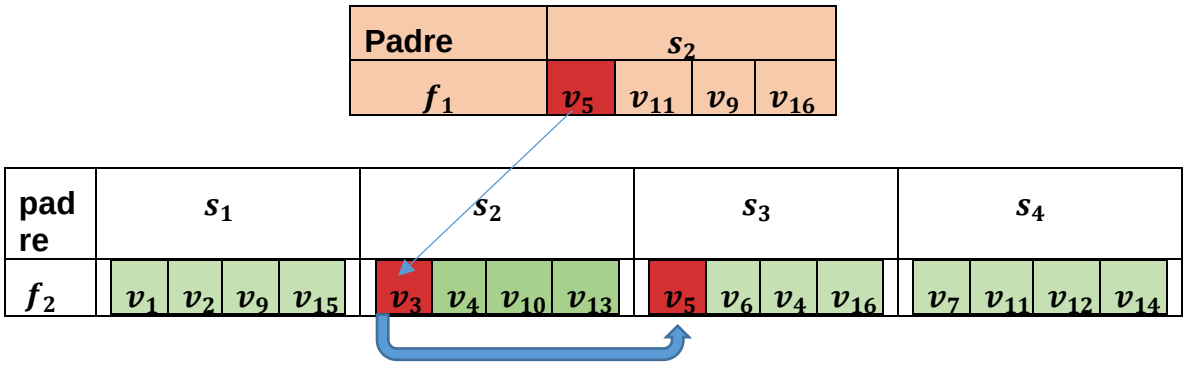


Figura 29. *Cruce de vigilantes (Paso 1).*

En la **Figura 30** se muestra como en el paso 2 el vigilante v_{4f_2} se cambia por el vigilante v_{11f_1} y a su vez el vigilante v_{11f_2} se cambiar por el vigilante v_{4f_1}.

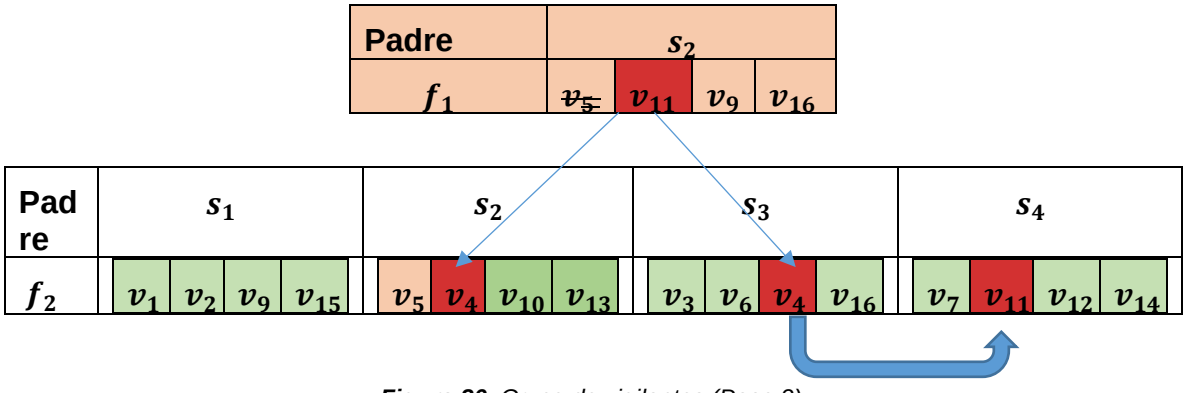


Figura 30. *Cruce de vigilantes (Paso 2).*

Después en la **Figura 31** se muestra como en el paso 3 el vigilante v_{10f_2} se cambia por el vigilante v_{9f_1}y a su vez el vigilante v_{9f_2} se cambiar por el vigilante v_{10f_1}.

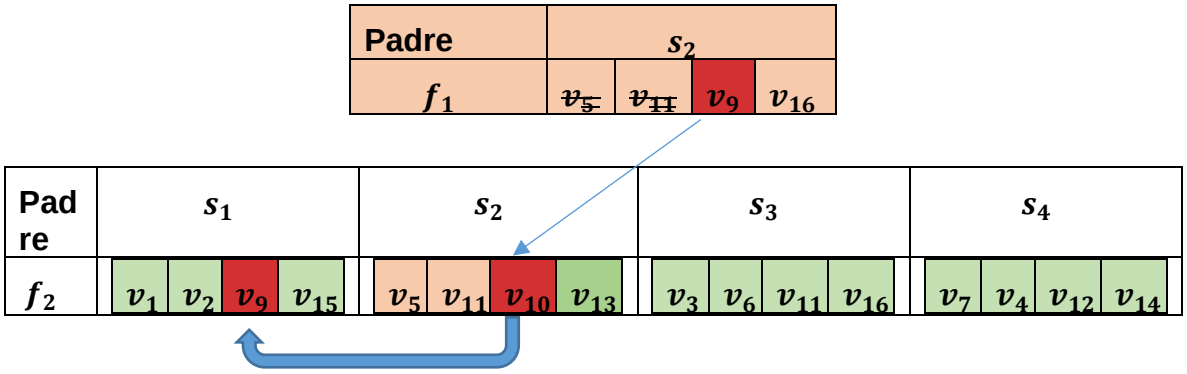


Figura 31. *Cruce de vigilantes (Paso 3).*

Luego en la **Figura 32** se muestra como en el paso 4 el vigilante v_{13f_2} se cambia por el vigilante v_{16f_1}y a su vez el vigilante v_{16f_2} se cambiar por el vigilante v_{13f_1}.

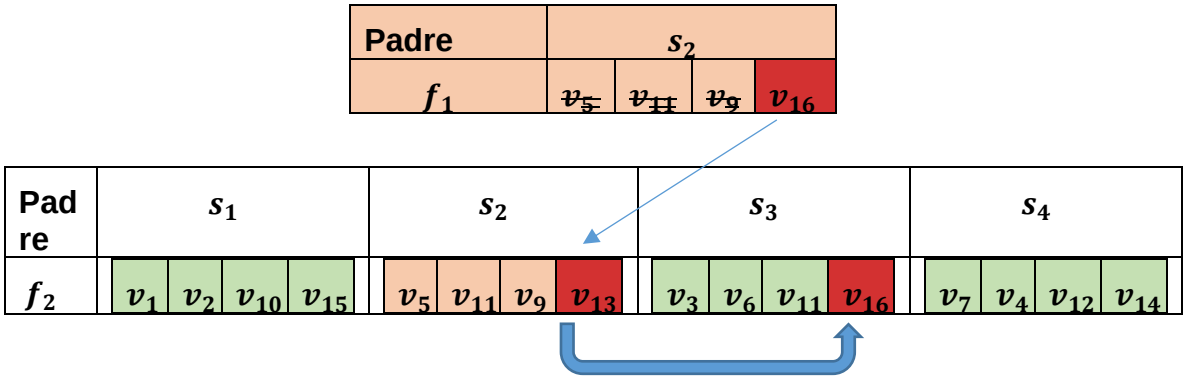


Figura 32. *Cruce de vigilantes (Paso 4).*

Y finalmente en la **Figura 33** se muestra el hijo resultante del cruce entre f_1 y f_2.

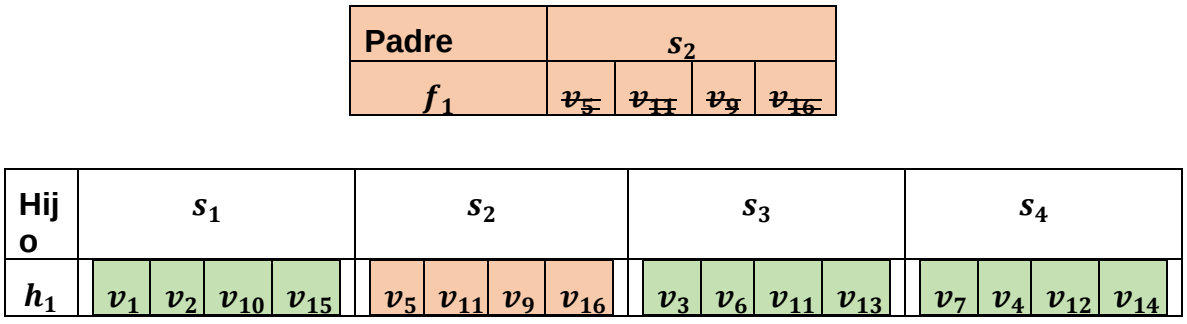


Figura 33. *Hijo resultante del cruce de vigilantes.*

En este punto ya se tiene los primeros descendientes de la población inicial, que se obtuvieron a partir de los diferentes mecanismos de cruce explicados anteriormente. Continuando con el **Algoritmo 3** el siguiente paso es hacer la unión de los padres

e hijos para luego ordenar la población basado en el Frente de Pareto el cual se explica en la siguiente sección**.**

4.5 FRENTE DE PARETO

Las soluciones que no son dominadas por ninguna otra solución en la población son las que forman el Frente de Pareto, y estas son las soluciones que el algoritmo retorna como las mejores soluciones encontradas [10][29]. El conjunto de soluciones se puede separar en rangos, el rango uno (1) corresponde al conjunto óptimo de soluciones o Frente de Pareto. Las soluciones que perteneces al conjunto del rango 2 son menos buenas que las soluciones del rango uno (1). Las soluciones del rango tres (3) tienen menor calidad que las del rango 1 y 2 y así sucesivamente con los rangos siguientes [29].

4.5.1 FAST NON-DOMINATED SORT

Para determinar los rangos se utilizó el algoritmo llamado **Ordenamiento No Dominado Rápido (Fast Non-Dominated Sort)** que es uno de los algoritmos más conocido y óptimos para el cálculo de estos rangos (ver **Algoritmo 6**) [21][29]. El algoritmo de divide en dos partes la primera de la línea 3 a la 21 que se encarga de hacer un conteo y la selección de las soluciones que son dominadas por otra solución. Por ejemplo, si se tienen una solución A, esta solución tiene una lista de las soluciones que A domina (**Dominados**), para esto se usa el algoritmo de dominancia que se presentó anteriormente. A además cuenta con un contador (**MeDominan**) donde se registra el número de soluciones que dominan a la solución A. Es así como las soluciones que no son dominadas por ninguna otra solución, es decir aquellas con MeDomina = 0, hacen parte del Frente de Pareto y por lo tanto se le asigna al rango el valor de uno (1) como se muestra en la línea 18 del algoritmo.

La segunda parte, de la línea 23 a la 34 hace la definición de los siguientes rangos (2, 3, en adelante). Para esto inicia seleccionado las soluciones del rango 1 y a todas las soluciones que son dominadas por las soluciones de este rango se les decrementa el contador **MeDominan** (línea 26), si una solución de esas llega a quedar en cero en este contador se asigna como parte del rango 2 (línea 27 a 30). Este proceso se repite hasta procesar todas las soluciones del rango 1 (línea 24 a 32), en ese punto se aumenta el rango a 2 y se repite el mismo proceso hasta que se llegue a un rango donde ya no haya soluciones (línea 23).

Entradas:	N: Tamaño de la población, P: Población actual de soluciones // Se modifican tres campos de cada solución, O: Lista de M Objetivos a optimizar // Se asume minimización de todos ellos
Salida:	Rangos de Pareto con las soluciones asignadas en cada uno de ellos
1.	**Inicio**
2.	Frentes = Nueva lista dinámica vacía
3.	**Para** i **desde** 1 **hasta** N **haga:**
4.	P[i]. Dominados = Nueva lista vacía
5.	P[i]. MeDominan = 0

```
6.          P[i]. Rango = -1
7.          Para j desde 1 hasta N haga:
8.              Si i == j entonces: Continuar Para
9.              Si P[i] domina a P[j] entonces:
10.                 Adicione P[j] a la lista P[i].Dominados
11.             Sino
12.                 Si P[j] domina a P[i] entonces:
13.                     P[i].MeDominan = P[i].MeDominan + 1
14.                 Fin Si
15.             Fin Si
16.         Fin Para
17.         Si P[i].MeDominan == 0 entonces:
18.             P[i]. Rango = 1
19.             Adicione P[i] en Frentes[1]
20.         Fin Si
21.     Fin Para
22.     rango = 1
23.     Mientras Frentes[rango] tenga soluciones haga:
24.         Para cada solución p del Frentes[rango] haga:
25.             Para cada solución q de p.Dominados haga:
26.                 q.MeDominan = q.MeDominan – 1
27.                 Si (q.MeDominan == 0) entonces:
28.                     q.Rango = rango + 1
29.                     Adicione q en Frentes[rango + 1]
30.                 Fin Si
31.             Fin Para
32.         Fin Para
33.         rango = rango + 1
34.     Fin Mientras
35.     Retornar Frentes
36. Fin
```

***Algoritmo 6**. Fast Non-Dominated Sort. Tomado de* [30]

La **Figura 34** a manera de ejemplo muestra el Frente de Pareto (problema de minimización) compuesto por las soluciones D, C, E, B y A, que no son dominadas por ninguna otra solución de la población. Las soluciones que corresponde al rango dos (2) son la F, G y H que solo son dominadas por algunas soluciones en Frente de Pareto o rango 1. Y las soluciones del rango tres (3), en este caso la I y la J son dominadas por algunas soluciones del rango 1 y 2.

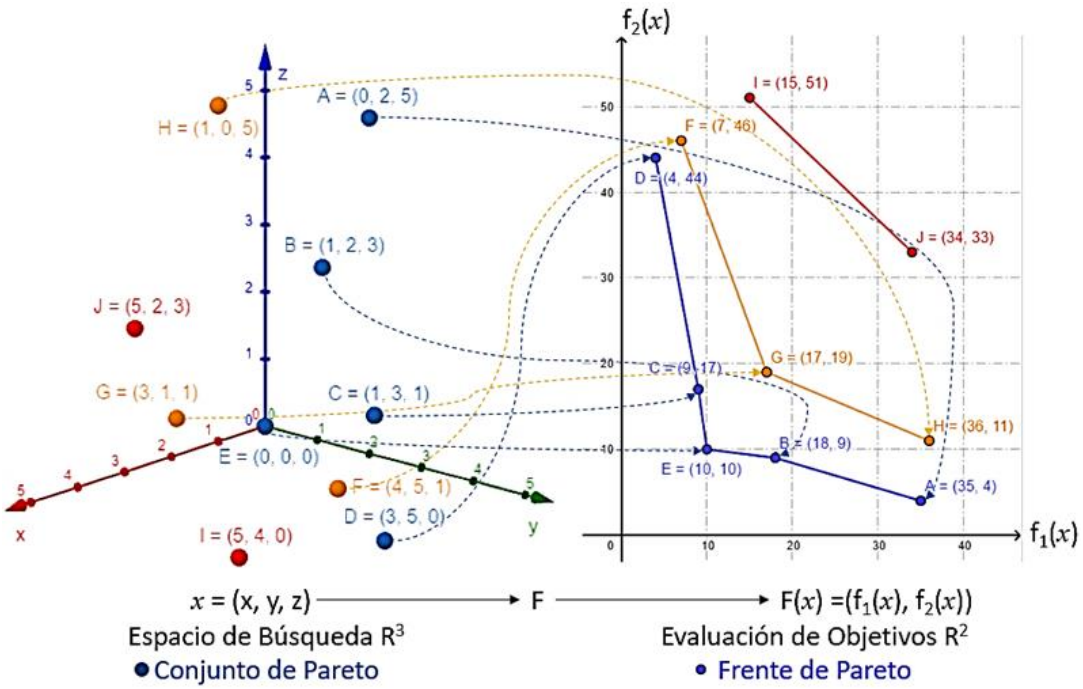


***Figura 34**. Conjuntos de Pareto y Frente de Pareto. Tomado de* [29] [30].

En la **Figura 34** se muestras 5 soluciones (D, C, E, B y A) en el Frente de Pareto si por alguna razón se requiere seleccionar sólo 4, una alternativa para hacer esto consiste en seleccionar las soluciones más separadas de las otras o las soluciones menos apiñadas a las otras. Para hacer esto se utiliza la Distancia de Crowding (ver **Algoritmo 7**).

Entradas: C: Número de soluciones en el Frente (rango) de Pareto que se va a procesar, F: Frente de Pareto con C soluciones // Se modifica un atributo de cada solución, O: Lista de M Objetivos a optimizar // Se asume minimización de todos ellos	
Salida: Frente (Rango) de Pareto con C soluciones y sus valores de Distancia de Crowding	
1.	**Inicio**
2.	Rangos = Lista de M valores con el rango de los datos de las soluciones en el Frente para cada objetivo. Es decir, para cada uno de los M objetivos se debe calcular el mayor valor y el menor valor de las soluciones del Frente y se almacena su rango (mayor - menor)
3.	**Para** i **desde** 1 **hasta** C **haga:**
4.	F.solucion[i].DistanciaCrowding = 0
5.	**Fin Para**
6.	**Para** j **desde** 1 **hasta** M **haga:**
7.	Ordene las soluciones del Frente de Pareto por el valor del objetivo j (O[j]) de menor a mayor
8.	F.solucion[1].DistanciaCrowding = Infinito
9.	**Para** i **desde** 2 **hasta** C-1 **haga:**
10.	F.solucion[i].DistanciaCrowding = F.solucion[i].DistanciaCrowding + (F.solucion[i+1].O[j] - F.solucion[i-1].O[j]) / Rangos[j]
11.	**Fin Para**

```
12.             F.solucion[N].DistanciaCrowding = Infinito
13.         Fin Para
14.         Retornar F
15.     Fin
```

Algoritmo 7. *Distancia de Crowding para un Rango de Pareto. Tomado de* [30].

CAPÍTULO 5

5 GRASP MULTIOBJETIVO

Este capítulo presenta la metaheurística GRASP y su adaptación al problema multiobjetivo de programación de personal de seguridad y vigilancia. Además, presenta el proceso de configuración y optimización para las diferentes soluciones obtenidas.

5.1 GRASP

El procedimiento de búsqueda adaptativa aleatoria codiciosa (GRASP) [22] es una metaheurística iterativa o de inicio múltiple en la que cada iteración consta de dos fases: construcción y búsqueda local. Este proceso se resume en la **Algoritmo 8**.

La fase de construcción (línea 4) tiene como objetivo crear una solución, si la solución no es factible, se aplica un procedimiento de reparación para buscar su factibilidad, en caso de que no se pueda alcanzar la factibilidad en GRASP la solución se descarta y se crea una nueva solución. Cuando se obtiene una solución factible, se procede con la fase de búsqueda local (línea 5), donde se buscan soluciones en las vecindades cercanas a la solución actual hasta encontrar una mejor solución, o hasta encontrar la mejor solución entre las vecindades. Luego, en la línea 6 se revisa si esa solución es mejor a la mejor solución que se ha encontrado en todo el proceso iterativo, si es así la remplaza con la solución actual (**Sol**).

Entradas: TLR: Tamaño de lista restringida, NV: Número de vecindades CE: Cantidad de elementos a construir	
Salida: Mejor solución encontrada	
1.	**Inicio**
2.	best = vacía
3.	**Mientras** el criterio de parada no se satisfaga **haga**
4.	Sol = Construir_solucion (TLR, CE) // Por un método aleatorizado y voraz
5.	Sol = Buscar_localmente (Sol, NV)
6.	Actualizar_solucion (Sol, best) // Si la nueva solución es mejor se remplaza por la anterior
7.	**Fin Mientras**
8.	**Retornar** best
9.	**Fin**

***Algoritmo 8**. GRASP.*

En un algoritmo codicioso, las soluciones se construyen progresivamente desde cero, en cada paso se incorpora un nuevo elemento seleccionado aleatoriamente del conjunto base de elementos (normalmente ordenados por un criterio codicioso que están en una lista de tamaño restringido) a la solución parcial que está en construcción, hasta obtener una solución factible completa.

Para la fase de construcción se empieza con una solución vacía y en cada iteración se van creando diferentes elementos (para el problema de programación de personal de seguridad y vigilancia se crean ***componentes***) de los cuales se crea una lista restringida de los TLR mejores elementos obtenidos. Luego, se escoge un elemento aleatoriamente de la lista restringida y se une a la solución. Este proceso se repite hasta que la solución esté completamente construida o el criterio de parada se cumpla, tal como se muestra en el **Algoritmo 9**.

Entradas: TLR: Tamaño de lista restringida CE: Cantidad de elementos a construir	
Salida: Solución construida	
1.	**Inicio**
2.	Sol = vacía
3.	**Mientras** la construcción de la solución no haya terminado **haga:**
4.	ELS = funcion_greedy (CE) // Función que crea los posibles elementos de una solución con base en un criterio voraz
5.	RCL = Crear_RCL (ELS, TLR) // Lista de candidatos restringida a un tamaño TLR
6.	e = Seleccionar_aleatoriamente_elemento (RCL) // Selecciona un elemento
7.	Sol = Sol U e // Se agrega el elemento a la solución
8.	**Fin Mientras**
9.	**Retornar** Sol
10.	**Fin**

Algoritmo 9*. Fase de construcción (Construir_solucion) en GRASP.*

Las soluciones generadas por algoritmos codiciosos no son necesariamente óptimas, incluso con respecto a los vecindarios más simples. La fase de búsqueda local busca mejorar este problema de forma iterativa reemplazando sucesivamente la solución actual por una mejor solución en alguna vecindad de la solución actual. Este proceso termina cuando no se encuentra una mejor solución en el vecindario o se cumple otro criterio de parada. El **Algoritmo 10** resume la fase de búsqueda local, la cual parte de una solución y hace uso de una estructura de vecindades para mejorar la solución en el problema.

La efectividad de un procedimiento de búsqueda local depende de varios aspectos, como por ejemplo la estructura del vecindario, la técnica de búsqueda del vecindario, la función de evaluación y la solución de partida. La búsqueda local puede implementarse usando una estrategia de mejora máxima o de mejora inicial.

En el caso de una mejor máxima, se investigan todos los vecinos y la actual solución es reemplazada por el mejor vecino. En el caso de una estrategia de mejora inicial la solución actual se mueve al primer vecino cuyo valor de la función de evaluación es mejor a la de la solución actual.

Entradas: Sol: Solución actual, NV: Número de vecindades	
Salida: Solución óptima local encontrada	
1.	**Inicio**
2.	**Mientras** Sol no sea localmente óptima o se halla intentado NV veces **haga:**
3.	SolN = Buscar_mejor_solucion (Sol, NV (Sol)) // la mejor solución debe estar en el vecindario de sol, NV (sol)
4.	Actualizar (Sol, SolN) // remplazar la solución sol con SolN
5.	**Fin Mientras**
6.	**Retornar** Sol // solución óptima local
7.	**Fin**

Algoritmo 10. *Fase de búsqueda local (Buscar_localmente) en GRASP.*

5.2 GRASP MULTIOBJETIVO

Dado que GRASP es un procedimiento mono objetivo, se realizaron modificaciones para que pueda ser usado en un ambiente multiobjetivo. En GRASP multiobjetivo se retorna un conjunto de soluciones que se ubican en el Frente de Pareto y que evolucionan en el proceso de búsqueda local. Este proceso se explica en el ***Algoritmo* 11**.

Entradas: N: Tamaño de soluciones en la población E: Número máximo de evoluciones de los objetivos CE: Cantidad de elementos a construir TLR: Tamaño de lista restringida NV: Número de vecindades	
Salida: Frente de Pareto encontrado	
1.	**Inicio**
2.	F = vacío //Población inicia vacía
3.	**Mientras** |F| <= N **haga:**
4.	Sol = Construir_solucion (TLR, NV) // Para este caso se utiliza el **Algoritmo 1** para crear una solución como función greedy
5.	F = F U Sol
6.	**Fin Mientras**
7.	**Mientras** no se hayan completado todas las evoluciones E **haga:**
8.	**Para** cada solución Sol en la población F **haga:**
9.	Sol2 = Buscar_localmente (Sol, NV) // Para nuestro caso se escoge aleatoriamente un objetivo a minimizar y se crean vecindades enfocadas en mejorar ese objetivo particular
10.	F = F U Sol2
11.	**Fin Para**
12.	F = Obtener_mejores_soluciones (F, N) // Se seleccionan las N mejores soluciones de la población con ayuda de los Frentes de Pareto
13.	**Fin Mientras**
14.	**Retornar** Frente de Pareto (Rango 1) de F
15.	**Fin**

Algoritmo 11. *GRASP Multiobjetivo (MGRASP).*

Primero, ya no se crea una única solución si no que se crea una población de $\mathbf{N}$ soluciones, siendo esta población la población inicial (líneas 3 a 5). En la función Greedy para el caso de estudio se utiliza el **Algoritmo 1** explicado previamente en el **Capítulo 3.**

Posteriormente se pone a evolucionar cada una de las soluciones obtenidas en la población con el proceso de búsqueda local (líneas 8 a 11). Para generar las vecindades cercanas a una solución se escoge aleatoriamente uno de los objetivos a minimizar en el problema (la cantidad de turnos no asignados, la cantidad de guardias de seguridad requeridos, las horas extras del personal de seguridad y la distancia entre el hogar del guardia de seguridad y el puesto de trabajo), y se crea una nueva solución vecina enfocada en optimizar el objetivo en concreto, si la nueva solución es mejor, se remplaza con la solución actual. Este proceso se repite hasta que se cumpla la condición de parada que se haya especificado en la fase de búsqueda local.

Al finalizar la fase de búsqueda local, la solución obtenida es añadida a la población (línea 10), y se repite el proceso hasta que se haya aplicado la fase de búsqueda local a cada una de las soluciones obtenidas en la población. De la población final obtenida se escoge las mejores N soluciones, y pasan a formar la nueva población a evolucionar (línea 12). Cuando se termina de realizar todas las evoluciones se retorna el Frente de Pareto de la población final (línea 14).

Para escoger las mejores $\mathbf{N}$ soluciones de la población se hace uso del algoritmo de ordenamiento rápido de soluciones no dominadas (**Algoritmo 6**) y la distancia de Crowding (**Algoritmo 7**). Para el criterio de búsqueda local se crearon 4 métodos de refinamientos que buscan mejorar en el espacio de búsqueda, uno para cada objetivo a minimizar. A cada refinamiento se le asigno un peso que refleja la probabilidad en que pueden ser escogidos.

5.2.1 REFINAMIENTO TURNOS SIN ASIGNAR

En este refinamiento se toma en cuenta los siguientes procesos para cada uno de los sitios que todavía tienen turnos por cubrir:

1. Asignar a los turnos los vigilantes que han trabajado menos de los periodos ideales y que se encuentran asignados en el mismo sitio.
2. Asignar horas extras a los vigilantes que se encuentran asignados en el mismo sitio.
3. Asignar a los turnos los vigilantes que han trabajado menos de los periodos ideales y que se encuentran asignados en un sitio diferente.
4. Asignar horas extras a los vigilantes que se encuentran asignados en un sitio diferente.

