Universidad del Bío-Bío.
Facultad de Ingeniería.
Departamento de Ingeniería Industrial.

Profesor Guía: Luis Ceballos A.

UNIVERSIDAD DEL BÍO-BÍO

"ASIGNACION DE HORARIOS DE TRABAJO, RESOLUCIÓN MEDIANTE ALGORITMO DE BUSQUEDA TABÚ"

"TRABAJO DE TITULACION PRESENTADO EN CONFORMIDAD A LOS REQUISITOS PARA OBTENER EL TÍTULO DE MAGISTER EN INGENIERÍA INDUSTRIAL"

Autor: Lorena Margarita Martínez Parra

Concepción, Diciembre 2013.

University of Bío-Bío.
Faculty of Engineering.
Department of Industrial Engineering.

Teacher Guide: Luis Ceballos A.

UNIVERSIDAD DEL BÍO-BÍO

"ASSIGNMENT OF LABOR SCHEDULING, RESOLUTION BY TABU SEARCH ALGORITHM"

"DEGREE WORK SUBMITTED PURSUANT TO THE REQUIREMENTS FOR THE TITLE OF MASTER IN INDUSTRIAL ENGINEERING"

Author: Lorena Margarita Martínez Parra

Concepción, December 2013.

DEDICATORIA

A mis hijos, Javiera y Tomás, que son mi razón de existir y que día a día llenan mi vida de emociones...

A mis padres, mi mamá Adelaida (†Q.E.P.D.) y papá Hugo, por su esfuerzo, entrega y amor incondicional hacia mí.

A mi familia, en especial a mi madrina Ma. Elena y mi tío Pedro, por sus sabios consejos, palabras de aliento y apoyo constante.

A Francisco, por quererme y apoyarme en momentos importantes de mi vida.

AGRADECIMIENTOS

Inicialmente, agradezco a mi casa de estudios, por otorgarme el Premio Universidad del Bio-Bío Año 2002, a través del rector de la época Sr. Hilario Hernández, ya que sin esta beca, no hubiese podido cursar este postgrado.

Agradezco también a quien fuera profesor y director del programa de Magíster en Ingeniería Industrial, Sr. Felipe Baesler Abufarde, quien me aceptó para ingresar al programa, y agradecer actualmente al profesor y nuevo director del programa Sr. Francisco Ramis Layon, por su consideración a mi proceso de titulación.

Mis agradecimientos al personal administrativo de la Facultad de Ingeniería Industrial, en especial a quienes fueran secretarias del programa de magíster, Sras. Anita Gallardo e Ingrid Troncoso, y actualmente Sra. Gladys Hernández, ya que ellas dan soporte a la relación alumno-profesor, y porque siempre me prestaron su ayuda ante cualquier contingencia.

Agradezco también al Sr. Fernando Mella Clark, por toda su asesoría profesional y técnica en el desarrollo práctico de este proyecto; sin su constante ayuda, este trabajo hubiese sido más complejo y difícil de realizar.

Por último, agradecer al cuerpo docente del programa por su gran calidad profesional, en especial a mi profesor guía Sr. Luis Ceballos Araneda, quien además de ser un gran académico, destaca por su calidad humana y sencillez; y porque constantemente me brindó palabras de aliento para que este trabajo llegara a su término.

RESUMEN

El objetivo de este trabajo es desarrollar un algoritmo de búsqueda tabú para dar solución al problema de asignación de horarios de trabajo de tal forma de poder cubrir los turnos y las necesidades en el mes del personal que se desempeña en el cargo de Mucama en una empresa del rubro entretención, hotelería y turismo.

Para ello, el trabajo se iniciará con la recopilación de información para el estudio, que incluye datos particulares del cargo Mucamas en la unidad: dotación según tipo de contrato, turnos a cubrir según horarios de demanda, cantidad de citaciones diarias de colaboradores por turno (definido como necesidad operativa), las restricciones operativas, legales y organizacionales de la compañía en el proceso de programación de turnos, entre otros aspectos relevantes. Dicha información será facilitada por el área de Planificación de Turnos de la compañía.

Luego, se realizará un estado del arte respecto al problema en cuestión y sus métodos de solución disponibles en la bibliografía existente.

Seguidamente, se elaborará el marco teórico del estudio donde se explique en detalle cómo opera el algoritmo de búsqueda tabú.

Posteriormente, se modelará el problema descrito preliminarmente, identificando la situación problema relativa al cargo Mucama y todas las restricciones asociadas; esto permitirá desarrollar el algoritmo a utilizar, detallando sus características y operatividad, el cual se programará en un ambiente computacional en lenguaje C#.

Finalmente, se analizarán los resultados obtenidos de estudios experimentales y se comparará con el actual método de la empresa para obtener las conclusiones y recomendaciones relevantes del estudio.

SUMMARY

The aim of this work is to develop a tabu search algorithm to solve the problem of assigning work schedules so as to meet the needs and shifts in the month of the personnel working in the office of Maid in a company the entertainment, hospitality and tourism field.

To do this, work will begin with the collection of information for the study, including specifics of the charge Maids in the unit: allocation by type of contract , to cover shifts as demand schedules, number of daily partners citations per shift (defined as operational need), operational, legal and organizational constraints of the company in the process of scheduling shifts, among other relevant aspects. Such information shall be provided by the Planning Shifts of the company.

Then, a state of the art regarding the problem in question and their solution methods available in the literature is performed.

Next, the theoretical framework of the study which explains in detail how to operate the tabu search algorithm is developed.

Subsequently the problem described preliminarily modeled, identifying the problem situation on the Maid charge and all associated restrictions; this will develop the algorithm to use, detailing its features and functionality, which will be scheduled in a computational environment in C # language.

Finally, the results of experimental studies will be discussed and compared with the existing method of the company to obtain the relevant conclusions and recommendations of the study.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE ECUACIONES

ÍNDICE DE GRÁFICOS

CAPÍTULO 1: INTRODUCCIÓN.

1.1. INTRODUCCIÓN

Informalmente, optimizar significa algo más que mejorar; sin embargo, en el contexto científico la optimización es el proceso de tratar de encontrar la mejor solución posible para un determinado problema.

El presente proyecto de título nace en tratar de optimizar el proceso de programación de turnos del personal en la compañía en la cual trabajo. Actualmente, me desempeño como Planning Controller en una empresa de servicios de entretención, hotelería y turismo[1], que funciona durante todo el año, los siete días de la semana y donde su operación principal se desarrolla en horario nocturno (a partir de las 22:00 hrs.).

El equipo de Planificación de Turnos, se compone de 6 planificadores. El objetivo principal del área es la confección de los turnos para 1000 colaboradores aproximadamente, cumpliendo con 3 objetivos básicos en la programación:

a) que la necesidad operativa de recursos humanos esté cubierta, es decir, si se necesitan "x" cantidad de colaboradores en determinados horarios, que efectivamente se encuentren con turno programado para cubrir la operación del negocio, de acuerdo a la demanda estimada de clientes, considerando también en paralelo que muchos cargos requieren que los colaboradores tengan habilidades y capacidades específicas para las tareas.

b) que se respete la normativa laboral en la programación de turnos, a fin de evitar exponer a la empresa ante eventuales multas ante una fiscalización de la Inspección del Trabajo, esto es, respetar el máximo de días corridos trabajados en la

[1] Por razones de confidencialidad de la información, no se puede proporcionar el nombre de la compañía.

programación, las horas de descanso entre turnos, el máximo de jornada semanal, entre otros aspectos legales.

c) que la programación de turnos sea equitativa para los colaboradores, esto es, que tengan la misma cantidad de turnos programados según su tipo de jornada (45 ó 30 horas semanales), la misma cantidad de días libres y la misma cantidad de turnos de mañana, tarde y noche durante cada mes.

Derivado de lo anterior, es típico dividir el trabajo diario en jornadas de mañana, tarde y noche y hacer una asignación de las jornadas de trabajo siguiendo un proceso rotativo por semanas.

Actualmente, este proceso se realiza de forma manual mediante una aplicación Excel que cuenta con validadores legales para programar (levanta alertas si no se cumple con la normativa legal), que contabiliza la cantidad de turnos programados por colaborador, que indica en porcentaje la carga de distribución de turnos en horario matutino, vespertino y nocturno para verificar la equidad de los turnos y que contrasta la necesidad operativa v/s la cantidad de colaboradores por turno a fin de chequear y validar que todas las posiciones requeridas para la operación diaria estén cubiertas.

Sabiendo que este problema real es factible de mejorar y buscar una mejor solución para el mismo mediante un método de optimización, a propuesta del docente Luis Ceballos, Director del Programa de Magíster en Ingeniería Industrial de la Universidad del Bío-Bío, el trabajo consistirá en desarrollar una heurística o metaheurística y compararla con trabajos similares ya realizados a fin de poder agregarle valor a la aplicación propuesta.

1.2. DESCRIPCIÓN DE LA EMPRESA

1.2.1. Identificación de la empresa

- Nombre de fantasía, Razón Social y Domicilio

Por razones de confidencialidad de la información, no se puede proporcionar identidad de la compañía.

- Rubro o giro de actividades

Casino de Juegos, Hotelería, Restaurantes, Centro de Entretención.

- Productos

Servicios de Entretención (Juegos y Espectáculos), Restoración (restaurantes y bares), Hotelería y Turismo (actividades outdoor).

- Clientes directos

Público en general, en sus diferentes categorías.

1.2.2. Breve reseña histórica

La compañía es una empresa de tipo familiar, que ha pasado de ser un operador de casinos, a ser pioneros en el modelo de entretención integral y a estar hoy considerado como la más prestigiosa e importante cadena del país y un referente en la industria de casinos a nivel latinoamericano por su exitoso modelo de negocio, su extraordinaria infraestructura y alta calidad de gestión.

La cadena hoy cuenta con una diversificada extensión geográfica con posiciones en el norte, centro y sur de Chile. En la actualidad opera 8 casinos y 5 hoteles. En los casinos existen cerca de 5.000 máquinas de azar, 300 mesas de juego y 1.100 posiciones de bingo. Además, tiene 24 restaurantes, 15 bares, 5 centros de convenciones con capacidad para más de 8.000 personas, discotheques, spa, business centers, salas de espectáculos, centros de entretención para niños y operador turístico que ofrece servicios de turismo en casi todas las unidades. Es decir, una variada oferta en un espacio único con infraestructura de primer nivel internacional, para ofrecer a los clientes disfrutar de experiencias inolvidables en un contexto de entretención con estilo.

1.2.3. Descripción de las principales unidades, funciones y recursos humanos

Cada uno de los servicios que la empresa entrega, son agrupados según sus características en alguna de las siguientes cuatro grandes áreas de negocios:

* CASINO: Compuesto por Mesas de Juegos, Máquinas Tragamonedas y Bingo, el casino es el principal lugar de encuentro de los clientes y es el *core* del negocio. En él se realizan los torneos, sorteos y promociones.

* AA&BB (Alimentos y Bebidas): Se compone de Restaurantes, Bares, Centro de Eventos, Centro de Entretención para niños y Discoteque. Es una gama de servicios que tienen un estilo para cada ocasión, en ambientes únicos para brindar entretención y placer.

* HOTEL: Habitaciones, SPA, Actividades Outdoor, son los servicios entregados por Hotelería y que brindan a los clientes descanso, elegancia y comodidad.

* APOYO: Compuesto por las áreas de Mantención, Personas, Comercial, Gerencia General y Tesorería Operativa, son claves para el resultado de las tres áreas antes mencionadas ya que entregan el soporte de mantención a toda la unidad.

La unidad está compuesta por un total de 1443 colaboradores, que se distribuyen dentro de las 4 unidades de negocios dependiendo de sus cargos, divisiones, subdivisiones y funciones, de la siguiente forma:

TABLA 1-1: DISTRIBUCIÓN DE RECURSOS HUMANOS, UNIDAD DE NEGOCIO

Unidad de Negocio	Total Colaboradores
AA&BB	560
APOYO	523
CASINO	257
HOTEL	103
Total Unidad de Negocio	**1443**

1.2.4. Estructura organizacional

La estructura organizacional de la empresa es de tipo funcional, y está liderada por un Gerente General y 6 Subgerencias de Área que reportan directamente al Gerente General y que corresponden a las gerencias de Juegos, Operaciones, Comercial, Mantención y Logística, Personas y Servicio. En conjunto, definen los lineamientos estratégicos de la organización.

Globalmente, la organización se estructura de la siguiente manera:

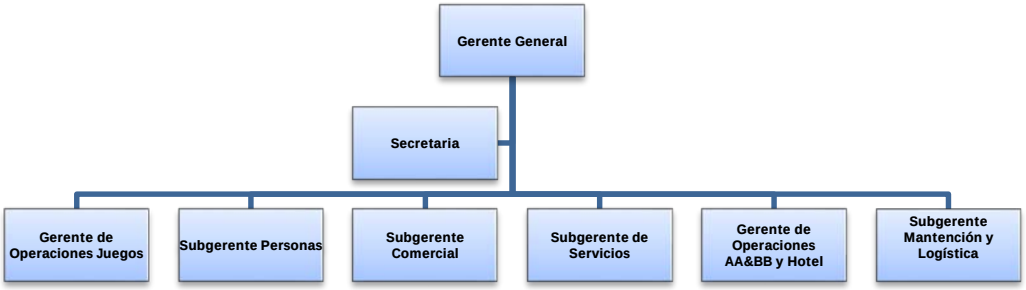


FIGURA 1.1: ORGANIGRAMA, UNIDAD DE NEGOCIO

CAPÍTULO 2: DESCRIPCIÓN DEL PROBLEMA Y ANTECEDENTES GENERALES DEL PROYECTO DE TÍTULO.

2.1. SITUACIÓN ACTUAL

La unidad de negocio tiene cerca de 1.200 colaboradores que trabajan bajo sistema rotativo de turnos, donde la programación de turnos del personal es realizada por un equipo de Planificación de Turnos compuesto de 6 planificadores y donde cada planificador genera programaciones para 200 colaboradores en promedio (una carga de trabajo considerable para cada planificador).

La programación de turnos del personal se genera de forma manual mediante una aplicación en programa Excel, donde el horizonte de confección de los turnos toma 1 mes calendario y donde se distinguen las siguientes actividades:

a) Días 1 al 6 de cada mes: Envío de parte de jefaturas de las restricciones personales de los colaboradores y necesidad operativa de su área.

Por *restricciones personales* se entenderán las solicitudes de días libres específicos en la programación de turnos de los colaboradores para que atiendan sus temas familiares o particulares (por ejemplo, cumpleaños de hijos, horas médicas), u horarios específicos de turnos en determinadas jornadas (sólo mañana, tarde o noche), de acuerdo a condiciones operativas (expertise de los colaboradores en una función determinada) o laborales (colaboradoras embarazadas, por ejemplo).

Necesidad operativa es la asignación de dotación (colaboradores) en determinados horarios de trabajo, para operar el negocio de forma efectiva. Esta definición de necesidades la realiza cada jefatura indicando cómo operará su área o punto de venta para el mes determinado. Cabe señalar que la necesidad operacional, no es un valor constante en el tiempo, es decir, puede cambiar de

acuerdo a los requerimientos de la operación o el comportamiento de demanda de los clientes, por lo cual debe ser constantemente evaluada y revisada por cada jefatura e informada al equipo de Planificación oportunamente.

Un ejemplo de necesidad operativa se visualiza en la siguiente figura:

Necesidad	7:00	7:30	8:00	8:30	9:00	9:30	10:00	10:30	11:00	11:30	12:00	12:30	13:00	13:30	14:00	14:30	15:00	15:30	16:00	16:30	17:00	17:30	18:00	18:30
Lun	1	1	2	2	3	3	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	1
Mar	1	1	2	2	3	3	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	1
Mie	1	1	2	2	3	3	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	1
Jue	1	1	2	2	3	3	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	1
Vie	1	1	1	1	2	2	8	8	8	8	8	8	8	8	8	6	4	4	6	6	6	6	6	2
Sab	1	1	1	1	2	2	8	8	8	8	8	8	8	8	8	6	4	4	6	6	6	6	6	2
Dom	1	1	1	1	2	2	6	6	6	6	8	8	8	8	8	8	2	2	2	2	5	5	5	5

Necesidad	19:00	19:30	20:00	20:30	21:00	21:30	22:00	22:30	23:00	23:30	0:00	0:30	1:00	1:30	2:00	2:30	3:00	3:30	4:00	4:30	5:00	5:30	6:00	6:30
Lun	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Mar	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Mie	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Jue	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Vie	2	2	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Sab	2	2	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Dom	3	3	2	2	2	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

FIGURA 2.1: NECESIDAD OPERATIVA, CARGO MUCAMA, UNIDAD DE NEGOCIO

b) Días 6 al 16 de cada mes: creación de propuestas de programación de turnos por el planificador.

Este proceso, es el que más consume tiempo en el trabajo de cada planificador, ya que la necesidad operativa que entrega cada jefatura debe ser transformada a citaciones y turnos "óptimos" que posteriormente se asignarán a los colaboradores.

La siguiente tabla ilustrativa (figura 2.2), muestra la cantidad de citaciones para el día simulado, la duración de los turnos (hora de inicio y término, respetando restricciones) para cada citación, horario asignado a colación (30 ó 60 minutos, marcados en color celeste) y los períodos de déficit (marcados en rojo) o exceso de dotación (marcado en amarillo si es adecuada, y en color verde si es demasiado).

Domingo

Citaciones	7:00	7:30	8:00	8:30	9:00	9:30	10:00	10:30	11:00	11:30	12:00	12:30	13:00	13:30	14:00	14:30
Citacion 1	1	1	1	c	c	1	1	1	1	1	1	1	1	1	1	1
Citacion 2				1	c	c	1	1	1	1	1	1	1	1	1	1
Citacion 3					1	1	1	1	1	1	1	1	1	1	1	1
Citacion 4					1	1	1	1	1	1	1	1	1	1	1	1
Citacion 5							1	1	1	1	1	1	1	1	1	1
Citacion 6							1	1	1	1	1	1	1	1	1	1
Citacion 7											1	1	1	1	1	1
Citacion 8											1	1	1	1	1	1
Citacion 9																
Citacion 10																
Necesidad Total	1	1	1	1	2	2	6	6	6	6	8	8	8	8	8	8
Total Posiciones	1	1	1	1	2	3	6	6	6	6	8	8	8	8	8	8
Diferencia	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

Citaciones	15:00	15:30	16:00	16:30	17:00	17:30	18:00	18:30	19:00	19:30	20:00	20:30	21:00	21:30	22:00	22:30
Citacion 1	1															
Citacion 2	1	1	1	1												
Citacion 3	c	c	1	1	1											
Citacion 4	c	c	1	1	1											
Citacion 5	1	c	c	1	1	1	1									
Citacion 6	1	1	c	c	1	1	1									
Citacion 7	1	1	c	c	1	1	1	1	1	1	1					
Citacion 8	c	c	1	1	1	1	1	1	1	1	1					
Citacion 9								1	1	1	1	1	1	1	1	1
Citacion 10																1
Necesidad Total	2	2	2	2	5	5	5	5	3	3	2	2	2	2	1	1
Total Posiciones	5	3	4	5	6	4	4	3	3	3	3	1	1	1	1	2
Diferencia	3	1	2	3	1	-1	-1	-2	0	0	1	-1	-1	-1	0	1

Citaciones	23:00	23:30	0:00	0:30	1:00	1:30	2:00	2:30	3:00	3:30	4:00	4:30	5:00	5:30	6:00	6:30	Hrs semana	Turnos
Citacion 1																	7,5	07:00 a 15:30
Citacion 2																	7,5	08:30 a 17:00
Citacion 3																	7,5	09:00 a 17::30
Citacion 4																	7,5	09:00 a 17::30
Citacion 5																	7,5	10:00 a 18:30
Citacion 6																	7,5	10:00 a 18:30
Citacion 7																	7,5	12:00 a 20:30
Citacion 8																	7,5	12:00 a 20:30
Citacion 9	1	1	c	c	1	1	1	1									7,5	18:30 a 03.00
Citacion 10	1	1	1	1	c	c	1	1	1	1	1	1	1	1	1	1	7,5	22:30 a 07:00
Necesidad Total	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	70	7x
Total Posiciones	2	2	1	1	1	1	2	2	1	1	1	1	1	1	1	1	75	
Diferencia	1	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0		

FIGURA 2.2: SIMULACIÓN DE CITACIONES Y TURNOS, DÍA DOMINGO, CARGO MUCAMA, UNIDAD DE NEGOCIO

Después de contar con todas las citaciones requeridas, para cubrir la necesidad operativa de manera óptima, se procede a la programación de los turnos. Esta consiste en asignar a los colaboradores con jornadas de 20, 30 y 45 horas semanales, las citaciones obtenidas de la simulación, con el objeto de asignar de manera eficiente y equitativa las citaciones, a los contratos existentes en la unidad.

En este proceso de asignación, de acuerdo a la legislación laboral vigente, existen también diferentes *restricciones legales* para los diferentes tipos de jornada antes mencionadas, que se deben cumplir en la programación. Resumidamente:

- No se pueden programar más de las horas establecidas por contrato (20, 30 ó 45 horas semanales, según corresponda).
- El período mínimo de descanso entre citaciones, es de 12 horas.
- El período de descanso entre una citación con día libre, seguido de una nueva citación, es de 30 horas.
- Los días laborales son considerados de manera correlativa, es decir, de 00:00 a 24:00, por lo que si una citación se inicia un día y termina al siguiente, deberá ser considerado como 2 días de trabajo.
- Como máximo cada colaborador puede trabajar 6 días seguidos (incluyendo la contabilización de días entre semanas distintas).
- Para las jornadas de 30 y 20 horas semanales, como máximo pueden trabajar 4 días por semana.
- Para la jornada de 45 horas semanales, su jornada se puede distribuir en un mínimo de 5 días y un máximo de 6 días por semana.
- Para las jornadas de 45 horas semanales, se deben tener 2 domingos libres al mes como mínimo (para los contratos de 20 y 30 horas con un máximo de 4 días trabajados en una semana, no es obligación entregar 2 domingos en el mes).
- Los turnos pueden durar un mínimo de 4 horas y un máximo de 10 horas ordinarias diarias.

En resumen, el planificador debe velar porque la programación cumpla:

- Con la cobertura de la operación
- Con la normativa legal
- Con las restricciones personales de los colaboradores (en lo posible, no es obligatorio)
- Con la equidad de los turnos entre colaboradores, esto es, diseñar mallas de turnos homogéneas que tengan la misma cantidad de turnos programados según su tipo de jornada (45 ó 30 horas semanales), la misma cantidad de días libres y la misma cantidad de turnos de mañana, tarde y noche durante cada mes. Este último aspecto, es difícil de cumplir.

c) Días 18 al 22: envío de propuesta a jefaturas, validación y correcciones de jefaturas.

Finalizada la programación de turnos de los colaboradores, se remite a cada jefatura para validarla, realizar correcciones y/o modificaciones, de acuerdo a nuevos antecedentes o situaciones que no se hayan considerado en la programación preliminar.

d) Días 23 y 24: Carga de turnos en sistema y publicación de turnos a los colaboradores de la unidad.

Este plazo es impostergable, ya que legalmente los turnos se deben disponibilizar a los colaboradores con 1 semana de anticipación, de lo contrario, la empresa se expone a multas de parte de la Inspección del Trabajo ante una eventual fiscalización.

La empresa utiliza sistema SAP para la carga y publicación de turnos.

e) Días 25 y siguientes: ajustes y cambios de turno (previo al inicio del mes de aplicación de la programación y durante su ejecución).

Posterior a la publicación, siempre ocurren situaciones imprevistas (por ejemplo, licencia de un colaborador) que alteran la programación inicial y la cual debe modificarse a último momento.