Para el primer caso se valida que los vigilantes asignados en el sitio aún no han cumplido con el número de periodos ideal a trabajar $\boldsymbol{n^{ideal}}$ en la semana, con el fin

de tratar de ver si pueden ser asignados en algún turno incompleto. En la

t_1	p_0	p_7	1	v_1
t_2	p_8	p_{15}	1	v_2
t_3	p_{16}	p_{23}	1	v_3
t_4	p_{24}	p_{31}	1	v_4
s_1				

t_1	p_0	p_7	2	v_5	v_6
t_2	p_8	p_{15}	2	v_7	
t_3	p_{16}	p_{23}	2		v_8
t_4	p_{24}	p_{31}	2	v_7	
s_2					

t_1	p_0	p_7	2	v_9	v_3
t_2	p_8	p_{15}	2	v_{10}	v_{11}
t_3	p_{16}	p_{23}	2	v_5	
t_4	p_{24}	p_{31}	2	v_{10}	
s_3					

Vigilante	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}
Periodos asignados en la semana	56	48	48	32	48	40	40	48	40	48	48

Figura 35, se muestra un caso en el que se tienen 3 sitios con sus respectivos turnos y que en algunos de ellos hay turnos que no se pudieron asignar (sombreado con color rojo), para el caso de ejemplo se toma el número de periodos ideal a trabajar en la semana $n^{ideal} = 48$, un numero de periodos máximo $n^{max} = 56$, con un tiempo de descanso $T^{rest} = 8$ y que los vigilantes han trabajado en la semana la cantidad de periodos representados en la

t_1	p_0	p_7	1	v_1
t_2	p_8	p_{15}	1	v_2
t_3	p_{16}	p_{23}	1	v_3
t_4	p_{24}	p_{31}	1	v_4
s_1				

t_1	p_0	p_7	2	v_5	v_6
t_2	p_8	p_{15}	2	v_7	
t_3	p_{16}	p_{23}	2		v_8
t_4	p_{24}	p_{31}	2	v_7	
s_2					

t_1	p_0	p_7	2	v_9	v_3
t_2	p_8	p_{15}	2	v_{10}	v_{11}
t_3	p_{16}	p_{23}	2	v_5	
t_4	p_{24}	p_{31}	2	v_{10}	
s_3					

Vigilante	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}
Periodos asignados en la semana	56	48	48	32	48	40	40	48	40	48	48

Figura 35.

t_1	p_0	p_7	1	v_1
t_2	p_8	p_{15}	1	v_2
t_3	p_{16}	p_{23}	1	v_3
t_4	p_{24}	p_{31}	1	v_4
s_1				

t_1	p_0	p_7	2	v_5	v_6
t_2	p_8	p_{15}	2	v_7	
t_3	p_{16}	p_{23}	2		v_8
t_4	p_{24}	p_{31}	2	v_7	
s_2					

t_1	p_0	p_7	2	v_9	v_3
t_2	p_8	p_{15}	2	v_{10}	v_{11}
t_3	p_{16}	p_{23}	2	v_5	
t_4	p_{24}	p_{31}	2	v_{10}	
s_3					

Vigilante	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}
Periodos asignados en la semana	56	48	48	32	48	40	40	48	40	48	48

Figura 35. *Refinamiento turnos sin asignar MGRASP.*

Como el sitio s_1 no tiene turnos faltantes, se revisa únicamente el sitio s_2 y s_3. Para el sitio s_2 falta cubrir los turnos t_{2,s_2}, t_{3,s_2} y t_{4,s_2}, y solamente los vigilantes v_6 y v_7

tienen asignados menos de los periodos ideales, por lo que se escoge uno de ellos aleatoriamente y se valida si puede ser asignado en el turno $\boldsymbol{t_{2,s_2}}$, en este caso ninguno de los vigilantes $\boldsymbol{v_6}$ y $\boldsymbol{v_7}$ pueden ser asignados debido a la restricción de descanso. Se procede entonces con el turno $\boldsymbol{t_{3,s_2}}$, en este caso, el vigilante $\boldsymbol{v_6}$ cumple con la restricciones y se le asigna el turno. Se escoge finalmente el turno $\boldsymbol{t_{4,s_2}}$, y ahora solamente el vigilante $\boldsymbol{v_7}$ tiene menos de los periodos esperados, pero no puede ser asignado debido a que ya se encuentra asignado en el turno. El proceso se repite para el sitio $\boldsymbol{s_3}$, en donde solamente el vigilante $\boldsymbol{v_9}$ tiene menos de las horas ideales, se escoge el turno $\boldsymbol{t_{3,s_3}}$ y se valida si puede ser asignado, como el vigilante $\boldsymbol{v_9}$ cumple con la restricciones se asigna al turno $\boldsymbol{t_{3,s_3}}$(ver **Figura 36**).

t_1	p_0	p_7	1	v_1
t_2	p_8	p_{15}	1	v_2
t_3	p_{16}	p_{23}	1	v_3
t_4	p_{24}	p_{31}	1	v_4
s_1				

t_1	p_0	p_7	2	v_5	v_6
t_2	p_8	p_{15}	2	v_7	
t_3	p_{16}	p_{23}	2	v_6	v_8
t_4	p_{24}	p_{31}	2	v_7	
s_2					

t_1	p_0	p_7	2	v_9	v_3
t_2	p_8	p_{15}	2	v_{10}	v_{11}
t_3	p_{16}	p_{23}	2	v_5	v_9
t_4	p_{24}	p_{31}	2	v_{10}	
s_3					

Vigilante	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}
Periodos asignados en la semana	56	48	48	32	48	48	40	48	48	48	48

Figura 36. *Refinamiento turnos sin asignar MGRASP (criterio 1).*

Como aún hay turnos que no están completos, se sigue con el segundo criterio que consiste en asignar horas extras a los vigilantes que se encuentran asignados en el mismo sitio (ver **Figura 37**). Para el sitio $\boldsymbol{s_2}$ todos los vigilantes tienen disponibilidad para asignarles horas extras, pero ninguno cumple con las restricciones de descanso en el turno $\boldsymbol{t_{2,s_2}}$ y $\boldsymbol{t_{4,s_2}}$. Se escoge el sitio $\boldsymbol{s_3}$, y allí todos los vigilantes tienen disponibilidad para asignarles horas extras, pero únicamente el vigilante $\boldsymbol{v_{11}}$ puede ser asignado.

t_1	p_0	p_7	1	v_1
t_2	p_8	p_{15}	1	v_2
t_3	p_{16}	p_{23}	1	v_3
t_4	p_{24}	p_{31}	1	v_4
s_1				

t_1	p_0	p_7	2	v_5	v_6
t_2	p_8	p_{15}	2	v_7	
t_3	p_{16}	p_{23}	2	v_6	v_8
t_4	p_{24}	p_{31}	2	v_7	
s_2					

t_1	p_0	p_7	2	v_9	v_3
t_2	p_8	p_{15}	2	v_{10}	v_{11}
t_3	p_{16}	p_{23}	2	v_5	v_9
t_4	p_{24}	p_{31}	2	v_{10}	v_{11}
s_3					

Vigilante	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}
Periodos asignados en la semana	56	48	48	32	48	48	40	48	48	48	56

Figura 37. *Refinamiento turnos sin asignar MGRASP (criterio 2).*

Como aún hay turnos que no están completos, se sigue entonces con el tercer criterio de asignar a los turnos los vigilantes que han trabajado menos de los periodos ideales y que se encuentran asignados en un sitio diferente. Para este

criterio se tiene que solo el vigilante $\boldsymbol{v_4}$ cumple con tener asignado menos de los periodos ideales, se escoge aleatoriamente el turno $\boldsymbol{t_{2,s_2}}$ y se asigna el vigilante $\boldsymbol{v_4}$ dado que cumple con las restricciones. Luego se escoge el turno $\boldsymbol{t_{4,s_2}}$ y aunque el vigilante $\boldsymbol{v_4}$ aún no cumple con los periodos ideales no puede ser asignado debido a que se encuentra asignado en el turno $\boldsymbol{t_{4,s_1}}$.

Finalmente se intenta con el ultimo criterio de asignar horas extras a los vigilantes que se encuentran asignados en un sitio diferente al actual, aquí se tiene disponibilidad de los vigilantes v_2, v_3, v_9, v_{10} pero solo el vigilante v_2 cumple la restricción de descanso y por esto se asigna (ver **Figura 38**).

t_1	p_0	p_7	1	v_1
t_2	p_8	p_{15}	1	v_2
t_3	p_{16}	p_{23}	1	v_3
t_4	p_{24}	p_{31}	1	v_4
s_1				

t_1	p_0	p_7	2	v_5	v_6
t_2	p_8	p_{15}	2	v_7	v_4
t_3	p_{16}	p_{23}	2	v_6	v_8
t_4	p_{24}	p_{31}	2	v_7	v_2
s_2					

t_1	p_0	p_7	2	v_9	v_3
t_2	p_8	p_{15}	2	v_{10}	v_{11}
t_3	p_{16}	p_{23}	2	v_5	v_9
t_4	p_{24}	p_{31}	2	v_{10}	v_{11}
s_3					

Vigilante	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}
Periodos asignados en la semana	56	56	48	40	48	48	40	48	48	48	56

Figura 38. *Refinamiento turnos sin asignar MGRASP (criterio 3 y 4).*

5.2.2 REFINAMIENTO DE LA CANTIDAD DE VIGILANTES ASIGNADOS

En este refinamiento se toma en cuenta los siguientes procesos para cada uno de los vigilantes que no cumplieron con la cantidad mínima de horas establecidas:

1. Transferir los turnos entre un vigilante que tiene menos de los periodos $\boldsymbol{n^{ideal}}$ contra otros con la misma característica.
2. Transferir los turnos entre un vigilante que tiene menos de los periodos $\boldsymbol{n^{ideal}}$ contra otro que tiene igual o más de los periodos $\boldsymbol{n^{ideal}}$.

Se toma de base el ejemplo de la **Figura 38** y se agrega otro sitio y vigilantes tal como se muestra en la **Figura 39** con el fin de facilitar la explicación de los 2 criterios anteriormente mencionados. El primer criterio consiste en transferir los turnos de los vigilantes que han sido asignados una cantidad de periodos menor a la $\boldsymbol{n^{ideal}}$ contra otros vigilantes con las mismas características.

t_1	p_0	p_7	1	v_1
t_2	p_8	p_{15}	1	v_2
t_3	p_{16}	p_{23}	1	v_3
t_4	p_{24}	p_{31}	1	v_4
s_1				

t_1	p_0	p_7	2	v_5	v_6
t_2	p_8	p_{15}	2	v_7	v_4
t_3	p_{16}	p_{23}	2	v_6	v_8
t_4	p_{24}	p_{31}	2	v_7	v_2
s_2					

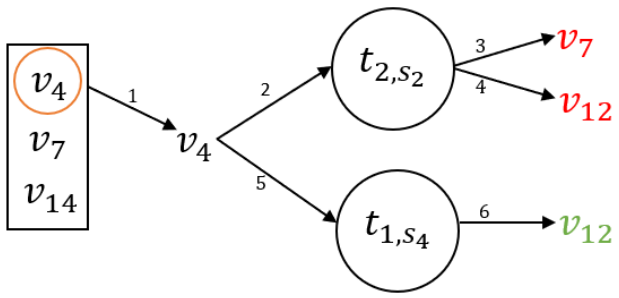
t_1	p_0	p_7	1	v_{12}
t_2	p_8	p_{15}	1	v_{13}
t_3	p_{16}	p_{23}	1	v_{14}
t_4	p_{24}	p_{31}	1	v_{13}
s_4				

t_1	p_0	p_7	2	v_9	v_3
t_2	p_8	p_{15}	2	v_{10}	v_{11}
t_3	p_{16}	p_{23}	2	v_5	v_9
t_4	p_{24}	p_{31}	2	v_{10}	v_{11}
s_3					

Vigilante	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}	v_{14}
Periodos asignados en la semana	56	56	48	40	48	48	40	48	48	48	56	40	48	48

Figura 39. *Refinamiento cantidad de vigilantes asignados MGRASP.*

Para el ejemplo se tiene que los vigilantes v_4, v_7, v_{12} no cumplen con los periodos $\boldsymbol{n^{ideal}}$, se toma aleatoriamente al vigilante $\boldsymbol{v_4}$ y se escoge primero de sus turnos el turno $\boldsymbol{t_{2,s_2}}$, para transferirlo a otros vigilantes, aleatoriamente se escoge el vigilante $\boldsymbol{v_7}$ y se valida si puede ser asignado en ese turno; como no se pude asignar, se valida contra el vigilante $\boldsymbol{v_{12}}$, el cual tampoco está disponible en esos periodos. Como no hay más vigilantes a comparar se escoge el turno $\boldsymbol{t_{4,s_1}}$; aleatoriamente se toma el vigilante $\boldsymbol{v_{12}}$ y en este caso si puede ser asignado, de manera que se hace la transferencia del turno $\boldsymbol{t_{4,s_2}}$ del vigilante $\boldsymbol{v_4}$ al vigilante $\boldsymbol{v_{12}}$. Debido a que el vigilante $\boldsymbol{v_4}$ ya no tiene más turnos entonces termina el proceso (ver Figura 40), en caso contrario se realizaría lo mismo con el siguiente turno.



t_1	p_0	p_7	1	v_1
t_2	p_8	p_{15}	1	v_2
t_3	p_{16}	p_{23}	1	v_3
t_4	p_{24}	p_{31}	1	v_{12}
s_1				

Vigilante	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}	v_{14}
Periodos asignados en la semana	56	56	48	32	48	48	40	48	48	48	56	48	48	48

***Figura 40**. Refinamiento cantidad de vigilantes asignados MGRASP (criterio 1).*

Para el segundo criterio el proceso es el mismo que el explicado en el párrafo anterior, la diferencia radica en que esta vez se compara contra los vigilantes que tienen igual o más de los periodos $\boldsymbol{n^{ideal}}$. Como ejemplo en la **Figura 41** se tiene que los vigilantes $\boldsymbol{v_4}$ y $\boldsymbol{v_7}$ son los únicos que no cumplen con él $\boldsymbol{n^{ideal}}$de periodos, y los vigilantes $v_3, v_5, v_6, v_8, v_9, v_{10}, v_{12}, v_{13}, v_{14}$tienen periodos disponibles para asignar. Dada la información se escoge aleatoriamente el vigilante $\boldsymbol{v_4}$, para el ejemplo solo se está mostrando una pequeña cantidad de los turnos en el que el vigilante pueda estar. Para el caso solo se cuenta con el turno $\boldsymbol{t_{2,s_2}}$ y en este turno no hay ningún vigilante ($v_3, v_5, v_6, v_8, v_9, v_{10}, v_{12}, v_{13}, v_{14}$) que pueda ser asignado en el sitio, de manera que se procede a seleccionar el vigilante $\boldsymbol{v_7}$ e intentar nuevamente con sus turnos, en los que tampoco es posible asignar algún vigilante. Como no hay más vigilantes a revisar el proceso finaliza aquí, pero en caso de que se hubiera encontrado algún vigilante se hubiera transferido el turno al vigilante disponible.

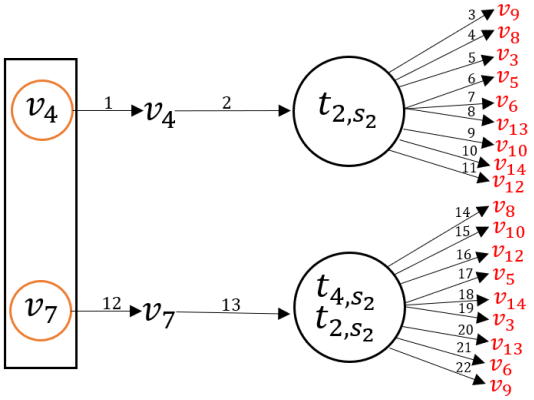


***Figura 41.** Refinamiento cantidad de vigilantes asignados MGRASP (criterio 2).*

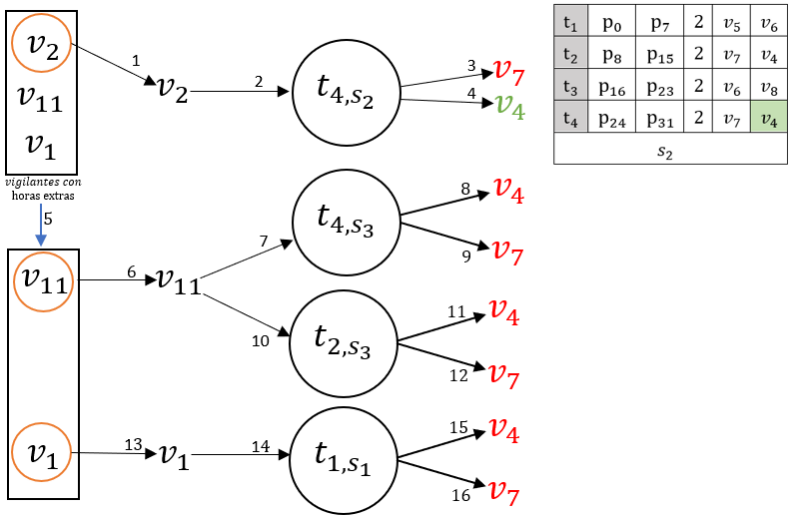
5.2.3 REFINAMIENTO POR HORAS EXTRAS ASIGNADAS

En este refinamiento se toma en cuenta los siguientes procesos para cada uno de los vigilantes que tienen horas extras asignadas:

1. Transferir un turno del vigilante con horas extras con aquellos vigilantes que tengan asignados entre 1 y $n^{ideal} - 1$ periodos.

2. Transferir un turno del vigilante con horas extras con aquellos vigilantes que no han sido asignados en ningún momento.

Usando el ejemplo de la Figura 40 se explicará el primer criterio que consiste en transferir algún turno de los vigilantes con horas extras con otro vigilante que tenga asignado menos de los periodos $\boldsymbol{n^{ideal}}$. Para el ejemplo en la **Figura 42** se tiene que los vigilantes v_1, v_2, v_{11} tienen asignados horas extras y los vigilantes v_4, v_7 no cumplen con la cantidad de periodos ideales. El primer paso consiste en tomar aleatoriamente uno de los 3 vigilantes con horas extras y revisar cuál de los vigilantes con menos periodos pueden ser asignados en algún turno del vigilante escogido. Suponiendo que el vigilante $\boldsymbol{v_2}$ fue escogido aleatoriamente, el vigilante tiene asignado entonces los turno $\boldsymbol{t_{2,s_1}, t_{4,s_2}}$, se escoge aleatoriamente el turno $\boldsymbol{t_{4,s_2}}$, se valida si el turno puede ser asignado entre el vigilante $\boldsymbol{v_4}$ o $\boldsymbol{v_7}$ se toma el vigilante $\boldsymbol{v_7}$ aleatoriamente, en este caso el vigilante no puede ser asignado por la restricción de periodos de descanso así que se valida con el vigilante $\boldsymbol{v_4}$ si puede ser asignado, se transfiere entonces el turno $\boldsymbol{t_{4,s_2}}$ al vigilante $\boldsymbol{v_4}$. Aquí ocurre que al transferir el turno del vigilante $\boldsymbol{v_2}$, el ya no tiene horas extras, si las tuviera se seguiría revisando en sus turnos en caso de que los tuviera. Como los vigilantes v_1, v_{11} aun cuentan con horas extras, se escoge aleatoriamente el vigilante $\boldsymbol{v_{11}}$ y se realiza nuevamente el proceso anteriormente mencionado. El vigilante $\boldsymbol{v_{11}}$ no puede ser asignando ni en el turno $\boldsymbol{t_{4,s_3}, t_{2,s_3}}$, así que se procede a escoger el vigilante $\boldsymbol{v_1}$, se toma el único turno en el que esta asignado $\boldsymbol{t_{1,s_1}}$ y se valida nuevamente si puede ser transferido contra v_4, v_7, caso en que tampoco se puede realizar. Como no hay más vigilantes con horas extras el proceso finaliza.



t_1	p_0	p_7	2	v_5	v_6
t_2	p_8	p_{15}	2	v_7	v_4
t_3	p_{16}	p_{23}	2	v_6	v_8
t_4	p_{24}	p_{31}	2	v_7	v_4
s_2					

Figura 42. *Refinamiento horas extras MGRASP (criterio 1).*

El criterio 2 consta de transferir los turnos del vigilante con horas extras contra aquellos vigilantes que no han sido asignados en ningún momento, es un caso que no se suele presentar mucho porque normalmente todos los vigilantes son usados, pero es posible. Continuando con el ejemplo en la **Figura 43** se muestra el estado actual de los turnos en los sitios, y como en el ejemplo no hay un vigilante con ningún periodo, se agrega uno para la demostración.

t_1	p_0	p_7	1	v_1
t_2	p_8	p_{15}	1	v_2
t_3	p_{16}	p_{23}	1	v_3
t_4	p_{24}	p_{31}	1	v_{12}
s_1				

t_1	p_0	p_7	2	v_5	v_6
t_2	p_8	p_{15}	2	v_7	v_4
t_3	p_{16}	p_{23}	2	v_6	v_8
t_4	p_{24}	p_{31}	2	v_7	v_4
s_2					

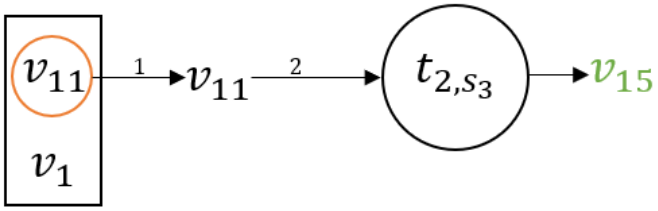
t_1	p_0	p_7	1	v_{12}
t_2	p_8	p_{15}	1	v_{13}
t_3	p_{16}	p_{23}	1	v_{14}
t_4	p_{24}	p_{31}	1	v_{13}
s_4				

t_1	p_0	p_7	2	v_9	v_3
t_2	p_8	p_{15}	2	v_{10}	v_{11}
t_3	p_{16}	p_{23}	2	v_5	v_9
t_4	p_{24}	p_{31}	2	v_{10}	v_{11}
s_3					

Vigilante	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}	v_{14}	v_{15}
Periodos asignados en la semana	56	48	48	32	48	48	40	48	48	48	56	48	48	48	0

Figura 43. *Estado actual de los turnos para el ejemplo MGRASP.*

En la **Figura 43** se tienen los vigilantes v_1, v_{11} con horas extras, y únicamente el vigilante $\boldsymbol{v_{15}}$ no ha sido asignado en ningún turno. Se toma aleatoriamente el vigilante $\boldsymbol{v_{11}}$ y de sus turnos $\boldsymbol{t_{2,s_3}}$,$\boldsymbol{t_{4,s_4}}$ se escoge $\boldsymbol{t_{2,s_3}}$ y se valida si $\boldsymbol{v_{15}}$ puede ser asignado en el turno (Si hubiera más vigilantes con cero periodos se escogería aleatoriamente). Finalmente, el turno $\boldsymbol{t_{2,s_3}}$ del vigilante $\boldsymbol{v_{11}}$ se transfiere al vigilante $\boldsymbol{v_{15}}$ y finaliza el proceso (ver **Figura 44**). Si hubiera otro vigilante que tenga cero periodos asignados, este criterio no se seguiría revisando con respecto a los otros vigilantes con horas extras porque se quiere evitar perjudicar el objetivo de minimizar los vigilantes utilizados, si se aplicara a muchos vigilantes estos podrían ser asignados solamente con un turno, la idea es que el criterio uno se encargue en mejorar las horas extras asignadas y solamente cuando no hay más posibilidades se use el aquí explicado (si a un vigilante sin periodos se le asigna un turno pasa automáticamente a hacer parte de los vigilantes 1 y $n^{ideal} - 1$ periodos).



t_1	p_0	p_7	2	v_9	v_3
t_2	p_8	p_{15}	2	v_{10}	v_{15}
t_3	p_{16}	p_{23}	2	v_5	v_9
t_4	p_{24}	p_{31}	2	v_{10}	v_{11}
s_3					

Figura 44. *Refinamiento horas extras MGRASP (criterio 2).*

5.2.4 REFINAMIENTO DE LA DISTANCIA RECORRIDA POR LOS VIGILANTES

En este refinamiento se toma en cuenta los siguientes procesos para cada uno de los vigilantes que quedaron asignados en un sitio diferente al más cercano o al por defecto:

1. Caso en el que el vigilante solo este asignado a un sitio.
2. Caso en el que el vigilante este asignado en múltiples sitios.

Para mejorar el caso 1 (**vigilante asignado únicamente en un sitio**) lo que se busca es intercambiar todos los turnos del vigilante con respecto a otro vigilante que se encuentre asignado solo a un sitio más cercano. A manera de ejemplo se tiene la **Figura 45** que representa la distancia que tiene un vigilante v hacia un sitio s y los turnos en que cada vigilante tiene asignado.

	s_1	s_2	s_3	s_4
v_1	2	1	3	0
v_2	0	3	2	1
v_3	0	3	1	2
v_4	2	3	1	0
v_5	3	2	0	1

Distancia del hogar de los vigilantes a los puestos de trabajo

v_1	t_{1,s_1}	t_{2,s_1}	t_{3,s_1}	t_{4,s_1}
v_2	t_{6,s_1}	t_{7,s_2}	t_{4,s_3}	t_{5,s_3}
v_3	t_{1,s_2}	t_{2,s_2}	t_{3,s_2}	t_{4,s_2}
v_4	t_{1,s_3}	t_{2,s_3}	t_{3,s_3}	t_{5,s_4}
v_5	t_{1,s_4}	t_{2,s_4}	t_{3,s_4}	t_{4,s_4}

Turnos asignados de los vigilantes

Figura 45. *Refinamiento de distancia MGRASP (datos de ejemplo).*

Para este caso lo primero es escoger los vigilantes que se encuentran asignados únicamente a un sitio (diferente al más cercano). Un vigilante se encuentra asignado únicamente a un sitio cuando todos sus turnos pertenecen al mismo sitio. En la **Figura 45** v_1, v_3, v_5 cumplen este criterio. Se escoge aleatoriamente el vigilante $\boldsymbol{v_3}$ que se encuentra asignado únicamente a $\boldsymbol{s_2}$, la distancia que tiene el vigilante $\boldsymbol{v_3}$ contra $\boldsymbol{s_2}$ es la peor posible, la idea es intercambiar todos sus turnos contra el vigilante que más le favorezca para disminuir la distancia (en la **Figura 46** se muestran los pasos de este ejemplo y el color representa la distancia del vigilante al sitio, entre más claro la distancia es menor). Se escoge $\boldsymbol{v_5}$ aleatoriamente y se valida si la distancia hacia $\boldsymbol{s_4}$ es menor que $\boldsymbol{s_2}$, como es menor se tiene que v_5 es la mejor opción para intercambiar. Se sigue con $\boldsymbol{v_1}$ y se valida si la distancia al sitio $\boldsymbol{s_1}$ es menor a la de $\boldsymbol{s_4}$, como es menor ahora $\boldsymbol{v_1}$ pasa a ser la mejor opción por intercambiar. Debido a que no hay más vigilantes con los cuales comparar, se hace el cambio de los turnos de $\boldsymbol{v_3}$ con $\boldsymbol{v_1}$ y se repite el proceso nuevamente. Se tiene ahora que v_1, v_5 aun no se encuentran en el sitio más cercano, se escoge aleatoriamente $\boldsymbol{v_5}$ para mejorar y se valida si la distancia contra el sitio $\boldsymbol{s_2}$ es menor (se compara contra $\boldsymbol{v_1}$ porque no hay más vigilantes); en este caso, aunque para el vigilante $\boldsymbol{v_1}$ la distancia al sitio s_4 es la mejor, para el vigilante $\boldsymbol{v_5}$ no lo es, por lo que no se efectúa el cambio. Dado que no hay más vigilantes con cual comparar se termina el proceso.

Vigilante y sitio escogido	Distancia al sitio actual	Sitio asignado del vigilante a intercambiar	Distancia del vigilante al sitio a intercambiar	Vigilante a intercambiar
v_3 s_2	3	s_4	2	v_5
		s_1	0	v_1
v_5 s_4	1	s_2	2	v_1

$\boldsymbol{v_1}$	t_{1,s_2}	t_{2,s_2}	t_{3,s_2}	t_{4,s_2}
$\boldsymbol{v_2}$	t_{6,s_1}	t_{7,s_2}	t_{4,s_3}	t_{5,s_3}
$\boldsymbol{v_3}$	t_{1,s_1}	t_{2,s_1}	t_{3,s_1}	t_{4,s_1}
$\boldsymbol{v_4}$	t_{1,s_3}	t_{2,s_3}	t_{3,s_3}	t_{5,s_4}
$\boldsymbol{v_5}$	t_{1,s_4}	t_{2,s_4}	t_{3,s_4}	t_{4,s_4}
Turnos asignados de los vigilantes				

Figura 46. *Refinamiento de distancia MGRASP (v. asignados a un sitio).*

Para el refinamiento del caso 2 (**Vigilante asignado en múltiples sitios**) se tienen dos criterios, uno en el que se hace el cambio total de los turnos y en otro que se cambian turnos específicos. Solo se aplica uno u otro dependiendo de la cantidad de turnos a cambiar.

Criterio 1 - el vigilante tiene la mayoría de sus turnos asignados a un sitio: Aquí se intercambian todos los turnos del vigilante asignado a múltiples sitios con otro vigilante que únicamente se encuentre asignado a un sitio más cercano. Continuando con el ejemplo de la **Figura 46** se tiene entonces que $\boldsymbol{v_4}$ tiene más de la mitad de los turnos asignados a $\boldsymbol{s_3}$ y un turno asignado a $\boldsymbol{s_4}$. De los vigilantes a intercambiar se tiene que v_1, v_5 están asignados únicamente a un sitio, $\boldsymbol{v_3}$ no se cuenta porque ya se encuentra asignado en su sitio esperado. En los pasos a seguir (ver **Figura 47**) como solo $\boldsymbol{v_4}$ cumple con tener la mayoría de los turnos asignados a un sitio será el escogido a mejorar (si hubiera más vigilantes se escogería aleatoriamente). De los vigilantes asignado únicamente a un sitio se escoge $\boldsymbol{v_1}$ y se valida si la distancia de $\boldsymbol{v_4}$ al sitio $\boldsymbol{s_2}$ es menor a $\boldsymbol{s_3}$ (aquí no importa la aleatoriedad dado que se compara con todos los vigilantes), como no lo es se continua con $\boldsymbol{v_5}$, y se valida la distancia. Dado que la distancia para $\boldsymbol{s_4}$ es menor y no hay más vigilantes asignados a un sitio se hace el intercambio con el vigilante. Si hubiera otra vigilante la comparación de la distancia se haría contra el ultimo sitio con la distancia menor y no con el sitio inicial (se compararía con la distancia de $\boldsymbol{s_4}$ y no de $\boldsymbol{s_3}$). Si hubiera otro vigilante con las características iniciales de $\boldsymbol{v_4}$ los pasos se repetirían para el vigilante.

Es posible que en otro escenario el intercambió no se pueda realizar debido a que no hay más vigilantes asignados únicamente a un sitio, o porque la distancia contra esos sitios es mayor con respecto al que se encuentra asignado. Cuando esto sucede se realiza el intercambio de los turnos solitariamente. En la **Figura 47** se tiene que $\boldsymbol{v_5}$ es el único vigilante que cumple con tener asignados la mayoría de sus turnos a un sitio y que $\boldsymbol{v_2}$ es el único que está asignado a múltiples sitios. En los pasos a seguir (**Figura 48**) como solo $\boldsymbol{v_5}$ cumple con tener la mayoría de los turnos asignados a un sitio será el escogido a mejorar (si hubiera más vigilantes se

escogería aleatoriamente). La idea es intercambiar todos los turnos que por una u otra razón no hacen parte de la mayoría. De $\boldsymbol{v_5}$ se tiene que $\boldsymbol{t_{5,s_4}}$ es el único turno que no pertenece al sitio ($\boldsymbol{s_3}$), si hubiera otro turno que no corresponde se aplicaría el mismo proceso que se hará continuación. Del turno seleccionado $\boldsymbol{t_{5,s_4}}$ se empieza a comparar en cual turno de los vigilantes asignados a múltiples sitios se puede hacer el intercambio (se escoge un vigilante aleatoriamente). Cabe resaltar que ambos vigilantes deben poder ser asignados en el turno del otro. Como $\boldsymbol{v_2}$ es el único vigilante con múltiples sitios se escogen sus turnos para validar la distancia. Se valida la distancia contra el turno $\boldsymbol{t_{6,s_1}}$ pero la distancia no es menor por lo que no se toma en cuenta, lo mismo para $\boldsymbol{t_{7,s_2}}$, en $\boldsymbol{t_{4,s_3}}$ la distancia ya es menor, pero $\boldsymbol{v_2}$ ya se encuentra asignado en los periodos de $\boldsymbol{t_{5,s_4}}$, así que se continua con $\boldsymbol{t_{5,s_3}}$ y en este caso la distancia es menor y los vigilantes se pueden intercambiar por lo que hasta el momento se tendrá en cuenta este turno para intercambiar, y la distancia a comparar será ahora a la del sitio $\boldsymbol{s_3}$. Como no hay más turnos con los cuales comparar se toma el ultimo vigilante asignado al turno más cercano ($\boldsymbol{t_{5,s_3}}$) y se realiza el intercambio y finaliza el proceso.

Vigilante y sitio escogido	Distancia al sitio actual	Sitio asignado del vigilante a intercambiar	Distancia del vigilante al sitio a intercambiar	Vigilante a intercambiar
v_4 s_3	1	S_2	3	v_1
		S_4	0	v_5

$\boldsymbol{v_1}$	t_{1,s_2}	t_{2,s_2}	t_{3,s_2}	t_{4,s_2}
$\boldsymbol{v_2}$	t_{6,s_1}	t_{7,s_2}	t_{4,s_3}	t_{5,s_3}
$\boldsymbol{v_3}$	t_{1,s_1}	t_{2,s_1}	t_{3,s_1}	t_{4,s_1}
$\boldsymbol{v_4}$	t_{1,s_4}	t_{2,s_4}	t_{3,s_4}	t_{4,s_4}
$\boldsymbol{v_5}$	t_{1,s_3}	t_{2,s_3}	t_{3,s_3}	t_{5,s_4}
Turnos asignados de los vigilantes				

Figura 47. *Refinamiento de distancia MGRASP (Caso A).*

Vigilante y turno escogido	Distancia al turno actual	Turno por intercambiar	Distancia del vigilante al sitio a intercambiar	Vigilante a intercambiar
v_5 t_{5,s_4}	1	t_{6,s_1}	3	v_2
		t_{7,s_2}	2	
		t_{5,s_3}	0	

$\boldsymbol{v_1}$	t_{1,s_2}	t_{2,s_2}	t_{3,s_2}	t_{4,s_2}
$\boldsymbol{v_2}$	t_{6,s_1}	t_{7,s_2}	t_{4,s_3}	t_{5,s_4}
$\boldsymbol{v_3}$	t_{1,s_1}	t_{2,s_1}	t_{3,s_1}	t_{4,s_1}
$\boldsymbol{v_4}$	t_{1,s_4}	t_{2,s_4}	t_{3,s_4}	t_{4,s_4}
$\boldsymbol{v_5}$	t_{1,s_3}	t_{2,s_3}	t_{3,s_3}	t_{5,s_3}
Turnos asignados de los vigilantes				

Figura 48. *Refinamiento de distancia MGRASP (Caso B).*

Criterio 2 – Intercambio de turnos: La idea del anterior caso es ayudar a que un vigilante se mantenga asignado a un único sitio y así evitar que por cubrir los turnos quede asignado a sitios diferentes, pero no siempre esto es posible de lograr, para estos casos lo que se busca entonces es al menos disminuir la distancia a los sitios. Para el ejemplo (**Figura 48**) se tiene que v_2 es el único con turnos en múltiples sitios, se agregara uno nuevo con sus características para la demostración tal como se muestras en la **Figura 49**.

	s_1	s_2	s_3	s_4
v_1	2	1	3	0
v_2	0	3	2	1
v_3	0	3	1	2
v_4	2	3	1	0
v_5	3	2	0	1
v_6	0	2	1	3

Distancia del hogar de los vigilantes a los puestos de trabajo

v_1	t_{1,s_2}	t_{2,s_2}	t_{3,s_2}	t_{4,s_2}
v_2	t_{6,s_1}	t_{7,s_2}	t_{4,s_3}	t_{5,s_4}
v_3	t_{1,s_1}	t_{2,s_1}	t_{3,s_1}	t_{4,s_1}
v_4	t_{1,s_4}	t_{2,s_4}	t_{3,s_4}	t_{4,s_4}
v_5	t_{1,s_3}	t_{2,s_3}	t_{3,s_3}	t_{5,s_3}
v_6	t_{6,s_2}	t_{1,s_1}	t_{4,s_2}	t_{5,s_3}

Turnos asignados de los vigilantes

Figura 49. *Refinamiento de distancia MGRASP (datos para intercambio turnos).*

Tanto v_2 y v_6 ahora son candidatos para hacer el intercambio dado que se encuentran asignados en múltiples sitios, el proceso consiste en intentar intercambiar turnos del vigilante seleccionado contra los turnos de otro vigilante asignado a múltiples sitios. Se toma aleatoriamente un vigilante en este caso v_6 y para cada turno en el que este asignado realizamos los siguientes pasos (a excepción de los turnos en que este asignado a su sitio más cercano). Primero seleccionamos un vigilante aleatoriamente con múltiples sitios (para el caso solo quedaría v_2) y se comienza a comparar la distancia contra cada uno de sus turnos siempre que ambos vigilantes estén disponibles para ser intercambiados. Se toma entonces el turno t_{6,s_2} y se compara contra $t_{6,s_1}, t_{7,s_2}, t_{4,s_3}, t_{5,s_4}$, dando como resultado que t_{6,s_1} tiene el sitio más cercano por lo que se intercambia el turno. Como se mencionó el proceso se repite para los demás turnos (como solo se tiene a v_2 la comparación en este ejemplo siempre será contra él), de t_{1,s_1} se tiene que ya está asignado en el sitio más cercano por lo que se omite, t_{2,s_4} se compara contra t_{7,s_2}, t_{4,s_3}, t_{5,s_4}, se verifica que en t_{4,s_3} la distancia es menor y se realiza el intercambio. Lo mismo se hace para t_{5,s_3} pero ningún turno tiene menor distancia. Como v_2 no tiene más turnos se finaliza el proceso (ver **Figura 50**).

Vigilante y turno escogido	Distancia al turno actual	Turno por intercambiar	Distancia del vigilante al sitio a intercambiar	Vigilante a intercambiar
v_6 t_{6,s_2}	2	t_{6,s_1}	0	v_2
		t_{7,s_2}	2	
		t_{4,s_3}	1	
		t_{5,s_4}	3	

v_6 t_{2,s_4}	3	t_{7,s_2}	2	v_2
		t_{4,s_3}	1	
		t_{5,s_4}	3	
v_6 t_{5,s_3}	1	t_{7,s_2}	2	v_2
		t_{2,s_4}	3	
		t_{5,s_4}	3	

$\boldsymbol{v_1}$	t_{1,s_2}	t_{2,s_2}	t_{3,s_2}	t_{4,s_2}
$\boldsymbol{v_2}$	t_{6,s_2}	t_{7,s_2}	t_{4,s_2}	t_{5,s_4}
$\boldsymbol{v_3}$	t_{1,s_1}	t_{2,s_1}	t_{3,s_1}	t_{4,s_1}
$\boldsymbol{v_4}$	t_{1,s_4}	t_{2,s_4}	t_{3,s_4}	t_{4,s_4}
$\boldsymbol{v_5}$	t_{1,s_3}	t_{2,s_3}	t_{3,s_3}	t_{5,s_3}
$\boldsymbol{v_6}$	t_{6,s_1}	t_{1,s_1}	t_{4,s_3}	t_{5,s_3}
Turnos asignados de los vigilantes				

***Figura 50**. Refinamiento de distancia MGRASP (intercambio de turnos).*

CAPÍTULO 6

6 MÉTRICAS Y AFINAMIENTO DE PARÁMETROS

6.1 MÉTRICAS

Cualquier algoritmo de optimización necesita medir su rendimiento. En escenarios multiobjetivo no se puede calcular la distancia a un "verdadero" óptimo, pero se puede considerar un conjunto de soluciones que se comportan mejor en uno u otro objetivo. En algunos casos ni siquiera se conoce el valor óptimo y se deben utilizar otras técnicas. Para este proyecto en particular se utilizaron dos métricas, el Hipervolumen y la Distancia Generacional Invertida (Inverted Generational Distance, IGD) las cuales han sido muy usadas en este tipo de problemas.

El indicador de hipervolumen introducido por Zitzler y Thiele [23], calcula el área/volumen que se encuentra dominado por un conjunto proporcionado de soluciones con respecto a un punto de referencia.

La **Figura 51** ilustra un ejemplo de dos objetivos (eje x y y) donde el área dominada por un conjunto de puntos se muestra en gris (se está minimizando). Los algoritmos buscan tener un valor alto de hipervolumen en la medida que se encuentran mejores soluciones que minimizan los objetivos. Para el problema de programación de personal de seguridad y vigilancia se tomó como referencia el punto $(1, 1, 1, 1)$ debido a que se está buscando minimizar todos los objetivos.

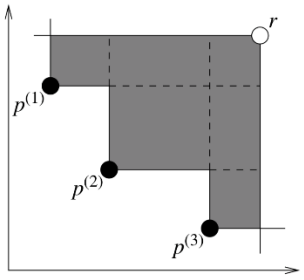


***Figura 51**. Hipervolumen.*

Entre otros indicadores, la distancia generacional invertida (IGD) es uno de los indicadores más frecuentemente utilizados para la evaluación del rendimiento de los algoritmos de optimización multiobjetivo [24]. IGD es la distancia promedio de cada punto de referencia a su solución más cercana del Frente de Pareto entregado por el algoritmo. Cuando se utiliza un conjunto de puntos de referencia grande en cantidad y bien distribuidos en todo el Frente de Pareto un valor pequeño del indicador IGD sugiere una buena convergencia de las soluciones reportadas por el

algoritmo y su distribución en todo el frente de Pareto (ver **Figura 52**). Para el problema de la presente investigación, la generación de los puntos de referencia se basó en tomar todos los puntos sobre los planos de origen de los ejes de coordenadas en un espacio de 4 dimensiones ya que se busca minimizar 4 objetivos. Dado que representar estos puntos de manera grafica es un trabajo complejo se muestra un ejemplo para un plano bidimensional donde se cubren los siguientes puntos de referencia: $(0,\ 0)$ $(0.1,\ 0)$ ... $(1,\ 0)$ $(0,\ 0.1)$ ... $(0{,}1)$ que corresponden en la figura a los puntos de color celeste.

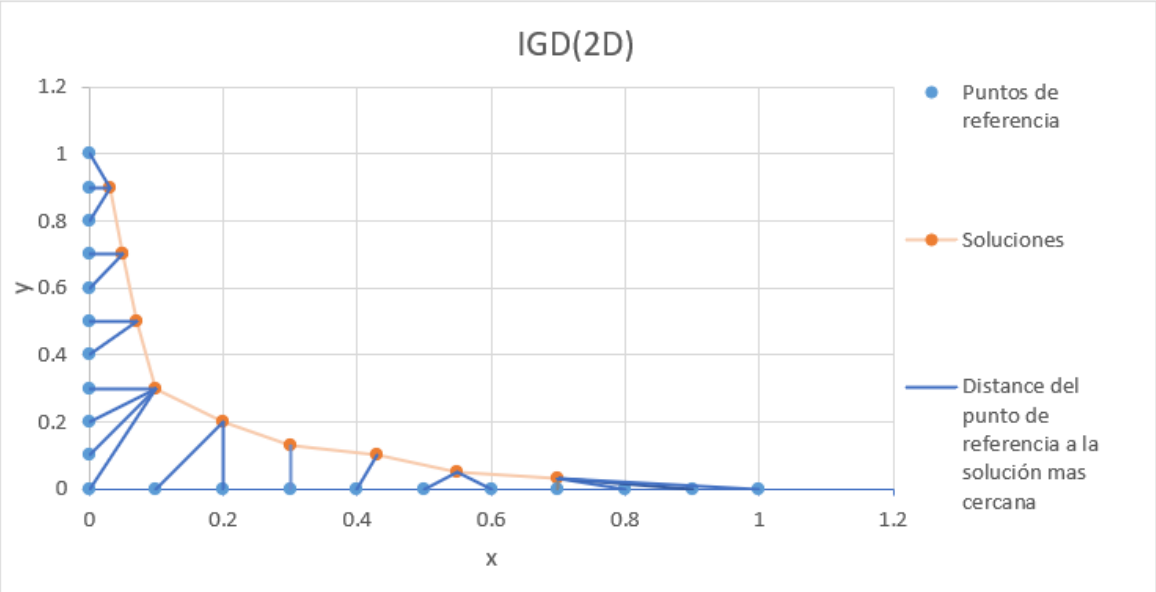


***Figura 52**. Distancia generacional invertida.*

6.2 AFINAMIENTO DE PARÁMETROS

Metaheurísticas como GRASP Y NSGA-II utilizan diferentes parámetros y estos deben configurarse apropiadamente para mejorar la manera en la que se construyen y se evolucionan las soluciones de un problema. Algunos de estos parámetros pueden ser por ejemplo el tamaño de la lista restringida que usa GRASP, el número de repeticiones para la búsqueda local, el número de hijos que debería tener una solución en NSGA II, el tamaño de la población que se debe manejar para un problema, entre otras. No se conocen reglas estrictas o determinadas para elegir los valores apropiados de los parámetros en las metaheurísticas, por lo que obtener un conjunto optimo o cercano al óptimo de los valores de los parámetros para un caso en concreto no sirve para generalizar todos los casos. Identificar los valores adecuados de los parámetros es frecuentemente manejado como un problema de prueba y error. Evidentemente repetir este proceso de prueba y error es tedioso y consume una cantidad considerable de tiempo, además, afinar los parámetros de a uno en uno no necesariamente permite obtener el mejor conjunto de parámetros, dado que los parámetros no son independientes entre sí, y muy por el contario pueden interactuar de manera compleja.

Existen diferentes enfoques para el afinamiento de parámetros [31], entre ellos, la meta optimización (una metaheurística optimiza los parámetros de la metaheurística que resuelve el problema concreto), el diseño de experimentos o DOE (con enfoque factorial, factorial completo, basado en arreglos de cobertura y acoplado con el análisis de regresión, el análisis de varianza o técnicas de aprendizaje de máquina) y la configuración de algoritmos sin modelo (basadas en pruebas de hipótesis estadísticas) [32].

El enfoque de meta optimización ha mostrado buenos resultados en diferentes aplicaciones y contextos y también se conoce con el nombre de meta-level genetic algorithm approach cuando la metaheurística que afina los parámetros es un algoritmo genético [25]. Este enfoque, sin embargo, implica un alto costo computacional.

En esta investigación se utilizó un algoritmo genético (GA) para optimizar los parámetros tanto de MGRASP como de NSGA-II, con diferentes data sets (problemas concretos de programación de personal de seguridad y vigilancia) para identificar el conjunto de parámetros que más se adaptan a los diferentes casos.

GA es una metaheurística basada en población inspirada en la evolución de las especies propuesta por Darwin que busca soluciones optimas o casi optimas a un problema. GA toma conceptos de la evolución como la reproducción y supervivencia, donde el más apto permanece y los demás son eliminados [33]. El proceso básico de este algoritmo se presenta en la **Figura 53** que empieza con una población y aquellas soluciones más aptas (padres) se reproducen con el fin de generar la próxima población más apta.

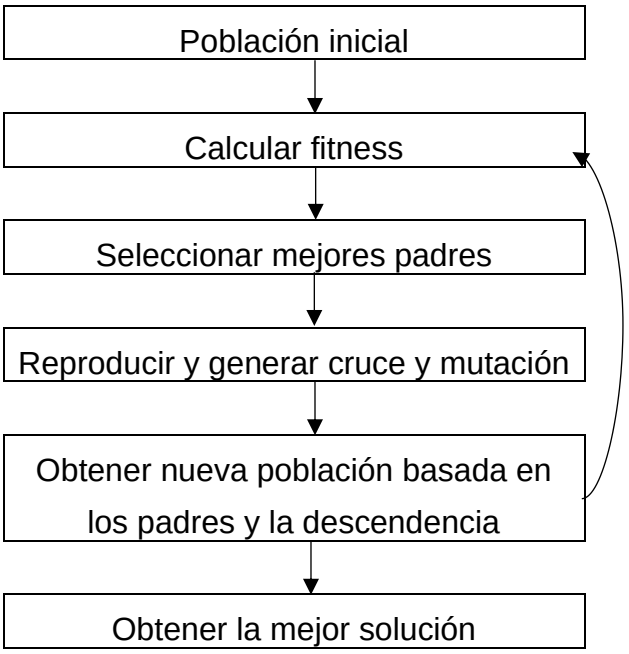


Figura 53. *Procedimiento del algoritmo genético (GA).*

Para la optimización de los parámetros la representación de la solución se basa en la unión de cada uno de los parámetros (genes) que se desean optimizar, siendo

p_n un parámetro (gen) de la metaheurística a optimizar como se muestra en la **Figura 54.**

$p_1 MGRASP$	$p_2 MGRASP$	...	$p_n GMGRASP$

$p_1 NSGA-II$	$p_1 NSGA-II$	...	$p_1 NSGA-II$

Figura 54. *Representación del individuo GA.*

El **Algoritmo 12** se usa para calcular la función de fitness (calidad) de cada solución (cromosoma) de GA. Para este cálculo se ejecuta la metaheurística (MGRASP o NSGA-II) con los parámetros de la solución una cantidad $\mathbf{R}$ *de* veces (se recomienda que sea más de 30 veces para que se cumpla el teorema del límite central y así el valor medio se comporte con una distribución normal) con diferentes semillas de valores aleatorios en un conjunto de problemas de programación de personal de seguridad y vigilancia. En cada una de estas repeticiones se calcula la métrica de hipervolumen ($\mathbf{HV}$) para medir la calidad del Frente de Pareto. Ejecutar la metaheurística $\mathbf{R}$ veces permite obtener el promedio de todos los $\mathbf{HV}$ para un problema; y luego el promedio de los hipervolúmenes promedio de todos los problemas será el resultado que se retornará. Este proceso, aunque costoso computacionalmente permite tener una visión más completa del comportamiento de la metaheurística con los parámetros.

Entradas: Sol: Soluciones (soluciones con los parámetros)
Pros: conjunto de problemas a resolver
M: Metaheurística (MGRASP o NSGA II)
R: Número de repeticiones de la metaheurística

Salida: fitness (resultados promedio del HV de cada solución)

```
1.  Inicio
2.      Fitness = [] // Lista con los resultados de calidad del HV para cada solución
3.      Para cada problema p en Pros haga:
4.          HVs = 0 // Suma de todos los hiper volúmenes en este problema
5.          Para la cantidad de repeticiones R del experimento haga:
6.              F = Ejecutar_metahuristica (M, Sol, p) // Se ejecuta la metaheurística
                    para resolver el problema p de programación de personal de
                    seguridad y vigilancia con los parámetros definidos en la solución
                    Sol y retorna el Frente de Pareto encontrado
7.              HV = Calcular_HV(F)
8.              HVs = HVs + HV
            Fin Para
9.          HVp = (HVs / R) // Se obtiene el promedio de los HVs
10.         Fitness = Fitness U HVp // Se agrega el hiper volumen promedio al fitness
11.     Fin Para
12.     Retornar Fitness / |Pro| // Se retorna el fitness promedio de todos los problemas
13. Fin
```

Algoritmo 12. *Calculo del fitness GA.*

Para empezar con el proceso de crucé y mutación primero se seleccionan las dos mejores soluciones obtenidas en el proceso anterior, las cuales serán los padres para generar la próxima población. El proceso de cruce consiste en tomar los parámetros (genes) de cada solución e intercambiarlos. En este caso se escoge el

punto medio sobre la cantidad de parámetros a optimizar y se intercambian los genes finales generando los nuevos hijos de la próxima población tal como se muestra en la **Figura 55** como ejemplo para la metaheurística MGRASP.

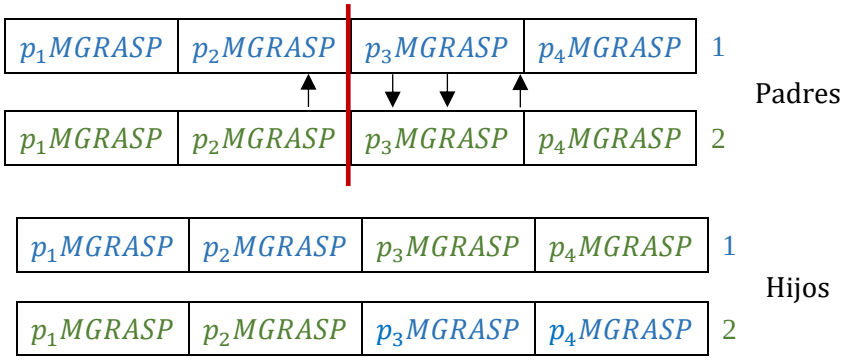


***Figura 55**. Cruce GA.*

Una vez se ha realizado el cruce, a cada uno de los hijos se les aplica el factor de mutación que consiste en modificar cada uno de los parámetros (genes), para este caso se escogió realizar 2 cambios sobre los parámetros de forma aleatoria como se muestra en la **Figura 56**.

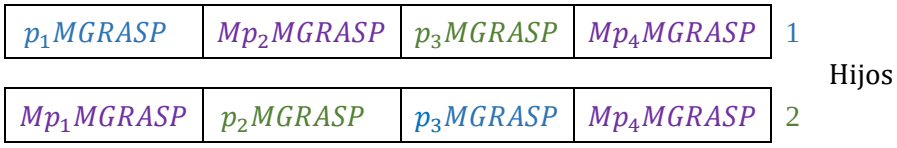


***Figura 56**. Mutación GA.*

Finalmente, se unen los padres con los hijos dando como resultado la población para la próxima generación.