2.2. PROBLEMAS DETECTADOS.

- La programación de turnos es realizada de forma manual mediante una herramienta básica (aplicación Excel). Si bien existe un procedimiento de tiempos asociados (fechas) al proceso de Planificación, no se dispone de un método que permita agilizar el proceso, optimizarlo y que sirva de apoyo a la toma de decisiones.

- Las programaciones con un mes de anticipación, no garantizan posteriormente que se ajusten a la realidad del mes para el cual fueron diseñadas. Como resultado, las variaciones del entorno y la demanda de los clientes, han generado un crecimiento importante de cambios de turno de los colaboradores, ya que la programación se cambia diariamente, según las urgencias y necesidades del momento, lo que comporta numerosos errores de previsión (al atender la urgencia se descuida lo importante). De hecho, el grado de cumplimiento de la programación inicial, es alrededor de un 85%; el 15% restante corresponde a modificaciones que se generan durante el mes de aplicación de dicha programación. Esto, además de generar un desorden operativo en el negocio, genera descontento en los colaboradores que ven alterada su vida personal por estar afectos a constantes cambios de turno en su trabajo.

- El tiempo que consumen las programaciones de turno en su confección, es muy extenso (30 días).

- Las programaciones de turnos actuales no garantizan la equidad de turnos entre los colaboradores, generando un descontento en los mismos y perjudicando el clima organizacional.

2.3. JUSTIFICACIÓN PARA ABORDAR EL PROBLEMA.

La compañía necesita optimizar en tiempo y recursos el proceso de programación de turnos de la unidad, ya que la herramienta actual con la que cuenta es muy básica (aplicación en Excel), y no permite hacer gestión del personal en un horizonte de mediano y largo plazo relacionados a minimizar los gastos de personal y costos de oportunidad esperados.

El proceso actual de programación de turnos es un proceso que se realiza manualmente, con un mes de anticipación, y si bien incorpora restricciones de tipo operativo, legal y equidad de turnos, sólo es aplicable para el mes al cual corresponde y no garantiza que se ajuste a la dinámica de la demanda de clientes, ya que es de tipo reactivo (cualquier desajuste en la programación se corrige mediante cambios de turno).

Es por ello, que el desarrollo que se pretende realizar (heurística o metaheurística), permitirá agilizar el proceso de programación de turnos y contar con una herramienta más robusta en resultados, ejecución y con mayor aporte de valor al negocio.

2.4. OBJETIVO GENERAL.

Asignación de turnos (horarios de trabajo) del personal, mediante algoritmo de Búsqueda Tabú.

2.5. OBJETIVOS ESPECÍFICOS.

1. Modelar el problema de asignación de horarios de trabajo, incluyendo criterios, restricciones y todas las variables asociadas en su diseño.

2. Elaborar un algoritmo de búsqueda tabú que dé solución al problema en estudio y que permita disminuir los tiempos actuales que utiliza la empresa en este proceso.

3. Analizar los resultados obtenidos con la herramienta actual de la empresa.

2.6. ÁMBITO DEL ESTUDIO.

El presente trabajo, considera las siguientes limitaciones en su desarrollo:

-El trabajo será aplicado en la programación de turnos de 1 cargo crítico en el modelo de servicio de la compañía: Mucama.

-Se elegirá 1 mes de programación de turnos para dicho cargo.

-Se considerarán las restricciones operativas, legales y organizacionales de la compañía en el proceso de programación de turnos.

-Se considerarán los datos particulares del cargo Mucama en la unidad: dotación según tipo de contrato, turnos a cubrir según horarios de demanda y cantidad de citaciones diarias de colaboradores por turno.

-El método y tiempo real que utiliza la empresa en el proceso de programación de turnos del cargo Mucama.

-La elección de la función objetivo es resolver la necesidad que tienen el casino en su dotación de personal, de forma que la asignación para un turno no le falte personal, pero tampoco que le sobre.

-La utilización de algoritmos de búsqueda tabú para dar solución al problema particular en cuestión.

2.7. ALCANCE.

El presente proyecto de título, es un estudio exploratorio de los diferentes métodos de resolución que existen al problema de asignación de horarios de trabajo del personal; por ende, no existe la necesidad de formular una hipótesis al problema descrito.

A partir de este estudio exploratorio, se analizará el método propuesto y se planteará la línea de investigación futura.

En este estudio no se pretende entregar un sistema de asignación y/o programa definitivo, dado que este es sólo el comienzo de varias posibilidades de estudio en la cual se puede mejorar la solución propuesta al problema.

CAPÍTULO 3: ESTADO DEL ARTE.

3.1. INTRODUCCIÓN A LOS PROBLEMAS MULTIOBJETIVO Y MÉTODOS DE RESOLUCIÓN

La planificación eficiente de horarios de trabajo es fundamental para mejorar la productividad en las empresas de servicios y minimizar los gastos de personal. Este problema es conocido tradicionalmente como *labor scheduling problem,* problema "difícil de resolver" que queda reflejado en el término científico y clasificado como tipo NP-hard, concepto utilizado en el contexto de complejidad algorítmica[2] y de gran tamaño, por lo que su resolución tradicionalmente se realiza mediante métodos heurísticos tradicionales para encontrar soluciones en tiempos computacionales razonables.

"*Labor scheduling* –el proceso que iguala durante las operaciones del día el número de empleados trabajando, con el número de trabajadores que se necesitan para proveer el nivel deseado de servicio al cliente– es frecuentemente un gran determinante de la eficiencia en la organización del servicio". (Thompson, 1995).

Complementando la definición anterior, muchas empresas de servicios tienen demandas que varían significativamente de una hora a otra y de un día a otro. Si dichas empresas carecen de capacidad para satisfacer la demanda cuando esta se presenta, pueden incurrir en pérdidas o aumentar sus costos (imagen, pérdida de clientes). De igual forma, un exceso de la capacidad de oferta sobre la demanda, puede conducir a un despilfarro de recursos.

[2]Un algoritmo es un conjunto ordenado y finito de operaciones que permiten solucionar un problema. Para la teoría de la complejidad computacional, la capacidad de un algoritmo para resolver un problema la determina el número de operaciones aritméticas necesarias para su ejecución. Un problema es fácil si existe un algoritmo que lo resuelve en tiempo polinomial; es decir, si el número de operaciones necesarias para que el algoritmo resuelva el problema es una función polinomial del tamaño del problema. Si esta función no es polinomial, se dice que el algoritmo es no polinomial y el problema se considera difícil.

Como hemos mencionado, el problema genérico de asignación de personal consiste en determinar las labores, horarios y días que serán realizadas por cada colaborador, bajo ciertas restricciones, que se pueden clasificar en:

- Restricciones propias del problema de asignación: que la necesidad operativa de recursos humanos esté cubierta, es decir, si se necesitan "x" cantidad de colaboradores en determinados horarios, que efectivamente se encuentren con turno programado para cubrir la operación del negocio, de acuerdo a la demanda estimada de clientes, asegurando que a cada tarea, sólo se asigne un colaborador capaz de realizarla y que todas las tareas sean realizadas.

- Restricciones de normativa legal y organizacional, esto es, respetar el máximo de días corridos trabajados en la programación, las horas de descanso entre turnos, el máximo de jornada semanal, la asignación de libres legales, entre otros aspectos legales.

- Restricción de mantener una distribución equitativa de la carga de trabajo entre todos los colaboradores, esto es, que tengan la misma cantidad de turnos programados según su tipo de jornada (45 ó 30 horas semanales), la misma cantidad de días libres y la misma cantidad de turnos de mañana, tarde y noche durante cada mes.

Dado que en la elección de la mejor decisión se han de tener en cuenta varios criterios y, por tanto, se desea alcanzar más de un objetivo, nos encontramos frente a un problema multiobjetivo.

F. Baesler et. al (2008)[3] indican que "La optimización multiobjetivo puede ser definida como un problema de optimización que presenta dos o más funciones

[3]BAESLERF, MORAGAR. y CORNEJOO. (2008). Artículo "Introducción de Elementos de Memoria en el Método Simulated Annealing para resolver problemas de programación multiobjetivo de máquinas paralelas", Revista Chilena de Ingeniería, Vol.16 nº 3, pp. 428-437.

objetivo. El inconveniente principal que presenta este tipo de problemas, en relación a un modelo de objetivo único, radica en la subjetividad de la solución encontrada. Un problema multiobjetivo no tiene una solución óptima única, más bien, genera un set de soluciones que no pueden ser consideradas diferentes entre sí. De esta manera, el conjunto de soluciones óptimas se denomina Frontera de Pareto. Esta frontera de soluciones contiene todos los puntos que no son superados en todos los objetivos por otra solución. Este concepto lleva el nombre de dominancia, por esta razón la frontera de Pareto consiste solo de soluciones no dominadas. Una solución domina a otra, si y sólo si, es al menos tan buena como las otras soluciones en todos sus objetivos, y es la mejor de todas en al menos un objetivo. La decisión final con respecto a cuál solución seleccionar de esta frontera de puntos depende de la perspectiva de cada tomador de decisiones (TD). Es decir, depende del nivel de compromiso que un tomador de decisiones otorga a cada objetivo dentro de su particular análisis".

Bajo esa perspectiva, los autores señalan que las técnicas para resolver problemas multiobjetivo dependerán de cómo se incorpora el punto de vista del TD en el problema, dividiendo dichas técnicas en tres grandes grupos:

- Técnicas de articulación previa de preferencias del TD, donde el TD entrega sus preferencias en relación a la importancia relativa de un objetivo con respecto al resto. Esta información es incorporada al modelo con el fin de encontrar la solución que satisface los requerimientos de ese TD en particular.

- Técnicas búsqueda interactivas que requieren que el TD se involucre en el proceso de búsqueda en forma constante.

- Técnicas de articulación posterior de preferencias del TD, que buscan generar la Frontera de Pareto en su totalidad con el fin que cada TD decida, de esta

gama de soluciones, cuál punto satisface sus requerimientos. Esta forma de ver el problema multiobjetivo es considerado un enfoque moderno para enfrentar este tipo de problemas y la mayoría de las metodologías que se han desarrollado en los últimos años se enmarcan en esta categoría.

De lo anterior, rara vez una decisión es tomada en base a un solo criterio. Por ello, el desafío en los problemas de asignación de personal es encontrar asignaciones eficientes[4] que permitan cumplir con la demanda existente a un costo aceptable y al mismo tiempo evitar violar las restricciones propuestas.

El problema de programación de tareas o *scheduling* puede definirse de manera muy general como el problema de organizar o secuenciar una serie de operaciones y ubicarlas en el tiempo sin violar ninguna de las restricciones, de precedencias y recursos, impuestas en el sistema.

La complejidad de este tipo de problemas plantea dificultades en su resolución. En primer lugar, el gran número de soluciones que ofrece la combinatoria hace inviable una exploración exhaustiva de las mismas. Por otro lado, al ser *NP-hard* hay serias sospechas de que no existen métodos para resolverlos de forma óptima en tiempo polinómico. Finalmente, otra dificultad añadida es la gran variedad de tipos de problemas que se presentan, siendo necesaria, en muchas ocasiones, la creación de nuevos procedimientos adaptados a la particularidad de cada caso.

[4]Se dice que una solución es eficiente o Pareto-óptima cuando es una solución factible tal que no existe otra solución factible que proporcione una mejora en un atributo cualquiera sin que produzca, de forma simultánea, un empeoramiento en, al menos, otro de los atributos. El término atributo hace referencia a las características que describen cada una de las alternativas disponibles en una situación de decisión. Para Hwang y Masud (1979), los atributos son las características, cualidades o parámetros de comportamiento de las alternativas.

Estos hechos, han motivado la investigación y desarrollo de métodos útiles para la secuenciación de operaciones, ya que son problemas representativos del área de optimización combinatoria[5].

Los métodos desarrollados en el campo de la secuenciación de operaciones se pueden dividir en dos categorías:

- Los **métodos *exactos***, que proporcionan una solución globalmente óptima, pero pueden requerir un tiempo de computación muy alto.

- Los **métodos aproximados**, que van eligiendo soluciones que cumplen determinados criterios y que habitualmente proporcionan una solución razonablemente buena en un tiempo aceptable. Aquí encontramos los ***métodos heurísticos*****,** y como mejora a los mismos, se generaron los ***métodos metaheurísticos*****.**

A continuación en el siguiente apartado, se muestra un resumen de los métodos más destacados.

- **Métodos Exactos**

Los métodos de resolución exactos son aquellos que garantizan el valor óptimo de la solución, pero normalmente a costa de un consumo computacional muy elevado, por lo que resultan adecuados para instancias pequeñas del problema. Según la clasificación establecida por Pinedo[6] se pueden distinguir los siguientes métodos:

[5]Dentro del área de Optimización (búsqueda de la mejor solución a un problema de entre todo un conjunto de posibles soluciones) podemos encontrar una diversidad de problemas que podemos clasificar en dos grupos principales de acuerdo al tipo de solución que se trata de encontrar: Optimización Numérica y Optimización Combinatoria. Por ejemplo: asumiendo que las variables son continuas en las funciones, el problema se considera de optimización numérica. En cambio, si las variables son de naturaleza discreta, el problema de encontrar soluciones óptimas es conocido como optimización combinatoria.

[6]PINEDO, M., *Planning and Scheduling in Manufacturing and Services*. Springer, (2005).

a) Programación lineal

Este procedimiento, introducido en la Segunda Guerra Mundial, resuelve un problema indeterminado, formulado a través de ecuaciones lineales, optimizando la función objetivo, también lineal, de tal forma que las variables de dicha función, estén sujetas a una serie de restricciones expresadas mediante un sistema de inecuaciones lineales. Un programa lineal entero es un programa lineal con la condición adicional de que las variables han de tomar valores enteros. Si entre las variables hay enteros y reales, se trata de un programa lineal mixto. Los programas lineales pueden resolverse en tiempo polinomial. Sin embargo, esto no sucede con los enteros ni los mixtos.

b) Ramificación y poda (*Branch & Bound*)

La técnica de *Branch & Bound* fue creada por Land y Doig[7] en 1960. Se suele interpretar como un árbol de soluciones, donde cada rama lleva a una posible solución posterior a la actual. La característica de esta técnica con respecto a otras anteriores es que el algoritmo se encarga de detectar en qué ramificación las soluciones dadas ya no son óptimas, para "podar" esa rama del árbol y no continuar desaprovechando recursos en casos que se alejan de la solución óptima. La búsqueda comienza en el nodo raíz y continúa hasta llegar a un nodo hoja. Desde un nodo no seleccionado, la operación de ramificación determina el siguiente conjunto de posibles nodos a partir del cual puede progresar la búsqueda. El procedimiento de poda selecciona la operación con la que continuar la búsqueda y se basa en una estimación de una cota inferior y la mejor cota superior alcanzada hasta el momento.

[7]LAND A.H., and DOIG, A.G. *An Automatic Method of Solving Discrete Programming Problems.* Econometrica, (1960).

Habitualmente estos algoritmos toman como solución inicial una obtenida como resultado de algún método heurístico, lo que permite ahorrar un tiempo de computación considerable.

c) Programación dinámica

En 1953 Bellman[8] ideó un método recursivo al que denominó Programación Dinámica. Este método plantea la solución como una sucesión de decisiones que intenta, a partir de un enfoque *divide y vencerás*, reducir el tiempo computacional necesario. Se trata de resolver una serie de etapas (subdivisiones del problema) en cada una de las cuales se toma una decisión simple hasta hallar una solución al problema original. Se determina la solución óptima para cada iteración, en base a la información obtenida en todas las etapas anteriores, y su contribución a la función objetivo.

Limitaciones Métodos Exactos

Como se definió anteriormente, los algoritmos de métodos exactos se caracterizan por realizar exploraciones exhaustivas, sin embargo son de los algoritmos más difundidos en las aplicaciones industriales. A la fecha, los métodos de búsqueda y corte son los más efectivos para resolver programas enteros de tamaños prácticos, siendo generalmente mejor el de *Branch & Bound*. Los métodos exactos pretenden hallar un plan jerárquico único analizando todos los posibles ordenamientos de las tareas o procesos involucrados en la línea de producción (exploración exhaustiva). Sin embargo, una estrategia de búsqueda y ordenamiento que analice todas las combinaciones posibles es computacionalmente costosa y sólo funciona para algunos tipos (tamaños) de instancia.

Los métodos exactos, serían los métodos idóneos si no tuvieran el inconveniente de la cantidad de tiempo necesaria para la resolución. El tiempo crece

[8]BELLMAN, R., *An introduction to the theory of dynamic programming*, The RAND Corporation, Report R-245, (1953).

exponencialmente con el tamaño del problema. En determinados casos, el tiempo de resolución podría llegar a ser de varios días, meses o incluso años, lo que provoca que el problema sea inabordable con estos métodos.

Dentro de la literatura, C. Coello (2003) indica que *"los algoritmos exactos muestran un rendimiento pobre para muchos problemas"* y los autores Yamada y Nakano (1999) indican que *"el algoritmo Branch & Bound para* problemas de programación de horarios o *Job Shop Scheduling Problem - JSSP…resuelve problemas JSP de hasta 15 * 14 (220 operaciones)"*; lo que muestra otra de las limitaciones de estos algoritmos respecto al número de variables que se pueden trabajar en su aplicación.

- **Métodos Aproximados**

Los métodos aproximados proporcionan una solución de alta calidad (razonablemente buena para los problemas combinatorios), aunque no necesariamente sea la óptima, al menos es factible, en un tiempo computacional aceptable.

Como indicamos, aquí encontramos los ***métodos heurísticos y metaheurísticos*****.**

a) Métodos heurísticos

El término *heurística*, proviene de la palabra griega *"heuriskein"*, cuyo significado está relacionado con el concepto de encontrar y se vincula a la famosa y supuesta exclamación de *¡eureka!* de Arquímedes. Con ese origen se han desarrollado un gran número de procedimientos heurísticos con mucho éxito para la resolución de problemas de optimización, de los que se intenta extraer lo mejor para emplearlos en otros problemas o contextos más extensos.

Los ***métodos heurísticos*** son estrategias generales de resolución y reglas de decisión utilizadas por los solucionadores de problemas, basadas en la experiencia previa con problemas similares. Estas estrategias indican las vías o posibles enfoques a seguir para alcanzar una solución.

La *heurística* comprende los métodos o algoritmos de exploración en los que las soluciones se hallan mediante la evaluación del progreso logrado en la búsqueda de un resultado final. Se califica de heurístico a un procedimiento que encuentra soluciones con un coste computacional razonable, aunque no se garantice su optimalidad.

De acuerdo con Monero y otros (1995) los procedimientos heurísticos son acciones que comportan un cierto grado de variabilidad y su ejecución no garantiza la consecución de un resultado óptimo.

Según Martí (2003), los algoritmos heurísticos suelen ser métodos de resolución muy útiles cuando se cumplen una o varias de las siguientes situaciones:

- La solución puede ser aproximada.
- El problema es de una naturaleza tal que no se conoce ningún método exacto para su resolución.
- Aunque existe un método exacto para resolver el problema, su uso es computacionalmente muy costoso o inviable.
- Como paso intermedio hacia otro procedimiento de resolución, normalmente un algoritmo metaheurístico.
- Cuando existen limitaciones de tiempo, espacio para almacenaje de datos, presupuesto, etc., que priorizan una solución rápida a costa de pérdida de precisión.

Clasificación de los métodos heurísticos

Existen muchos métodos heurísticos de naturaleza muy diferente, por lo que es complicado dar una clasificación completa. Además, muchos de ellos han sido diseñados para un problema específico sin posibilidad de generalización o aplicación a otros problemas similares.

Martí (2003), en su artículo *Algoritmos Heurísticos,* presenta el siguiente esquema de clasificación que trata de dar unas categorías amplias, no excluyentes, en donde ubicar a los heurísticos más conocidos:

-Métodos de Descomposición: El problema original se descompone en subproblemas más sencillos de resolver, teniendo en cuenta, aunque sea de manera general, que ambos pertenecen al mismo problema.

-Métodos Inductivos: La idea de estos métodos es generalizar de versiones pequeñas o más sencillas al caso completo. Propiedades o técnicas identificadas en estos casos más fáciles de analizar pueden ser aplicadas al problema completo.

-Métodos de Reducción: Consiste en identificar propiedades que se cumplen mayoritariamente por las buenas soluciones e introducirlas como restricciones del problema. El objeto es restringir el espacio de soluciones simplificando el problema. El riesgo obvio es dejar fuera las soluciones óptimas del problema original.

-Métodos Constructivos: Consisten en construir literalmente paso a paso una solución del problema. Usualmente son métodos deterministas y suelen estar basados en la mejor elección en cada iteración.

-Métodos de Búsqueda Local: A diferencia de los métodos anteriores, los procedimientos de *búsqueda o mejora local* comienzan con una solución del problema y la mejoran progresivamente. El procedimiento realiza en cada paso un

movimiento de una solución a otra con mejor valor. El método finaliza cuando, para una solución, no existe ninguna solución accesible que la mejore.

Medidas de calidad de los métodos heurísticos[9]

Al resolver un problema de forma heurística se debe de medir la calidad de los resultados puesto que, como ya se ha mencionado, la optimalidad no está garantizada. Se debe, pues, medir la calidad y eficiencia de un algoritmo para poder determinar su valía frente a otros. Un buen algoritmo heurístico debe de tener las siguientes propiedades:

***-Eficiente*.** Un esfuerzo computacional realista para obtener la solución.
***-Bueno*.** La solución debe de estar, en promedio, cerca del óptimo.
***-Robusto*.** La probabilidad de obtener una mala solución (lejos del óptimo) debe ser baja.

Para tal fin existen diversos procedimientos, entre los que se encuentran los siguientes:

- ***Comparación con la solución óptima, si se conoce, o con la mejor solución disponible.*** Aunque normalmente se recurre al algoritmo aproximado por no existir un método exacto para obtener el óptimo, o por ser éste computacionalmente muy costoso, en ocasiones puede que dispongamos de un procedimiento que proporcione el óptimo para un conjunto limitado de ejemplos (usualmente de tamaño reducido). Este conjunto de ejemplos puede servir para medir la calidad del método heurístico. Normalmente se mide, para cada uno de los ejemplos, la desviación porcentual de la solución heurística frente a la óptima, calculando posteriormente el promedio de dichas desviaciones. Si llamamos c_h al coste de la solución del algoritmo heurístico y

[9] Martí (2003).

c_{opt} al coste de la solución óptima de un ejemplo dado, en un problema de minimización la desviación porcentual viene dada por la expresión:

$$\frac{c_h - c_{opt}}{c_{opt}} * 100$$

- ***Comparación con una cota.*** En ocasiones el óptimo del problema no está disponible ni siquiera para un conjunto limitado de ejemplos. Un método alternativo de evaluación consiste en comparar el valor de la solución que proporciona el heurístico con una cota del problema (inferior si es un problema de minimización y superior si es de maximización). Obviamente la bondad de esta medida dependerá de la bondad de la cota (cercanía de ésta al óptimo), por lo que, de alguna manera, tendremos que tener información de lo buena que es dicha cota. En caso contrario, la comparación propuesta no tiene demasiado interés.

- ***Comparación con un método exacto truncado.*** Un método enumerativo como el de Ramificación y Acotación explora una gran cantidad de soluciones, aunque sea únicamente una fracción del total, por lo que los problemas de grandes dimensiones pueden resultar computacionalmente inabordables con estos métodos. Sin embargo, podemos establecer un límite de iteraciones (o de tiempo) máximo de ejecución para el algoritmo exacto. También podemos saturar un nodo en un problema de maximización cuando su cota inferior sea menor o igual que la cota superior global más un cierto valor *a* (análogamente para el caso de minimizar). De esta forma, se garantiza que el valor de la mejor solución proporcionada por el procedimiento no dista más de *a* del valor óptimo del problema. En cualquier caso, la mejor solución encontrada con estos procedimientos truncados proporciona una cota con la que contrastar el heurístico.