$p_1MGRASP$	$p_2MGRASP$	$p_3MGRASP$	$p_4MGRASP$	s_1
$p_1MGRASP$	$p_2MGRASP$	$p_3MGRASP$	$p_4MGRASP$	s_2
$p_1MGRASP$	$Mp_2MGRASP$	$p_3MGRASP$	$Mp_4MGRASP$	s_3
$Mp_1MGRASP$	$p_2MGRASP$	$p_3MGRASP$	$Mp_4MGRASP$	s_4
⋮	⋮	⋮	⋮	⋮
$p_1MGRASP$	$Mp_2MGRASP$	$Mp_3MGRASP$	$p_4MGRASP$	s_n

***Figura 57**. Generación final GA.*

Para el afinamiento de parámetros se escogieron 3 problemas para el parámetro Pros del **Algoritmo 12**, 1 caso por cada dificultad, teniendo en cuenta los posibles horarios que una empresa de vigilancia pueda tener. En concreto los casos de prueba fueron ejecutados para un mes y se pueden observar en la **Tabla 7**.

La **Tabla 8** muestra los parámetros que se optimizaron para cada metaheurística.

Tabla 7. Casos de optimización.

Dificultad	N. de sitios	N. de vigilantes	N. de vigilantes esperados a utilizar	Horarios	N. de vigilantes requeridos en el horario	Tiempo Máximo de ejecución
Baja	2	15	14	10 p.m. - 6 a.m.	2	60s
				6 a.m. – 6.a.m		
Media	30	200	195	10 p.m. - 6 a.m.	2	1800s
				10 p.m. - 6 a.m.	4	
				6 a.m. – 6.a.m	3	
Alta	50	375	365	10 p.m. - 6 a.m.	2	3600s
				10 p.m. - 6 a.m.	4	
				6 a.m. – 6.a.m	3	
				6 a.m. – 6.a.m	4	

Tabla 8. *Parámetros de optimización.*

Metaheurística	p_1	p_2	p_3	p_4
MGRASP	Cantidad de componentes a crear	Tamaño de la lista restringida	Iteraciones de búsqueda local en las vecindades	Tamaño de la población
NSGA-II	Cantidad de hijos a generar	Tamaño de la lista restringida (%)	Intentos para obtención de un gen	Tamaño de la población

El algoritmo genético de optimización fue ejecutado con una población de 10 individuo y 3 generaciones. La razón de que fuera un número tan bajo de generaciones se debió al tiempo que toma realizar la unión de todas las ejecuciones por cada solución, ya que se realizan 30 ejecuciones por cada metaheurística, 10 ejecuciones por cada nivel de dificultad, dando como resultado un total de 900 ejecuciones permitiendo obtener un promedio de los parámetros en diferentes circunstancias. De la **Tabla 9** se puede apreciar que los resultados de una misma metaheurística no son muy diferentes, al igual que su diferencia no es tan significa entre las dos metaheurísticas.

De lo resultados obtenidos se identifica que para MGRASP la mejor solución encontrada fue la numero 4 de la evolución 3, representando respectivamente una cantidad de componentes a crear de 20, un tamaño de la lista restringida igual a 19, un número de iteraciones de búsqueda local de las vecindades de 18, y un tamaño de la población igual a 10.

Para NSGA-II la mejor solución encontrada fue la numero 8 de la evolución 3, representando respectivamente una cantidad de hijos a generar de 30, un tamaño de la lista restringida (%) de 1.8, un número de intentos para obtener un gen de 15 y un tamaño de la población igual a 10.

Cabe destacar que los parámetros encontrados fueron los utilizados en los experimentos que se presentan más adelante en el Capítulo 8.

***Tabla 9**. Resultados optimizaciones.*

MGRASP

E	Sol	p_1	p_2	p_3	p_4	Hv promedio
1	1	20	4	10	10	0.739
1	2	20	4	10	25	0.737
1	3	30	24	10	25	0.741
1	4	20	19	18	10	0.742
1	5	20	4	16	40	0.775
1	6	20	4	10	20	0.732
1	7	20	20	10	25	0.737
1	8	20	4	20	15	0.746
1	9	28	4	10	30	0.734
1	10	30	10	10	10	0.738
2	1	20	4	10	10	0.789
2	2	20	4	10	25	0.744
2	3	30	24	10	25	0.74
2	4	20	19	18	10	0.732
2	5	20	4	16	40	0.745
2	6	20	4	10	20	0.738
2	7	20	20	10	25	0.754
2	8	20	4	20	15	0.75
2	9	28	4	10	30	0.755
2	10	30	10	10	10	0.741
3	1	20	4	10	10	0.765
3	2	20	4	10	25	0.758
3	3	30	24	10	25	0.739
3	4	20	19	18	10	0.797
3	5	20	4	16	40	0.741
3	6	20	4	10	20	0.766
3	7	20	20	10	25	0.739
3	8	20	4	20	15	0.784
3	9	28	4	10	30	0.738
3	10	30	10	10	10	0.789

NSGA-II

E	Sol	p_1	p_2	p_3	p_4	Hv promedio
1	1	2	0.941	13	10	0.731
1	2	9	0.018	11	14	0.737
1	3	2	0.941	11	24	0.734
1	4	9	0.018	29	10	0.736
1	5	2	18.941	11	14	0.734
1	6	9	0.018	27	10	0.738
1	7	2	0.941	11	26	0.737
1	8	9	19.018	13	10	0.729
1	9	15	0.941	11	14	0.739
1	10	28	0.018	13	10	0.739
2	1	2	0.941	13	10	0.735
2	2	9	0.018	11	14	0.656
2	3	2	0.941	11	24	0.666
2	4	9	0.018	29	10	0.738
2	5	2	18.941	11	14	0.724
2	6	9	0.018	27	10	0.73
2	7	2	0.941	11	26	0.723
2	8	9	19.018	13	10	0.733
2	9	15	0.941	11	14	0.691
2	10	28	0.018	13	10	0.743
3	1	15	0.941	13	14	0.681
3	2	9	0.018	11	10	0.728
3	3	28	0.018	13	10	0.743
3	4	9	0.018	29	10	0.732
3	5	28	0.941	15	10	0.723
3	6	9	0.018	30	10	0.731
3	7	9	0.018	30	14	0.739
3	8	30	0.018	15	10	0.745
3	9	2	18.941	15	15	0.724
2	10	28	0.018	13	10	0.743

Esta página ha sido dejada intencionalmente en blanco.

CAPÍTULO 7

7 APLICACIÓN WEB

En esta sección se muestra el software desarrollado que soporta desde la carga de los datos del problema hasta la presentación de los resultados.

Un requerimiento esencial para que las metaheurísticas funcionen correctamente era contar con un data set correctamente estructurado y con datos que reflejen el problema que se quiere resolver. La información que maneja una empresa de seguridad y vigilancia puede ser sencilla o muy compleja, esto depende de la cantidad de clientes que la empresa maneje y de sus necesidades; mayor volumen de clientes (sitios, turnos, personal requerido, entre otros) vuelve el proceso de la definición de los datos más tedioso y repetitivo, dando como resultado posibles errores en su construcción. Por esta razón, la creación de una interfaz sencilla que ayude al usuario con el proceso de construcción de los datos para el uso del programa fue esencial.

Un requerimiento básico que se tuvo en cuenta fue el de definir una interfaz que cualquier tipo de usuario pudiera usar y adaptarse rápidamente. Esto porque cada empresa de seguridad maneja sus datos de diferente forma y crear una interfaz personalizada por cada empresa es costoso y hasta cierto punto inviable.

La interfaz diseñada se dividió en dos secciones, una para gestionar (crear, mostrar, modificar y borrar, CRUD) los datos relacionados con los diferentes sitios y horarios que la empresa debe vigilar (ver **Figura 58**) y la otra con la gestión de los datos de los vigilantes (ver **Figura 60**).

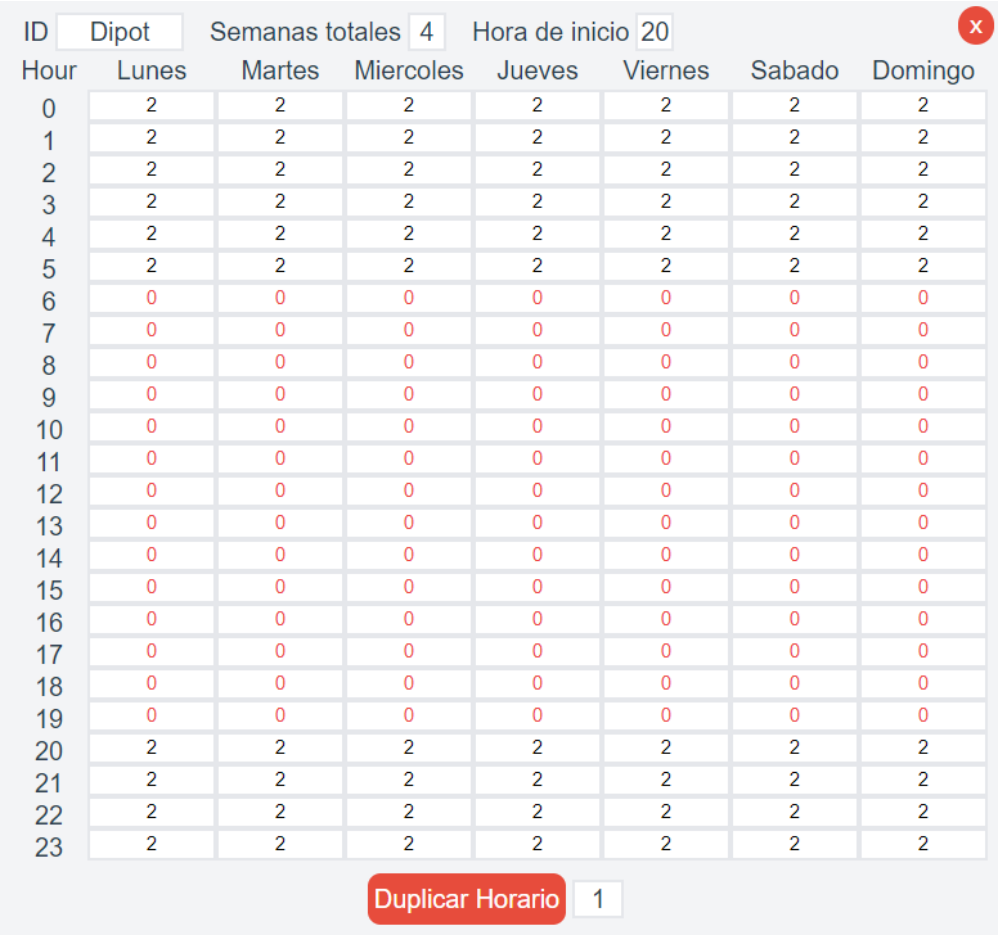
***Figura 58**. Gestión de sitios y horarios.*

Los datos necesarios para crear el horario que se debe vigilar en un sitio es:

- ID: Identificador del sitio.
- Numero de semanas: Numero de semanas que se desean cubrir para el sitio.

- Hora de inicio del horario: Inicio de hora para el primer día.
- Horario: Se refiere a el horario que se debe cubrir, está representado según un cronograma básico de lunes a domingo con bloques de 1 hora correspondientes a cada hora del día, en cada bloque se especifica el número de guardias que se necesitan para esa hora y en caso contrario de no requerir ninguno, un 0 representa que en esa hora no se necesita vigilancia.

Además, se incluye la opción de hacer una copia del horario del sitio, caso tal que se manejen los mismos datos en diferentes sitios, ayudando al manejo de datos repetidos y reduciendo así el tiempo de registro de datos de parte de los usuarios (ver **Figura 59**).



Hour	Lunes	Martes	Miercoles	Jueves	Viernes	Sabado	Domingo
0	2	2	2	2	2	2	2
1	2	2	2	2	2	2	2
2	2	2	2	2	2	2	2
3	2	2	2	2	2	2	2
4	2	2	2	2	2	2	2
5	2	2	2	2	2	2	2
6	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0
20	2	2	2	2	2	2	2
21	2	2	2	2	2	2	2
22	2	2	2	2	2	2	2
23	2	2	2	2	2	2	2

***Figura 59**. Opción para duplicar horarios de un sitio a otro.*

En la segunda sección se solicitan los datos de los vigilantes, tales como (ver **Figura 60)**:

- ID: Identificador del vigilante.
- Sitio por defecto a vigilar: Hace referencia al sitio que por una u otra razón el vigilante en particular debe ser asignado obligatoriamente (normalmente por solicitud expresa del cliente).
- Distancias: Hace referencia a la distancia en km que hay desde el hogar del vigilante a los diferentes sitios de vigilancia.

10 Agregar vigilantes

ID	Sitio por defecto a vigilar	dipot	Campos	Bosques	Kuter	La paz	Campo bello
106123345	dipot	23	12	32	23	12	2
164562867	dipot	34	34	43	1	2	3
2712345	Ninguno	75	23	23	453	2	90
2486789	Ninguno	234	5	6	12	86	1
103546545	Campos	34	45	43	32	12	75
2131232	Ninguno	45	214	86	75	78	76
106123345	Ninguno	12	33	34	41	23	45
106123345	Ninguno	67	86	45	3	2	4
106156765	La paz	36	23	52	8	7	7
10712345	Ninguno	68	3	6	41	45	1
108526345	Kuter	1	2	3	5	6	7

***Figura 60**. Gestión de vigilantes.*

Con los datos completamente registrados, el usuario puede generar los cronogramas, los cuales serán entregados una vez el programa finalice su ejecución. Es preciso comentar que la interfaz permite guardar la información en un archivo de Excel con el fin de realizar más pruebas o facilitar otros procesos que la empresa desarrollo. Este archivo (plantilla) de Excel con los datos de los vigilantes y los sitios también puede ser cargada a través de la interfaz de la aplicación, evitando crear los datos nuevamente.

Cuando el programa finalmente ha terminado de procesar los datos suministrados y ha finalizado con la generación de los cronogramas se procede a guardar toda la información del procesamiento desde las soluciones hasta los datos relevantes para el usuario y la empresa (ver **Figura 61**).

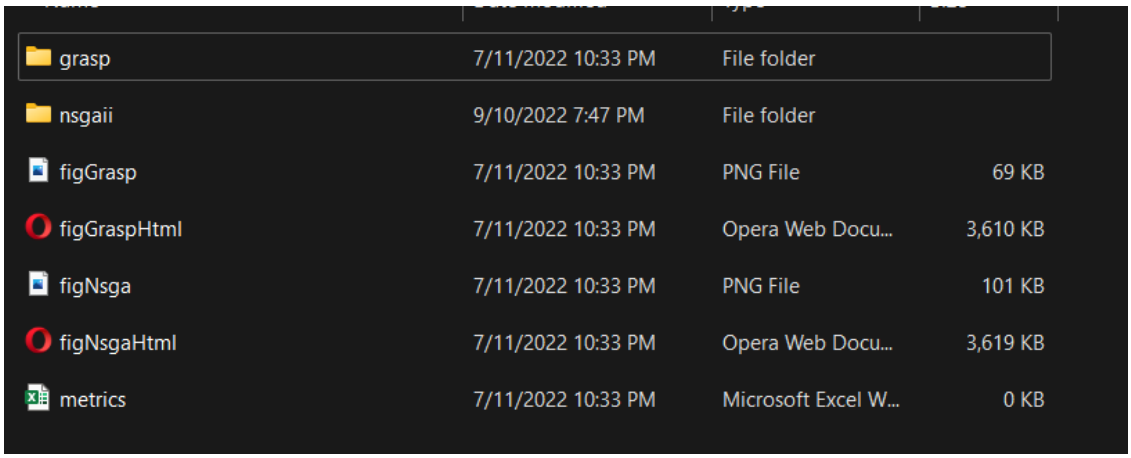


***Figura 61**. Estructura general de los resultados entregados.*

Tanto para MGRASP como para NSGA-II se genera una carpeta con la cantidad de las **N** soluciones que se hayan generado (ver **Figura 62**).

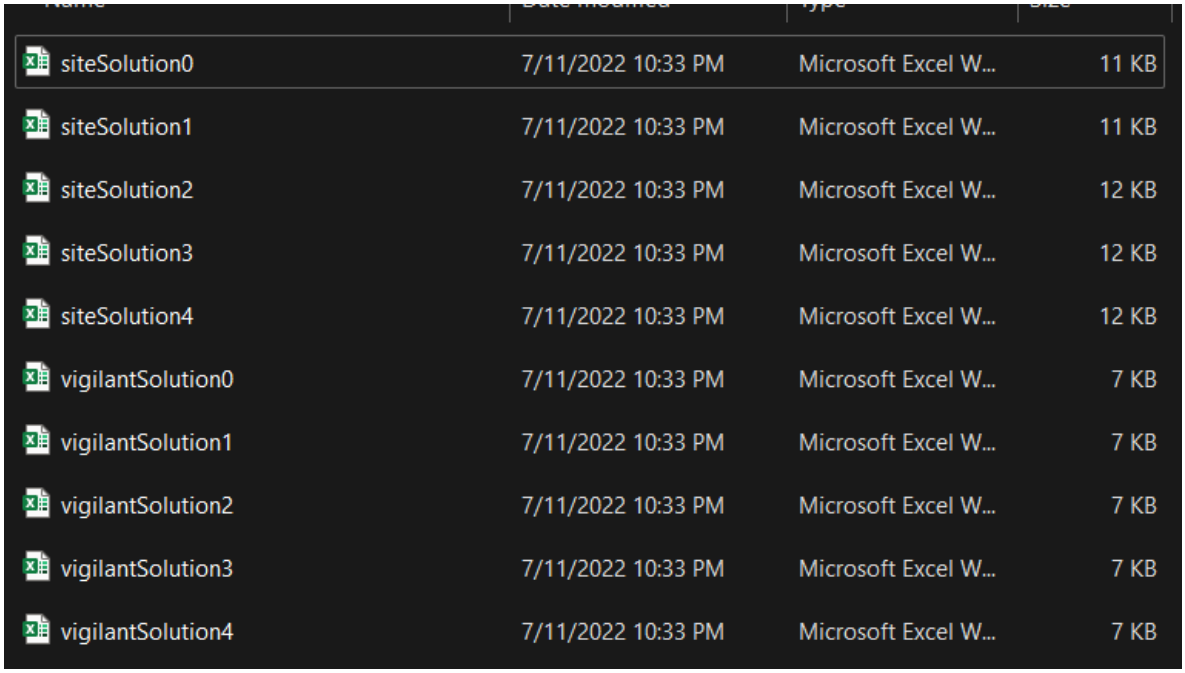


***Figura 62**. Soluciones generadas para la metaheurística.*

El cronograma se plasma en dos archivos Excel, uno conteniendo la información de los sitios y otro con la información de los vigilantes. La **Figura 63** muestra la programación de sitios. Cada hoja del archivo contiene el cronograma de un sitio, y la información de cada campo representa el id de los vigilantes que fueron asignados en ese sitio en el día d y en el periodo p, por ejemplo, los campo *I8:I15* muestra que el vigilante *12* y *10* fueron asignados el día 8 (lunes) desde las 6:00 a.m. hasta las 2:00 p.m.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Hora	dia 1	dia 2	dia 3	dia 4	dia 5	dia 6	dia 7	dia 8	dia 9	dia 10	dia 11	dia 12	dia 13	dia 14	dia 15
2	0	[]	[9, 12]	[11, 12]	[12, 1]	[12, 1]	[10, 4]	[10, 9]	[9, 3]	[11, 3]	[12, 3]	[4, 12]	[3, 4]	[9, 4]	[10, 1]	[10, 3]
3	1	[]	[9, 12]	[11, 12]	[12, 1]	[12, 1]	[10, 4]	[10, 9]	[9, 3]	[11, 3]	[12, 3]	[4, 12]	[3, 4]	[9, 4]	[10, 1]	[10, 3]
4	2	[]	[9, 12]	[11, 12]	[12, 1]	[12, 1]	[10, 4]	[10, 9]	[9, 3]	[11, 3]	[12, 3]	[4, 12]	[3, 4]	[9, 4]	[10, 1]	[10, 3]
5	3	[]	[9, 12]	[11, 12]	[12, 1]	[12, 1]	[10, 4]	[10, 9]	[9, 3]	[11, 3]	[12, 3]	[4, 12]	[3, 4]	[9, 4]	[10, 1]	[10, 3]
6	4	[]	[9, 12]	[11, 12]	[12, 1]	[12, 1]	[10, 4]	[10, 9]	[9, 3]	[11, 3]	[12, 3]	[4, 12]	[3, 4]	[9, 4]	[10, 1]	[10, 3]
7	5	[]	[9, 12]	[11, 12]	[12, 1]	[12, 1]	[10, 4]	[10, 9]	[9, 3]	[11, 3]	[12, 3]	[4, 12]	[3, 4]	[9, 4]	[10, 1]	[10, 3]
8	6	[3, 1]	[4, 3]	[3, 9]	[9, 11]	[3, 13]	[12, 1]	[12, 4]	[12, 10]	[10, 1]	[11, 1]	[11, 1]	[11, 12]	[11, 3]	[4, 11]	[4, 13]
9	7	[3, 1]	[4, 3]	[3, 9]	[9, 11]	[3, 13]	[12, 1]	[12, 4]	[12, 10]	[10, 1]	[11, 1]	[11, 1]	[11, 12]	[11, 3]	[4, 11]	[4, 13]
10	8	[3, 1]	[4, 3]	[3, 9]	[9, 11]	[3, 13]	[12, 1]	[12, 4]	[12, 10]	[10, 1]	[11, 1]	[11, 1]	[11, 12]	[11, 3]	[4, 11]	[4, 13]
11	9	[3, 1]	[4, 3]	[3, 9]	[9, 11]	[3, 13]	[12, 1]	[12, 4]	[12, 10]	[10, 1]	[11, 1]	[11, 1]	[11, 12]	[11, 3]	[4, 11]	[4, 13]
12	10	[3, 1]	[4, 3]	[3, 9]	[9, 11]	[3, 13]	[12, 1]	[12, 4]	[12, 10]	[10, 1]	[11, 1]	[11, 1]	[11, 12]	[11, 3]	[4, 11]	[4, 13]
13	11	[3, 1]	[4, 3]	[3, 9]	[9, 11]	[3, 13]	[12, 1]	[12, 4]	[12, 10]	[10, 1]	[11, 1]	[11, 1]	[11, 12]	[11, 3]	[4, 11]	[4, 13]
14	12	[3, 1]	[4, 3]	[3, 9]	[9, 11]	[3, 13]	[12, 1]	[12, 4]	[12, 10]	[10, 1]	[11, 1]	[11, 1]	[11, 12]	[11, 3]	[4, 11]	[4, 13]
15	13	[3, 1]	[4, 3]	[3, 9]	[9, 11]	[3, 13]	[12, 1]	[12, 4]	[12, 10]	[10, 1]	[11, 1]	[11, 1]	[11, 12]	[11, 3]	[4, 11]	[4, 13]
16	14	[11, 10]	[10, 1]	[4, 10]	[10, 4]	[11, 9]	[3, 11]	[11, 1]	[4, 13]	[4, 13]	[10, 13]	[13, 10]	[13, 1]	[12, 13]	[12, 8]	[12, 1]
17	15	[11, 10]	[10, 1]	[4, 10]	[10, 4]	[11, 9]	[3, 11]	[11, 1]	[4, 13]	[4, 13]	[10, 13]	[13, 10]	[13, 1]	[12, 13]	[12, 8]	[12, 1]
18	16	[11, 10]	[10, 1]	[4, 10]	[10, 4]	[11, 9]	[3, 11]	[11, 1]	[4, 13]	[4, 13]	[10, 13]	[13, 10]	[13, 1]	[12, 13]	[12, 8]	[12, 1]
19	17	[11, 10]	[10, 1]	[4, 10]	[10, 4]	[11, 9]	[3, 11]	[11, 1]	[4, 13]	[4, 13]	[10, 13]	[13, 10]	[13, 1]	[12, 13]	[12, 8]	[12, 1]
20	18	[11, 10]	[10, 1]	[4, 10]	[10, 4]	[11, 9]	[3, 11]	[11, 1]	[4, 13]	[4, 13]	[10, 13]	[13, 10]	[13, 1]	[12, 13]	[12, 8]	[12, 1]
21	19	[11, 10]	[10, 1]	[4, 10]	[10, 4]	[11, 9]	[3, 11]	[11, 1]	[4, 13]	[4, 13]	[10, 13]	[13, 10]	[13, 1]	[12, 13]	[12, 8]	[12, 1]
22	20	[11, 10]	[10, 1]	[4, 10]	[10, 4]	[11, 9]	[3, 11]	[11, 1]	[4, 13]	[4, 13]	[10, 13]	[13, 10]	[13, 1]	[12, 13]	[12, 8]	[12, 1]
23	21	[11, 10]	[10, 1]	[4, 10]	[10, 4]	[11, 9]	[3, 11]	[11, 1]	[4, 13]	[4, 13]	[10, 13]	[13, 10]	[13, 1]	[12, 13]	[12, 8]	[12, 1]
24	22	[9, 12]	[11, 12]	[12, 1]	[12, 1]	[10, 4]	[10, 9]	[9, 3]	[11, 3]	[12, 3]	[4, 12]	[3, 4]	[9, 4]	[10, 1]	[10, 3]	[3, 10]
25	23	[9, 12]	[11, 12]	[12, 1]	[12, 1]	[10, 4]	[10, 9]	[9, 3]	[11, 3]	[12, 3]	[4, 12]	[3, 4]	[9, 4]	[10, 1]	[10, 3]	[3, 10]
26	Periodos faltantes	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

***Figura 63**. Programación de sitios y horarios.*

Para los vigilantes se tiene la información de los turnos que deben cubrir y del número de horas que trabajo en cada semana (ver **Figura 64**). Una celda representa el identificador del sitio que debe vigilar y el horario del turno para un día d, por ejemplo, la celda J11 muestra que el vigilante 10 debe cubrir un turno de 6:00 a.m. a 2:00 p.m. en el sitio 1.

	A	B	C	D	E	F	G	H	I	J	K
1	**Vigilante**	**Horas trabajadas por semana**	**dia 1**	**dia 2**	**dia 3**	**dia 4**	**dia 5**	**dia 6**	**dia 7**	**dia 8**	**dia 9**
2	**1**	[48, 40, 48, 40]	[1, [6, 13]]	[1, [14, 21]]	[1, [22, 5]]	[1, [22, 5]]	[]	[1, [6, 13]]	[1, [14, 21]]	[]	[1, [6, 13]]
3	**2**	[48, 56, 56, 48]	[2, [22, 5]]	[]	[2, [6, 13]]	[2, [6, 13]]	[2, [22, 5]]	[2, [22, 5]]	[2, [22, 5]]	[2, [22, 5]]	[2, [22, 5]]
4	**3**	[48, 48, 48, 40]	[1, [6, 13]]	[1, [6, 13]]	[1, [6, 13]]	[]	[1, [6, 13]]	[1, [14, 21]]	[1, [22, 5]]	[1, [22, 5]]	[1, [22, 5]]
5	**4**	[40, 48, 48, 48]	[]	[1, [6, 13]]	[1, [14, 21]]	[1, [14, 21]]	[1, [22, 5]]	[]	[1, [6, 13]]	[1, [14, 21]]	[1, [14, 21]]
6	**5**	[48, 48, 56, 56]	[2, [14, 21]]	[2, [22, 5]]	[2, [22, 5]]	[]	[2, [6, 13]]	[2, [6, 13]]	[2, [6, 13]]	[2, [6, 13]]	[]
7	**6**	[48, 48, 48, 56]	[2, [14, 21]]	[2, [22, 5]]	[2, [22, 5]]	[]	[2, [6, 13]]	[2, [14, 21]]	[2, [14, 21]]	[2, [14, 21]]	[2, [14, 21]]
8	**7**	[48, 48, 56, 56]	[2, [6, 13]]	[2, [6, 13]]	[2, [14, 21]]	[2, [22, 5]]	[]	[2, [6, 13]]	[2, [6, 13]]	[2, [6, 13]]	[2, [6, 13]]
9	**8**	[48, 48, 56, 48]	[2, [6, 13]]	[2, [6, 13]]	[2, [14, 21]]	[2, [22, 5]]	[]	[2, [14, 21]]	[2, [14, 21]]	[2, [14, 21]]	[2, [14, 21]]
10	**9**	[48, 40, 48, 48]	[1, [22, 5]]	[]	[1, [6, 13]]	[1, [6, 13]]	[1, [14, 21]]	[1, [22, 5]]	[1, [22, 5]]	[]	[2, [6, 13]]
11	**10**	[48, 48, 48, 40]	[1, [14, 21]]	[1, [14, 21]]	[1, [14, 21]]	[1, [14, 21]]	[1, [22, 5]]	[1, [22, 5]]	[]	[1, [6, 13]]	[1, [6, 13]]
12	**11**	[48, 48, 8, 40]	[1, [14, 21]]	[1, [22, 5]]	[]	[1, [6, 13]]	[1, [14, 21]]	[1, [14, 21]]	[1, [14, 21]]	[1, [22, 5]]	[]
13	**12**	[48, 48, 48, 48]	[1, [22, 5]]	[1, [22, 5]]	[1, [22, 5]]	[1, [22, 5]]	[]	[1, [6, 13]]	[1, [6, 13]]	[1, [6, 13]]	[1, [22, 5]]
14	**13**	[8, 48, 48, 48]	[]	[]	[]	[]	[1, [6, 13]]	[]	[]	[1, [14, 21]]	[1, [14, 21]]
15	**14**	[48, 48, 56, 48]	[2, [22, 5]]	[]	[2, [6, 13]]	[2, [6, 13]]	[2, [22, 5]]	[2, [22, 5]]	[2, [22, 5]]	[2, [22, 5]]	[2, [22, 5]]

***Figura 64**. Programación de vigilantes.*

En cada ejecución se pueden obtener diferentes soluciones y contar con información de todo el proceso es vital para entender la evolución y o estancamiento del algoritmo al resolver el problema. Por esta razón, se proporciona la información gráfica de las evoluciones y los resultados de cada una de las soluciones, permitiendo visualizar las ventajas y desventajas de cada una de las soluciones de forma más clara (ver **Figura 65**). En la gráfica se muestran los valores normalizados de cada objetivo a minimizar, teniendo como máximo los valores de las violaciones calculadas en el **Capítulo 3** y como mínimo el fitness de cada solución para el objetivo minimizado (representado en porcentaje %).

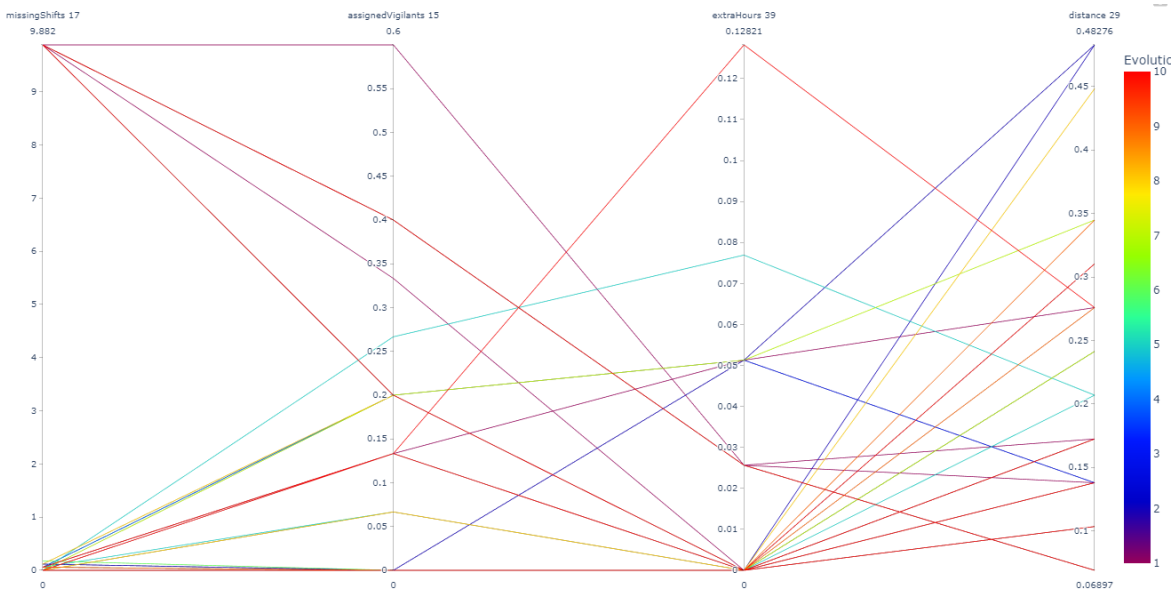


***Figura 65**. Evolución de las soluciones.*

Finalmente, también se guarda datos del proceso como los resultados en cada uno de los cuatro objetivos para cada solución, el hipervolumen y la distancia generacional invertida y el tiempo de ejecución de la Metaheurística (ver **Figura 66**).

	Evolución	Solución	Periodos faltantes (%)	Vigilantes asignados (%)	Horas extras (%)	Distancia (%)	HV Metahuristica	IGD Metahuristica	Tiempo Metahuristica
0	0	1	0.015	0.015	0.333	0.225	0.572	0.764	01:00:00.153
1	0	2	0.033	0.016	0.229	0.419	0.572	0.764	01:00:00.153
2	0	3	0.048	0.016	0.229	0.375	0.572	0.764	01:00:00.153
3	0	4	0.018	0.015	0.313	0.450	0.572	0.764	01:00:00.153
4	0	5	0.021	0.016	0.375	0.369	0.572	0.764	01:00:00.153
5	0	6	0.012	0.015	0.292	0.343	0.572	0.764	01:00:00.153
6	0	7	0.018	0.016	0.333	0.338	0.572	0.764	01:00:00.153
7	0	8	0.024	0.015	0.229	0.312	0.572	0.764	01:00:00.153
8	0	9	0.024	0.015	0.271	0.345	0.572	0.764	01:00:00.153
9	0	10	0.015	0.015	0.333	0.265	0.572	0.764	01:00:00.153
10	1	1	0.015	0.015	0.333	0.225	0.705	0.753	01:00:00.153
11	1	2	0.015	0.013	0.188	0.371	0.705	0.753	01:00:00.153
12	1	3	0.012	0.013	0.208	0.252	0.705	0.753	01:00:00.153
13	1	4	0.024	0.013	0.083	0.427	0.705	0.753	01:00:00.153
14	1	5	0.024	0.013	0.208	0.177	0.705	0.753	01:00:00.153
15	1	6	0.033	0.016	0.229	0.419	0.705	0.753	01:00:00.153
16	1	7	0.048	0.016	0.229	0.375	0.705	0.753	01:00:00.153
17	1	8	0.018	0.015	0.313	0.450	0.705	0.753	01:00:00.153
18	1	9	0.021	0.016	0.375	0.369	0.705	0.753	01:00:00.153
19	1	10	0.012	0.015	0.292	0.343	0.705	0.753	01:00:00.153

***Figura 66**. Datos estadísticos de la ejecución del algoritmo.*

CAPÍTULO 8

8 EXPERIMENTACIÓN Y RESULTADOS

En orden de verificar la efectividad de los algoritmos y las técnicas propuestas, el problema de asignación de vigilantes con horarios flexibles fue creado y optimizado con base en 6 casos de prueba, cubriendo diferentes dificultades y necesidades (fácil, media y difícil), y con base en la información de vigilantes que se maneja actualmente en una empresa.

8.1 EXPERIMENTACIÓN CON CASOS SINTÉTICOS

Se realizaron 6 casos de prueba, con el fin de validar cómo se comporta el software en diferentes problemas, los casos de prueba diseñados simulan los problemas que una empresa de seguridad de vigilancia puede presentar, como, por ejemplo: variación de los horarios, diferente cantidad de vigilantes, y variación en el número de vigilantes requeridos por turnos en los sitios. La información de los casos utilizadas se encuentra en la **Tabla 10**. Con respecto a la cantidad de periodos n^{ideal},n^{max}, T^{rest} se utilizaron los valores de $n^{ideal} = 48$, $n^{max} = 56$ y $T^{rest} = 12$, valores que son los frecuentemente utilizados en las empresas de seguridad y vigilancia.

Tabla 10. *Casos sintéticos.*

Caso	Cantidad de sitios	Vigilantes	Vigilantes esperados	Horario (1 mes)	Vigilantes requeridos en el horario	Tiempo de ejecución	Número de ejecuciones
1	2	14	11	10 p.m. a 6 a.m.	2	15 minutos	10
2	2	5	3	6 a.m. a 6 a.m.		15 minutos	
3	30	210	205	10 p.m. a 6 a.m.		30 minutos	
4	30	70	65	6 a.m. a 6 a.m.		30 minutos	
5	50	350	345	6 a.m. a 6 a.m.		1 hora	
6	50	116	110	10 p.m. a 6 a.m.		1 hora	

De los casos ejecutados, y por los tamaños contenidos, se mostrará el mejor resultado de las iteraciones de cada caso en concreto.

Caso 1

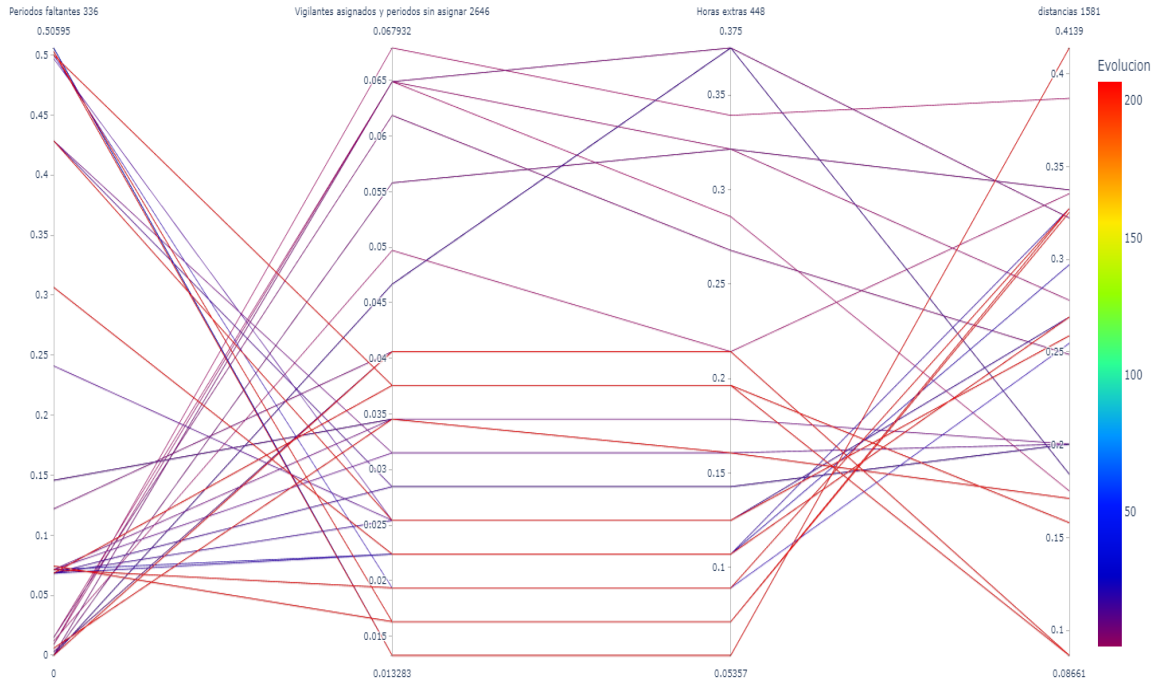


Figura 67. *NSGA-II evolución de los objetivos (Caso 1).*

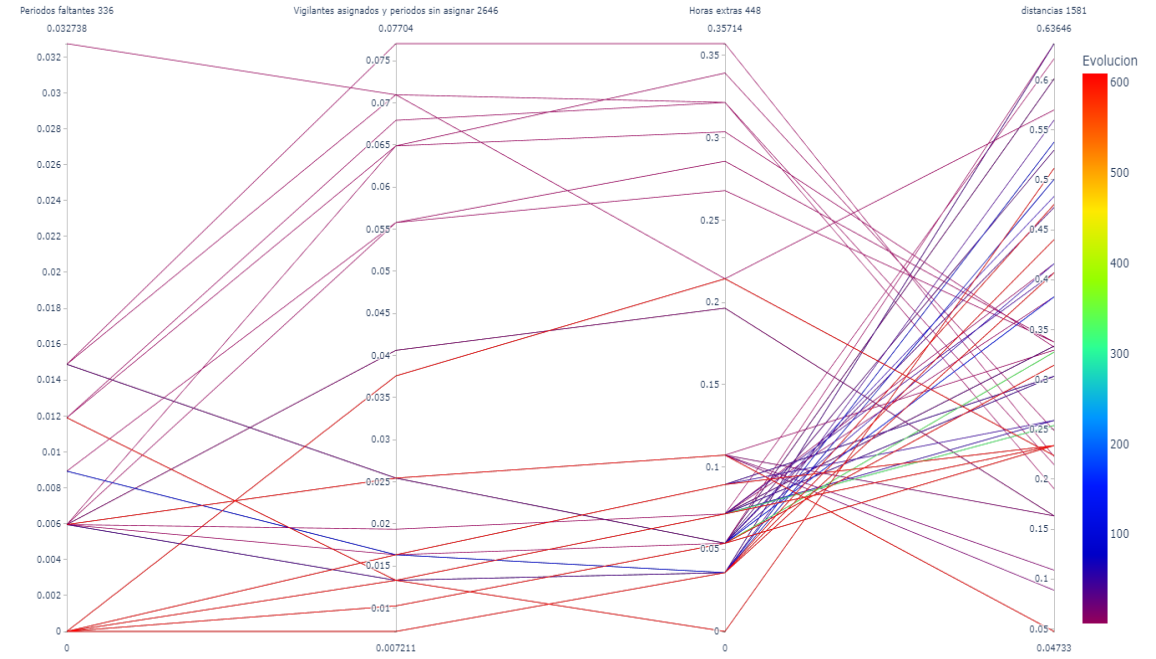


Figura 68. *GRASP Multiobjetivo evolución de los objetivos (Caso 1)*

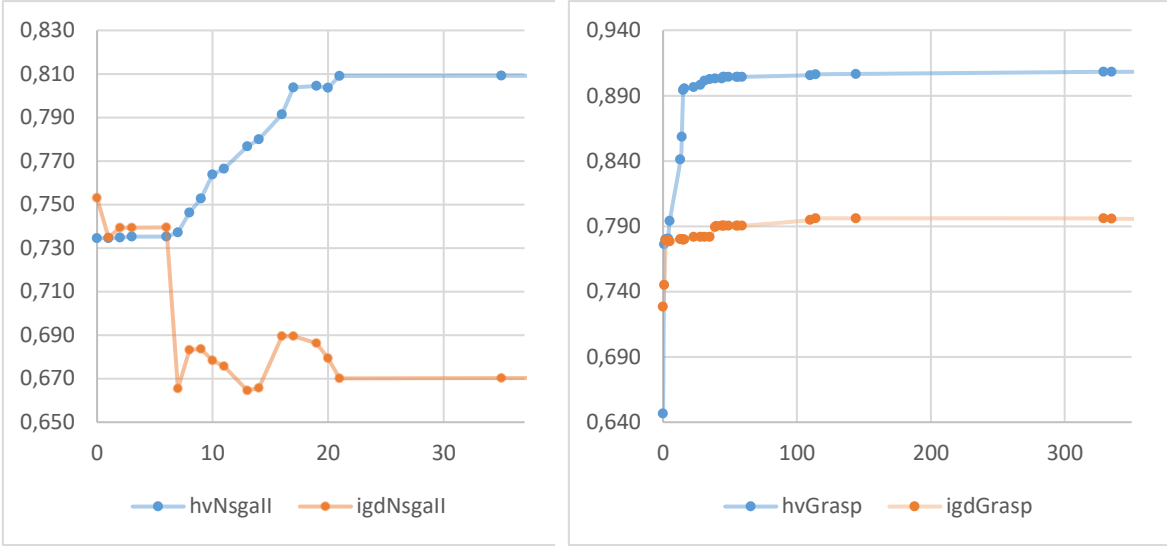


Figura 69. *Comparación de las métricas de evolución (Caso 1).*

Del primer caso, se observa que ambas metaheurísticas (ver **Figura 67** y **Figura 68**) pudieron cumplir con la cobertura total de turnos, y van evolucionando con el tiempo. En la **Figura 69**, se puede observar como el **HV** va aumento tras cada evolución, MGRASP particularmente obtiene un mejor resultado que NSGA-II, optimizando de mejor manera los objetivos locales, pero en la métrica **IGD**, NSGA-II es la ganadora, la cual mantiene un mayor espacio de búsqueda que MGRASP (ver **Tabla 11**). En la **Tabla 12** se muestran los resultados de los objetivos máximos y mininos obtenidos al inicio y al final del proceso de evolución. De MGRASP, se observa que al final de la evolución reduce los valores máximos y mínimos de cada objetivo, mientras que NSGA-II aumenta el rango máximo para el objetivo de minimización de turnos sin asignar, obteniendo diferentes soluciones que minimizan los rangos de los demás objetivos.

Tabla 11*. Resultado de métricas de evolución (Caso 1).*

Metaheurística	Evolución final	HV inicial	HV final	IGD inicial	IGD final	IGD promedio
MGRASP	609	0.646	0.908	0.728	0.795	0.794
NSGA-II	192	0.735	0.809	0.753	0.670	0.673

Tabla 12*. Resultado de objetivos máximos y mínimos de la evolución (Caso 1).*

Metaheurística	Turnos sin asignar		Vigilantes asignados y periodos sin utilizar		Horas Extras		Distancia	
	Max (%)	Min (%)	Max (%)	Min (%)	Max (%)	Min (%)	Max (%)	Min (%)
MGRASP Inicial	0.03	0	0.077	0.038	0.357	0.196	0.57	0.163
MGRASP Final	0.012	0	0.038	0.007	0.214	0	0.512	0.047
NSGA-II Inicial	0.015	0	0.068	0.035	0.375	0.161	0.414	0.087
NSGA-II Final	0.503	0	0.041	0.013	0.214	0.054	0.414	0.087

Caso 2

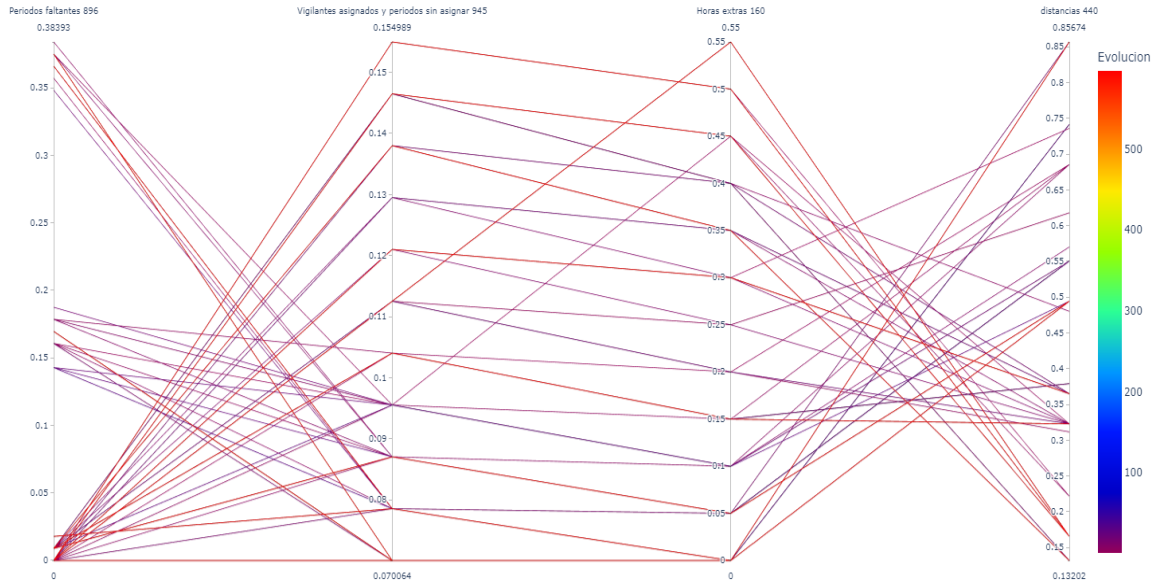


Figura 70. *NSGA-II evolución de los objetivos (Caso 2).*

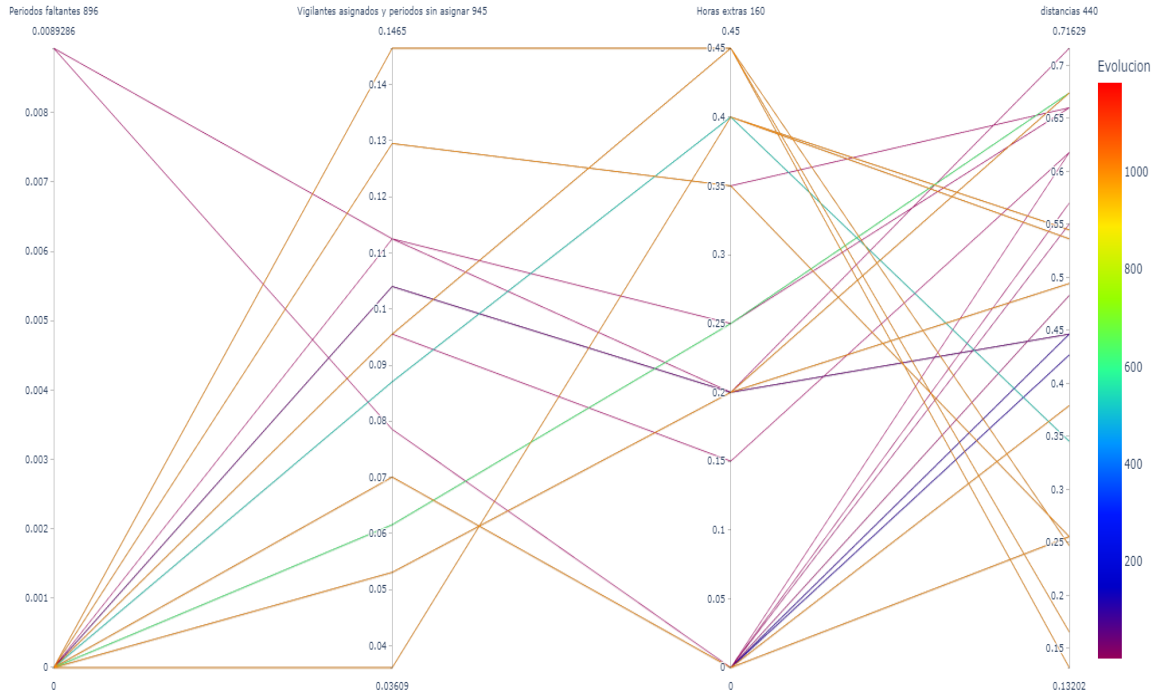


Figura 71. *GRASP Multiobjetivo evolución de los objetivos (Caso 2).*

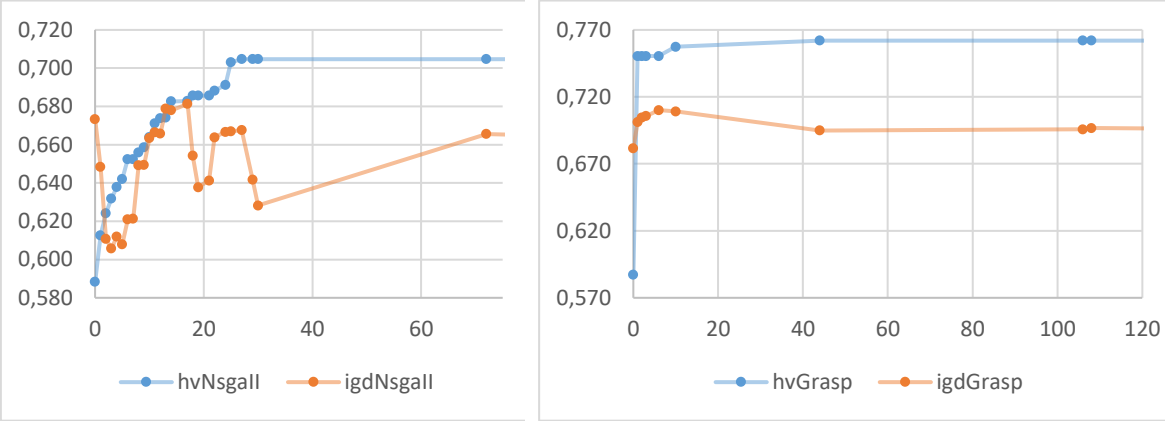


Figura 72. *Comparación de las métricas de evolución (Caso 2).*

En el caso 2, nuevamente ambas metaheurísticas mantienen resultados muy bajos en cada objetivo (ver **Figura 70** y **Figura 71**), MGRASP evoluciona el **HV** de una forma rápida, quedando estancado en sus primeras evoluciones. NSGA-II, por otra parte, evoluciona de una forma más lenta, y genera soluciones más dispersas en el espacio de búsqueda (ver **Figura 72**). En la **Tabla 13**, se observa como MGRASP mantiene mejores resultados en el **HV**, mientras que NSGA-II lo tiene en la métrica de **IGD**. MGRASP desde el inicio optimiza mejor los objetivos que NSGA-II (ver **Tabla 14**), y mantiene esta diferencia al final del procedimiento. Al igual que el caso anterior, MGRASP optimiza mejor los objetivos locales, generando rangos máximos y mínimos más reducidos, mientras NSGA-II mantiene un mayor espacio de búsqueda, especialmente en el primer objetivo de minimización de turnos sin asignar.