- ***Comparación con los resultados de otras heurísticas.*** Este es uno de los métodos más empleados en problemas difíciles (NP hard) sobre los que se ha

trabajado durante tiempo y para los que se conocen algunos buenos heurísticos. Al igual que ocurre con la comparación con las cotas, la conclusión de dicha comparación está en función de la bondad del heurístico escogido.

- ***Análisis del peor caso.*** Uno de los métodos que durante un tiempo tuvo bastante aceptación es analizar el comportamiento en el peor caso del algoritmo heurístico; esto es, considerar los ejemplos que sean más desfavorables para el algoritmo y acotar analíticamente la máxima desviación respecto del óptimo del problema. Lo mejor de este método es que acota el resultado del algoritmo para cualquier ejemplo; sin embargo, por esto mismo, los resultados no suelen ser representativos del comportamiento medio del algoritmo. Además, el análisis puede ser muy complicado para los heurísticos más sofisticados. Aquellos algoritmos que, para cualquier ejemplo, producen soluciones cuyo coste no se aleja de un porcentaje ε del coste de la solución óptima, se llaman *Algoritmos ε-Aproximados*. Esto es; en un problema de minimización se tiene que cumplir para un ε>0 que: $c_h \leq (1 + \varepsilon)c_{opt.}$

Limitaciones Métodos Heurísticos

El principal problema que presentan los algoritmos heurísticos es su incapacidad para escapar de los óptimos locales. En la Figura 3.1 se muestra como para una vecindad dada, el algoritmo heurístico basado en un método búsqueda local se quedaría atrapado en un máximo local.

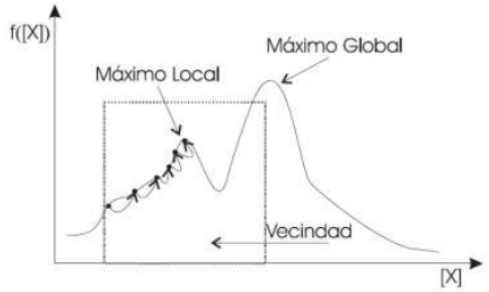


FIGURA 3.1: EJEMPLO DE ENCASILLAMIENTO DE LOS ALGORITMOS HEURÍSTICOS EN ÓPTIMOS LOCALES (FUNCIÓN MULTIMODAL).

b) Métodos Metaheurísticos

Si bien todos estos métodos han contribuido a ampliar el conocimiento, en los últimos veinte años una nueva clase de algoritmos de aproximación han surgido que tratan de combinar métodos básicos de heurísticas en altos niveles de estructuras y que apuntan a explorar eficaz y eficientemente el espacio de búsqueda. Estos métodos son comúnmente llamados hoy ***"Metaheurísticas"***, concepto introducido por primera vez por Fred Glover en 1986, obtenido de la composición de 2 palabras griegas: H*eurística que significa "Encontrar" y* el sufijo *Meta* que significa *"más allá"* o *"a un nivel superior"*[10]*.*

Las concepciones actuales de lo que es una metaheurística están basadas en las diferentes interpretaciones de lo que es una forma inteligente de resolver un problema.

Los profesores Osman y Kelly (1995) introducen la siguiente definición[11]: "*Los procedimientos Metaheurísticos son una clase de métodos aproximados que están diseñados para resolver problemas difíciles de optimización combinatoria, en los que los heurísticos clásicos no son efectivos. Los Metaheurísticos proporcionan un marco general para crear nuevos algoritmos híbridos combinando diferentes conceptos*

[10]Antes de que este término fuera ampliamente adoptado, las Metaheurísticas fueron frecuentemente llamadas Heurísticas Modernas (Reeves, 1993).
[11]Osman, I.H. and Kelly, J.P. (eds.). *Meta-Heuristics: Theory and Applications*, Boston USA Ed. Kluwer Academic, (1996).

derivados de la inteligencia artificial, la evolución biológica y los mecanismos estadísticos".

Una metaheurística se define como "Un proceso de generación iterativa que guía una heurística subordinada combinando diferentes conceptos para explorar y explotar el espacio de búsqueda, aprendiendo estrategias usadas para estructurar información y encontrar eficientemente soluciones (cercanas) óptimas evitando de caer en los óptimos locales y que pueden ser aplicadas a diferentes problemas de optimización permitiendo modificaciones para adaptarlas también a problemas específicos[12]".

Las metaheurísticas se sitúan conceptualmente "por encima" de las heurísticas en el sentido que guían el diseño de éstas, pueden estar compuestas por una combinación de algunas heurísticas, por ejemplo una metaheurística puede usar una heurística constructiva para generar una solución inicial y luego usar otra heurística de búsqueda para encontrar una mejor solución.

Resumidamente, las propiedades que caracterizan las Metaheurísticas son las siguientes:

- Son estrategias que "guían" los procesos de búsqueda.
- La meta es explorar eficientemente el espacio de búsqueda para encontrar soluciones (cercanas) óptimas.
- Técnicas que constituyen una gama de algoritmos metaheurísticos desde procedimientos de búsqueda local a complejos procesos de aprendizaje.
- Los algoritmos metaheurísticos son aproximados y usualmente no-determinísticos.

[12]Elaboración propia, en base a definiciones de la bibliografía analizada.

- Pueden incorporar mecanismos para evitar quedar atrapados en áreas confinadas del espacio de búsqueda.
- Los conceptos básicos de las Metaheurísticas permiten una descripción a nivel abstracto.
- Son de uso genérico, no específicos a una clase de problemas.
- Pueden hacer uso de conocimiento de dominio específico en la forma de heurísticas que son controladas por el nivel superior de la estrategia.
- Hoy, más metaheurísticas avanzadas usan experiencia de búsqueda (personificada en alguna forma de memoria) para guiar la búsqueda.

Se puede decir entonces que las metaheurísticas son estrategias de nivel superior para explorar espacios de búsqueda usando diferentes métodos.

También es importante mencionar el balance dinámico dado entre Diversificación e Intensificación. Diversificación generalmente se refiere a la Exploración del espacio de búsqueda (promover al proceso de búsqueda a examinar regiones no visitadas del espacio de búsqueda, para generar soluciones que difieran de manera significativa de las soluciones ya vistas), mientras que la Intensificación se refiere a la Explotación de la experiencia acumulada de búsqueda (enfocar la búsqueda en examinar soluciones que pertenezcan a la vecindad de las mejores soluciones encontradas). Con la Exploración se pretende identificar rápidamente regiones en el espacio de búsqueda con soluciones de alta calidad y con la Explotación, no desperdiciar mucho tiempo en regiones del espacio de búsqueda que fueron ya exploradas o que no proporcionan soluciones de alta calidad. Existe un amplio consenso en que estas dos características deben modelarse adecuadamente para conseguir el éxito práctico de las aplicaciones de las metaheurísticas.

Las estrategias de búsqueda de diferentes metaheurísticas son altamente dependientes de la filosofía de la metaheurística en sí. Algunas de estas filosofías pueden ser vistas como extensiones "inteligentes" de algoritmos de búsqueda local.

La meta de esta clase de metaheurísticas es escapar de mínimos locales para avanzar en la exploración del espacio de búsqueda y moverse para encontrar otro mínimo local esperanzadamente mejor. Este es el caso de las metaheurísticas basadas en métodos de trayectoria como Búsqueda Tabú (TS-Tabu Search), Búsqueda Local Iterativa (ILS-Iterated Local Search), Búsqueda Variable de Vecindario (VNS-Variable Neighborhood Search), GRASP (Greedy Randomized Adaptive Search Procedure) y Recocido Simulado (SA-Simulated Annealing) que trabajan sobre uno o varias estructuras de vecindario impuestas sobre los miembros (las soluciones) del espacio de búsqueda.

Otra filosofía diferente se encuentra en los algoritmos como Optimización de Colonias de Hormigas (Ant Colony Optimization-ACO) y Computación Evolutiva (Evolutionary Computation-EC) que incorporan un componente de aprendizaje e implícitamente o explícitamente tratar de establecer correlaciones entre las variables de decisión para identificar áreas de alta calidad en el espacio de búsqueda.

Medición de la calidad y eficiencia de los métodos metaheurísticos[13]

Son propiedades deseables todas aquellas que favorezcan el interés práctico y teórico de las metaheurísticas, pero no será posible mejorar todas las propiedades a la vez, dado que algunas son parcialmente contrapuestas.

Una relación de tales propiedades debe incluir las siguientes:

- ***Simple.*** La metaheurística debe estar basada en un principio sencillo y claro; fácil de comprender.
- ***Precisa.*** Los pasos y fases de la metaheurística deben estar formulados en términos concretos.

[13]Melián, Belén. Perez, Jose A. et al. "Metaheurísticas: una visión global". *Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial*. N.19 pp. 7-28 ISSN: 1137-3601. © AEPIA(2003).http://www.aepia.org/revista.

- ***Coherente.*** Los elementos de la metaheurística deben deducirse naturalmente de sus principios.
- ***Eficaz.*** La probabilidad de alcanzar soluciones óptimas de casos realistas con la metaheurística debe ser alta.
- ***Eficiente.*** La metaheurística debe realizar un buen aprovechamiento de recursos computacionales: tiempo de ejecución y espacio de memoria.
- ***General.*** La metaheurística debe ser utilizable con buen rendimiento en una amplia variedad de problemas.
- ***Adaptable.*** La metaheurística debe ser capaz de adaptarse a diferentes contextos de aplicación o modificaciones importantes del modelo.
- ***Robusta.*** El comportamiento de la metaheurística debe ser poco sensible a pequeñas alteraciones del modelo o contexto de aplicación.
- ***Interactiva.*** La metaheurística debe permitir que el usuario pueda aplicar sus conocimientos para mejorar el rendimiento del procedimiento.
- ***Múltiple.*** La metaheurística debe suministrar diferentes soluciones alternativas de alta calidad entre las que el usuario pueda elegir.
- ***Autónoma.*** La metaheurística debe permitir un funcionamiento autónomo, libre de parámetros o que se puedan establecer automáticamente.

Clasificación de las Metaheurísticas

Los criterios de clasificación más importantes de las metaheurísticas son:

a) Inspiradas v/s No Inspiradas en la Naturaleza (sistemas biológicos, físicos o sociales)

Existen algoritmos inspirados en la naturaleza como los Algoritmos Genéticos (Evolución de las especies) y Colonias de Hormigas; y algoritmos no basados en la naturaleza como Búsqueda Tabú y Búsqueda Local Iterativa.

Esta clasificación no se considera significativa por 2 razones: Muchos algoritmos híbridos no pueden clasificarse en ninguna clase o incluso pueden clasificarse en ambas clases al mismo tiempo. Por otro lado, es difícil clarificar los atributos de las clases para clasificar los algoritmos.

b) Basadas en Poblaciones v/s Basadas en Puntos Individuales de Búsqueda

Los algoritmos que trabajan sobre soluciones individuales se denominan Métodos de Trayectoria y Metaheurísticas de Búsqueda Local como TS, ILS y VNS que describen una trayectoria en el espacio de búsqueda durante el mismo proceso. Las metaheurísticas basadas en poblaciones desarrollan procesos de búsqueda las cuales describen la evolución de un conjunto de puntos en el espacio de búsqueda, es decir, buscan el óptimo a través de un conjunto de soluciones.

c) Funciones Objetivos Estáticas v/s Funciones Objetivos Dinámicas

Algunos algoritmos mantienen la función objetivo dada en la representación del problema y otros la modifican incorporando información acumulada durante el proceso de búsqueda para escapar de mínimos locales.

d) Una v/s Varias Estructuras de Vecindario

Algunas metaheurísticas utilizan una estructura de vecindario que no cambia durante el curso del algoritmo, mientras que otras como VNS utilizan un conjunto de estructuras de entorno que dan la posibilidad de diversificar la búsqueda.

e) Con Uso de Memoria v/s Sin Uso de Memoria

Diversas metaheurísticas se refieren al uso de información sobre las características y propiedades comunes a soluciones de alta calidad o sobre las decisiones de mejora adoptadas durante el proceso de solución. Esta información permite mejorar el rendimiento de la búsqueda de arranque múltiple ajustando los parámetros que modulan la exploración y la explotación del proceso. Se incluyen en las Metaheurísticas de *aprendizaje* ya que son capaces de emplear información obtenida en la aplicación del propio procedimiento a un problema específico. El uso de memoria es hoy reconocido como uno de los elementos fundamentales de una buena metaheurística.

A continuación, se describirán las metaheurísticas más importantes bajo el criterio basado en el número de soluciones, esto es, *Metaheurísticas basadas en Trayectoria y las basadas en Poblaciones*[14].

Esta taxonomía se muestra de forma gráfica en la siguiente figura donde se presentan las principales metaheurísticas que se describirán en los apartados siguientes.

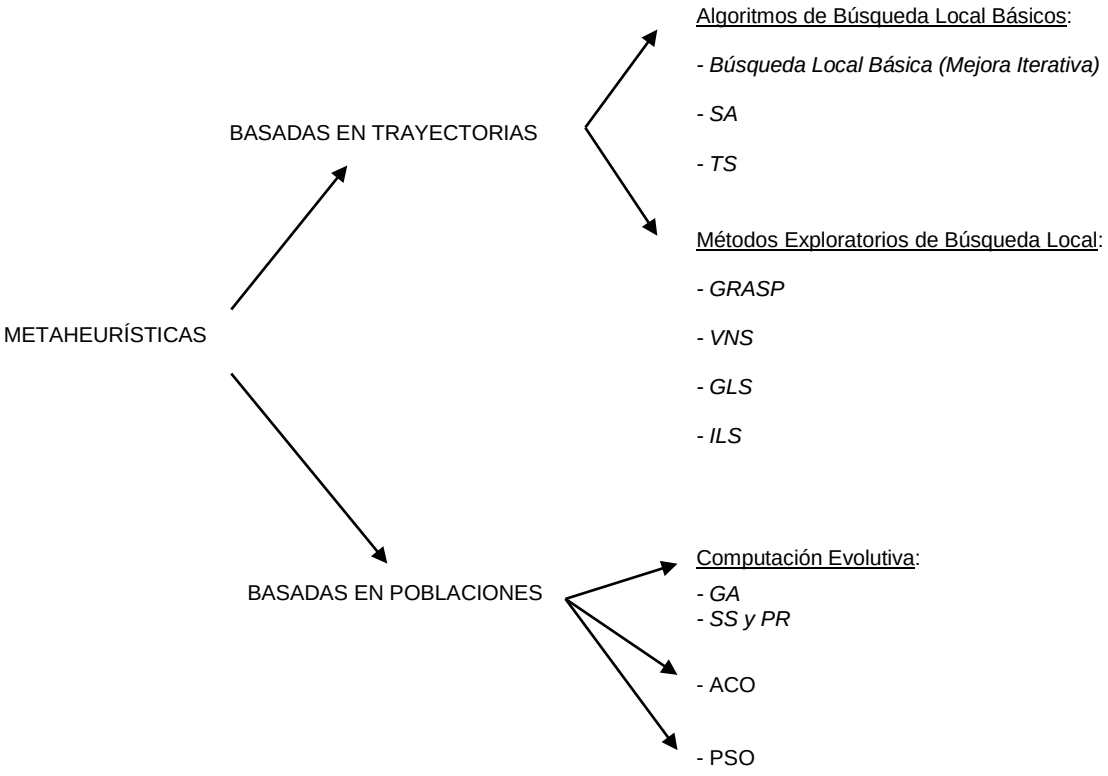


FIGURA 3.2: CLASIFICACIÓN DE LAS METAHEURÍSTICAS [15].

[14]Se elige esta clasificación porque es ampliamente utilizada en la comunidad científica, además de ser muy coherente.

[15] Para abreviaturas, ver Glosario de Términos al final del estudio.

a) Metaheurísticas basadas en trayectorias

El término métodos de trayectoria es usado porque el proceso desarrollado de búsqueda se caracteriza por una trayectoria en el espacio de búsqueda donde una solución sucesora puede o no pertenecer al vecindario de la solución actual.

El algoritmo comienza desde un estado inicial (la solución inicial) y describe una trayectoria en estado del espacio. La dinámica del sistema depende de la estrategia usada; algoritmos simples generan una trayectoria compuesta de dos partes: una fase de tránsito seguida por una atracción (un punto fijo, un ciclo o un atractor complejo). Los algoritmos con estrategias avanzadas generan trayectorias más complejas las cuales no pueden ser subdivididas en estas dos fases.

La característica de la trayectoria es que proporciona información acerca de la conducta del algoritmo y su efectividad respecto al ejemplo que se aborda. Vale la pena destacar que la dinámica es el resultado de la combinación del algoritmo con la representación del problema y la instancia del problema. En efecto, la representación del problema junto a las estructuras de vecindario definen el horizonte de búsqueda; el algoritmo describe la estrategia usada para explorar el horizonte y finalmente, las características del espacio actual de búsqueda son definidas por la instancia del problema a resolver.

A continuación, se describirán primero los algoritmos de búsqueda local básicos, y luego, se tratarán algoritmos con estrategias exploratorias generales que pueden incorporar métodos de trayectoria como componentes.

a.1) Algoritmos de Búsqueda Local Básicos

- ***Búsqueda Local Básica (Mejora Iterativa)***

La Búsqueda Local Básica es una de las primeras metaheurísticas. Se trata de un procedimiento de búsqueda que parte de una solución *s* y realiza movimientos desde esta solución a soluciones vecinas de la misma según una cierta definición de entorno *N (s).* Se llama también Mejora Iterativa porque cada nuevo movimiento[16] requiere que la solución resultante sea mejor que la solución actual. El algoritmo se detiene tan pronto cuando encuentra una solución mejor que todas sus vecinas, es decir, cuando encuentra un óptimo local.

Al ir explorando exhaustivamente el vecindario puede ocurrir que se retorne a una de las soluciones que arroje un valor más bajo de la función objetivo *f*, quedando atrapado en un mínimo local. Por lo tanto, su desarrollo depende fuertemente de la definición de la solución *s,* la función objetivo *f* y el entorno *N (s)*.

El desarrollo de los procedimientos de mejora iterativa en problemas de optimización combinatoria es usualmente bastante insatisfactorio ya que el algoritmo siempre se detiene al encontrar un óptimo local, que a veces está bastante alejado del óptimo global del problema. Por ello, varias técnicas han sido desarrolladas para prevenir que los algoritmos queden atrapados en mínimos locales y que les permitan escapar. Esto también implica que las condiciones de término del algoritmo metaheurístico son más complejos que el simple alcance respecto al mínimo local. En efecto, las posibles condiciones de término incluyen: tiempo máximo de procesamiento (CPU), un número máximo de iteraciones, entre otros.

[16]Un movimiento es la elección de una solución *s'* desde el vecindario *N (s)* de una solución *s*.

- ***Recocido Simulado (SA-Simulated Annealing)***

SA es una de las más populares metaheurísticas por ser uno de los primeros algoritmos que tiene una estrategia explícita para escapar de mínimos locales. La idea fundamental es que permite movimientos que generan soluciones de peor calidad que la solución actual para escapar de los mínimos locales. Los orígenes del algoritmo están en la mecánica estadística (algoritmo Metropolis) y fue por primera vez presentado por Kirkpatrick et al. (1983) y Cerny (1985) como un algoritmo de búsqueda para problemas de optimización combinatoria (por ejemplo, Quadratic Assignment Problem – QAP y Job Shop Scheduling –JSS).

El término de recocido simulado proviene de la estrecha analogía que guarda con el proceso del recocido tal y como se usa en metalurgia. Éste consiste en el enfriamiento progresivo de un metal de manera que sus moléculas van adoptando poco a poco una configuración de mínima energía. A medida que la temperatura disminuye se va ralentizando el movimiento de las moléculas y éstas tienden a adoptar paulatinamente las configuraciones de menor energía.

El espacio de configuraciones (posiciones de las moléculas) en el caso del *Job-Shop* viene determinado por los valores de una variable de interés, normalmente la secuencia de operaciones que se desean procesar, mientras que el papel de la energía lo asume la función que se intenta minimizar.

El recocido simulado ha sido vastamente aplicado con considerable éxito. Su naturaleza aleatoria permite convergencia asintótica a la solución óptima bajo condiciones moderadas. Desafortunadamente, la convergencia requiere típicamente de tiempo exponencial, convirtiendo el recocido simulado en impráctico como instrumento para procurarse soluciones óptimas. En cambio, como muchos algoritmos de búsqueda local, resulta eficiente como método de aproximación, para lo cual presenta una tasa de convergencia más aceptable.

- ***Búsqueda Tabú (TS-Tabu Search)***

Este algoritmo, desarrollado por Glover (1986), utiliza una búsqueda local con memoria a corto plazo que le permite "escapar" de mínimos locales y evita ciclos. La memoria de corto plazo está representada por la lista Tabú la cual registra las últimas soluciones "visitadas" e impide volver a ellas en los próximos movimientos. La lista Tabú se actualiza normalmente en forma FIFO (First In-First Out, el primer elemento en entrar es el primero en salir), porque de esta forma, una vez hecho el movimiento inicial desde la solución actual *s* a la nueva solución *s'*, el movimiento inverso desde *s'* a *s,* se prohíbe al menos durante los siguientes movimientos.

El largo de la lista tabú controla la memoria del proceso de búsqueda. Una lista tabú corta, controla áreas reducidas del espacio de búsqueda y una larga, fuerza a una búsqueda en áreas mayores.

El largo de la lista puede cambiar durante el proceso de búsqueda. Por razones de eficiencia, no se guarda la solución completa en la lista tabú, sino una parte de sus atributos, se define una lista tabú de condiciones. En este proceso se pierde información y buenas soluciones pueden ser excluidas del conjunto permitido. Para reducir este problema, se define un criterio de aspiración que permitiría a una solución estar dentro del conjunto de soluciones permitidas aun cuando figure en la lista tabú.

a.2) Métodos Explorativos de Búsqueda Local

A continuación se presenta el Procedimiento de Búsqueda Miope Aleatorizado y Adaptativo (GRASP-Greedy Randomized Adaptive Search Procedure), Búsqueda por Entornos Variable (VNS-Variable Neighborhood Search); Búsqueda Local Guiada (GLS- Guided Local Search) y Búsqueda Local Iterativa (ILS-Iterated Local Search)

- ***GRASP (Greedy Randomized Adaptive Search Procedure)[17]***

Es una metaheurística para encontrar soluciones aproximadas (sub-óptimas de buena calidad, pero no necesariamente óptimas) a problemas de optimización combinatoria. Se basa en la premisa de que soluciones iniciales diversas y de buena calidad juegan un papel importante en el éxito de métodos locales de búsqueda.

Un GRASP es un método multi-arranque, en el cual cada iteración GRASP consiste en la construcción de una solución miope aleatorizada seguida de una búsqueda local usando la solución construida como el punto inicial de la búsqueda local. Este procedimiento se repite varias veces y la mejor solución encontrada sobre todas las iteraciones GRASP se devuelve como la solución aproximada.

De lo anterior, se distinguen dos pasos: una fase de construcción y otra de mejora. En la fase de construcción se aplica una heurística constructiva para obtener una solución inicial y en la segunda fase dicha solución se mejora mediante un algoritmo de búsqueda local. En cada iteración la elección del próximo elemento a ser añadido a la solución parcial está determinada por una función greedy (voraz o golosa), la cual elige el elemento que da mejor resultado inmediato sin tener en cuenta una perspectiva más amplia. Se dice que se adapta porque en cada iteración se actualizan los beneficios obtenidos al añadir el elemento seleccionado a la solución parcial. Se denomina aleatorizado porque no selecciona al mejor candidato

[17] Feo y Resende (1995).

sino que, con el objeto de diversificar y no repetir soluciones, se construye una lista de los mejores candidatos, de entre los cuales, en función de su bondad, se elige uno al azar. En la fase de mejora se realiza un proceso de búsqueda local a partir de la solución construida hasta que no se pueda mejorar más.