Tabla 13*. Resultado de métricas de evolución (Caso 2).*

Metaheurística	Evolución final	HV inicial	HV final	IGD inicial	IGD final	IGD promedio
MGRASP	1182	0.587	0.762	0.682	0.686	0.691
NSGA-II	596	0.588	0.705	0.673	0.628	0.634

Tabla 14*. Resultado de objetivos máximos y mínimos de la evolución (Caso 2).*

Metaheurística	Turnos sin asignar		Vigilantes asignados y periodos sin utilizar		Horas Extras		Distancia	
	Max (%)	Min (%)	Max (%)	Min (%)	Max (%)	Min (%)	Max (%)	Min (%)
MGRASP Inicial	0.009	0	0.146	0.07	0.45	0.15	0.716	0.132
MGRASP Final	0	0	0.146	0.036	0.45	0	0.674	0.132
NSGA-II Inicial	0.009	0	0.155	0.096	0.550	0.2	0.736	0.132
NSGA-II Final	0.366	0	0.155	0.07	0.550	0	0.857	0.132

Caso 3

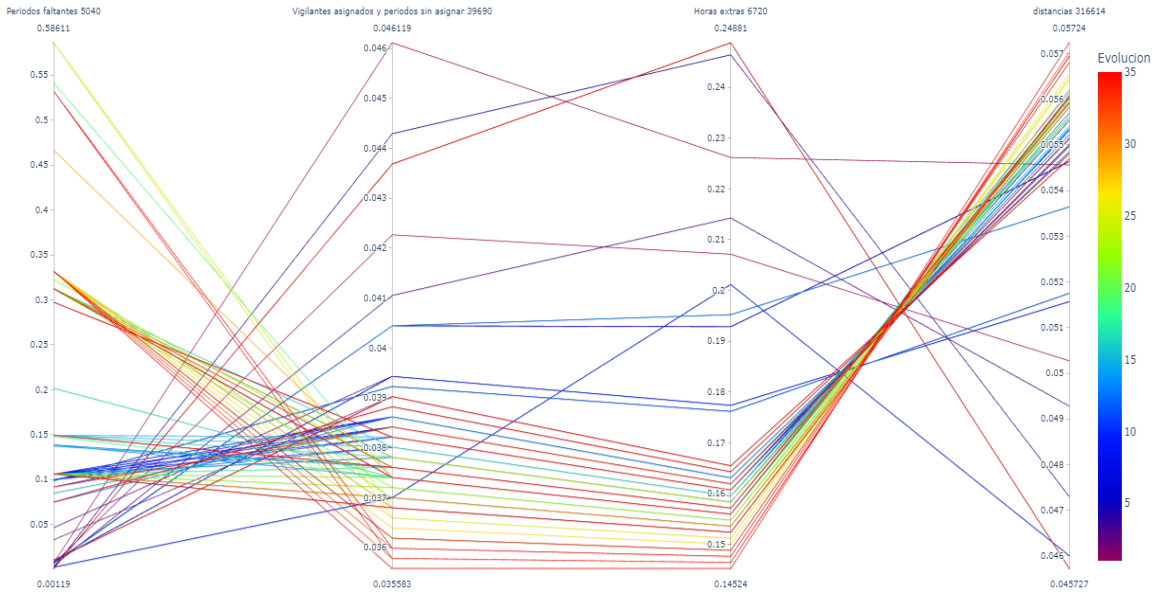


Figura 73. *NSGA-II evolución de los objetivos (Caso 3).*

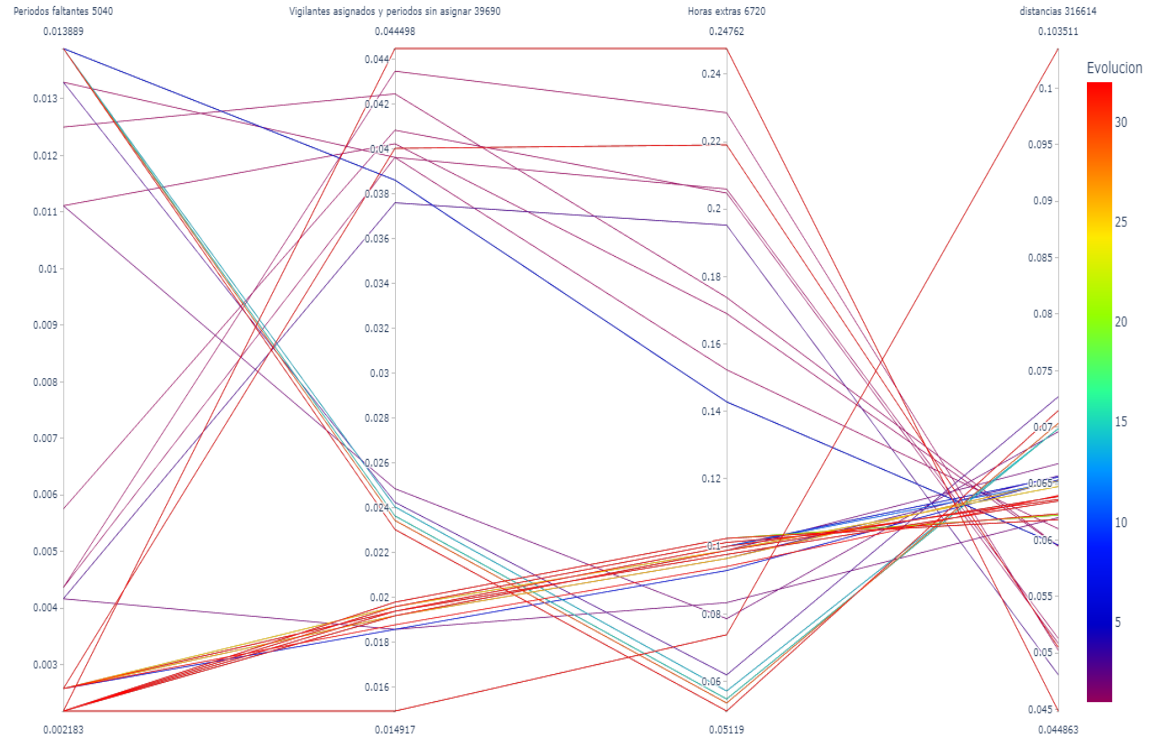


Figura 74. *GRASP Multiobjetivo evolución de los objetivos (Caso 3).*

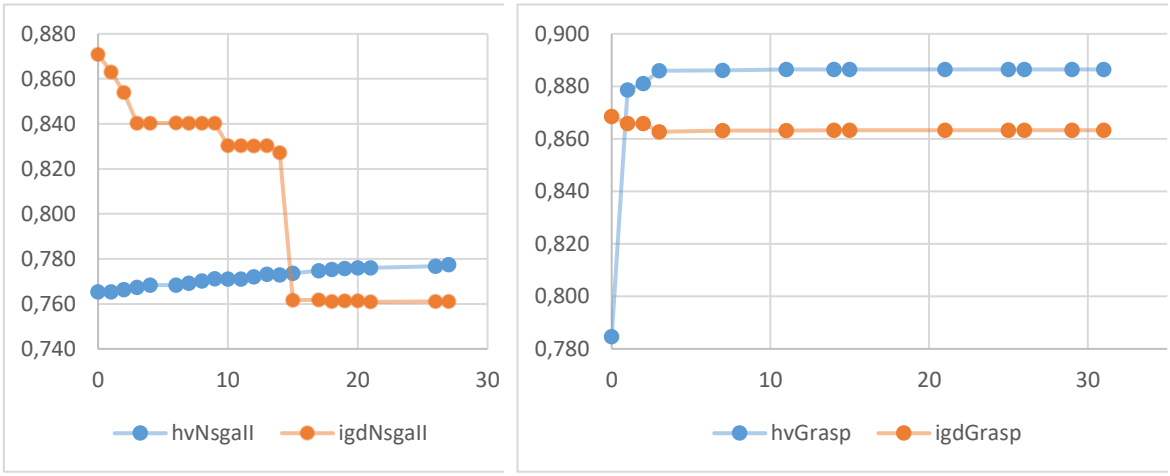


Figura 75. *Comparación de las métricas de evolución (Caso 3).*

En la **Figura 73** y **Figura 74**, se comienza a evidenciar el costo computacional, y el tiempo que toma resolver el problema de asignación de seguridad y vigilancia cuando la cantidad de sitios, y vigilantes empieza a ser grande. Por un lado, se comienza a reducir considerablemente el número de evoluciones, y los resultados de cada objetivo empiezan a aumentar, aunque mantienen el mismo comportamiento de los casos anteriores. En la **Figura 75**, se observa que el **HV** de MGRASP y NSGA-II, aumenta al mismo tiempo que la **IGD** va disminuyendo. MGRASP en el proceso evolutivo hace una gran mejora al inicio, y después se mantiene más estable, tal como sucede con los casos anteriores, mientras que NSGA-II, aunque aumenta el **HV**, su cambio no es tan significante a comparación de la **IGD**. En la **Tabla 15**, se observan a más detalle el resultado inicial y final de cada métrica. En la **Tabla 16**, se observa que ninguna metaheurística logro cubrir en totalidad los turnos, aunque quedan muy cerca de lograrlo, siendo NSGA-II la mejor opción para este objetivo, para los demás objetivos MGRASP mantiene resultados menores y rangos más bajos.

Tabla 15. *Resultado de métricas de evolución (Caso 3).*

Metaheurística	Evolución final	HV inicial	HV final	IGD inicial	IGD final	IGD promedio
MGRASP	31	0.785	0.886	0.869	0.863	0.863
NSGA-II	34	0.765	0.777	0.871	0.761	0.804

Tabla 16. *Resultado de objetivos máximos y mínimos de la evolución (Caso 3).*

Metaheurística	Turnos sin asignar		Vigilantes asignados y periodos sin utilizar		Horas Extras		Distancia	
	Max (%)	Min (%)	Max (%)	Min (%)	Max (%)	Min (%)	Max (%)	Min (%)
MGRASP Inicial	0.014	0.002	0.044	0.038	0.248	0.143	0.061	0.045
MGRASP Final	0.014	0.002	0.044	0.015	0.248	0.051	0.104	0.045
NSGA-II Inicial	0.011	0.001	0.046	0.037	0.2249	0.165	0.055	0.046
NSGA-II Final	0.434	0.001	0.044	0.036	0.249	0.149	0.057	0.046

Caso 4

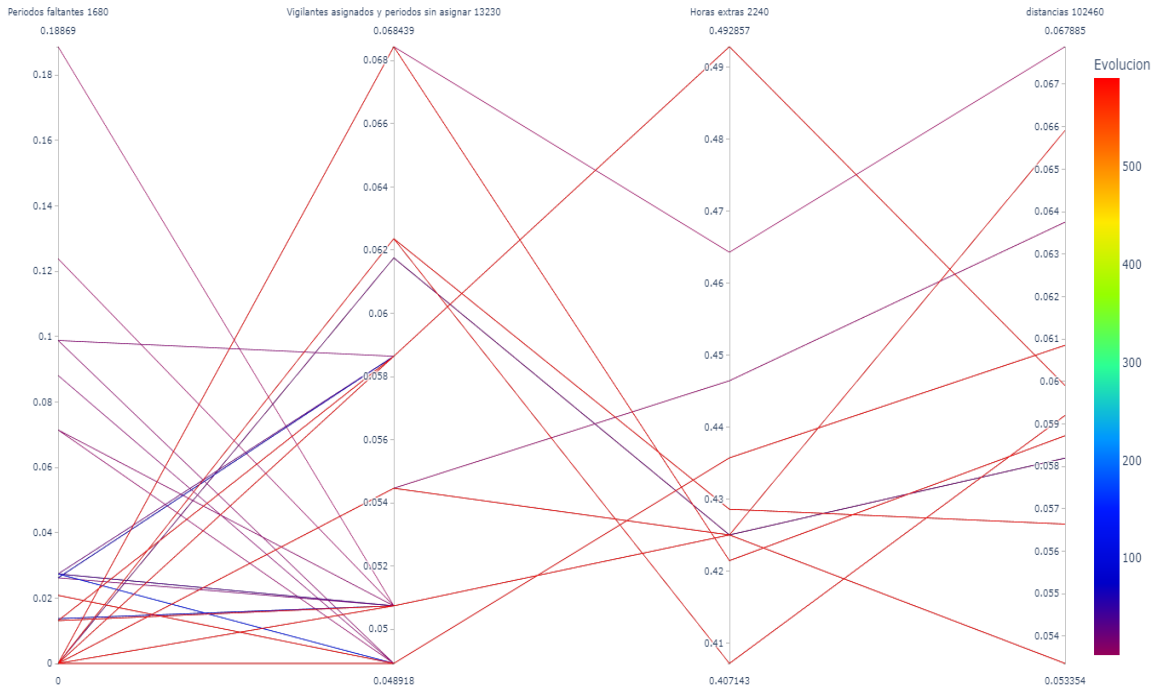


Figura 76. *NSGA-II evolución de los objetivos (Caso 4).*

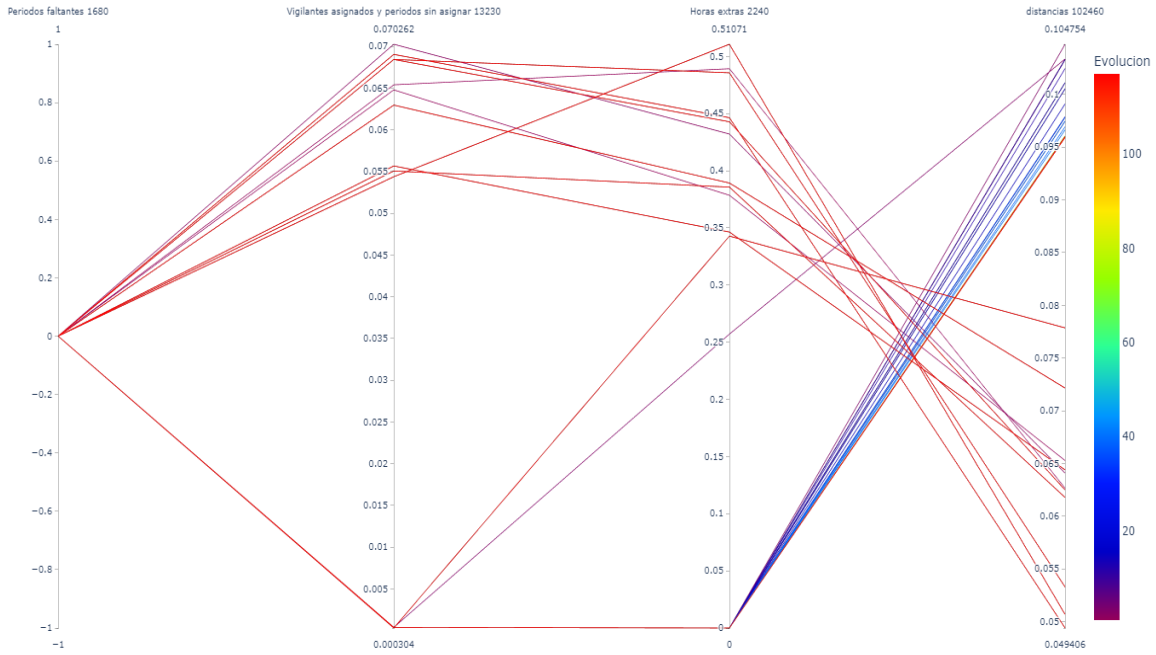


Figura 77. *GRASP Multiobjetivo evolución de los objetivos (Caso 4).*

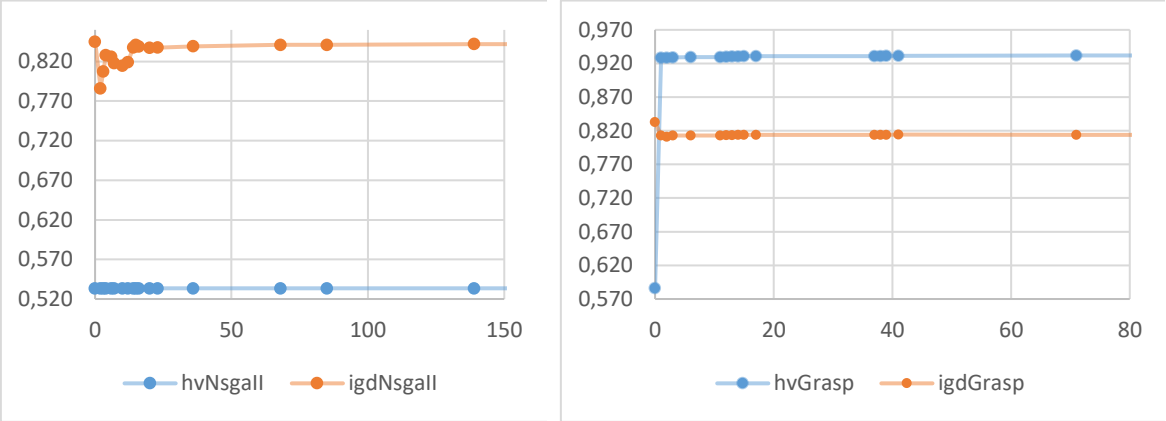


Figura 78. *Comparación de las métricas de evolución (Caso 5).*

Es preciso recordar que, para este caso, existe un único turno por día (turno de 8 horas), limitando las opciones y evoluciones de búsqueda. En la **Figura 76** y **Figura 77**, se observa que, para este tipo de casos, las evoluciones son propensas a quedar atrapados en óptimos locales. Anteriormente en el caso 2, NSGA-II presento que tras un numero de evoluciones se quedaba estancado, para el caso aquí mencionado, NSGA-II no logra mejorar las soluciones no dominadas de la primera evolución. En la **Figura 78**, se puede ver como MGRASP en la primera evolución mejora el **HV**, después, sus mejoras son mínimas, NSGA-II por lo contrario no logra encontrar mejores soluciones en los óptimos locales. NSGA-II en las primeras evoluciones amplia el rango de las soluciones, y se mantiene con ellas, impidiendo obtener mejores soluciones para el primer frente. En la **Tabla 17**, se describen los resultados de cada métrica. En la **Tabla 18**, se observa como MGRASP al final del proceso evolutivo minimiza los resultados de cada objetivo, a excepción del objetivo de distancia que se mantiene igual. NSGA-II no logra minimizar ningún objetivo, sino que amplía el rango para el objetivo turnos sin asignar y de distancia.

Tabla 17. *Resultado de métricas de evolución (Caso 4).*

Metaheurística	Evolución final	HV inicial	HV final	IGD inicial	IGD final	IGD promedio
MGRASP	116	0.586	0.932	0.833	0.814	0.814
NSGA-II	590	0.533	0.533	0.845	0.842	0.841

Tabla 18. *Resultado de objetivos máximos y mínimos de la evolución (Caso 4).*

Metaheurística	Turnos sin asignar		Vigilantes asignados y periodos sin utilizar		Horas Extras		Distancia	
	Max (%)	Min (%)	Max (%)	Min (%)	Max (%)	Min (%)	Max (%)	Min (%)
MGRASP Inicial	0	0	0.070	0.054	0.511	0.346	0.072	0.049
MGRASP Final	0	0	0.069	0	0.511	0	0.096	0.049
NSGA-II Inicial	0	0	0.068	0.049	0.493	0.407	0.068	0.053
NSGA-II Final	0.021	0	0.068	0.049	0.493	0.407	0.066	0.053

Caso 5

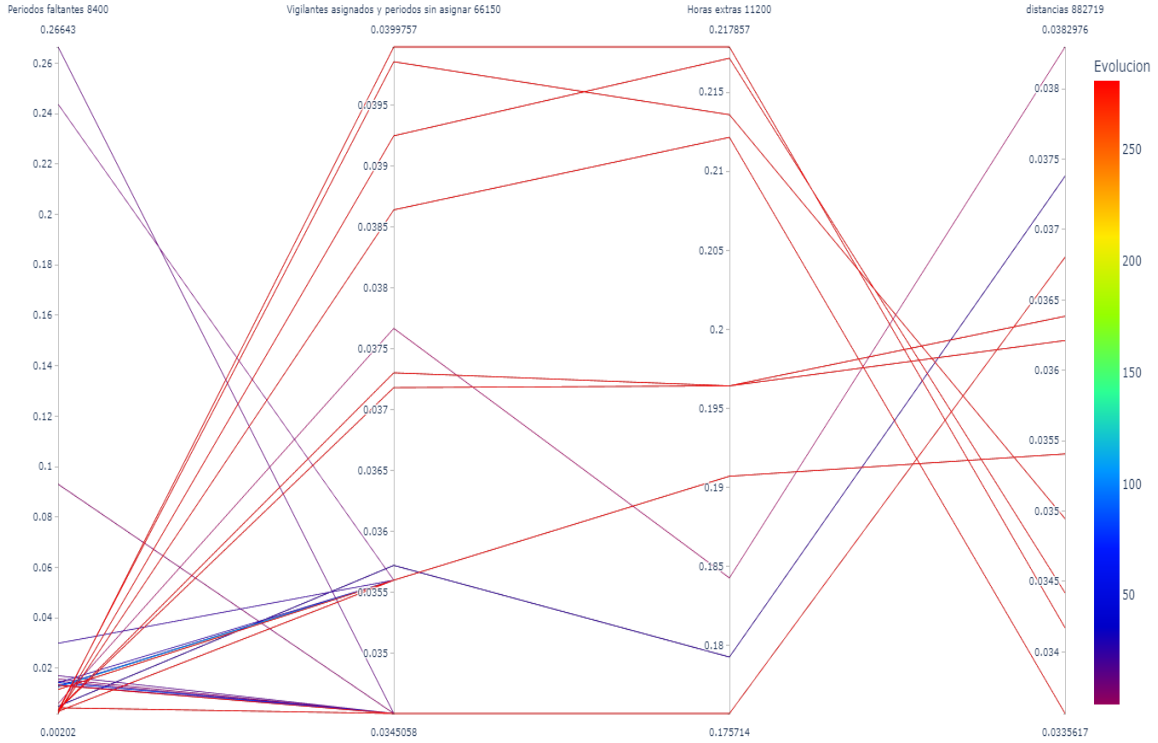


Figura 79. *NSGA-II evolución de los objetivos (Caso 5).*

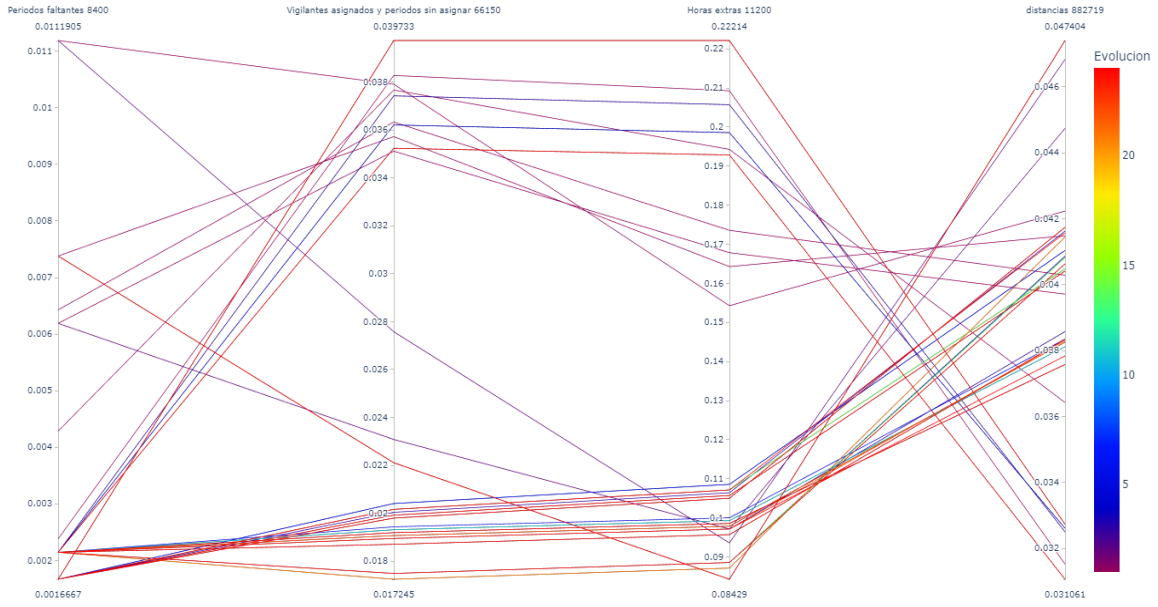


Figura 80. *GRASP Multiobjetivo evolución de los objetivos (Caso 5).*

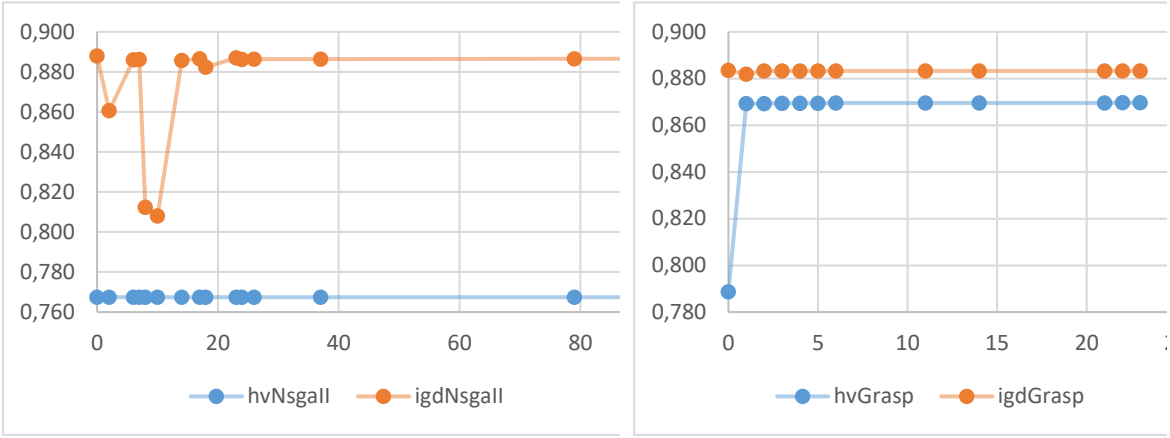


Figura 81. *Comparación de las métricas de evolución (Caso 5).*

Para los problemas de asignación de personal y vigilancia que son muy grandes, se identifica que MGRASP se queda atrás respecto al número de evoluciones que maneja NSGA-II (ver **Figura 79**, **Figura 80**). MGRASP que debe construir muchos componentes, y de estos escoger los mejores para unir a la solución, le toma demasiado tiempo en problemas muy grandes (ver **Figura 81**), sin embargo, ambas terminan basándose en soluciones que ya cumplen con un nivel de factibilidad, por lo que las evoluciones de NSGA-II se asemejan a la cantidad de componentes que debe generar MGRASP. En la **Tabla 19**, se muestra el resultado de las métricas del proceso evolutivo, y se observa que NSGA-II, al igual que el caso anterior, en problemas grandes no puede mejorar su **HV**, quedándose en cada evolución con el mismo frente. En la **Tabla 20**, se muestra que a NSGA-II le es difícil ampliar o minimizar los rangos, a comparación de los casos anteriores, permitiendo que MGRASP mantenga mejores resultados en cada uno de los objetivos.