Es uno de los métodos más populares debido a su sencillez y facilidad de implementación.

- ***Búsqueda por Entornos Variable (VNS-Variable Neighborhood Search)***[18]

Es una metaheurística reciente que consiste en cambiar de forma sistemática la estructura de entorno para escapar de los mínimos locales. El VNS básico obtiene una solución del entorno de la solución actual, ejecuta una búsqueda monótona local desde ella hasta alcanzar un óptimo local, que reemplaza a la solución actual si ha habido una mejora y modifica la estructura de entorno en caso contrario.

Una de sus variantes es la búsqueda descendente por entornos variables *(VND-Variable Neighborhood Descent)* que aplica una búsqueda monótona por entornos cambiando de forma sistemática la estructura de entornos cada vez que se alcanza un mínimo local.

Además de reiniciar la búsqueda y modificar la estructura de entornos, la otra vía para evitar quedarse atrapados en un óptimo local es controlar la aceptación de movimientos que no sean de mejora para que, al menos a la larga, se vayan mejorando las soluciones encontradas, y utilizar información histórica del proceso de búsqueda para controlar cuando el recorrido se está estancando en un mínimo local y evitar la formación de ciclos. Las metaheurísticas fundamentales que aplican estas estrategias son el Recocido Simulado y la Búsqueda Tabú.

[18]Mladenovic N. y Hansen P. (1997)

- ***Búsqueda Local Guiada (GLS-Guided Local Search)***[19]

La metaheurística consiste básicamente en una secuencia de procedimientos de búsqueda local; al finalizar cada uno de ellos se modifica la función objetivo penalizando determinados parámetros que aparecen en el óptimo local obtenido en el último paso, estimulando de esta forma la diversificación de la búsqueda.

- ***Búsqueda Local Iterativa (ILS-Iterated Local Search)***

ILS propone una estrategia exploratoria de mejora que actúa de la siguiente forma: dada una solución obtenida por la aplicación de la heurística base, se aplica un cambio o alteración que da lugar a una solución intermedia. La aplicación de la heurística base a esta nueva solución aporta una nueva solución que, si supera un test de aceptación, pasa a ser la nueva solución alterada. En resumen, en este algoritmo, los movimientos se realizan sólo si se mejora la solución (ver figura 3.3).

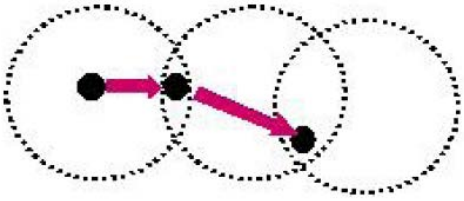


FIGURA 3.3: MOVIMIENTOS EN LA BÚSQUEDA LOCAL ITERATIVA.

Tiene la ventaja de ser flexible y aplicable con gran generalidad, pues sólo requiere de unas cuantas especificaciones como representación de la solución, una función de evaluación y un método eficiente para explorar entornos. No obstante puede quedar estancada en óptimos locales fácilmente. En este sentido, se ha investigado mucho respecto del alcance de la exploración que provea soluciones de alta calidad en tiempos razonablemente bajos.

[19] Voudouris y Tsang (1995 y 1999).

Esta metaheurística se suele utilizar también como entorno, o incluida como componente de otras (por ejemplo, VNS).

b) Metaheurísticas basadas en Poblaciones

En una búsqueda basada en poblaciones se sustituye la solución actual que recorre el espacio de soluciones, por un conjunto de soluciones que lo recorren conjuntamente interactuando entre ellas.

A continuación se describen los métodos más estudiados en optimización combinatoria: Computación Evolutiva (EC-Evolutionary Computation), Optimización de Colonias de Hormigas (ACO-Ant Colony Optimization) y los algoritmos basados en Cúmulos de Partículas (PSO-Particle Swarm Optimization).

b.1) Computación Evolutiva (EC-Evolutionary Computation)

Amplio espectro de técnicas heurísticas que funcionan emulando mecanismos de evolución natural. Trabaja sobre una población de individuos o conjunto de soluciones que evoluciona utilizando mecanismos de selección y construcción de nuevas soluciones candidatas mediante recombinación de características de las soluciones seleccionadas.

Etapas del mecanismo evolutivo:

a) Evaluación: función de fitness.
b) Selección: de individuos adecuados (de acuerdo al fitness) para la aplicación de operadores evolutivos.
c) Aplicación de operadores evolutivos.
d) Reemplazo o recambio generacional.

El proceso evolutivo decide en cada iteración qué individuos entrarán en la próxima población y los operadores evolutivos determinan el modo en que el algoritmo explora el espacio de búsqueda.

Un ejemplo de estos algoritmos son los Algoritmos Genéticos (GA-Genetic Algorithms), Búsqueda Dispersa (SS-Scatter Search) y Reencadenamiento de Camino (PR-Path Relinking).

- ***Algoritmos Genéticos (GA-Genetic Algorithms)***

Los Algoritmos Genéticos surgieron en 1975 de la mano de Holland[20], e imitan el procedimiento de la selección natural sobre el espacio de soluciones del problema considerado. Estos algoritmos hacen evolucionar una población de individuos sometiéndola a acciones aleatorias semejantes a las que actúan en la evolución biológica (mutaciones y recombinaciones genéticas), así como también a una selección de acuerdo con algún criterio, en función del cual se decide cuáles son los individuos más adaptados, que sobreviven, y cuáles los menos aptos, que son descartados.

El esquema de trabajo de los algoritmos genéticos ha sido aplicado a un gran número de problemas donde han aflorado buenos resultados, incluido el problema de programación de horarios. Los algoritmos genéticos conforman una extensa clase de la cual emergen muchos enfoques muy parecidos entre sí.

Dentro de las principales dificultades de estos algoritmos, como en casi todas las metaheurísticas, es que los parámetros iniciales de ejecución dependen en gran medida de pruebas empíricas por parte del usuario para llevar a buen éxito las soluciones.

[20]HOLLAND. H. (1975), *Adaptation in Natural and Artificial Systems*. University of Michigan Press Ann Arbor.

- ***Búsqueda Dispersa (SS-Scatter Search) y Reencadenamiento de Camino (PR-Path Relinking)***

El enfoque de *Búsqueda Dispersa* contempla el uso de un conjunto de referencia de buenas soluciones dispersas que sirve tanto para conducir la búsqueda mejorando las herramientas para combinarlas adecuadamente, como para mantener un grado satisfactorio de diversidad. La Búsqueda Dispersa se distingue de otros procedimientos en los mecanismos de intensificación y diversificación que explotan la memoria adaptada recurriendo a los fundamentos que unen el Scatter Search a la Búsqueda Tabú.

La Búsqueda Dispersa es un método evolutivo que ha resultado muy efectivo en la resolución de problemas de optimización. Al igual que los algoritmos genéticos, el método que nos ocupa se basa en mantener un conjunto de soluciones y realizar combinaciones con éstas; pero a diferencia de éstos, no está fundamentado en la aleatorización sobre un conjunto relativamente grande de soluciones sino en las elecciones sistemáticas y estratégicas sobre un conjunto pequeño (como ilustración basta decir que los algoritmos genéticos suelen considerar una población de 100 soluciones mientras que en la búsqueda dispersa es habitual trabajar con un conjunto de tan sólo 10 soluciones).

El "*Path Relinking*" es una metaheurística asociada a la búsqueda dispersa que utiliza la información que se obtiene de las mejores soluciones. Esta información se aprovecha en las mejoras de otras soluciones que se encuentran posteriormente. Básicamente, se trata de generar soluciones explorando las trayectorias que conectan soluciones de alta calidad. Partiendo de una de estas soluciones, se genera un camino de soluciones hacia la otra solución incorporando a la primera atributos de la segunda. Este camino se construye tomando cada vez el atributo de la segunda solución que lo hace más cercano a ella. A continuación se toman, como

puntos de arranque para nuevas fases de mejora, una o varias de las soluciones del recorrido anterior.

- ***Optimización Colonias de Hormigas (ACO-Ant Colony Optimization)***

Optimización por Colonia de Hormigas es una meta-heurística, propuesta por Dorigo et al. (1996) que se inspira directamente en el comportamiento de las colonias reales de hormigas[21] para solucionar problemas de optimización combinatoria.

ACO se basa en una colonia de hormigas artificiales, esto es, unos agentes computacionales simples que trabajan de manera cooperativa y se comunican indirectamente mediante rastros de feromona artificiales.

Los algoritmos de ACO son esencialmente *algoritmos constructivos*: en cada iteración del algoritmo, cada hormiga construye una solución al problema recorriendo un grafo de construcción. Cada arista del grafo, que representa los posibles pasos que una hormiga puede dar, tienen asociados dos tipos de información que guían el movimiento de cada hormiga:

-*La Visibilidad*, que mide la preferencia heurística de moverse desde un nodo a otro nodo teniendo en cuenta valores de costos.

-*Los rastros de feromona artificiales*, que mide la "deseabilidad aprendida" del movimiento de un nodo a otro nodo. Imita a la feromona real que depositan las hormigas. Esta información se modifica durante la ejecución del algoritmo, dependiendo de las soluciones que van siendo encontradas por las hormigas.

[21]Si bien cada hormiga tiene individualmente capacidades básicas, la colonia en conjunto logra un comportamiento inteligente. A pesar de ser insectos casi ciegos logran encontrar el camino más corto entre el hormiguero y una fuente de comida y regresar, dada una comunicación en base a feromonas, que dejan un "rastro" que sirve de referencia a otras hormigas entre ellas de origen químico que se potencia con la presencia simultánea de muchas hormigas.

Existen actualmente nuevas versiones mejoradas de ACO como Ant Colony System (ACS) y MAX-MIN Ant System (MMAS).

- ***Algoritmos Basados en Cúmulos de Partículas (PSO-Particle Swarm Optimization)[22]***

Son técnicas metaheurísticas inspiradas en el comportamiento social del vuelo de las bandadas de aves o el movimiento de los bancos de peces. Se fundamenta en los factores que influyen en la toma de decisión de un agente que forma parte de un conjunto de agentes similares. La toma de decisión por parte de cada agente se realiza conforme a una componente social y una componente individual, mediante las que se determina el movimiento (dirección) de este agente para alcanzar una nueva posición en el espacio de soluciones. Simulando este modelo de comportamiento se obtiene un método para resolver problemas de optimización.

[22]Kennedy J. et al. (2001), Swarm Intelligence. San Francisco: Morgan Kaufmann Publishers.

De los métodos disponibles referidos en los apartados anteriores y analizando su adecuación al problema a resolver, la tabla siguiente resume las principales ventajas e inconvenientes de los tipos de métodos aplicables:

TABLA 3-1: CUADRO COMPARATIVO DE LOS MÉTODOS DE RESOLUCIÓN

Métodos	Ventajas	Desventajas	Idoneidad
Exactos	Solución óptima Exploración exhaustiva de las soluciones Precisos	Tiempo computacional elevado No adecuados para problemas *NP-hard* Limitados a *Job-Shop* reducidos Dificultad de implementación	Descartado
Heurísticos	Tiempo computacional razonable Adecuados en problemas *NP-hard* Proporcionan varias soluciones Sencillez de implementación	No siempre garantizan el óptimo Exploración de soluciones no exhaustiva	Adecuado
Metaheurísticos	Tiempo computacional razonable Adecuados en problemas *NP-hard* Proporcionan varias soluciones Sencillez de implementación Mejoran los resultados heurísticos	No siempre garantizan el óptimo Exploración de soluciones no exhaustiva	Adecuado

Como resultado de los criterios anteriores se descartan directamente los métodos exactos de resolución, puesto que demuestran ser intratables para las dimensiones del problema de estudio, consumen un tiempo computacional muy elevado y son difíciles de implementar.

Para el desarrollo de este proyecto, se optará por utilizar un método aproximado (heurística o metaheurística).

La elección de este de método se basa en los siguientes motivos:

- En primer lugar, la solución puede ser aproximada, siempre que satisfaga de forma razonable el objetivo perseguido.

- En segundo lugar, aunque siempre que esté disponible y se pueda costear es preferible un método exacto frente a uno aproximado, en nuestro caso no se puede aplicar un método exacto dada la naturaleza *hard* del problema, puesto que dicho método o no existe o dispara el consumo de tiempo computacional. La empresa dispone de limitaciones económicas y de tiempo, por lo que prevalece la obtención de un método rápido que mejore la planificación actual, aunque sea menos preciso.

- Por último, una ventaja importante es que suelen ser más fáciles de entender (por parte de los directivos de las empresas y gente no experta), sin mencionar que generalmente ofrecen más de una solución, permitiendo así ampliar las posibilidades de elección.

Finalizado el estudio teórico de los métodos de resolución, como mencionamos anteriormente, son pocos los trabajos existentes en la literatura que abordan los problemas de *labor scheduling problem.*

En este punto, se plantea la interrogante ¿Qué tipo de heurística o metaheurística aplicar?

Dantzing (1954) fue el primero en formular el problema de *labor scheduling* como un problema matemático. El objetivo de este modelo era minimizar el costo de los turnos programados en el horizonte temporal considerado (ya sea un día o una semana) sujeto a que haya un suficiente número de trabajadores en todos los períodos para satisfacer la demanda.

Otros modelos de *labor scheduling* son los de Moondra (1976), Baker (1976), Keith (1979), Betchold y Jacobs (1990) y (1991), Betchold y otros (1991) y Thompson (1990), (1992), (1995) y (1997).

Dado que generalmente las empresas (públicas o privadas) mantienen un elevado nivel de flexibilidad en sus horarios, el número de turnos disponibles suele ser muy elevado. Además, Bartholdi (1981) mostró que estos problemas son NP-completos. Es decir, se requiere de un tiempo elevadísimo para obtener la solución óptima incluso en problemas de pequeño tamaño. Esto motiva el desarrollo de los convencionales métodos de solución heurísticos y metaheurísticos (más rápidos que los métodos óptimos y que dan lugar a soluciones seudo óptimas). Varias empresas de servicios como L.L. Bean (Andrews y Parsons, 1989 y 1993, y Quinn y otros, 1991), United Airlines (Holloran y Byrn, 1986) y el Servicio de Policía de San Francisco (Taylor y Huxley, 1989) han experimentado importantes actuaciones de mejora después de adoptar estas técnicas.

Tradicionalmente el heurístico más efectivo y convencional para este tipo de problemas combina la programación lineal con algunas formas de Búsqueda Local (Hill Climbing) (Bechtold et. al, 1991, Easton,1991 y Aykin, 1996). Sin embargo, las técnicas de Hill Climbing usadas con los heurísticos más convencionales en general "prohíben" las soluciones infactibles. Esto limita sus caminos de búsqueda a regiones factibles del espacio de búsqueda, sin la posibilidad de "saltar" de una región factible a otra a través de regiones infactibles. Más problemática sin embargo es la tendencia de los métodos de Hill Climbing de converger a mínimos locales no globales. Estas limitaciones sugirieron un mayor potencial para obtener mejoras en las soluciones heurísticas. Recientemente, algunos investigadores (mencionados seguidamente) han propuesto para este trabajo diferentes técnicas metaheurísticas como el Temple Simulado, Búsqueda Tabú y Algoritmos Genéticos para superar algunas limitaciones de los heurísticos convencionales.

Concretamente, en Brusco y Jacobs (1993a y 1993b), Brusco y otros (1995) y Thompson (1996), proponen algoritmos basados en Temple Simulado; en Easton y Mansour (1999) se diseña un Algoritmo Genético; y en Glover y Mc Millan (1986), Taylor y Huxley (1989), Easton y Rossin (1996), Dowsland (1998) y Alvarez-Valdes et. al (1999) se diseñan procedimientos de Búsqueda Tabú.

S. Casado et. al (2003), realizan un estudio comparativo de diferentes estrategias metaheurísticas para la resolución del *labor scheduling problem.* La aportación de este trabajo ha sido el diseño y posterior estudio comparativo de una amplia gama de metaheurísticos para este problema, además de un eficaz tipo de movimientos vecinales. Concretamente, las estrategias que se usan en este trabajo son *GRASP*, *Búsqueda en Entornos Variables, Temple Simulado, Búsqueda Tabú, Algoritmos Genéticos y Algoritmos Meméticos*. Para realizar esta comparación se hacen pruebas con diferentes instancias ficticias con un horizonte temporal de una semana. Hay que indicar que muchas de las estrategias propuestas están basadas en movimientos vecinales, e incluso incorporan procedimientos de Búsqueda Local basados en estos movimientos. Los movimientos vecinales que se proponen en este

trabajo son una cadena o composición de movimientos simples. Además resultan eficaces ya que por una parte permiten visitar regiones no factibles, dando más flexibilidad a la búsqueda, a la vez que se va obteniendo en cada momento la "proyección" de dichas soluciones en la región factible (es decir, la solución factible más cercana). De la comparación realizada entre estos métodos, con problemas de diferente tamaño, destaca el basado en *Búsqueda Tabú*, tanto en calidad de la solución como en rapidez para obtenerla.

Respecto a métodos actuales para la resolución del problema de asignación de personal, se encuentran los métodos exactos como Branch and Bound o Branch & Price.

A. Ernst et. al (2004), presenta una revisión sobre diversas publicaciones, a partir de la cual los autores construyen un modelo para el problema de asignación de personal, separando en módulos.

En los estudios de C. Azmat et. al (2004), G. Eitzen et. al (2004) M. Moz y M. Vaz Pato (2004) y S. Topaloglu et. al (2004) se proporcionan diversos enfoques de programación lineal y métodos exactos, de la Programación Entera Mixta, Programación por Metas, Flujos Multicomodity, Generación de Columnas y Branch& Price, aplicados en distintos tipos de situaciones y problemas, y resueltos utilizando CPLEX de Ilog.

Los artículos de A. Ernst et. al (2004) y A. Corominas et. al (2004) están centrados en el uso de softwares comerciales, tales como: *Carmen Crew Rostering* que fue especialmente diseñado para ser usado por aerolíneas y *Scheduler* en aplicaciones que van desde el cuidado de ancianos hasta producciones de televisión.

En los últimos años se dispone del artículo J. Júdice y otros (2005), que analiza la distribución de correspondencia, donde los objetos de correo son tratados en un proceso de producción en cadena.

S. Sampson (2006) indica que el problema de la asignación de trabajo de voluntarios (VLA) es notablemente diferente al problema de la asignación tradicional de trabajo (TLA). Una diferencia principal pertenece a la estructura de costos de trabajo, en TLA procuran reducir al mínimo costos de trabajo. Otra diferencia es el tamaño asumido del fondo de trabajo: TLA típicamente asume que el trabajo suficiente cubra exigencias de tarea, mientras que el fondo de trabajo de VLA es limitado por el número de los voluntarios que pueden ser reclutados.

S. Casado et. al (2006) desarrollan un sistema para la resolución del problema de programación de horarios de trabajo en un modelo de flujo de pasajeros en un aeropuerto. Concretamente, el trabajo trata de racionalizar los costes del personal de facturación y de seguridad, y a la vez garantizar un nivel de fluidez mínimo requerido en el tránsito de los pasajeros, medido en términos de los tiempos de espera. El sistema está compuesto por tres herramientas principales: un simulador, un método para optimizar simulaciones y un método para resolver el problema de programación de horarios propiamente dicho. Este sistema permite la obtención de una gran diversidad de soluciones facilitando así la tarea del decisor que puede seleccionar la alternativa más adecuada dependiendo de sus preferencias. Las soluciones generadas se aproximan a la curva de eficiencia teniendo en cuenta los objetivos de coste de personal y nivel de servicio ofrecido, utilizando un algoritmo de tipo Búsqueda Dispersa (SS).

D. Ballesteros et. al (2007), señalan que los modelos de programación de la fuerza de trabajo tienden a ser bastante diferentes de los modelos de programación de máquinas. La programación de la mano de obra implica cualquier cambio en la planificación en instalaciones del servicio (por ejemplo, un centro de atención) o planificación de la tripulación en un medio de transporte. Indican también que los modelos de programación de turnos son los más fáciles para formular: para cada intervalo de tiempo hay requerimientos que consideran el número de personal que debe estar presente. Un intervalo de tiempo *i* requiere b_i personas. El personal puede

contratarse para los diferentes turnos y hay un costo asociado con cada tipo de contrato. El objetivo es minimizar el costo total.

Se podría argumentar entonces que la programación por turnos es similar a la programación de máquinas. Durante un intervalo de tiempo i, b_i tareas deben realizarse. En este sentido, la fuerza de trabajo es equivalente a un número de máquinas en paralelo donde se procesan las diferentes tareas. Al menos los recursos b_i deben estar disponibles para hacer estas tareas. Existen restricciones y costos asociados con el mantenimiento de estos recursos, por lo tanto el objetivo de estos modelos es minimizar el costo total.

En la práctica, la programación de la fuerza de trabajo puede estar entrelazada con otras funciones de programación. Por ejemplo, la programación de máquinas puede depender de la programación de turnos y la programación de flotas puede depender de la programación de la tripulación.

La información que se puede analizar está asociada con la duración de la actividad (tiempo de proceso p_{ij}), tiempo de iniciación más temprano (fecha de liberación r_j), tiempo tardío de terminación (o fecha de entrega d_j), nivel de prioridad (ponderación w_j). El nivel de prioridad de una actividad depende del beneficio o costo. Una actividad puede tener más parámetros que especifican, por ejemplo, los recursos adicionales que se exigen para su ejecución. En las empresas de servicios los recursos pueden ser un salón de clase, una sala de juntas, un cuarto de hotel, un estadio, una agencia que alquila vehículos, una sala de cirugía.

L. Pradenas et. al (2008), proponen una metaheurística tipo Tabu Search para la resolución del problema de asignación de supervisores forestales donde la función objetivo consistía en lograr una distribución equitativa de la carga laboral. Este problema consistía en la programación de turnos de trabajos para personas que tienen la misma habilidad, con la incorporación de restricciones que son tanto propias del problema matemático, como las que tienen que ver con los aspectos

organizacionales. El algoritmo propuesto se implementó en la programación de turnos en una empresa de servicios logísticos, que presta servicios a empresas forestales y sus cadenas de suministro.

Finalmente, del análisis bibliográfico presentado, se desprende que:

- Desde 1954, se ha investigado el problema de asignación de horarios de trabajo del personal y en los años siguientes han surgido metodologías y algoritmos que buscan dar solución al problema en distintos tipos de empresas (públicas, privadas, de servicios, etc.).
- Existen softwares comerciales en el mercado, que buscan simplificar el problema de asignación de horarios de trabajo en las empresas.
- En la mayoría de los métodos propuestos, la función objetivo del problema penaliza la violación de restricciones.
- Tabu Search es una buena herramienta para entregar buenas soluciones, en bajo tiempo computacional. Se puede aplicar a problemas como asignación de enfermeras en un hospital, formación de grupos de mantención, asignación de trabajos part-time, comida rápida, horario profesores, etc. Para problemas reales, la inclusión de nuevas restricciones es permitida porque el algoritmo de solución es genérico. El no cumplimiento de restricciones blandas, propias de la organización, implicaría una penalización a la función objetivo. También propone un efectivo tipo de movimientos vecinales entre soluciones.
- Para la comparación con el Algoritmo Tabu Search a elaborar para el problema de asignación de horarios de trabajo, se utilizará la técnica actual con la que cuenta la empresa, dado que es el único sistema que existe para poder comparar.

Derivado del análisis bibliográfico, a continuación, se estudiará en detalle el Algoritmo de Búsqueda Tabú.

3.2. ALGORITMO DE BÚSQUEDA TABÚ

3.2.1. Introducción

Esta técnica fue propuesta por Glover en 1986. Su nombre está relacionado con las acepciones de *tabú* como "prohibición impuesta por las costumbres sociales como una medida de prevención" y "prohibición de algo que constituye un riesgo". Los tabúes sociales suelen pervivir durante un tiempo y luego acaban por extinguirse. Sin embargo, mientras están vigentes, los tabúes no operan como prohibiciones cuando la alternativa prohibida resulta mucho más interesante que el resto.

En el caso de esta técnica, los tabúes se traducen en determinadas prohibiciones en el proceso de búsqueda y el riesgo del que se pretende huir es que la búsqueda quede atrapada en un conjunto de soluciones sin poder salir de él. Para ello, a lo largo de proceso, se almacena información sobre las características de las soluciones visitadas y sobre las operaciones realizadas para obtener dichas soluciones, el proceso tiene memoria de lo ocurrido. Como señalan Glover y Laguna (1997), la memoria, que permite guiar el proceso de búsqueda, es la principal característica de esta técnica; en este sentido, una mala elección puede ofrecer más información que una elección aleatoria y, por tanto, puede ser preferible. Con la memoria es posible orientar la búsqueda hacia soluciones con buenas características a la vez que se exploran nuevas de regiones del espacio de soluciones que son potencialmente interesantes. Existen dos tipos de memoria: la memoria a corto plazo y la memoria a largo plazo. Simplemente con el empleo de la memoria a corto plazo, es posible diseñar un buen proceso de búsqueda, aunque la mejor forma de explotar todas las posibilidades que ofrece esta técnica es el empleo de los dos tipos de memoria.

A veces, como en la analogía del campo social, las prohibiciones dejarán de serlo cuando se cumplan determinadas condiciones y no operarán como tales prohibiciones.

Básicamente, la técnica opera como sigue: dada una solución, se examina un conjunto de soluciones vecinas de aquella y se selecciona la mejor de entre las vecinas, que pasa a ser la nueva solución (aun cuando no sea mejor que la solución de partida). Sin embargo, no todas las soluciones vecinas son soluciones candidatas a ser nueva solución, algunas están *prohibidas*. Este proceso se repite iterativamente hasta que se cumple alguna condición establecida anteriormente.

Con el procedimiento de la búsqueda tabú, que se detalla en los próximos apartados, se pretende evitar la convergencia en torno a óptimos locales. Para ello, se veta el acceso a determinadas soluciones, evitando así (con más o menos éxito, según configuren los parámetros del método) la repetición cíclica de un conjunto de soluciones próximas a un óptimo local y se alienta la exploración de regiones de soluciones potencialmente interesantes. Como indica Glover (1989), con el procedimiento de la búsqueda tabú se trata de evitar la vuelta a *estados de soluciones* anteriores, es decir, a soluciones visitadas anteriormente cuyo mejor movimiento a partir de ella es el mismo que el de la vez anterior en que tal solución fue visitada.

La mayoría de las aplicaciones de la búsqueda tabú se han comenzado a desarrollar a partir de finales de los años ochenta. La búsqueda tabú se ha aplicado, entre otros, a los siguientes problemas: telecomunicaciones, finanzas, ingeniería de organización, medicina, etc.

Se debe apuntar que la búsqueda tabú no es una técnica rígida, sino que se debe confeccionar para cada problema concreto. La forma que adopta el algoritmo depende en gran medida del problema que se pretenda resolver y de la representación que del mismo se adopte. Por lo tanto, este apartado (de carácter

descriptivo) describe los elementos más significativos de la metodología de forma genérica y, en algunos casos, cita ejemplos concretos que sirven para ilustrarla.

3.2.2. Búsqueda Tabú Simplificada. Memoria de Corto Plazo.

En este apartado se describe la modalidad más sencilla de la búsqueda tabú, que se caracteriza por el empleo, únicamente, de la memoria a corto plazo.

El procedimiento parte de una solución inicial, en la forma de un vector ***x***, a partir de la cual obtiene una nueva solución efectuando cambios elementales, que se denominan ***movimientos,*** en la solución de partida. A su vez, a partir de esta solución se obtiene, de la misma forma, otra y así sucesivamente.

Dada una solución, el conjunto de soluciones a las que se puede acceder mediante los movimientos considerados se denomina ***vecindario.***

De entre todas las soluciones que componen el vecindario, existe un conjunto de ellas que están prohibidas (bien por sus características o bien por el movimiento elemental que condujo a las mismas): son las soluciones tabú y durante un determinado número de iteraciones están vetadas.

La nueva solución, en cada transición, es la mejor solución de entre las vecinas. No es una condición necesaria (a diferencia de otros procedimientos) que la nueva solución sea mejor que la existente.

Esta técnica también tiene en cuenta la posibilidad de que los movimientos prohibidos puedan restringir el área de búsqueda y que ciertas regiones que podrían arrojar buenos resultados queden sin explorar. Para evitar que esto ocurra, existe un ***criterio de aspiración:*** es posible aceptar una solución clasificada como tabú como nueva solución si cumple con alguna determinada condición.

Por último, el proceso de búsqueda se detiene cuando se cumple algún criterio de finalización.

A continuación se describen los elementos (tipo de solución inicial, tipo de vecindario, etc.) que integran la búsqueda tabú y algunas de las posibles alternativas para cada uno estos elementos.

Para ilustrar algunos aspectos de la búsqueda tabú, se considerará el problema del viajante con diez ciudades.

- ***Solución Inicial***

Cada una de las posibles soluciones del problema debe poder ser expresada en forma de un vector o una matriz, ***x***. En el problema del viajante, el elemento *x(i)* puede representar la posición en la que se visita la ciudad cuyo índice es *i* o el índice de la ciudad que se visita en lugar *i-ésimo* (esta última será la que emplearemos).

Para comenzar a aplicar esta técnica es necesario disponer de una solución inicial. Esta solución se puede obtener a partir de una heurística (típicamente de aplicación más sencilla), de forma aleatoria o, sencillamente, a partir de una solución conocida (si el problema se refiere a un sistema ya en funcionamiento, la solución del sistema real es una posible solución de partida).

- ***Función Objetivo***

Se trata de la función cuyo valor se pretende hacer máximo o mínimo. En cada una de las iteraciones del procedimiento es necesario calcular el valor de la función objetivo de las soluciones candidatas a ser la nueva solución.

- ***Tipos de Movimientos***

Como se ha indicado anteriormente, la búsqueda tabú explora posibles soluciones a partir de una dada mediante movimientos sencillos. Estos pueden ser, básicamente, de dos tipos:

-Inserción ("*insert*" en la literatura sajona), en la que un determinado elemento (componente de un vector, columna en una matriz, etc.) de la solución, *x(i)*, pasa a ocupar otra posición, *j*. El resto de los elementos de la solución quedan desplazados, como se puede apreciar en la figura 3.4.

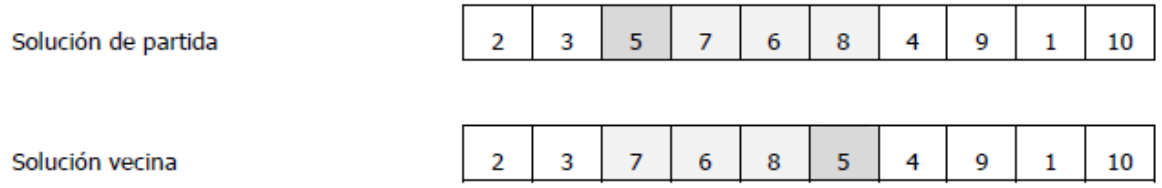


FIGURA 3.4: MOVIMIENTO DE INSERCIÓN

-Intercambio *("swap")*, en el que dos elementos de la solución intercambian su posición (figura 3.5).

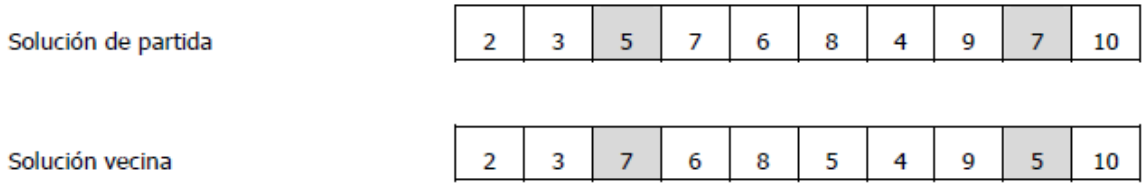


FIGURA 3.5: MOVIMIENTO DE INTERCAMBIO

En ocasiones, para simplificar, se consideran solamente movimientos de inserción, ya que un intercambio se puede considerar como un caso especial de inserción.

En cada caso, dependiendo de la forma que adopte la solución del problema considerado, se debe definir qué significa un movimiento de intercambio o uno de inserción. Además, es posible restringir el abanico de posibilidades permitiendo sólo la selección de movimientos que afecten a elementos de la solución que no estén alejados más de una determinada *distancia.* La distancia significará en cada caso una cosa diferente.

- ***Lista Tabú***

La lista tabú constituye la memoria a corto plazo del proceso de búsqueda.

Mediante la lista se alienta la exploración de diferentes regiones de soluciones y se dificulta la convergencia en torno a un óptimo local.

En cada iteración existirá un conjunto de soluciones tabú. Las soluciones son tabú en función de las características de la misma o en función de las soluciones por las que se haya transitado en el pasado reciente. A continuación se explican los dos posibles motivos por los cuales una solución es tabú.

-En primer lugar, una solución puede ser tabú si posee alguna característica que no está permitida. Estas características se llaman atributos. En cada iteración existe un conjunto de atributos que no están permitidas, son los denominados ***atributos tabú,*** por ejemplo, el valor de un determinado elemento de la solución.

En la configuración de la búsqueda tabú empleada en cada problema, se debe definir de qué manera se convierten en tabú determinados atributos (características de la solución), de tal manera que al pasar de una solución x_1 a otra x_2 mediante un determinado movimiento, se genere de forma automática el o los atributos tabú asociados a la transición anterior. Asimismo se debe definir la longitud de la lista tabú, es decir, el número de iteraciones durante el cual dicho atributo va a ser tabú.

Al pasar de una solución a otra mediante un movimiento de inserción, como en la figura, se puede generar el atributo tabú consistente en que la ciudad 5 no puede ser visitada en tercer lugar.

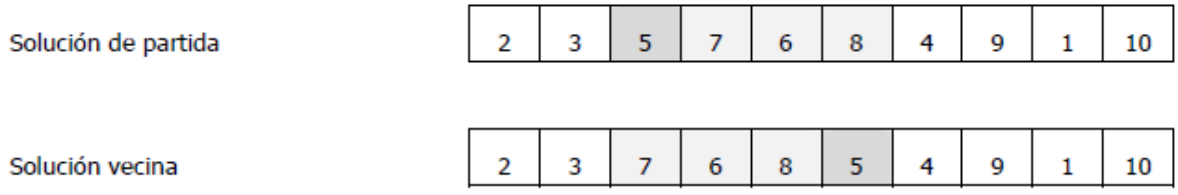


FIGURA 3.6: ATRIBUTOS TABÚ

-En segundo lugar, una solución puede ser tabú si para llegar a ella se ha realizado un movimiento no permitido. Entonces se habla de ***movimiento tabú***.

Como en el caso anterior, la definición de la búsqueda tabú debe señalar el tipo de movimientos tabú asociados a cada iteración y el número de iteraciones durante el que dicho movimiento es, efectivamente, tabú.

En el ejemplo anterior, un posible movimiento tabú generado a partir de la transición considerada puede ser el consistente en insertar la ciudad que ocupe el lugar sexto en el tercero.

A continuación se citan algunos ejemplos de atributos tabú:

- que los elementos que ocupan las posiciones *i* y *j* en la solución tomen los valores *a* y *b* respectivamente;
- que el elemento que ocupe la posición *i* tome el valor *a* o que el elemento que ocupe la posición *j* tome el valor *b*;
- que elemento que ocupe la posición *i* sea mayor que un determinado valor *a*.

Igualmente, los siguientes son ejemplos de movimientos tabú:

− intercambiar los elementos de las posiciones *i* y *j*;
− mover el elemento que está en las posición *i*;
− mover los elementos de las posiciones *i* y *j*;
− mover el elemento *i* a una posición anterior a la posición *a.*

En cada aplicación concreta se debe definir si las soluciones se convierten en tabú por haber llegado a ellas mediante movimientos tabú o porque existen atributos tabú. Lo más habitual es trabajar con atributos tabú y no con movimientos tabú, aunque, de hecho, la definición de un atributo tabú está relacionada con los movimientos que hacen que una determinada solución tenga un atributo o no.

Tanto si el algoritmo trabaja con atributos tabú como si lo hace con soluciones tabú, existe una lista denominada ***lista tabú***. Esta lista registra la información pertinente para identificar las soluciones tabú y no tabú en cada una de las iteraciones. El número de iteraciones que un movimiento o un atributo es tabú viene dado por la longitud de la lista tabú.

La lista tabú puede ser de dos tipos:

− Lista tabú explícita, en la que se almacena el conjunto de todas las soluciones que están prohibidas porque contienen atributos tabú. Esta alternativa consume una cantidad muy elevada de recursos aunque puede ser interesante cuando se utilizan estrategias que registran y analizan ciertas *"soluciones especiales*".

− Lista de atributos, en la que sólo se registran los atributos tabú en cada una de las iteraciones. Por ejemplo, una lista tabú de atributos puede ser como la de la figura 3.7:

Posición en la solución	2	4	1	3	6	8	1	4	2	3
Valor del elemento de la solución	4	5	1	6	2	4	5	7	8	2

FIGURA 3.7: LISTA TABÚ DE ATRIBUTOS

Se trata de una lista tabú de longitud igual a cinco formada por atributos tabú. Las dos primeras columnas indican, por ejemplo, que una solución que contenga el valor 4 en la segunda posición y el 5 en la cuarta se considerará tabú. Cada par de columnas se interpretan de la misma manera. Cuando se alcanza una nueva solución, aparecen dos nuevas columnas, correspondientes a los atributos de la solución que se abandona que pasan a ser tabú y desaparecen las dos columnas correspondientes a la iteración más antigua.

La lista tabú es uno de los elementos más relevantes de esta técnica metaheurística. Con ella se trata de impedir procesos de búsqueda cíclicos en torno a un óptimo local. Se debe tener presente que una lista tabú muy larga puede

impedir que el heurístico explore regiones de soluciones interesantes, mientras que una lista tabú muy pequeña puede conducir a bucles en torno a un óptimo local.

Glover (1989) señala que en las diferentes aplicaciones de la búsqueda tabú se observa que existe un rango de valores amplio para la longitud de la lista tabú que hace que el proceso de búsqueda evite búsquedas cíclicas y, a la ve, ofrezca buenos resultados. Conviene, por ello, al tratar de resolver un problema, realizar una serie de ensayos previos para tratar de buscar cotas a la longitud de la búsqueda tabú. Con estos ensayos se observará que para valores muy pequeños de la longitud, el proceso de búsqueda convergerá muy pronto entorno a óptimos locales; esto se observa cuando aparecen repeticiones periódicas del valor de la función objetivo.

Al contrario, para longitudes muy elevadas se observará un rápido empeoramiento de las soluciones por las que pasa el proceso de búsqueda.

El número de iteraciones durante el cual permanecerá un atributo tabú puede ser diferente. Con ello se pretende alentar la búsqueda en nuevas regiones (cuando el número de iteraciones durante el cual el atributo es tabú es elevado) o para intensificarla en la región en la que está en un determinado instante (cuando el número es pequeño). Esta estrategia constituye una primera aproximación al concepto de estrategia de diversificación y de intensificación, que se comentarán más adelante.

Por ejemplo, si la memoria es explícita, es posible saber si la solución seleccionada en cada iteración ha sido previamente seleccionada, de manera que se puede multiplicar por un determinado factor el tiempo durante el cual el atributo correspondiente será tabú, para hacer menos probable que la búsqueda realice un nuevo ciclo. Para evitar que el número de iteraciones durante el cual un atributo es tabú, L, crezca indefinidamente, es necesario reducir dicho valor con un determinado criterio, por ejemplo, cada vez que transcurra un número determinado de iteraciones (dividiendo L, por ejemplo, por un determinado factor).

- ***Criterio de Aspiración***

Existen casos en los que una solución tabú perteneciente al vecindario puede dejar de serlo si cumple con un determinado criterio de aspiración. Al permitir la selección de soluciones tabú, la búsqueda gana flexibilidad y permite la exploración de regiones de soluciones que pueden arrojar buenos resultados. A continuación se describen algunos criterios de aspiración:

− Criterio de aspiración por defecto. Se trata del caso más simple, según el cual si todas las soluciones en un determinado instante son tabú se elegirá aquella que ha sido tabú durante un mayor número de operaciones desde que fue catalogada como tal, es la solución "*menos tabú*" porque es aquella a la que le quedan menos iteraciones para dejar de ser tabú.

− Criterio de aspiración determinado por la función objetivo. Una solución dejará de ser tabú si el valor de la función objetivo correspondiente es mejor que el de la mejor solución encontrada hasta el momento (carácter global) o que la mejor solución del espacio de soluciones consideradas en ese instante (carácter local).

− Criterio de aspiración según la dirección de búsqueda. Un movimiento *m* que introduce un atributo tabú se acepta siempre y cuando si, primero, introduzca una mejora en la función objetivo y, segundo, que en la transición que dio lugar a que dicho movimiento *m* fuera tabú se realizó un movimiento que mejorara la función objetivo.

− Criterio de aspiración según la influencia del movimiento. Existen movimientos cuya influencia sobre la evolución del algoritmo es mayor que la de otros. Es decir, determinados movimientos conducen a nuevas regiones de soluciones mientras que otros no; los primeros tienen mayor influencia sobre la evolución de la búsqueda que los primeros. En este caso, se debe definir y

cuantificar el nivel de influencia de un determinado movimiento. Según este criterio, un movimiento (o un atributo correspondiente a un movimiento) de poca influencia deja de serlo si tras él se ha realizado otro de mayor influencia. Esto es así porque la posibilidad de que tenga lugar un proceso cíclico es mucho menor.

Por ejemplo, en el problema del viajero, si representamos la solución como la ordenación de las ciudades por las que se desplaza, podemos cuantificar la influencia de un movimiento a partir de la distancia entre las ciudades que se ven afectadas por el movimiento.

- ***Lista de Candidatos. Exploración del Vecindario.***

En cada iteración es necesario considerar un conjunto de soluciones de entre las cuales se seleccionará la próxima solución. La exploración de todas las soluciones candidatas puede exigir mucho tiempo. En estos casos conviene explorar el vecindario de una manera más eficiente. A continuación se citan algunas de las posibles formas para generar las candidatas en cada iteración.

− Exploración completa. Una primera forma de estudiar el vecindario consiste en evaluar todas y cada una de las soluciones vecinas que son accesibles mediante los movimientos definidos. En ocasiones esto puede ser muy costoso en términos de tiempo y por eso se emplean otros métodos más eficientes.

− Exploración aleatoria, en la que se acepta como nueva solución la primera solución vecina que se encuentre mejor que la actual. De esta forma es más probable acceder a ciertas regiones con buenas soluciones.

− Exploración por aspiración[23]. En este caso se estudia el vecindario solución por solución hasta que se encuentra una solución vecina cuyo valor de la función

[23]En la literatura sajona, este método de exploración se conoce como *"aspiration plus".*

objetivo supera un determinado umbral. Una vez obtenida esta solución, se estudian otras *M* soluciones más y se escoge la mejor de entre todas las estudiadas. Generalmente se obliga a este método a estudiar un número mínimo y un número máximo de soluciones.

− Exploración mediante una lista de movimientos de élite. En una determinada iteración se estudia un número elevado de soluciones vecinas. De éstas, se seleccionan las *k* mejores y se almacenan los movimientos que han conducido a la obtención de dichas soluciones en una lista de movimientos de élite. En la presente iteración se utilizará el primer movimiento de la lista, en la siguiente el segundo y así sucesivamente hasta que finalmente un movimiento cae por debajo de un determinado umbral o transcurre un determinado número de iteraciones. Cuando una de estas condiciones se cumple se construye de nuevo la lista de movimientos de élite y se procede de la misma manera.

− Exploración mediante abanicos de soluciones. Con este método, en una determinada iteración se examina un conjunto de soluciones de las que se escogen las *p* mejores. A su vez se repite el proceso con estas soluciones obteniendo de nuevo las *p* mejores. Así un número de veces determinado, al cabo de las cuales se obtiene un conjunto de *p* caminos a partir de la solución inicial. El camino que se escoge es el mejor de todos ellos. A partir de la solución en la que termina el proceso comienza de nuevo la exploración. Con este método, en ocasiones, los caminos tendrán alguna solución en común, por lo tanto, será necesario general alguna solución adicional para mantener siempre *p* caminos.

- ***Criterio de Detención***

En cada transición se debe estudiar si el proceso de búsqueda se debe detener o no. Existen cuatro criterios que se utilizan típicamente para finalizar el proceso de búsqueda y admitir una solución como buena. Son los siguientes:

1. haber efectuado un determinado número de iteraciones desde el comienzo del proceso de búsqueda;
2. haber realizado un determinado número de iteraciones sin mejorar la solución;
3. haber alcanzado un determinado tiempo de computación;
4. haber logrado un valor de la función objetivo suficientemente bueno.

En la figura 3.8 se muestra un diagrama de flujo en el que se muestra cómo opera la búsqueda tabú con memoria a corto plazo.

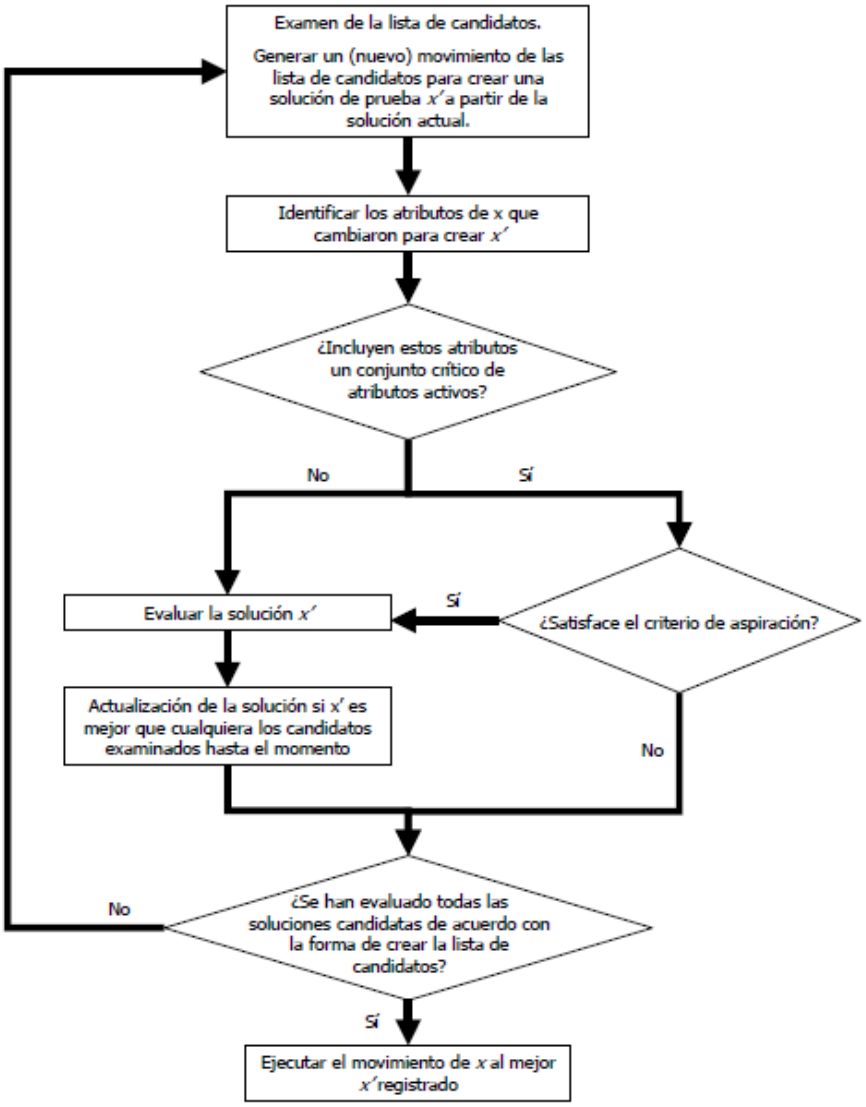


FIGURA 3.8: DIAGRAMA DE FLUJO DE LA MEMORIA A CORTO PLAZO

3.2.3. Búsqueda Tabú y Memoria de Largo Plazo.

La búsqueda tabú, tal y como se ha formulado en el apartado anterior, incorpora una memoria (en forma de la lista tabú). Esta memoria se refiere sólo al corto plazo y, en ocasiones, puede ser adecuada para la resolución de determinados problemas.