Tabla 19. *Resultado de métricas de evolución (Caso 5).*

Metaheurística	Evolución final	HV inicial	HV final	IGD inicial	IGD final	IGD promedio
MGRASP	23	0.789	0.870	0.884	0.883	0.883
NSGA-II	280	0.767	0.767	0.888	0.886	0.885

Tabla 20. *Resultado de objetivos máximos y mínimos de la evolución (Caso 5).*

Metaheurística	Turnos sin asignar		Vigilantes asignados y periodos sin utilizar		Horas Extras		Distancia	
	Max (%)	Min (%)	Max (%)	Min (%)	Max (%)	Min (%)	Max (%)	Min (%)
MGRASP Inicial	0.011	0.002	0.040	0.035	0.222	0.154	0.042	0.031
MGRASP Final	0.007	0.002	0.040	0.017	0.222	0.084	0.047	0.031
NSGA-II Inicial	0.006	0.002	0.040	0.035	0.218	0.176	0.038	0.034
NSGA-II Final	0.013	0.002	0.040	0.035	0.218	0.176	0.037	0.034

Caso 6

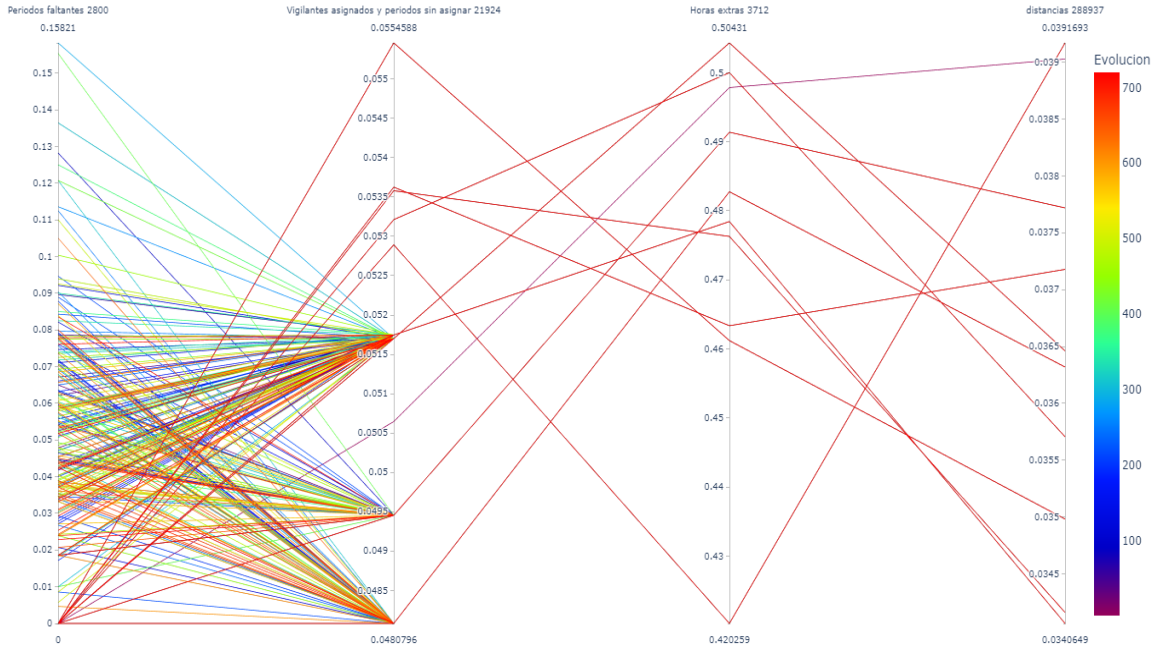


Figura 82. *NSGA II evolución de los objetivos (Caso 6).*

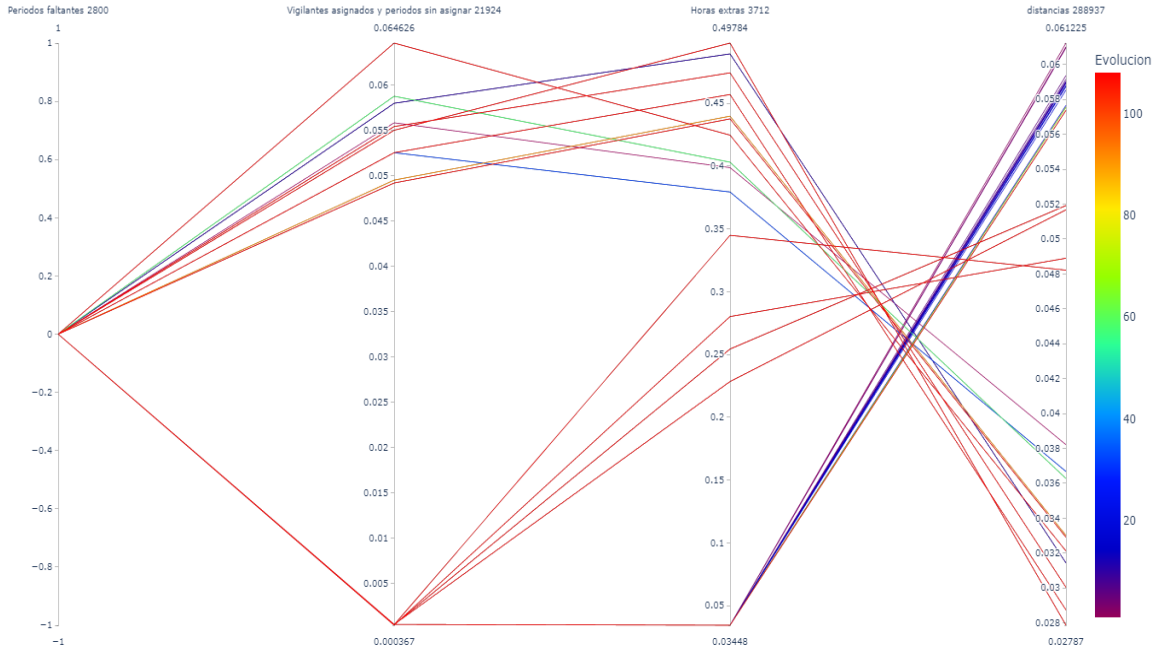


Figura 83 *GRASP Multiobjetivo evolución de los objetivos (Caso 6).*

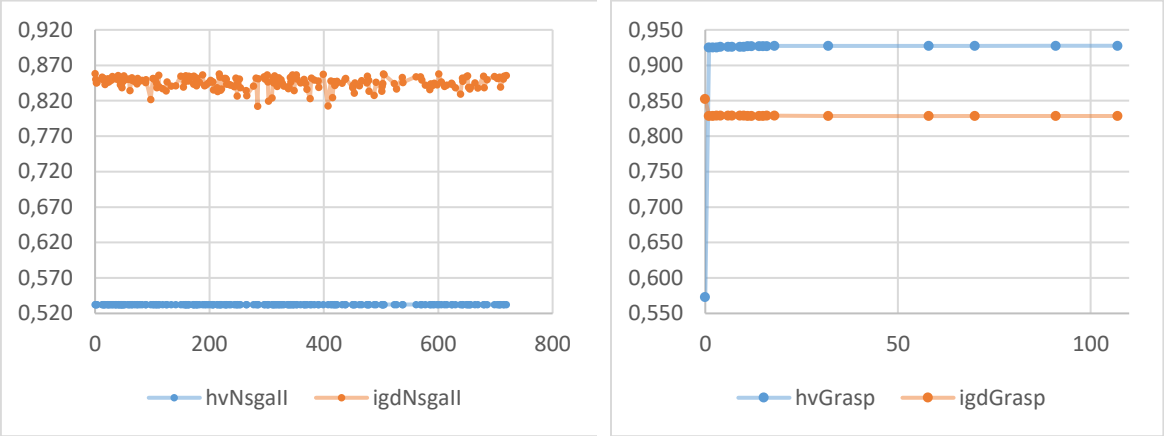


Figura 84. *Comparación de las métricas de evolución (Caso 6).*

En la **Figura 82** y **Figura 83**, se puede ver más claro las limitaciones anteriormente mencionadas del caso 4, respecto a la exploración de soluciones en el espacio de búsqueda. Al tener limitantes en los turnos, y al tener gran cantidad de sitios y vigilantes, las metaheurísticas son propensas a quedarse en los óptimos locales. MGRASP, es la que mayor impacto tiene en este tema, debido a su fase de búsqueda local, que busca soluciones en sus vecinos más cercanos. Esta estrategia limita el espacio de búsqueda, a diferencia de NSGA-II que explora con diferentes soluciones (ver **Figura 84**), sin embargo, la adaptación aquí investigada no logra mejorar el frente en problemas muy grandes. En este caso particularmente, NSGA-II amplía el rango de búsqueda del objetivo de minimización de turnos sin asignar en el proceso, y de esta toma las soluciones con mayor dispersión. En la **Tabla 21**, se muestra el resultado de cada métrica. De los resultados, se tiene que MGRASP mejora el HV de manera considerable para los casos donde solo se tiene un turno en el horario. La **Tabla 22** muestra los resultados de cada objetivo, donde MGRASP obtiene mejores resultados en la minimización de cada uno de los objetivos.

Tabla 21. *Resultado de métricas de evolución (Caso 6).*

Metaheurística	Evolución final	HV inicial	HV final	IGD inicial	IGD final	IGD promedio
MGRASP	107	0.573	0.928	0.852	0.828	0.829
NSGA-II	719	0.533	0.533	0.858	0.856	0.848

Tabla 22. *Resultado de objetivos máximos y mínimos de la evolución (Caso 6).*

Metaheurística	Turnos sin asignar		Vigilantes asignados y periodos sin utilizar		Horas Extras		Distancia	
	Max (%)	Min (%)	Max (%)	Min (%)	Max (%)	Min (%)	Max (%)	Min (%)
MGRASP Inicial	0	0	0.065	0.049	0.499	0.379	0.038	0.028
MGRASP Final	0	0	0.065	0	0.498	0.034	0.057	0.028
NSGA-II Inicial	0	0	0.055	0.048	0.504	0.42	0.039	0.034
NSGA-II Final	0.025	0	0.055	0.048	0.504	0.42	0.039	0.034

De los resultados obtenido se encuentra que, NSGA-II realiza y comienza a evolucionar más rápido que MGRASP a partir de los casos de dificultad media, esto hace sentido ya que NSGA-II parte de una única población inicial, para luego explorar en los espacios de búsqueda, a diferencia de MGRASP, que al ser una metaheurística voraz, va a tomarle más tiempo en crear y recorrer las soluciones para la población inicial a evolucionar. En general, ambas metaheurísticas minimizan el objetivo de asignación de turnos, por lo menos para estos casos, siendo uno de los objetivos más importantes, sino que el más importante para una empresa de seguridad y vigilancia. MGRASP en los casos aquí propuestos, tiene mejores resultados en la optimización de cada uno de los objetivos, pero abre paso a que quede atrapado localmente en cada uno de estos. Por otro lado, NSGA-II, se destaca por abrir y buscar más soluciones en el espacio de búsqueda, con la ayuda de las soluciones iniciales, las cuales mantiene durante el proceso evolutivo, logrando así una mayor dispersión en las soluciones.

8.2 EXPERIMENTNACION CON DATOS DE LA EMPRESA

Con el fin de validar la eficiencia de las metaheurísticas se realizó un caso de estudio con datos reales de una empresa encargada a la asignación de vigilantes en el departamento del Cauca, pero por motivos de seguridad prefieren mantener en el anonimato. Se proporciono un caso en concreto que la empresa manejo y diseño para el presente año, el caso consta de 14 sitios a cubrir, y 42 vigilantes, siendo 36 vigilantes dedicados a vigilar puestos físicos y 6 al área de monitoreo. Este cronograma fue diseñado para 1 mes, cubriendo cada día de la semana.

A diferencia de la investigación aquí realizada, los turnos que manejan la empresa son fijos y estos aplican para cada sitio a vigilar, en cada turno solamente se asigna un vigilante a excepción del sitio de monitoreo que requiere 2 vigilantes por turno. El horario de vigilancia se realiza un total de 24 horas cada día de la semana, y los turnos definidos para cubrir este horario se pueden observar en la **Tabla 23**.

Tabla 23. *Turnos definidos de la empresa.*

Lunes	6 a.m. - 2 p.m.	2 p.m. - 10 p.m.	10 p.m. - 6 a.m.
Martes	6 a.m. - 6 p.m.	6 a.m. - 6 p.m.	
Miércoles	6 a.m. - 6 p.m.	6 a.m. - 6 p.m.	
Jueves	6 a.m. - 6 p.m.	6 a.m. - 6 p.m.	
Viernes	6 a.m. - 2 p.m.	2 p.m. - 10 p.m.	10 p.m. - 6 a.m.
Sabado	6 a.m. - 2 p.m.	2 p.m. - 10 p.m.	10 p.m. - 6 a.m.
Domingo	6 a.m. - 2 p.m.	2 p.m. - 10 p.m.	10 p.m. - 6 a.m.

Según la información suministrada el cálculo total de periodos a vigilar por semana es de $24*14*7 = 2352$, si se divide entre los 42 vigilantes que maneja la empresa, da como resultado que cada vigilante debe trabajar por semana 56 horas. Esto quiere decir que todos los vigilantes están trabajando 8 horas extras por semana.

Para la empresa ésta es la soluciona ideal y está totalmente optimizada para cubrir los turnos.

Como se evidencia el cronograma de la empresa no tiene turnos flexibles, con el propósito de experimentar diferentes casos de la vida real se propone tres escenarios:

1. En el primer caso se propone correr con el dataset de la empresa de vigilancia sin ninguna modificación en el algoritmo.
2. El segundo caso se propone hacer una adaptación a los turnos que maneja la empresa y desactivar el objetivo de horas extras.
3. El tercer caso plantea desactivar el objetivo de horas extras con el fin de darles a los algoritmos mayor flexibilidad en la asignación de los vigilantes.

8.2.1 CASO SIN ADAPTAR LAS METAHEURISTICAS

Para este escenario, se corren los algoritmos sin nungun tipo de adaptacion al problema de la empresa de seguridad y vigilancia. En la **Figura 85** y **Figura 86**, se evidencia que los algoritmos obtienen buenos resultados, mejorando todos los objetivos a minimizar, pero no logran asignar el 100% los turnos, esto se debe en parte a que los algoritmos están adaptados para horarios flexibles, y no fijos como se maneja en la empresa. Además, el programa está diseñado para optimizar más objetivos de los que se maneja en la empresa, como lo es la cantidad de horas extra, siendo este el objetivo que más perjudica a que no se cubran la totalidad de los turnos para este problema en específico.

En la **Figura 85,** MGRASP encuentra varios puntos óptimos en diferentes objetivos, por ejemplo, en las evoluciones de las líneas color azul, se observa que encontró los mejores valores para el objetivo de distancia, pero descuida los objetivos de asignación de vigilantes y turnos sin asignar, además, también se puede evidenciar que la cantidad de horas extras en las últimas evoluciones tiene picos muy altos, esto se debe a que los datos suministrados por la empresa están totalmente ajustados para que cada vigilante trabaje 56 horas cada semana, es decir, que en las últimas evoluciones MGRASP busca encontrar la solución óptima para la empresa, incrementando las horas extras para hacer posible la asignación total de turnos para todos los sitios.

En la **Figura 86**, NSGA-II al igual que MGRASP le es imposible optimizar las horas extras debido a la cantidad de vigilantes disponibles, además, el algoritmo que se encarga de crear las soluciones iniciales no está adaptado para asignar más de 48 horas por semana. Cuando el algoritmo finaliza, no alcanza a cubrir el total de periodos de cada sitio, por esto, las solución iniciales finalizan con turnos sin asignar, afectando al resultado del objetivo de minimización de turnos sin asignar, NSGA-II, para optimizar este objetivo, empieza a asignar horas extras a los vigilantes en un diferente sitio, afectando negativamente el objetivo de distancia, dado que los vigilantes empiezan a trabajar en más de un sitio. Como el proceso evolutivo de NSGA-II intenta mejorar los genes menos adaptados, se hace una exploración más exhaustiva en el objetivo de minimización de turnos faltantes y de distancia.

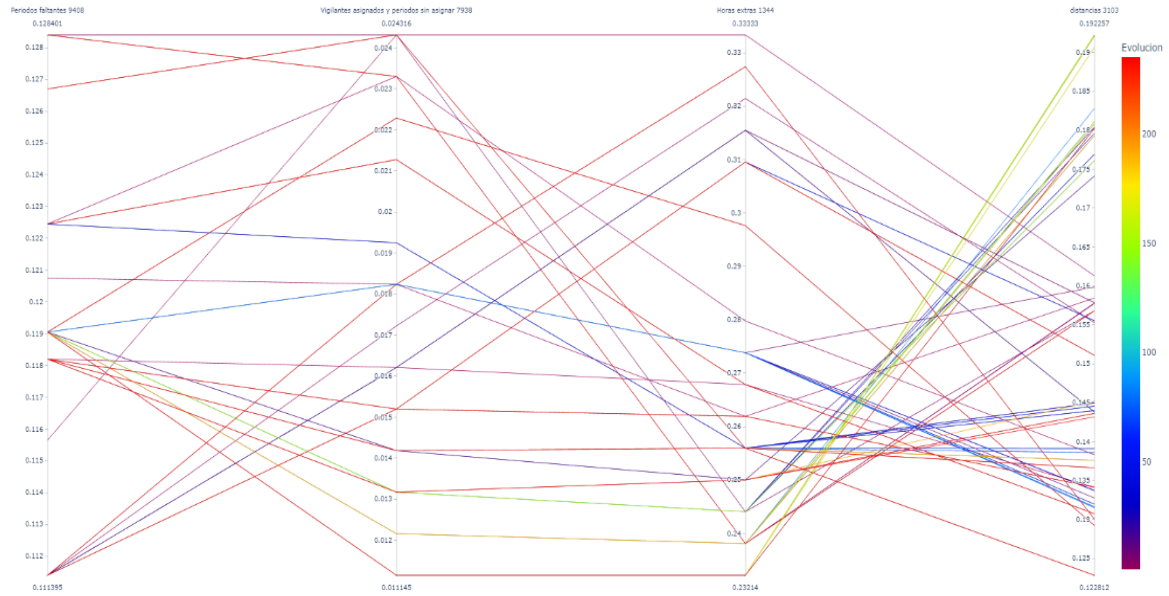


Figura 85. *Turnos flexibles MGRASP (caso 1).*

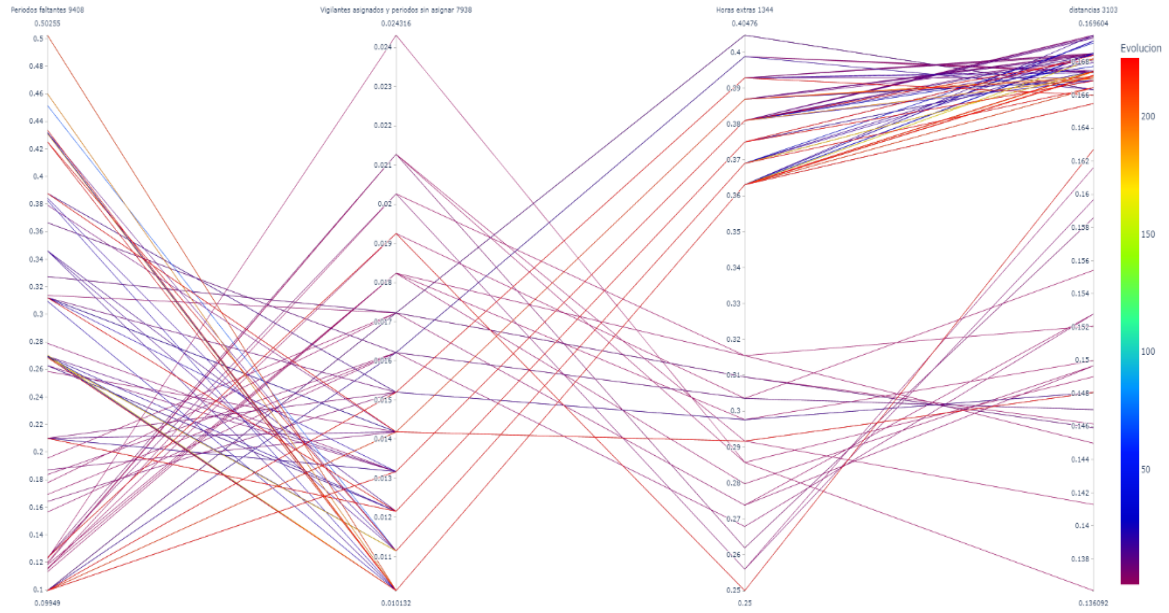


Figura 86. *Turnos flexibles NSGA-II (caso 1).*

En la **Tabla 24** se muestran los resultados encontrados por las dos metaheurísticas para cada objetivo, mayormente se tendrán soluciones muy buenas optimizando un objetivo, y fallando en los demás. Para este caso se tomará la mejor solución encontrada por cada una de las metaheurísticas según las necesidades de la empresa, es decir cada solución tiene sus ventajas y desventajas, por ejemplo, en la solución de MGRASP la distancia esta mejor optimizada que en la solución de NSGA-II, pero menos optimizada en el objetivo de turnos sin asignar, se puede decir, que una solución se clasifica como optima o no dependiendo de las

necesidades. Para este problema, se toma como mejor solución aquella que reporte mejor resultado en la minimización de los objetivos de turnos sin asignar, vigilantes asignados y distancias, para el objetivo de horas extras no se tendrá relevancia dado que la empresa manifiesta que le interesa tener más horas extras en sus vigilantes, siempre y cuando no sobrepase los límites legales establecidos.

Tabla 24. *Mejor solución MGRASP y NSGA-II (caso 1).*

Objetivos	Turnos sin asignar 9408 (100%)	Vigilantes asignados y periodos sin utilizar 7938 (100%)	Horas extras 1344 (100%)	Distancias 3103(100%)
MGRASP	0,111395 %	0,015 %	0,31 %	0,151 %
NSGA-II	0,0995 %	0,013 %	0,38 %	0,168 %

8.2.2 CASO ADAPTADO A TURNOS FIJOS DE LA EMPRESA SIN HORAS EXTRAS

Para este escenario, se adapta los algoritmos para hacer uso de los turnos que maneja la empresa de vigilancia, y se desactiva el objetivo de minimización de horas extras, con el fin de dar mayor posibilidades a las metaheuristicas de cubrir todos los turnos que maneja la emprea. Se debe mencionar, que en la **Figura 87** y **Figura 88**, no se debe tener en cuenta el objetivo de horas extras. En estas figuras se evidencia que el espacio de busqueda de los dos algoritmos tienen comportamientos similares, en la **Figura 87**, se pude mirar de forma sencilla como MGRASP no explora muy bien por el espacio de busqueda, quedando en optimos locales, debido a su naturaleza como algoritmo voraz y de su fase de busqueda entre vencidades cercanas. NSGA-II obtiene mejores resultados, y logra evolucionar de forma satisfactoria en todos sus objetivos, especialmente en el objetivo de minimizacion de turnos sin asignar, donde incluso encontro mejores resultados que MGRASP.

En la **Tabla 25** se muestra la mejor solución encontrada por cada una de las metaheurísticas del caso 2.

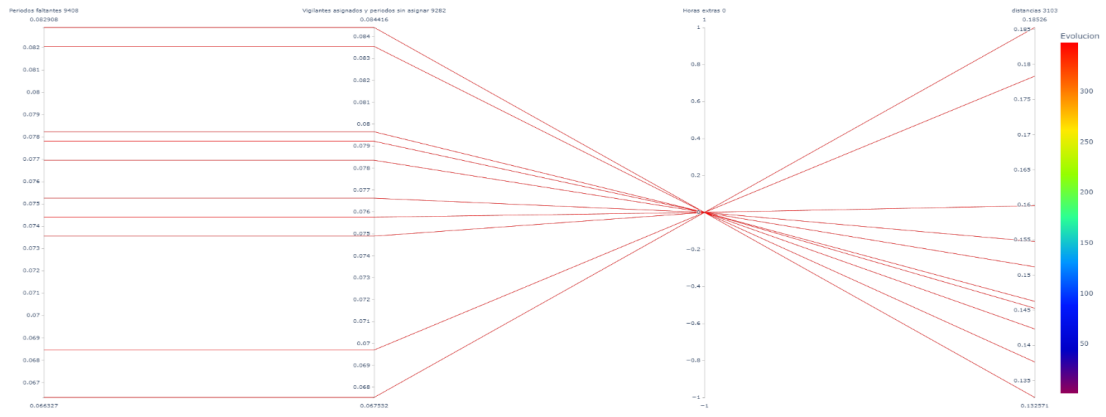


Figura 87. *MGRASP adaptado a turnos de la empresa (caso 2).*

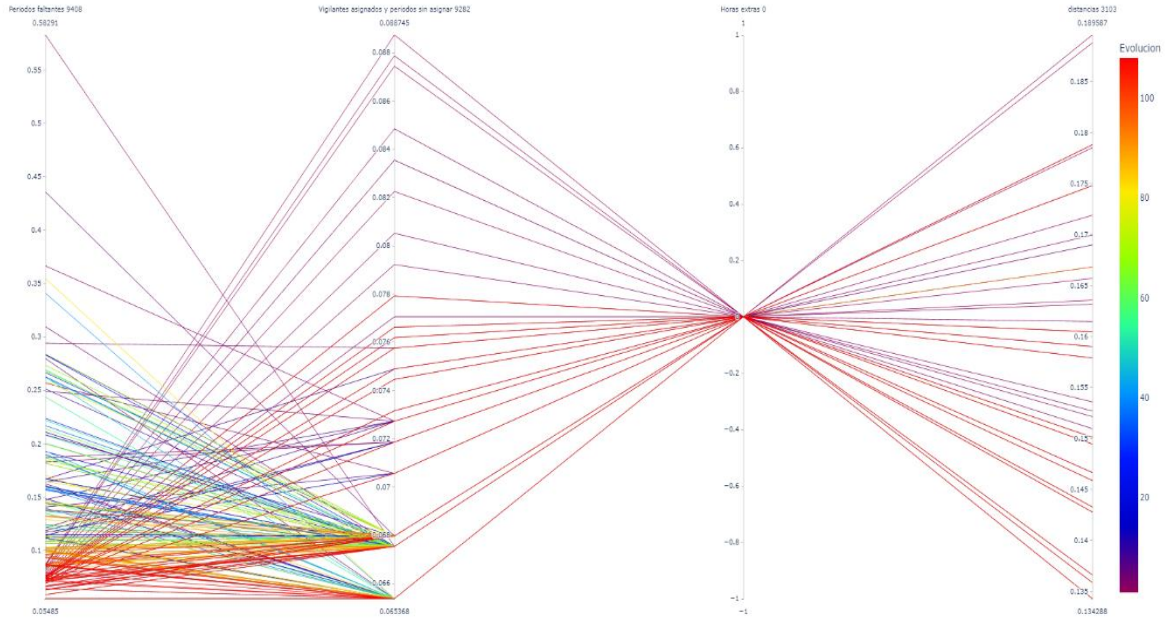


Figura 88. *NSGA-II adaptado a turnos de la empresa (caso 2).*

Tabla 25. *Mejor solución NSGA-II y MGRASP (caso 2).*

Objetivos	Turnos sin asignar 9408 (100%)	Vigilantes asignados y periodos sin utilizar 7938 (100%)	Horas extras 1344 (100%)	Distancias 3103(100%)
MGRASP	0,066327	0,067532	Vacio	0.143
NSGA-II	0,05485	0,065368	Vacio	0,161

8.2.3 CASO CON 45 VIGILANTES DISPONIBLES

Para este caso, no se hace ninguna adaptacion, es decir, se ejecutan las metaheuristicas con los turnos por defecto inicialmente planteados, y se mantiene activo el objetivo de horas extras, sin embargo, se incrementaran tres vigilantes al problema, con el fin de observar el comportamiento de las metahuristicas aquí planteadas, y ademas, validar que mejora el objetivo de minimizacion de turnos sin asignar. Para la **Figura 89**, se observa que MGRASP encuentra buenos resultados desde las primeras evoluciones, al igual que NSGA-II (ver **Figura 90**), sin embargo**,** no le va muy bien con el objetivo de minimizacion de distancia. De NSGA-II, se observa que explora muy bien todos los objetivos, excepto el objetivo de distancia donde queda atrapapo un tiempo en un optimo local (lineas moradas), pero logra salir en sus ultimas evoluciones, permitiendo encontrando resultados para este objetivo (lineas rojas).