Sin embargo, la búsqueda tabú puede ser más potente si, además de la memoria acorto plazo, se incorporan estructuras de memoria a largo plazo que permitan seguir estrategias más eficientes.

Con el empleo de la memoria a largo plazo se registran soluciones denominadas *de élite* con objeto de intensificar la búsqueda hacia regiones potencialmente atractivas (estrategias de intensificación) o bien para orientar la búsqueda hacia regiones no suficientemente exploradas (estrategias de diversificación).

- ***Memoria de Frecuencia***

La memoria de frecuencia a largo plazo consta de dos indicadores: el indicador de permanencia y el indicador de transición. Ambos indicadores se expresan en forma de ratios. En el caso del indicador de permanencia, el numerador del ratio es el número de iteraciones durante las cuales la solución seleccionada poseía un determinado atributo. En el indicador de transición, el numerador es el número de veces que un determinado movimiento ha tenido lugar o un determinado atributosale o entra de la solución.

El denominador, en ambos casos puede ser (1) el número total de iteraciones realizadas en el proceso de búsqueda, (2) el valor del máximo numerador o (3) la suma de todos los numeradores.

En el problema del viajante, un indicador de permanencia puede ser el número de veces que la ciudad C_1 ha ocupado el lugar *i-ésimo* en la solución. Un posible indicador de transición es el número de veces que intercambian su posición las ciudades C_1 y C_2.

- ***Estrategias de Intensificación***

Las estrategias de intensificación modifican las reglas de selección de soluciones vecinas para explorar con mayor profundidad determinadas regiones del espacio de soluciones:

− alentando la aparición de determinadas características de las nuevas soluciones o fomentando determinados movimientos que presumiblemente conduzcan a regiones atractivas o

− volviendo a regiones de soluciones ya visitadas con intención de explorarlas de forma más exhaustiva.

Una forma incorporar las estrategias de intensificación consiste en tener un registro de *k* soluciones, las *k* mejores visitadas durante el proceso. Cuando se encuentra una solución mejor que cualquier otra visitada hasta entonces se añade al final de la lista y se toma la primera solución $\boldsymbol{x}^1$ de la lista como punto de partida del proceso de búsqueda. Además de los atributos activos que estén operando en ese momento, se tienen en cuenta los atributos tabú asociados a la solución $\boldsymbol{x}^1$ que se generaron cuando esa solución se aceptó como nueva solución en el proceso de búsqueda. De esta manera el proceso de búsqueda vuelve a una solución visitada anteriormente pero, sin embargo, el camino iniciado es otro y los atributos tabú son, en general, diferentes, de modo que cabe esperar que de esta forma se exploren soluciones del entorno de una solución interesante no exploradas con anterioridad.

Otra forma que adoptan las estrategias de intensificación son las denominadas *estrategias de intensificación por descomposición.* Estas estrategias consisten en la

construcción de soluciones de partida del proceso de búsqueda mediante la combinación de elementos o atributos interesantes de las soluciones visitadas. Para ello es útil la memoria a largo plazo: los indicadores de permanencia indican si un atributo es atractivo o no, dependiendo de que cuando aparezca, las soluciones que lo contienen sean de calidad o no respectivamente. Si un atributo aparece ligado sistemáticamente con soluciones de buena calidad conviene intensificar la búsqueda en regiones de soluciones que ofrezcan dicho atributo y, al contrario, si las soluciones son de mala calidad conviene abandonar la búsqueda en tales regiones.

Un ejemplo de esta segunda opción para el problema del viajante es la siguiente consiste en identificar las ciudades que están en los extremos del recorrido de muchas soluciones de elevada calidad y dejarlos "atrapados" en el interior del recorrido para modificar otras partes de la solución.

- ***Estrategias de Diversificación***

Con las estrategias de diversificación se persigue la exploración de nuevas regiones. Para ello, en primer lugar, se puede modificar el valor de la función objetivo de manera que aumente o disminuya según los atributos que contenga dicha solución hayan aparecido en pocas o en numerosas ocasiones a lo largo de la exploración. En un problema de maximización, se define el valor de un movimiento como el incremento de la función objetivo. Cuanto mayor es el valor de un movimiento, más atractiva es la solución a la que conduce ese movimiento.

El valor del movimiento puede modificarse de la siguiente manera:

$$valor\ del\ movimiento = valor\ del\ movimiento - d \cdot penalización$$

donde la *penalización* suele depender de la frecuencia del atributo (medida según los ratios descritos antes) y *d* es un parámetro de diversificación. Las penalizaciones negativas o *incentivos* tienen el objetivo de alentar la aparición

determinados atributos y las penalizaciones positivas, inhibir la aparición de dichos atributos. Por su lado, los valores elevados de *d* indican una mayor tendencia a la diversificación (a explorar nuevas regiones). Para diversificar el proceso de búsqueda también se puede, periódicamente, reiniciar el proceso con una nueva solución de partida construida a partir de la información almacenada de la memoria a largo plazo, fomentando la aparición de atributos que han aparecido con poca frecuencia e inhibiendo la de aquellos que han aparecido en más ocasiones.

Por otro lado, si la permanencia de un determinado atributo conduce a soluciones tanto de buena como de mala calidad, el atributo puede estar *anclando* la búsqueda a una región sin resultados interesantes y conviene deshacerse de él. Al contrario, si la aparición poco frecuente de un atributo conduce al mismo resultado de antes quizá la búsqueda esté siendo restringida al no incluir regiones con dicho atributo y convenga introducirlo. En ambos casos se trata de diversificar la búsqueda.

Existen diferentes formas de realizar diferentes etapas, unas en las que se intensifica el proceso de búsqueda y otras en las que se diversifica. Sin embargo, no se conoce suficientemente bien cómo combinar intensificación y diversificación de la manera más eficiente

- ***Estructura del Proceso***

Las estrategias de diversificación y de intensificación se introducen en el proceso de búsqueda de forma conjunta con los pasos descritos anteriormente para la búsqueda tabú con memoria de corto plazo.

Al final de cada iteración se comprueba si se satisface alguna determinada condición que active una fase de diversificación o de intensificación. Por ejemplo, si transcurrido un determinado número de iteraciones la solución no ha mejorado, se inicia una fase de diversificación: se modifican el valor de los movimientos según se ha descrito anteriormente durante un número de iteraciones o hasta que no se

cumpla alguna condición, o se reinicializa el proceso generando una nueva solución, etc.

Igualmente, en determinados instantes y cuando se cumpla una determinada condición, se inicia una etapa de intensificación utilizando, por ejemplo, alguna de las medidas descritas más arriba.

3.2.4. Oscilación Estratégica.

El objetivo es permitir la búsqueda en determinadas regiones no exploradas mediante la admisión como soluciones actuales soluciones no factibles. Las soluciones no factibles tienen una penalización. La penalización se modifica según el número de veces que el algoritmo haya transitado por soluciones no factibles.

3.2.5. Conclusiones del Algoritmo de Búsqueda Tabú.

La búsqueda tabú ofrece un conjunto muy amplio de posibilidades para la resolución de problemas combinatorios. El método se basa en la acumulación de información sobre el proceso de búsqueda.

De forma sucinta comentaremos que, con respecto a la convergencia de la búsqueda tabú, el proceso de búsqueda ordinario, de carácter determinista, no converge a un óptimo local. Sin embargo, se puede demostrar que la búsqueda converge asintóticamente al introducir elementos estocásticos (definiendo funciones de probabilidad que gobiernen el proceso de selección de candidatos, el grado de reversibilidad de determinados movimientos así como la aplicación del criterio de aspiración).

Es posible diseñar procesos de búsqueda sencillos que ofrecen ventajas frente a otros métodos. Se pueden diseñar búsquedas más elaboradas cuyo desarrollo exige más tiempo pero son más eficientes.

En este capítulo introductorio se han comentado los aspectos más relevantes, pero existen muchas otras consideraciones que dan lugar a diferentes alternativas que, en cada caso, hacen que la búsqueda tabú adopte formas particulares.

Por ello, lo que refuerza la elección del algoritmo de Búsqueda Tabú para la resolución del problema, se fundamenta en los siguientes aspectos:

- Porque es el método más abordado en la literatura existente para problemas de asignación de horarios de trabajo, además de su aplicación en empresas reales.

- Porque a través de los movimientos vecinales permite visitar regiones no factibles dando más flexibilidad a la búsqueda.

- Porque permite entregar buenas soluciones en un tiempo computacional razonable.

- Por la validación de autores al realizar estudios comparativos de diferentes estrategias metaheurísticas para la resolución del *labor schedulling problem*.

- Por la visualización personal del problema planteado.

CAPÍTULO 4: MODELAMIENTO DEL PROBLEMA MEDIANTE ALGORITMO DE BÚSQUEDA TABÚ.

En este capítulo se describe la aplicación del algoritmo elaborado para la asignación de los colaboradores (mucamas) a los turnos necesarios por la empresa en un período de cuatro semanas. Inicialmente, esta labor es realizada por diferentes colaboradores de forma manual y está basada en la experiencia que tenga cada uno de ellos en esta labor, la cual implica un tiempo de aproximadamente un mes para realizar esta asignación. Para la solución del problema en cuestión se elaboró una metodología que permitió la aplicación del algoritmo Tabu Search, el cual asignó a los colaboradores de forma de poder cubrir la necesidad de personal necesaria en bloques de media hora. De esta forma, el objetivo establecido fue disminuir la brecha entre lo asignado y lo necesitado en cada bloque, esto considerando la situación en cuanto existe más personal que lo necesario, así como el personal faltante en un bloque o turno determinado.

4.1. DESCRIPCIÓN DEL PROBLEMA

El problema está dado de la siguiente manera; un gerente está encargado de definir la necesidad de personal que es necesaria para los distintos días de la semana, los cuales se repiten durante el mes. Conocida la necesidad por turno que requiere para una sucursal en particular, existe una persona que en base a su dotación de personal, realiza la programación de turnos para la necesidad entregada. Para realizar esta programación se debe tener en cuenta todos los aspectos legales que impidan que un trabajador pueda ser asignado a un turno determinado. Por otro lado, se trabajó con el mismo sistema de representación con que se trabaja actualmente de la forma manual, esto es, por cada media hora existe una necesidad de personal a cubrir por los colaboradores por lo que existe 1343 bloques de media

hora para cubrir las cuatro semanas los cuales contienen una necesidad que debe ser cubierta por un colaborador.

Por otro lado, se definió que cada trabajador tendrá turnos de 8.5 horas lo que equivale a 17 medias horas, y dentro de esta 17 medias horas, existe 1 hora para colación la cual debe ser tomada de forma continua.

Además, de las restricciones propias a lo legal, existen restricciones de operación como son que un colaborador no pueda trabajar debido a que se encuentra con permiso o de cumpleaños. Para este caso, la asignación no puede considerar a dicho colaborador en un día determinado por 24 horas o 48 bloques de media hora el cual será fijado con el valor -1.

4.2. INFORMACIÓN DE ENTRADA

Dado lo anterior, la información de entrada con la cual funcionará el problema viene dado por (Anexo 8.1):

N_Colaboradores: Indica el número de colaboradores con lo que cuenta el programador de turnos en un mes específico.

Necesidad: Esto es la necesidad de personal que necesita cada bloque de media hora según el día de la semana.

L: Necesidad requerida de Lunes a Jueves en el mes
V: Necesidad requerida para el día viernes en el mes.
S: Necesidad requerida para el día Sábado en el mes.
D: Necesidad requerida para el día Domingo en el mes.

Restricciones: Marcadas con las siglas RT Indica que trabajador no puede trabajar un determinado día.

Contrato: Indica el tipo de contrato el cual posee el colaborador los cuales pueden ser de 45, 30 y 20 hrs. de trabajo semanales.

4.3. MODELO DE SIMULACIÓN

4.3.1. Heurística Constructiva

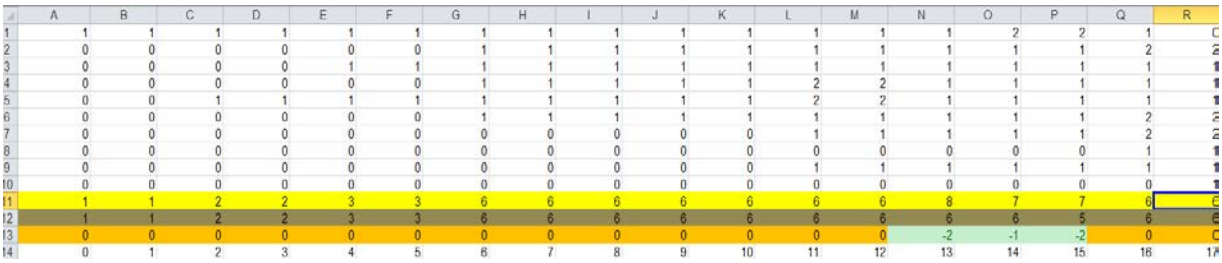
Se realiza la asignación según la necesidad de cada bloque y los colaboradores asignados son escogidos de forma aleatoria según su disponibilidad, esto es, que puedan ser asignados sin que rompan ninguna restricción de descanso o máximo de días trabajados por semana.

En caso de que ninguno cumpla para poder ser asignado, el proceso se termina en ese bloque y se continúa asignando con el siguiente bloque. Cada asignación será marcada con un 1.

TABLA 4-1: BLOQUE DE UNA NECESIDAD DE 1/2 HORA.

Colaborador 1	1
Colaborador 2	1
Colaborador 3	1
Colaborador 4	1
Colaborador 5	1
Colaborador 6	1
Colaborador 7	1
Colaborador 8	0
Colaborador 9	1
Colaborador 10	0
Necesidad	8

Pero también hay que tener en cuenta que una asignación por bloque no implica un bloque, si no, 17 bloques continuos, por tanto al buscar asignaciones para el bloque en cuestión, afecta indirectamente a 17 bloques más adelante.



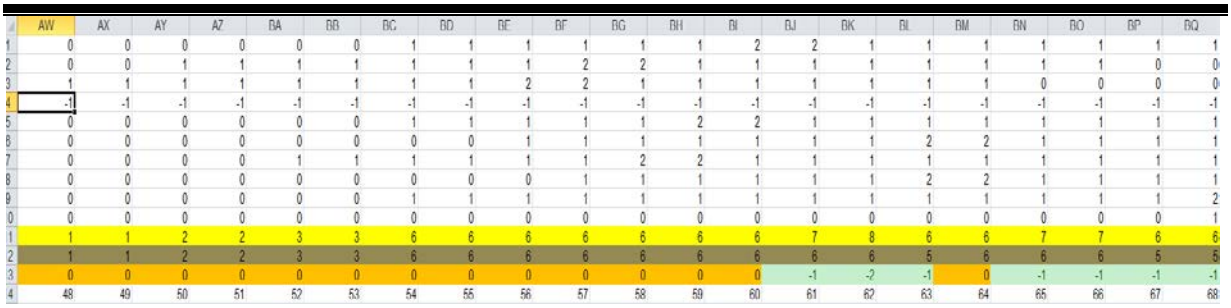
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2	2	1	C
2	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	2	2
3	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
4	0	0	0	0	0	0	1	1	1	1	1	2	2	1	1	1	1	1
5	0	0	1	1	1	1	1	1	1	1	1	2	2	1	1	1	1	1
6	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	2	2
7	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	2	2
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
9	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
11	1	1	2	2	3	3	6	6	6	6	6	6	6	8	7	7	6	E
12	1	1	2	2	3	3	6	6	6	6	6	6	6	6	6	5	6	E
13	0	0	0	0	0	0	0	0	0	0	0	0	0	-2	-1	-2	0	C
14	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17

FIGURA 4.1: ASIGNACIÓN DE 17 BLOQUES

En la figura (ver Figura 4.1), cada fila representa a un colaborador y cada columna a un bloque horario dentro del mes, en ella se puede apreciar que en la fila13, columna N, las asignaciones fueron consecuencia de otras asignaciones realizadas con anterioridad y además estas quedaron con un exceso de personal (celda verde). Para este caso, la heurística constructiva se salta ese bloque donde necesitaba 6 personas (celda Café), dado que tiene asignadas a 8 personas para cubrir ese bloque (necesidad con celda amarilla).

Notar también que en las secuencias se pueden apreciar números con el valor 2, esto simplemente significa que en esa hora el colaborador se encuentra en colación, por tanto no es sumado este valor como asignación.

Por otro lado, la ley exige que los contratos que tienen 45 hrs. semanales, a lo menos deban tener dos domingos libres en el mes. Para esto se eligió aleatoriamente los dos domingos libres y además se eligió aleatoriamente los dos días libres faltantes dentro de la semana. Por simplicidad del problema, una vez fijados estos días libres, son inamovibles, por tanto no se puede insertar una secuencia dentro de estos días. A estos días al igual que los días con permiso se les marco con un -1 en su asignación (días no asignables, Ver Figura 4.2).



	AW	AX	AY	AZ	BA	BB	BC	BD	BE	BF	BG	BH	BI	BJ	BK	BL	BM	BN	BO	BP	BQ
1	0	0	0	0	0	0	1	1	1	1	1	1	2	2	1	1	1	1	1	1	1
2	0	0	1	1	1	1	1	1	1	2	2	1	1	1	1	1	1	1	1	0	0
3	1	1	1	1	1	1	1	1	2	2	1	1	1	1	1	1	1	0	0	0	0
4	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
5	0	0	0	0	0	0	1	1	1	1	1	2	2	1	1	1	1	1	1	1	1
6	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	2	2	1	1	1	1
7	0	0	0	0	1	1	1	1	1	1	2	2	1	1	1	1	1	1	1	1	1
8	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	2	2	1	1	1	1
9	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
11	1	1	2	2	3	3	6	6	6	6	6	6	6	7	8	6	6	7	7	6	6
12	1	1	2	2	3	3	6	6	6	6	6	6	6	6	6	5	6	6	6	5	5
13	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	-2	-1	0	-1	-1	-1	-1
14	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68

FIGURA 4.2: ASIGNACIÓN DE DÍA LIBRE (-1)

4.3.2. Función Objetivo y Penalización

Para la función objetivo, se realizó una minimización de la sumatoria de las asignaciones v/s la necesidad operativa de cada bloque, esto es:

$$\text{Minimizar Objetivo} \sum_{k=0}^{1343} (N_k - A_k) * \alpha \begin{cases} N_k - A_k < 0; & \alpha = 1.5 \\ N_k - A_k > 0 \ \wedge \ A_k = 0\,; & \alpha = 3 \\ N_k - A_k > 0 \ \wedge \ A_k \neq 0; & \alpha = 2 \end{cases}$$

ECUACIÓN 4-1: FUNCIÓN OBJETIVO

En donde N_k es la necesidad del bloque k y A_k es la cantidad asignados en el bloque k, de esta forma si existen más asignados que lo que necesitamos, el excedente lo multiplicaremos por 1.5. En caso de que exista menos cantidad de lo asignado, separamos dos casos; si no existe nada asignado en ese bloque significa que no hay ningún trabajador en ese turno, lo que es sancionado con un valor de 3 y en caso de que existan colaboradores asignados a un bloque, pero aun así falte personal, esto se penaliza con 2.

Además de lo anterior, existe otra penalización que es de tipo prohibida, y es que en algunos movimientos de mejoramiento se dará la situación que una o varias asignaciones fueron insertadas en lugares donde se rompen restricciones de tipo

legal. En esta situación cada una de estas asignaciones son penalizadas con un valor de 20.

Por tanto la función objetivo quedaría se la siguiente forma

$$\text{Minimizar Objetivo} \sum_{k=0}^{1343} (N_k - A_k) * \alpha \ * (p * 20)$$

ECUACIÓN 4-2: FUNCION OBJETIVO PENALIZADA

Donde p es el número de infracciones encontradas en la asignación global

4.3.3. Mejoramiento y Búsqueda Tabú

Para el mejoramiento de realizar el movimiento entre vecinos, se estudió una serie de movimientos experimentales, pero a medida que se fue evaluando el problema, se estableció que los movimientos verticales entre las asignaciones no modificaban en nada la función objetivo dado que si una asignación de un trabajador era movida hacia otro trabajador, la función objetivo permanecía intacta dado que no modificaba en nada el número de la asignación o era muy poco el cambio, además de ser casi de inmediato la solución penalizada, dado que una inserción dentro de la semana a otro trabajador, era agregarle un día más de trabajo y la heurística de construcción habitualmente dejaba al máximo la cantidad permitida de asignaciones a un trabajador. Por tanto, este tipo de movimiento provocaba que el algoritmo fuese sumamente ineficiente.

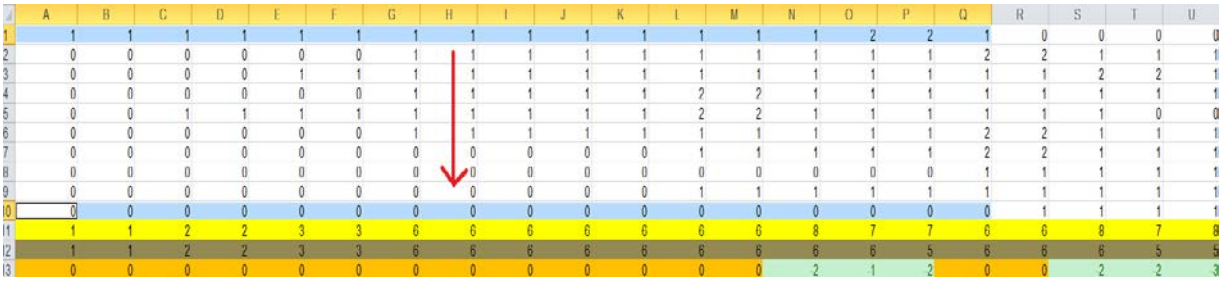


FIGURA 4.3: MOVIMIENTOS DE BLOQUES INVALIDOS

En la Figura 4.3, se puede apreciar que el movimiento de la secuencia del trabajador 1 al ser movido al trabajador 10, la sumatoria de los trabajadores asignados seguirá siendo la misma y además si fuese insertada provocaría de inmediato una infactibilidad dado que después de una asignación debe haber por lo menos 12 horas de descanso, lo que aquí no ocurriría dado que en la columna R existe inmediatamente una asignación.

Por tanto los únicos movimientos permitidos y que cumplen con el vecindario es mover la secuencia dentro del mismo trabajador y dentro de la misma semana (Ver Figura 4.4)

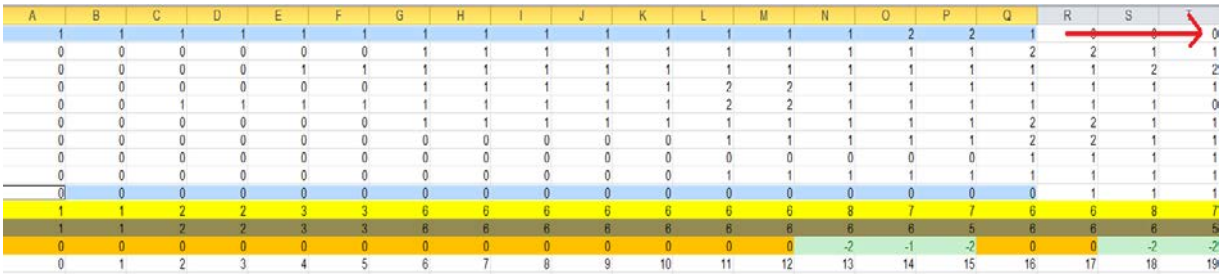


FIGURA 4.4: MOVIMIENTOS DE BLOQUES VALIDOS

De esta forma cambia la función objetivo y se producen mínimas penalizaciones por quiebres de restricciones. Este movimiento de desplazamiento lo realizará en todos los que pueda y se insertará entre asignaciones, siempre y cuando este dentro de la misma semana y cumpla con las restricciones legales.

La mejor función objetivo de los "n" movimientos realizados o "n" vecinos se guardará como lista tabú y con esta solución se buscará mover otra secuencia en otro trabajador, los cuales son elegidos de forma aleatoria.

La cantidad de movimientos que se puedan realizar en un mismo origen o solución inicial estará dado por parámetros, así como también el largo de la lista tabú y el número de iteraciones (soluciones iniciales) a realizar.

El algoritmo realizado viene dado por la siguiente diagrama de flujo (ver figura 4.5), el cual muestra la secuencia de cómo opera el algoritmo de forma sencilla.

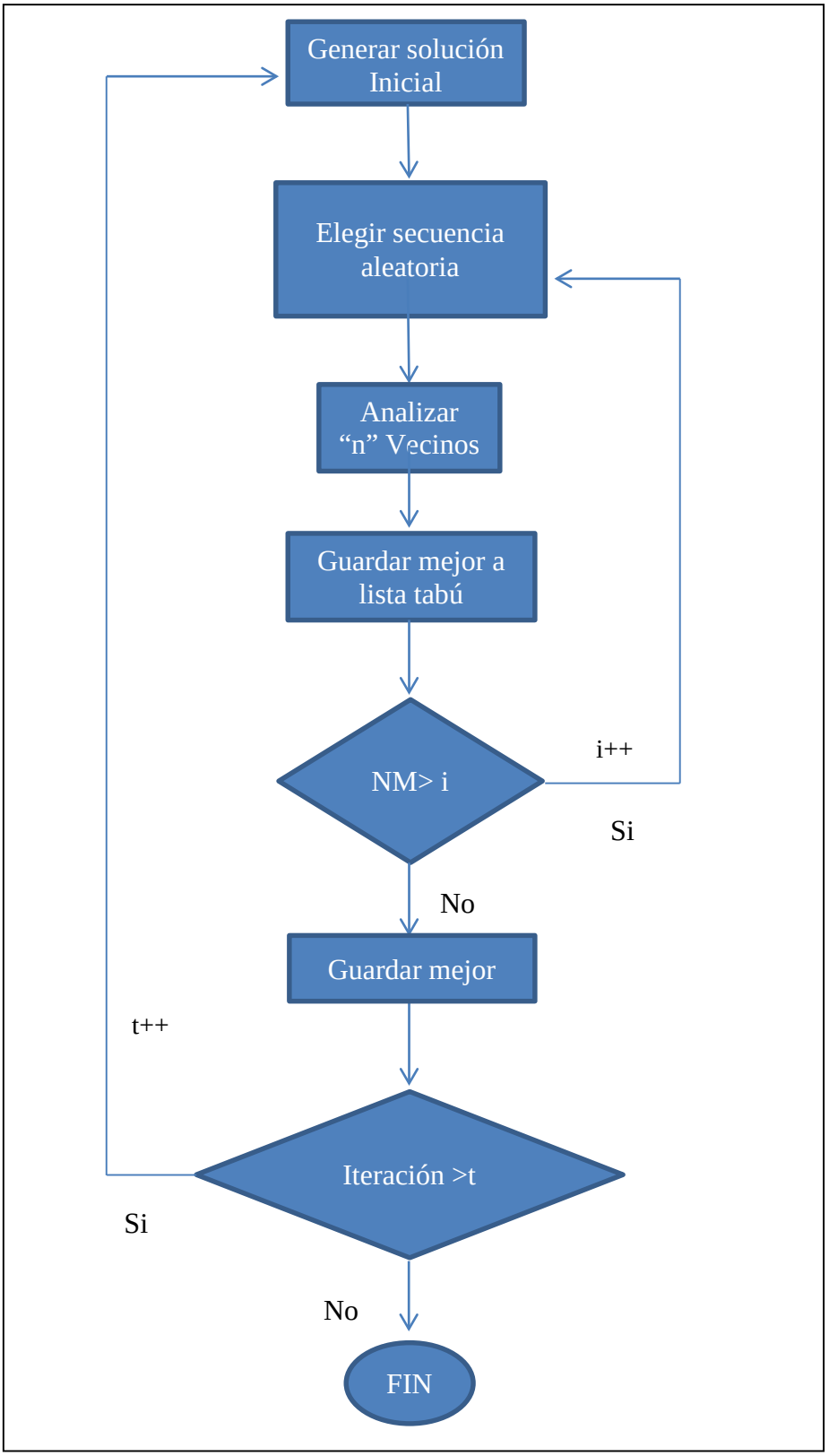


FIGURA 4.5: ALGORITMO TABU

4.4. PARÁMETROS DEL ALGORITMO Y EJECUCIÓN

Los parámetros necesarios para correr el algoritmo son:

Número de Iteraciones (t): es el número de soluciones iniciales que se generará de forma aleatoria, esto de forma que todas las soluciones base sean distintas y de esta forma provocar la diversificación.

Número de Movidas (NM): Número de secuencias que serán elegidas para realizar inserción con sus vecinos.

Lista tabú: Tamaño que tendrá la lista tabú para recordar intercambios ya vistos.

A objeto de encontrar las mejores soluciones al problema de reasignación de los turnos, se realizaron las pruebas correspondientes para ajustar los parámetros que utiliza el algoritmo para realizar los cálculos. Esto se hizo variando los parámetros correspondientes al número de iteraciones, al número de movimientos y el largo de la lista tabú, con el fin de determinar los valores de los parámetros con los que el algoritmo entregó los mejores resultados. El buen ajuste de parámetros permite que las soluciones encontradas sean de mejor calidad en un menor tiempo computacional, lo que significa que un acertado ajuste significa mejores valores para el objetivo planteado. Se realizaron 20 corridas por cada prueba con la finalidad de visualizar con mayor efectividad los resultados que entregaron cada combinación de los parámetros en evaluación.

La Tabla 4-2 presenta los ajustes de parámetros finales con los cuales se encontraron las mejores soluciones:

TABLA 4-2: PARÁMETROS FINALES. [ELABORACIÓN PROPIA]

PARÁMETROS	Algoritmo Tabu
Número de Iteraciones	100
Número de Movidas	200
Lista Tabú	20

A pesar que con estos parámetros se obtuvieron los mejores resultados, resulta interesante mencionar que en este punto no se logró la convergencia, si no que esta se logró con los parámetros mostrados en la Tabla 4-3, pero estos parámetros no lograron los mejores resultados, lo que es un punto de estudio para futuras investigaciones.

TABLA 4-3: PARÁMETROS CONVERGENCIA. [ELABORACIÓN PROPIA]

PARÁMETROS	Algoritmo Tabu
Número de Iteraciones	100
Número de Movidas	400
Lista Tabú	20

Para la ejecución del algoritmo, se utilizó un PC Intel Core i5-3210M de 2.50 GHz con 4 Gb de RAM y 1 Tb de disco duro. El tiempo de ejecución del programa, teniendo en consideración los parámetros que se señalarán en el próximo Capítulo, es de 15 segundos aproximadamente.

4.5. RESULTADOS OBTENIDOS Y ANÁLISIS COMPARATIVO

Se dispone de los datos correspondientes a la necesidad operativa de las mucamas (anexo 8.1) para un mes determinado, en donde el programador de turnos dispone de 10 colaboradores. Se dispone de las necesidades de personal para cada media hora de lunes a jueves, viernes, sábado y domingos. Además el programador de turnos maneja la información que los trabajadores numerados con los valores 1,5 y 8 no podrán asistir a trabajar los días del mes 27, 3 y 19 respectivamente.

Cada uno de las 20 pruebas, se corrió con los parámetros indicados en las tablas, esto es por ejemplo en la Tabla 4.4, se corrió con 100 iteraciones, 100 movidas y 20 el tamaño de la lista tabú (ver tabla 4.4). De esta forma, se obtuvieron los siguientes resultados por cada 20 muestras con parámetros diferentes:

TABLA 4-4: PARAMETROS PRUEBA 1

Iteraciones	Movidas	Tabu
100	100	20
7 seg		
1	fitness:	3269
2	fitness:	3276
3	fitness:	3279
4	fitness:	3367,5
5	fitness:	3389,5
6	fitness:	3397
7	fitness:	3400
8	fitness:	3404
9	fitness:	3405
10	fitness:	3412
11	fitness:	3419,5
12	fitness:	3426,5
13	fitness:	3436
14	fitness:	3440
15	fitness:	3450
16	fitness:	3465,5
17	fitness:	3466
18	fitness:	3484
19	fitness:	3488
20	fitness:	3528
	Promedio	3410,125

TABLA 4-5: PARAMETROS PRUEBA 2

Iteraciones	Movidas	Tabu
100	100	100
7 seg		
1	fitness:	3289
2	fitness:	3356,5
3	fitness:	3364,5
4	fitness:	3365,5
5	fitness:	3366,5
6	fitness:	3373
7	fitness:	3375,5
8	fitness:	3377
9	fitness:	3388,5
10	fitness:	3404,5
11	fitness:	3417
12	fitness:	3430
13	fitness:	3438,5
14	fitness:	3444,5
15	fitness:	3462
16	fitness:	3463
17	fitness:	3463
18	fitness:	3463,5
19	fitness:	3474,5
20	fitness:	3478
	Promedio	3409,725

TABLA 4-6: PARAMETROS PRUEBA 3

Iteraciones	Movidas	Tabu
100	200	20

11 seg

1	fitness:	3139
2	fitness:	3230,5
3	fitness:	3246,5
4	fitness:	3258
5	fitness:	3258,5
6	fitness:	3286,5
7	fitness:	3297,5
8	fitness:	3303,5
9	fitness:	3308
10	fitness:	3311,5
11	fitness:	3337,5
12	fitness:	3361,5
13	fitness:	3373
14	fitness:	3381
15	fitness:	3404,5
16	fitness:	3425
17	fitness:	3436
18	fitness:	3441,5
19	fitness:	3449,5
20	fitness:	3497
	Promedio	3337,3

TABLA 4-7 PARAMETROS PRUEBA 4

Iteraciones	Movidas	Tabu
100	200	100

11 seg

1	fitness:	3179
2	fitness:	3254,5
3	fitness:	3261
4	fitness:	3261
5	fitness:	3264
6	fitness:	3316
7	fitness:	3321
8	fitness:	3322,5
9	fitness:	3332
10	fitness:	3349,5
11	fitness:	3356
12	fitness:	3357
13	fitness:	3365
14	fitness:	3371,5
15	fitness:	3385,5
16	fitness:	3388
17	fitness:	3408,5
18	fitness:	3431
19	fitness:	3446
20	fitness:	3452
	Promedio	3341,05

TABLA 4-4: PARAMETROS PRUEBA 5

Iteraciones	Movidas	Tabu
300	200	20

27 seg

1	fitness:	3132,5
2	fitness:	3275
3	fitness:	3305
4	fitness:	3333
5	fitness:	3337
6	fitness:	3345,5
7	fitness:	3354
8	fitness:	3365
9	fitness:	3384,5
10	fitness:	3391,5
11	fitness:	3398,5
12	fitness:	3401,5
13	fitness:	3413
14	fitness:	3414,5
15	fitness:	3432,5
16	fitness:	3449
17	fitness:	3451,5
18	fitness:	3460
19	fitness:	3474,5
20	fitness:	3528
	Promedio	3382,3

TABLA 4-5: PARAMETROS PRUEBA 6

Iteraciones	Movidas	Tabu
100	400	20

2 Min

1	fitness:	3315,5
2	fitness:	3315,5
3	fitness:	3330,5
4	fitness:	3330,5
5	fitness:	3348
6	fitness:	3348
7	fitness:	3361,5
8	fitness:	3390,5
9	fitness:	3399
10	fitness:	3399
11	fitness:	3404
12	fitness:	3420
13	fitness:	3420
14	fitness:	3450
15	fitness:	3459
16	fitness:	3459
17	fitness:	3459,5
18	fitness:	3459,5
19	fitness:	3506
20	fitness:	3547
	Promedio	3406,1

El desempeño obtenido por la Metaheurística fue muy por debajo de lo que se esperaba dado que la programación realizada manualmente por el programador de turnos obtuvo un fitness de 1100, y como se puede apreciar en los resultados, el que obtuvo menor valor fue la programación con un fitness de 3132. Además de estos resultados, se pueden apreciar que a la hora de mejorar entre ellos, su diferencia no

es mayor que 300, lo que implica que difícilmente podrá llegar a un fitness cercano o superior a 1100 que era lo que se esperaba.

Si miramos más detenidamente el problema y analizamos lo ocurrido en la secuencia, nos damos cuenta que la mayoría de las veces los días sábado y domingos están en rojo (ver Figura 4.6: Análisis días sin asignación), esto es, que faltó personal por asignar ese día.

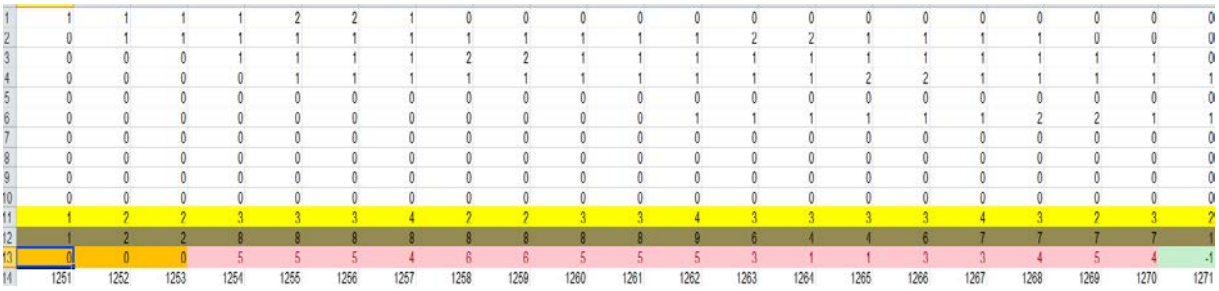


FIGURA 4.6: ANALISIS DIAS SIN ASIGNACIÓN

Por otro lado, si vemos lo ocurrido en un día martes o miércoles, ocurre siempre que existen muy pocos días en que falta asignación en un bloque (ver figura 4.7), lo que nos lleva a pensar que a pesar de la aleatoriedad para construir la solución inicial, esta solución está fuertemente condicionada a las primeras asignaciones que se realizan en la heurística constructiva y por ende, le es muy difícil salir a explorar otro espacio de soluciones, dado que esta es una Metaheurística basada en la búsqueda de óptimos locales, lo que implica que para mejorar la soluciones de esta Metaheurística, se debe elaborar una heurística constructiva eficiente y que no condicione de sobre manera los últimos días de las asignaciones, que dicho de paso, son los días en los cuales se requiere más dotación de personal.

	ANP	ANQ	ANR	ANS	ANT	ANU	ANV	ANW	ANX	ANY	ANZ	AOA	AOB	AOC	AOD	AOE	AOF	AOG	AOH	AOI	AOJ
1	0	0	0	0	0	0	1	1	1	1	1	1	1	1	2	2	1	1	1	1	1	1
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	1	1	1	1	2	2	1	1	1	1	1	1	1	1	1
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2	2	1	1	0
7	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
8	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	2	2	1	1	1
9	0	1	1	1	1	1	1	2	2	1	1	1	1	1	1	1	1	1	1	0	0	0
10	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	2	2	1	1
11	0	1	1	2	2	3	3	6	6	7	7	6	6	6	6	6	7	5	4	5	6	5
12	0	1	1	2	2	3	3	6	6	6	6	6	6	6	6	6	6	5	6	6	6	5
13	0	0	0	0	0	0	0	0	0	-1	-1	0	0	0	0	-1	0	2	1	0	0	0
14	1055	1056	1057	1058	1059	1060	1061	1062	1063	1064	1065	1066	1067	1068	1069	1070	1071	1072	1073	1074	1075	

FIGURA 4.7: ANALISIS DIAS CON ASIGNACIÓN

A pesar de lo anterior, es importante recalcar dentro de este análisis que no todo es negativo, considerando que la programación obtenida manualmente es resultado de un mes de trabajo en contraste con el programa que entrega una solución en aproximadamente 27 segundos, además, las programaciones entregadas por el programa siempre cumplió con las restricciones legales, ya sea con la cantidad de días trabajados en la semana, cantidad de descanso obligatorios, etc., lo que a diferencia de la programación manual este proceso debe realizarse en etapas, chequeando cada vez, por semana y por cada colaborador, que la normativa legal se cumpla.

Ahora, si se compara el comportamiento de la metaheurística con la herramienta actual de la empresa, la cobertura operacional de turnos de los días Lunes a Miércoles, es prácticamente perfecta, sin sobredotación ni faltante significativo de cobertura operacional, a diferencia de la programación manual de la empresa que tiene sobredotación de colaboradores en largos tiempos de la jornada laboral.

Pero como indicamos anteriormente, los últimos días de las asignaciones, quedan condicionados a las primeras asignaciones de la heurística, por ende, desde el día Jueves hasta el término de la semana (día Domingo), las coberturas operacionales se deterioran (pero aún en algunos días, por ejemplo, Viernes y

Sábado, siguen siendo mejores los resultados del algoritmo respecto a la programación de la empresa.

Todo lo anterior se puede apreciar en los siguientes gráficos:

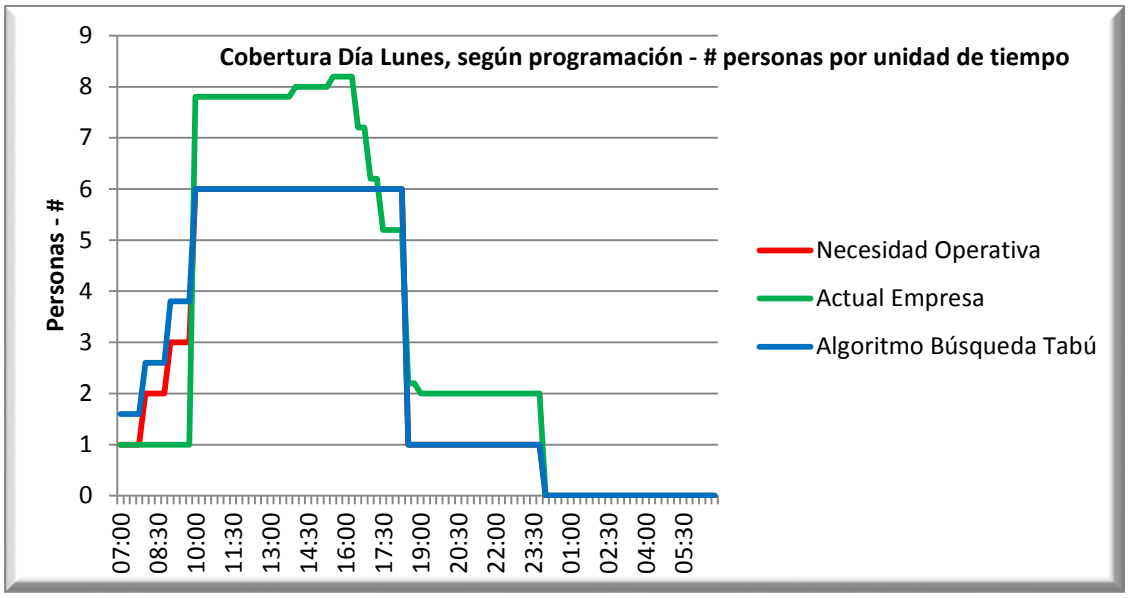


GRÁFICO 4.8: COMPARACIÓN COBERTURA OPERACIONAL DÍA LUNES.

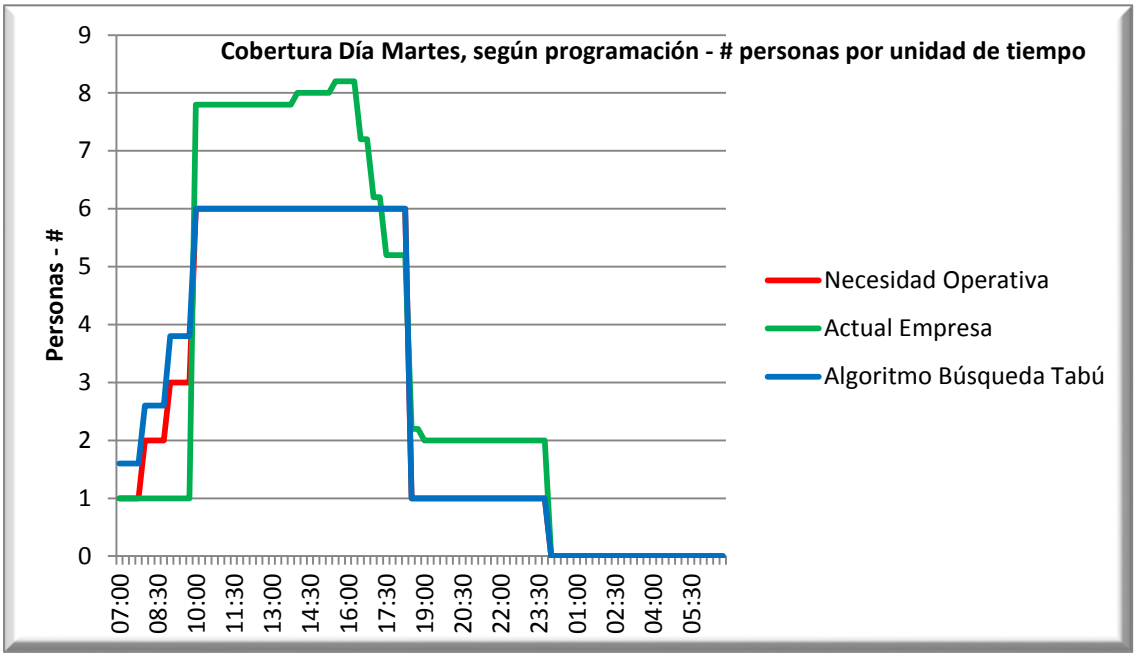


GRÁFICO 4.2: COMPARACIÓN COBERTURA OPERACIONAL DÍA MARTES.