En la **Tabla 26**, se muestra la mejor solución encontrada por cada una de las metaheurísticas del caso 3. Demostrando que las metaheurística NSGA-II, y MGRASP generan buenas soluciones y mejores resultados que los del caso 1, pero a pesar de esto, las metaheurísticas no logran asignar todos los turnos respectivos del cronograma de la empresa de seguridad. Demostrando por un lado que a las

metaheurística tiene limitantes al asignar horas extra en los vigilantes. Por otro lado, también demuestra la gran carga que tienen los vigilantes de la empresa de seguridad, dado que, para resolver el problema para este caso, mínimamente un vigilante tendría que trabajar aproximadamente 52 horas semanales si se quisiera cubrir todos los periodos ($24*7*14/45$).

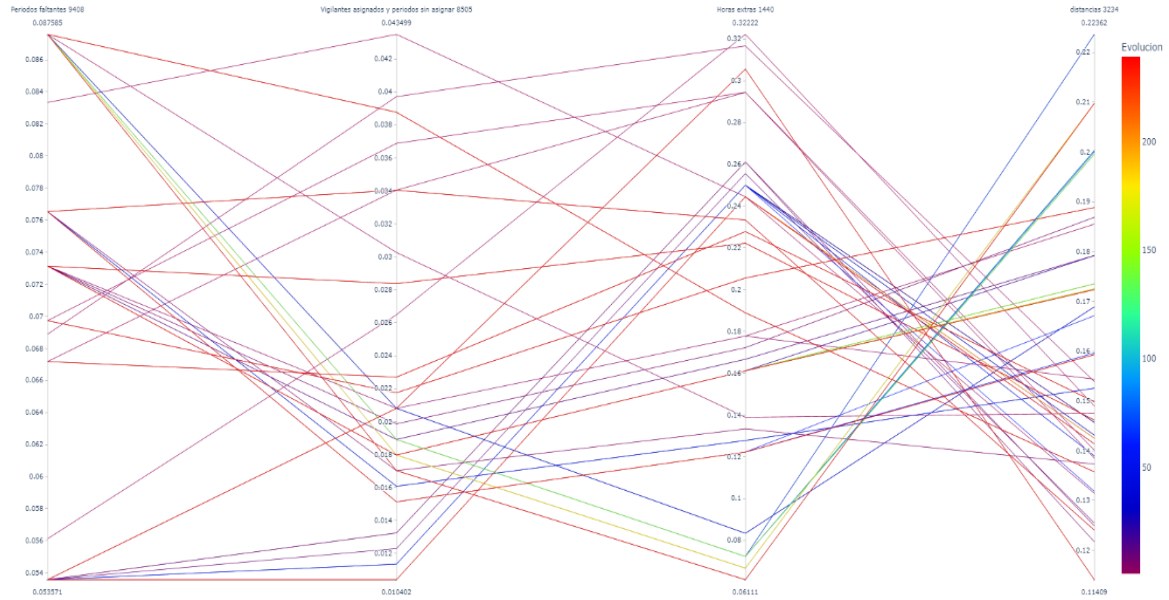


***Figura 89**. MGRASP sin objetivo de minimización de horas extras (caso 3).*

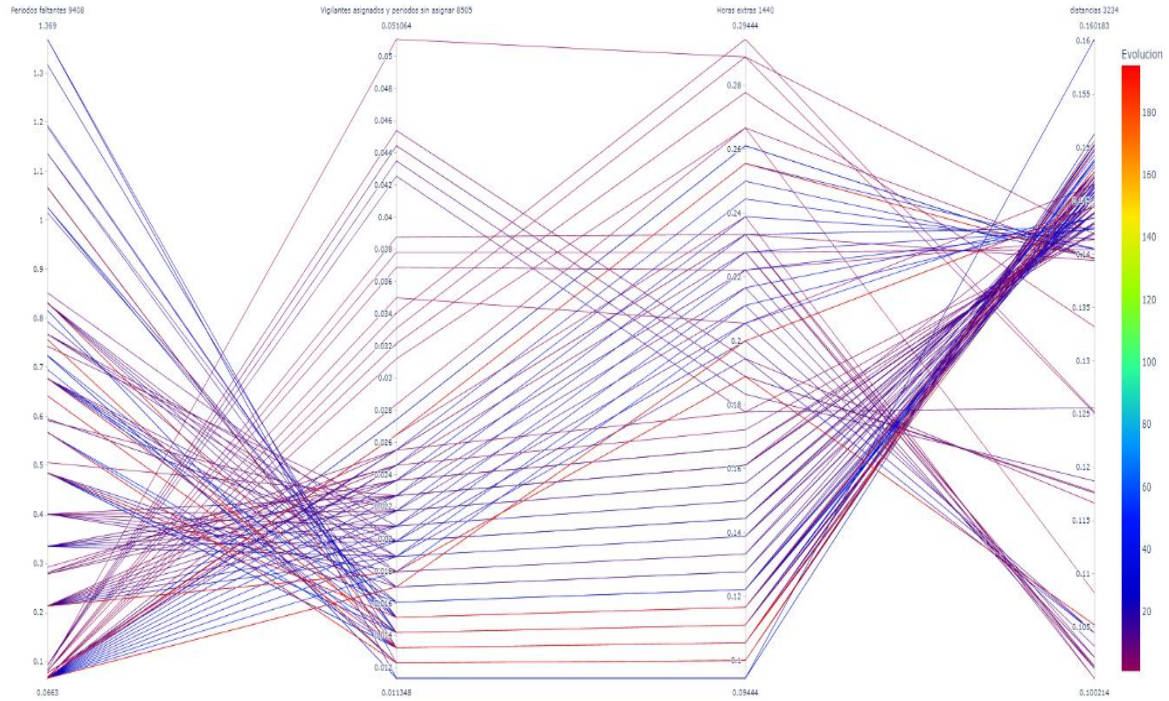


***Figura 90**. NSGA-II sin objetivo de minimización de horas extras (caso 3).*

Tabla 26. *Mejor solución NSGA-II y MGRASP (Caso 3).*

Objetivos	Turnos sin asignar 9408 (100%)	Vigilantes asignados y periodos sin utilizar 7938 (100%)	Horas extras 1344 (100%)	Distancias 3103(100%)
MGRASP	0,053571	0,010402	0.0245	0.15
NSGA-II	0,0663	0,016.5	0.19	0.106

En la **Tabla 27**, se muestra la comparativa entre los resultados de NSGA-II, MGRASP, y la empresa de vigilancia, tomando como base la mejor solución encontrada hasta el momento para cada metaheurística, que fueron las ejecutadas en el caso 2, en la cual se adaptan los turnos del programa a los usados por la empresa, motivo por el que en la **Tabla 27** no se tendrá en cuenta el objetivo de minimización de horas extras. En la primera fila se muestra el tiempo que toma resolver el problema, la empresa por lo general le toma 72 horas en hacer un cronograma para el mes, mientras que las metaheurísticas fueron ejecutadas durante una hora. Las demás filas comparan los resultados faltantes para cada objetivo a minimizar.

De NSGA-II se encuentra un margen de error del 0.05485 para el objetivo de minimización turnos sin asignar, 0.065368 para el objetivo de asignación de vigilantes, y 0.161 para el objetivo de minimización de distancia. Para MGRASP se tiene que la mejor solución encontrada tuvo un margen de error de 0,053571 para el objetivo de minimización de turnos sin asignar, 0,010402 para el objetivo de asignación de vigilantes, y 0.15 para el objetivo de minimización de la distancia. En la **Tabla 27**, los valores de las columnas 4 y 5 muestran los valores que faltan para optimizar los objetivos. Este valor sale de multiplicar el valor máximo esperado para un objetivo por el porcentaje mínimo encontrado en el objetivo de la solución en particular. Por ejemplo, para el objetivo de turnos sin asignar se espera como máximo 9408 periodos que equivales al total de periodos de la empresa es decir 14 sitios x 24 periodos x 7 días x 4 semanas, y este valor se multiplica por el valor mínimo encontrado, y se obtuvo un porcentaje de 0.05485 dando como resultado que faltan 516,0288 periodos por asignar para NSGA-II.

Tabla 27. *Comparación de mejores resultados.*

Numero	Variable	Empresa de seguridad	NSGA -II	MGRASP
1	Tiempo	72 horas	1 hora	1 hora
2	Turnos sin asignar(**9408**)	0	516,0288	503,995968
3	Vigilantes asignados y periodos sin utilizar (**7938**)	0	518,891184	82,571076
4	Distancias(**3103**)	225,4	499,583	465,45

Finalmente, a continuación, en la **Figura 91** se deja un ejemplo de uno de los cronogramas con mejores resultados obtenidos.

Vigilante	Horas por semana	Dia 1	Dia 2	Dia 3	Dia 4	Dia 5
1	[50, 56, 56, 54]	[5, [22, 5]]	[5, [22, 5]]	[5, [22, 5]]	[5, [22, 5]]	[5, [22, 5]]
2	[56, 56, 56, 48]	[12, [6, 13]]	[12, [6, 13]]	[12, [6, 13]]	[12, [6, 13]]	[12, [6, 13]]
3	[50, 56, 56, 55]	[14, [22, 5]]	[14, [22, 5]]	[14, [22, 5]]	[14, [22, 5]]	[14, [22, 5]]
4	[56, 56, 56, 48]	[1, [6, 13]]	[1, [6, 13]]	[1, [6, 13]]	[1, [6, 13]]	[1, [6, 13]]
5	[50, 56, 56, 55]	[4, [22, 5]]	[4, [22, 5]]	[4, [22, 5]]	[4, [22, 5]]	[4, [22, 5]]
6	[56, 56, 56, 49]	[8, [14, 21]]	[8, [14, 21]]	[8, [14, 21]]	[8, [14, 21]]	[8, [14, 21]]
7	[50, 56, 56, 54]	[1, [22, 5]]	[1, [22, 5]]	[1, [22, 5]]	[1, [22, 5]]	[1, [22, 5]]
8	[56, 56, 56, 49]	[11, [6, 13]]	[11, [6, 13]]	[11, [6, 13]]	[11, [6, 13]]	[11, [6, 13]]
9	[56, 56, 56, 48]	[1, [14, 21]]	[1, [14, 21]]	[1, [14, 21]]	[1, [14, 21]]	[1, [14, 21]]
10	[50, 56, 56, 55]	[12, [22, 5]]	[12, [22, 5]]	[12, [22, 5]]	[12, [22, 5]]	[12, [22, 5]]
11	[56, 56, 56, 49]	[12, [14, 21]]	[12, [14, 21]]	[12, [14, 21]]	[12, [14, 21]]	[12, [14, 21]]
12	[56, 56, 56, 49]	[6, [6, 13]]	[6, [6, 13]]	[6, [6, 13]]	[6, [6, 13]]	[6, [6, 13]]
13	[56, 56, 56, 48]	[3, [6, 13]]	[3, [6, 13]]	[3, [6, 13]]	[3, [6, 13]]	[3, [6, 13]]
14	[56, 56, 56, 48]	[10, [14, 21]]	[10, [14, 21]]	[10, [14, 21]]	[10, [14, 21]]	[10, [14, 21]]
15	[56, 56, 56, 49]	[3, [14, 21]]	[3, [14, 21]]	[3, [14, 21]]	[3, [14, 21]]	[3, [14, 21]]
16	[50, 56, 56, 54]	[2, [22, 5]]	[2, [22, 5]]	[2, [22, 5]]	[2, [22, 5]]	[2, [22, 5]]
17	[50, 56, 56, 54]	[3, [22, 5]]	[3, [22, 5]]	[3, [22, 5]]	[3, [22, 5]]	[3, [22, 5]]
18	[56, 56, 56, 48]	[7, [6, 13]]	[7, [6, 13]]	[7, [6, 13]]	[7, [6, 13]]	[7, [6, 13]]
19	[56, 56, 56, 49]	[4, [6, 13]]	[4, [6, 13]]	[4, [6, 13]]	[4, [6, 13]]	[4, [6, 13]]
20	[50, 56, 56, 54]	[6, [22, 5]]	[6, [22, 5]]	[6, [22, 5]]	[6, [22, 5]]	[6, [22, 5]]
21	[56, 56, 56, 48]	[14, [14, 21]]	[14, [14, 21]]	[14, [14, 21]]	[14, [14, 21]]	[14, [14, 21]]
22	[56, 56, 56, 48]	[2, [6, 13]]	[2, [6, 13]]	[2, [6, 13]]	[2, [6, 13]]	[2, [6, 13]]
23	[56, 56, 56, 48]	[2, [14, 21]]	[2, [14, 21]]	[2, [14, 21]]	[2, [14, 21]]	[2, [14, 21]]
24	[56, 56, 56, 48]	[6, [14, 21]]	[6, [14, 21]]	[6, [14, 21]]	[6, [14, 21]]	[6, [14, 21]]
25	[56, 56, 56, 49]	[8, [6, 13]]	[8, [6, 13]]	[8, [6, 13]]	[8, [6, 13]]	[8, [6, 13]]
26	[56, 56, 56, 49]	[9, [14, 21]]	[9, [14, 21]]	[9, [14, 21]]	[9, [14, 21]]	[9, [14, 21]]
27	[50, 56, 56, 55]	[7, [22, 5]]	[7, [22, 5]]	[7, [22, 5]]	[7, [22, 5]]	[7, [22, 5]]
28	[50, 56, 56, 55]	[11, [22, 5]]	[11, [22, 5]]	[11, [22, 5]]	[11, [22, 5]]	[11, [22, 5]]
29	[56, 56, 56, 49]	[9, [6, 13]]	[9, [6, 13]]	[9, [6, 13]]	[9, [6, 13]]	[9, [6, 13]]
30	[50, 56, 56, 55]	[8, [22, 5]]	[8, [22, 5]]	[8, [22, 5]]	[8, [22, 5]]	[8, [22, 5]]
31	[56, 56, 56, 48]	[5, [6, 13]]	[5, [6, 13]]	[5, [6, 13]]	[5, [6, 13]]	[5, [6, 13]]
32	[50, 56, 56, 54]	[9, [22, 5]]	[9, [22, 5]]	[9, [22, 5]]	[9, [22, 5]]	[9, [22, 5]]

Figura 91. *Cronograma generado por el aplicativo.*

Esta página ha sido dejada intencionalmente en blanco.

CAPÍTULO 9

9 CONCLUSIONES Y TRABAJO FUTURO

En el presente trabajo de grado se propuso adaptar dos algoritmos, uno basado en GRASP y otro en NSGA-II, para abordar el problema de programación del personal de seguridad y vigilancia para las empresas de seguridad basados en horarios flexibles, buscando minimizar simultáneamente cuatro objetivos (la cantidad de turnos no asignados, la cantidad de guardias de seguridad requeridos, las horas extras del personal de seguridad, y la distancia entre el hogar del guardia de seguridad y el puesto de trabajo). La solución propuesta además se adaptó a un caso real de una empresa de la ciudad de Popayán con horarios o turnos fijos.

Para abordar el problema de asignación de vigilantes se propuso una solución inicial que permita a las metaheurísticas partir de una solución factible que cumple con las restricciones duras del problema. La metaheurística GRASP se adaptó al problema multiobjetivo para que genere un conjunto de soluciones en el Frente de Pareto y que evolucionan en el proceso de búsqueda local. De NSGA-II, se crearon unos procesos de cruce adaptados al problema de asignación de vigilantes que ayudaron a encontrar soluciones factibles en el espacio de búsqueda.

Para ayudar a afinar los parámetros de las metaheurísticas propuestas (GRASP Multiobjetivo y NSGA-II adaptados) se implementó un algoritmo genético (GA). Este proceso de optimización ejecutó las metaheurísticas 30 veces sobre 3 problemas por cada solución (conjunto de parámetros) de manera que se cumpliera el teorema de limite central en los resultados obtenidos. Los resultados mostraron que a pesar de que el GA se ejecutó pocas generaciones los resultados mejoraban de una generación a otra. Los parámetros obtenidos fueron los usados en la fase de experimentación

Para evaluar los resultados de las metaheurísticas propuestas se crearon 6 casos sintéticos de prueba con diferentes niveles de dificultad con el fin de evaluar la factibilidad de las soluciones generadas por las metaheurísticas en diferentes horarios y circunstancias. Además, se evaluó los resultados generados por las metaheurísticas contra un cronograma real proporcionado por una empresa de seguridad y vigilancia.

De la evaluación realizada se puede concluir que las metaheurísticas se pueden adaptar a diferentes problemas de asignación y sus resultados dependerán de los recursos del personal disponible de la empresa de seguridad y vigilancia. Cuando los recursos son limitados los objetivos tienden a competir en demasía unos con otros evitando encontrar soluciones optimas, como sucedió con la información suministrada por la empresa de vigilancia dado que las metaheurísticas no lograron

cubrir en su totalidad los turnos para todos los sitios. También hay que añadir que en este momento la empresa de seguridad, y vigilancia presenta una sobrecarga de trabajo en sus vigilantes, haciendo que estos tengan que trabajar horas extras cada semana. Se debe mencionar, además, que la solución inicial está enfocada para resolver el problema de asignación de turnos flexibles y no en turnos fijos como lo maneja la empresa.

Para hacer el uso del programa más accesible se desarrolló una interfaz web que permite ingresar los datos necesarios para obtener los cronogramas que son generados por un servicio REST Api que ejecuta y procesa las metaheurísticas propuestas en esta investigación.

Como trabajo futuro se busca mejorar la creación de la solución inicial propuesta de manera que cuando se construya esta genere más variabilidad en los turnos asignados manteniendo la factibilidad. Además, se busca crear nuevos refinamientos que permitan lograr una mejor explotación del espacio de búsqueda. También agregar nuevos objetivos que ayuden a mejorar las condiciones de trabajo de los vigilantes como por ejemplo las preferencias de turnos, preferencias en las fechas de vacaciones, dominicales y festivos, en la cantidad de horas extras y ayude a las preferencias de la empresa como por ejemplo rotaciones entre diferentes sitios y asignación de turnos fijos para puestos de trabajo específicos.

CAPÍTULO 10

10 REFERENCIAS BIBLIOGRÁFICAS

[1] S. Lan, W. Fan, T. Liu, and S. Yang, “A hybrid SCA–VNS meta-heuristic based on Iterated Hungarian algorithm for physicians and medical staff scheduling problem in outpatient department of large hospitals with multiple branches,” *Applied Soft Computing Journal*, vol. 85, Dec. 2019, doi: 10.1016/j.asoc.2019.105813.

[2] E. H. Özder, E. Özcan, and T. Eren, “A Systematic Literature Review for Personnel Scheduling Problems,” *Int J Inf Technol Decis Mak*, vol. 19, no. 06, pp. 1695–1735, Nov. 2020, doi: 10.1142/S0219622020300050.

[3] Z. A. Abdalkareem, A. Amir, M. A. Al-Betar, P. Ekhan, and A. I. Hammouri, “Healthcare scheduling in optimization context: a review,” *Health Technol (Berl)*, vol. 11, no. 3, pp. 445–469, May 2021, doi: 10.1007/s12553-021-00547-5.

[4] H. K. Alfares and A. S. Alzahrani, “Optimum workforce scheduling for multiple security gates,” *INFOR: Information Systems and Operational Research*, vol. 58, no. 3, pp. 438–455, Jul. 2020, doi: 10.1080/03155986.2019.1629770.

[5] M. Cildoz, F. Mallor, and P. M. Mateo, “A GRASP-based algorithm for solving the emergency room physician scheduling problem,” *Appl Soft Comput*, vol. 103, May 2021, doi: 10.1016/j.asoc.2021.107151.

[6] M. Erhard, “Flexible staffing of physicians with column generation,” *Flex Serv Manuf J*, vol. 33, no. 1, pp. 212–252, Mar. 2021, doi: 10.1007/s10696-019-09353-8.

[7] C. Faycal, M. E. Riffi, and B. Ahiod, “HYBRID GENETIC ALGORITHM AND GREEDY RANDOMIZED ADAPTIVE SEARCH PROCEDURE FOR SOLVING A NURSE SCHEDULING PROBLEM,” *J Theor Appl Inf Technol*, vol. 73, no. 2, 2015, [Online]. Available: www.jatit.org

[8] H. Ishibuchi, H. Masuda, Y. Tanigaki, and Y. Nojima, “Modified Distance Calculation in Generational Distance and Inverted Generational Distance,” 2015, pp. 110–125. doi: 10.1007/978-3-319-15892-1_8.

[9] Z. Sinuany-Stern and Y. Teomi, “Multi-Objective Scheduling Plans for Security Guards,” 1986. [Online]. Available: www.jstor.org

[10] M. Ohki, “Effectiveness of NSGA-II with Linearly Scheduled Pareto-Partial Dominance for Practical Many-Objecitve Nurse Scheduling,” in *7th International Conference on Control, Decision and Information Technologies, CoDIT 2020*, Jun. 2020, pp. 581–586. doi: 10.1109/CoDIT49905.2020.9263847.

[11] A. Wibowo* and Y. Lianawati, “A Multi-objective Genetic Algorithm for Optimizing the Nurse scheduling Problem,” *International Journal of Recent Technology and Engineering (IJRTE)*, vol. 8, no. 3, pp. 5409–5414, Sep. 2019, doi: 10.35940/ijrte.C6204.098319.

[12] M. Hamid, R. Tavakkoli-Moghaddam, F. Golpaygani, and B. Vahedi-Nouri, “A multi-objective model for a nurse scheduling problem by emphasizing human factors,” *Proc Inst Mech Eng H*, vol. 234, no. 2, pp. 179–199, Feb. 2020, doi: 10.1177/0954411919889560.

[13] M. Cildoz, F. Mallor, and P. M. Mateo, “A GRASP-based algorithm for solving the emergency room physician scheduling problem,” *Appl Soft Comput*, vol. 103, p. 107151, May 2021, doi: 10.1016/j.asoc.2021.107151.

[14] D. H. Wolpert and W. G. Macready, “No free lunch theorems for optimization,” *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, Apr. 1997, doi: 10.1109/4235.585893.

[15] FREDERICK S. HILLIER and GERALD J. LIEBERMAN, “Introduction to operations research,” vol. 7th Edition, 1967.

[16] J. O. Brunner, J. F. Bard, and R. Kolisch, “Flexible shift scheduling of physicians,” *Health Care Manag Sci*, vol. 12, no. 3, pp. 285–305, Jul. 2009, doi: 10.1007/s10729-008-9095-2.

[17] S. Tan and Y. Luo, “Flexible Security Guard Scheduling to Satisfy Defensive Power by Tabu-Search Algorithm,” in *Proceedings of the 2nd International Conference on Vision, Image and Signal Processing*, Aug. 2018, pp. 1–5. doi: 10.1145/3271553.3271586.

[18] H. K. Alfares and A. S. Alzahrani, “Optimum workforce scheduling for multiple security gates,” *INFOR*, vol. 58, no. 3, pp. 438–455, 2020, doi: 10.1080/03155986.2019.1629770.

[19] R. C. Carrasco, “Long-term staff scheduling with regular temporal distribution,” *Comput Methods Programs Biomed*, vol. 100, no. 2, pp. 191–199, Nov. 2010, doi: 10.1016/j.cmpb.2010.03.015.

[20] A. Fügener and J. O. Brunner, “Planning for overtime: The value of shift extensions in physician scheduling,” *INFORMS J Comput*, vol. 31, no. 4, pp. 732–744, Sep. 2019, doi: 10.1287/IJOC.2018.0865.

[21] M. Ohki, “Many-Objective Nurse Scheduling using NSGA-II based on Pareto Partial Dominance with Linear Subset-size Scheduling,” in *Proceedings of the 10th International Joint Conference on Computational Intelligence*, 2018, pp. 118–125. doi: 10.5220/0006894501180125.

[22] S. Lan, W. Fan, T. Liu, and S. Yang, “A hybrid SCA–VNS meta-heuristic based on Iterated Hungarian algorithm for physicians and medical staff scheduling problem in outpatient department of large hospitals with multiple branches,” *Appl Soft Comput*, vol. 85, p. 105813, Dec. 2019, doi: 10.1016/j.asoc.2019.105813.

[23] A. A. Soukour, L. Devendeville, C. Lucet, and A. Moukrim, “Staff scheduling in airport security service,” in *IFAC Proceedings Volumes (IFAC-PapersOnline)*, 2012, vol. 14, no. PART 1, pp. 1413–1418. doi: 10.3182/20120523-3-RO-2023.00169.

[24] S. M. Al-Yakoob and H. D. Sherali, “Mixed-integer programming models for an employee scheduling problem with multiple shifts and work locations,” *Ann Oper Res*, vol. 155, no. 1, pp. 119–142, 2007, doi: 10.1007/s10479-007-0210-4.

[25] S. Liu, T. Zhang, P. Feng, Y. Zheng, and W. Chen, “Hierarchical Staffing Problem by Shift Design in Nursing Homes: A Two-stage Method,” in *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*, Aug. 2020, pp. 1013–1018. doi: 10.1109/CASE48305.2020.9216768.

[26] J. Y. Shiau, M. K. Huang, and C. Y. Huang, “A hybrid personnel scheduling model for staff rostering problems,” *Mathematics*, vol. 8, no. 10, pp. 1–20, Oct. 2020, doi: 10.3390/math8101702.

[27] F. Guerriero and R. Guido, “Modeling a flexible staff scheduling problem in the Era of Covid-19,” *Optim Lett*, vol. 16, no. 4, pp. 1259–1279, May 2022, doi: 10.1007/s11590-021-01776-3.

[28] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, “A fast and elitist multiobjective genetic algorithm: NSGA-II,” *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002, doi: 10.1109/4235.996017.

[29] C. Alberto Cobos Lozada, *Algoritmos Metaheurísticos para Optimización de Muchos Objetivos*, Primera Edicion. Popayán: 2019, 2019.

[30] C. Alberto Cobos Lozada, *Algoritmos Metaheurísticos para Optimización de Muchos Objetivos*, Primera Edicion. Popayán: 2019, 2019.

[31] S. K. Joshi and J. C. Bansal, “Parameter tuning for meta-heuristics,” *Knowl Based Syst*, vol. 189, p. 105094, 2020, doi: https://doi.org/10.1016/j.knosys.2019.105094.

[32] I. Negrin, D. Roose, E. Chagoyén, and K. U. Leuven, “PARAMETER TUNING STRATEGIES FOR METAHEURISTIC METHODS APPLIED TO DISCRETE OPTIMIZATION OF STRUCTURAL DESIGN.”

[33] J. H. Holland, “Adaptation in Natural and Artificial Systems,” 1975.