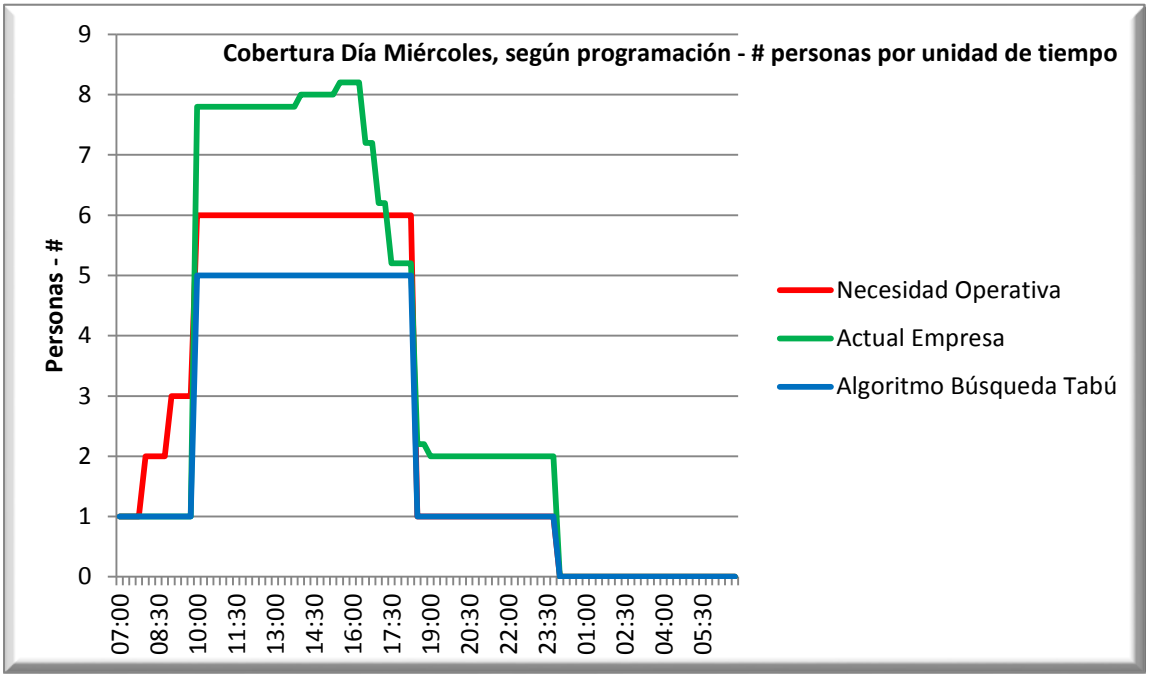


GRÁFICO 4.3: COMPARACIÓN COBERTURA OPERACIONAL DÍA MIÉRCOLES.

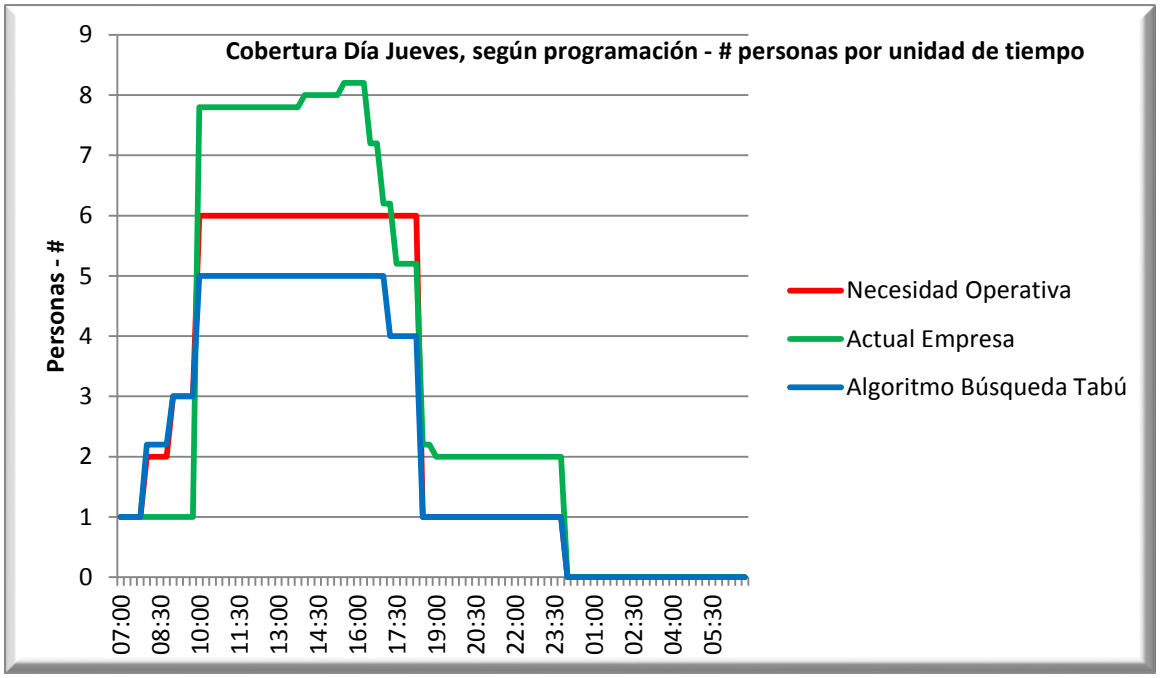


GRÁFICO 4.4: COMPARACIÓN COBERTURA OPERACIONAL DÍA JUEVES.

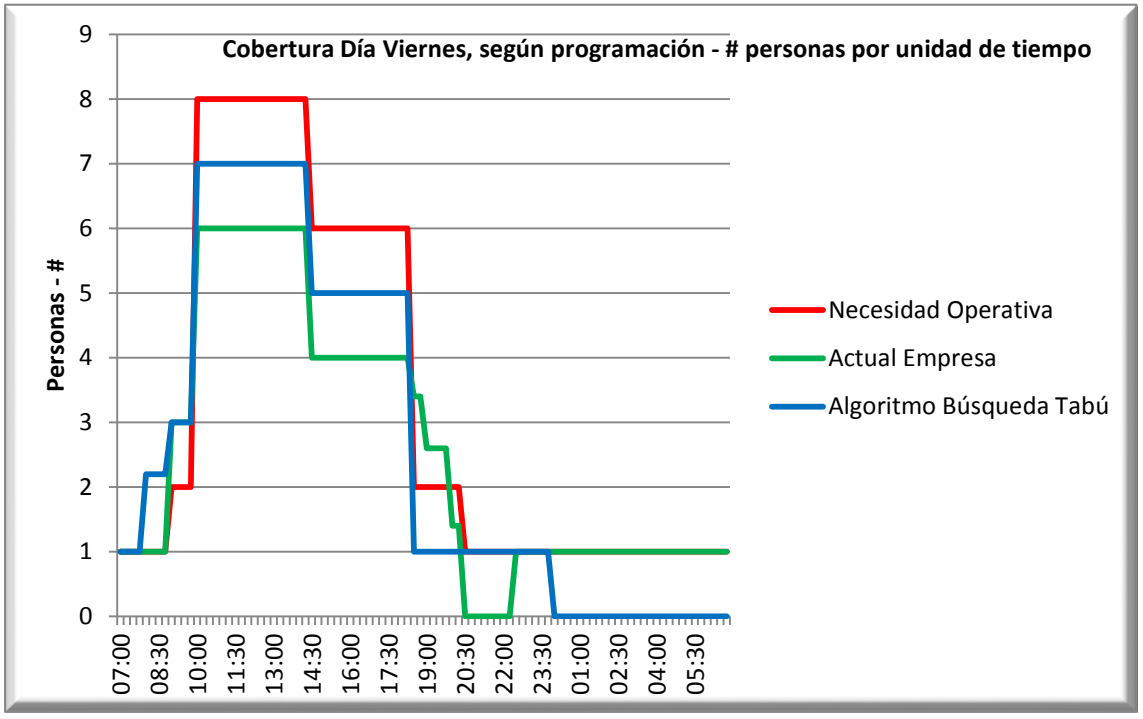


GRÁFICO 4.5: COMPARACIÓN COBERTURA OPERACIONAL DÍA VIERNES.

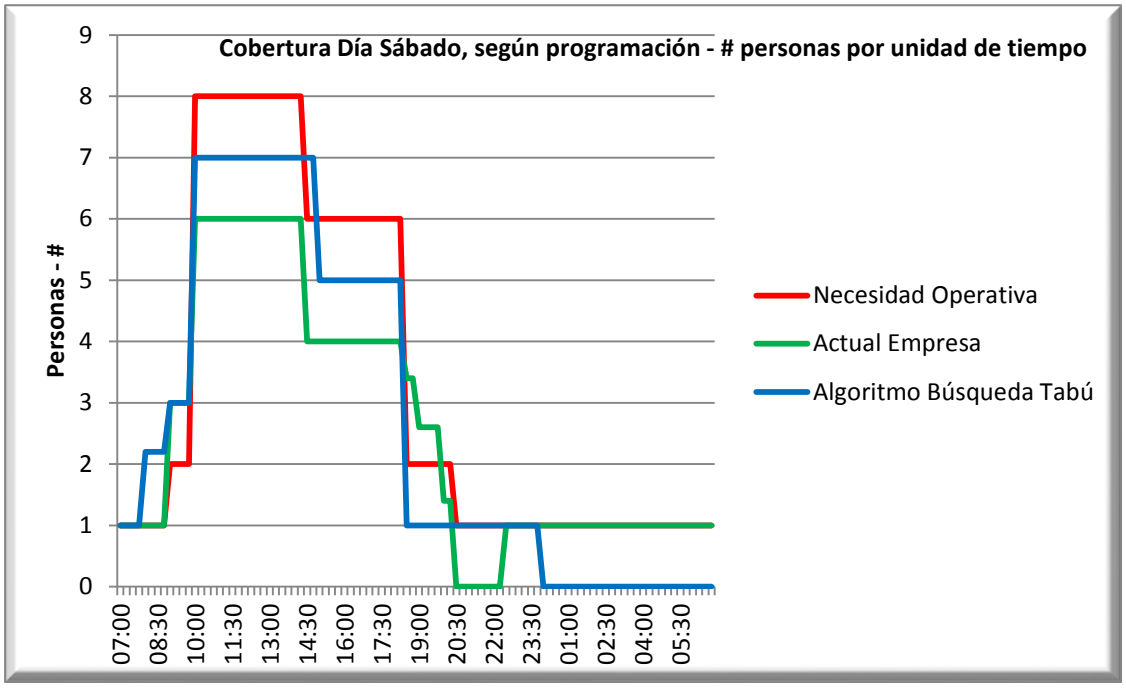


GRÁFICO 4.6: COMPARACIÓN COBERTURA OPERACIONAL DÍA SÁBADO.

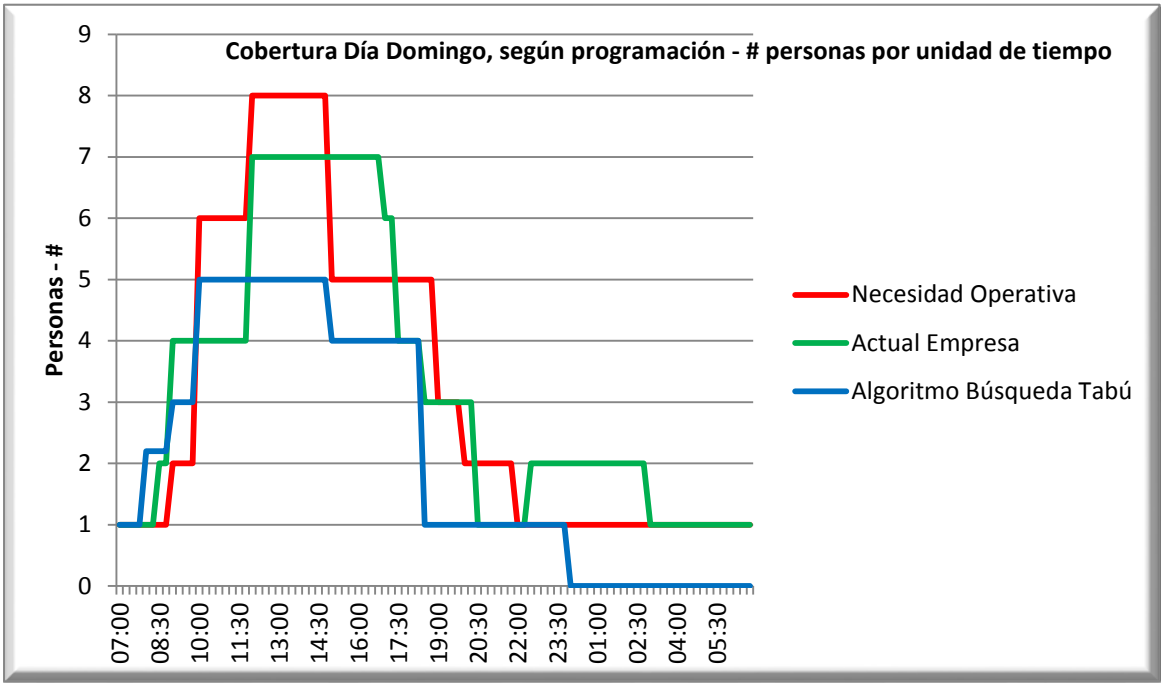


GRÁFICO 4.7: COMPARACIÓN COBERTURA OPERACIONAL DÍA DOMINGO.

CAPÍTULO 5: CONCLUSIONES Y RECOMENDACIONES.

5.1. CONCLUSIONES

Se elaboró un algoritmo metaheurístico basado en Tabu Search con el fin de mejorar y automatizar la asignación de turnos para mucamas.

Este algoritmo, a través del modelo de búsqueda en el que se basa, busca mejorar distintas soluciones en base a sus vecinos; lo que permitió efectivamente dar soluciones, aunque de baja calidad al problema planteado.

Al utilizar un método aproximado como lo es Tabu Search en este problema de optimización combinatoria, permitió proponer soluciones en un tiempo de respuesta óptimo en contraste a la manera actual de trabajo, esto es, alrededor de 27 segundos de procesamiento.

Tabu Search logró encontrar propuestas de soluciones de asignación de turnos, por lo que es un algoritmo aplicable al problema asignación, lo que concuerda con lo descrito en la literatura respecto de que para las Metaheurísticas, no existe un marco teórico que las sustente, sino que es a través de los buenos resultados experimentales donde encuentran su justificación.

Se hace evidente con los resultados que es de suma importancia tener una buena heurística constructiva para poder encauzar de cierta manera el algoritmo y de esta forma encontrar buenos resultados.

5.2. RECOMENDACIONES

Se recomienda realizar nuevos estudios abordando el problema de Asignación de Turnos de manera que esta logre a lo menos acercarse a lo programado de forma manual, dado que el ahorro de tiempo y dinero para la empresa en cuestión sería de millones de pesos mensuales, sin considerar que de esta forma no se incurrirían en faltas a la ley del trabajador.

Es importante evaluar otras Metaheurísticas dado que cada una de estas son adaptadas al problema y por tanto, todas no funcionan de la misma manera para un tipo de problema.

Para obtener una mejora en las soluciones, es importante realizar un ajuste de parámetros antes de ejecutar el programa, ya que existe una interacción directa entre éstos y las soluciones que se van obteniendo en cada iteración del algoritmo. Por ello, se recomienda trabajar en la elaboración de una aplicación que ajuste automáticamente los mejores parámetros para el algoritmo en cuestión y de esta forma se integre a los algoritmos.

Dado que las primeras asignaciones en la semana condicionan fuertemente las asignaciones posteriores en cada bloque, se recomienda, implementar una heurística constructiva que utilice una estrategia de inicio dirigida a cubrir primero los días con mayor necesidad de personal y no de forma secuencial como se planteó en este estudio.

En relación a la comparación a Tabu Search, con este estudio y para este tipo de problema queda demostrado que sus resultados quedan muy dependientes a la solución inicial y por tanto, le cuesta mucho la diversificación, por lo que se recomienda para este tipo de problemas, buscar estrategias que ayuden al algoritmo poder diversificar más ampliamente sus soluciones.

CAPÍTULO 6: GLOSARIO.

ACO: Ant Colony Optimization. Metaheurística Optimización de Colonias de Hormigas.

EC: Evolutionary Computation. Computación Evolutiva.

GA: Genetic Algorithms. Algoritmos Genéticos.

GLS: Guided Local Search. Metaheurística Búsqueda Local Guiada.

GRASP: Greedy Randomized Adaptive Search Procedure. Metaheurística Búsqueda Miope, Aleatorizado y Adaptativo.

ILS: Iterated Local Search. Metaheurística Búsqueda Local Iterativa.

PR: Path Relinking. Metaheurística Reencadenamiento de Camino.

PSO: Particle Swarm Optimization. Algoritmos Basados en Cúmulos de Partículas.

TS: Tabu Search. Metaheurística Búsqueda Tabú.

VNS: Variable Neighborhood Search. Metaheurística Búsqueda Variable de Vecindario.

SA: Simulated Annealing. Metaheurística Recocido Simulado.

SS: Scatter Search. Metaheurística Búsqueda Dispersa.

CAPÍTULO 7: FUENTES BIBLIOGRÁFICAS.

ALVAREZ-VALDES R., CRESPO E. and TAMARIT J.M. (1999). Labor Scheduling at an Airport Refuelling Installation. Journal of the Operational Research Society, Vol.50, nº3, pp.211- 218.

ANDREWS B. and PARSONS H. (1989). L.L. Bean Chooses a Telephone Agent Scheduling System. Interfaces, Vol.19, nº6, pp.1-9.

ANDREWS B. and PARSONS H. (1993). Establishing Telephone-Agent Staffing Levels Through Economic Optimization. Interfaces, Vol.23, nº2, pp.14-20.

AYKIN T. (1996). Optimal Shift Scheduling with Multiple Break Window. Management Science, Vol.42, nº4, pp.591-602.

AZMAT C., HÛRLIMANN T. and WIDMER M. (2004). "Mixed Integer Programming to Schedule a Single-Shift Workforce under Annulized Hours". Annals of Operations Research. Vol.128, pp. 199-215.

BAESLER F., MORAGA R. y CORNEJO O. (2008). Artículo "Introducción de Elementos de Memoria en el Método Simulated Annealing para resolver problemas de programación multiobjetivo de máquinas paralelas", Revista Chilena de Ingeniería, Vol.16, nº 3, pp. 428-437.

BAKER K.R. (1976). Workforce Allocation in Cyclical Scheduling Problems: A survey. Operational Research Quarterly, 27, pp.155-167.

BARTHOLDI, J.J. (1981). A Guaranteed-Accuracy Round-off Algorithm for Cyclic Scheduling and Set Covering. Operations Research, Vol.29, nº3, pp.501-510

BECHTOLD S.E. and JACOBS L.W. (1990). Implicit optimal modeling of flexible break assignments in labor staffing decisions for service operations. Management Science, 36, 11, pp.1339-1351.

BECHTOLD S.E. and JACOBS L.W. (1991). Improvement of labor utilization in shift scheduling for services with implicit optimal modeling. International J. of Oper. An Production Management, 11, 2, pp.54-69.

BETCHOLD S.E., BRUSCO M. and SHOWALTERM. (1991). A Comparative Evaluation of Labor Tour Scheduling Methods. Decision Science, 22, pp. 683-699.

BRUSCO M. and JACOBS L. (1993a). A Simulated Annealing Approach to the Cyclic Staff-Scheduling Problem. Naval Research Logistics, 40(1), pp.69-84.

BRUSCO M. and JACOBS L. (1993b). A Simulated Annealing Approach to the Solution of Flexible Labour Scheduling Problems. Journal of the Operational Research Society, Vol. 44, nº 12, pp.1191-1200.

BRUSCO M., JACOBS L., BONGIORNO R., LYONS D. and TANG B (1995). Improving Personnel Scheduling at Airline Stations. Operations Research 43(5), pp.741-751.

CASADO YUSTA S. y PACHECO BONROSTRO J. (2003). Facultad de CCEE. y EE. Universidad de Burgos. Artículo "Estudio comparativo de diferentes estrategias metaheurísticas para la resolución del labor scheduling problem", Estudios de Economía Aplicada Vol. 21-3, pp.537-557

COELLO C. (2003). Introducción a la Computación Evolutiva - Notas de Curso. CINVESTAVIPN Departamento de Ingeniería Eléctrica, Sección de Computación, Mayo, 2003.

COROMINAS A., LUSA A. and PASTOR R. (2004). "Planning Annualised Hours with a Finite Set of Weekly Working Hours and Joint Holidays". Annals of Operations Research. Vol. 128, pp. 217-233.

DANTZIG G.B. (1954). A Commenton Edie´s `Traffic Delays at Toll Booths´. Operations Research, 2,3, pp.339-341.

DOWNSLAND K.A. (1998). Nurse Scheduling with Tabu Search and Strategic Oscillation. European Journal of Operational Research, 106, pp.393-407.

EASTON F. (1991). Modular IP Aproximation Procedures for Tour Scheduling Problems. Proceedings of the National Conference, Decision Sciences Institute, Atlanta, GA.

EASTON F. and MANSOURN. (1999). A Distributed Genetic Algorithm for Employee Staffing and Scheduling Problems. In: S. Forrest (Ed.), Proceedings of Fifth International Conference on Genetic Algorithms, Morgan-Kaufman, San Mateo, CA, pp.360-367.

EASTON F. and ROSSIND. (1996). A Stochastic Goal Program for Employee Scheduling. Decision Sciences 27(3), pp.541-568.

EITZEN G., PANTON D. and MILLS G. (2004). "Multi-Skilled Workforce Optimization". Annals of Operations Research. Vol. 127, pp. 359-372.

ERNST A., JIANG H, KRISHNAMOORTHY M. and SIER D. (2004). "Staff scheduling and rostering: A review of applications, methods and models". European Journal of Operational Research. Vol. 153, pp. 3-27.

JÚDICE J., MARTINS P., NUNES J. (2005). "Workforce planning in a lot sizing mail processing problem". Computers & Operations Research. Vol.32, Issue 11, pp. 3031-3058. November 2005.

KEITH E.G. (1979). Operator Scheduling. AIIE Transactions,11,1, pp.37-41.

MARTÍ CUNQUERO R. (2003), Artículo "Algoritmos Heurísticos", Departamento de Estadística e Investigación Operativa, Universidad de Valencia, España. pp. 2-3.

MOONDRAS. L. (1976). An L.P. Model for Work Force Scheduling for Banks. J. Bank Res., 7,4, pp. 299-301

MOZ M. and VAZ PATO M. (2004) "Solving the Problem of Rerostering Nurse Schedules with Hard Constraints: New Multicommodity Flow Models". Annals of Operations Research. Vol.128, pp. 179-197.

QUINN P., ANDREWS B. and PARSONS H. (1991). Allocating Telecommunications Resources at L.L. Bean. Interfaces, Vol.21, nº1, pp.75-91.

PRADENAS L., HIDALGO S. y JENSEN M. (2008). Artículo "Asignación de Supervisores Forestales. Resolución mediante un Algoritmo Tabu Search", Revista Chilena de Ingeniería, Vol.16, nº 3, pp.404-414. Octubre-Diciembre 2008

SAMPSON S. (2006) "Optimization of volunteer labor assignments" Journal of Operations Management. Vol.24, Issue 4, pp.363-377. June 2006.

TAYLOR P. and HUXLEY S. (1989). A Break from Tradition for the San Francisco Police: Patrol Officer Scheduling Using an Optimization-Based Decision Support System. Interfaces, Vol.19, nº1, pp.4-24.

THOMPSON G.M. (1990). Shift Scheduling when Employees Have Limited Availability: an L.P. Approach. J. of Oper. Management, 9,3, pp.352-370.

THOMPSON G.M. (1992). Improving the Utilization of Front-Line Service Delivery System Personnel. Decision Sciences, 23,5, pp.1072-1098.

THOMPSON, G.M. (1995). Improved implicit optimal modeling of the labor shift scheduling problem. Management Science, 41(4), pp.595-607.

THOMPSON, G.M. (1996). A Simulated Annealing Heuristic for Shift Scheduling Using non-Continuously Available Employees. Computers and Operations Research, Vol.23, nº3, pp.275-278.

THOMPSON, G.M. (1997). Labor Staffing and Scheduling Models for Controlling Service Levels. Naval Research Logistics, Vol.44, pp.719-740.

TOPALOGLU S., OZKARAHAN I. (2004). "An Implicit Goal Programming Model for the Tour Scheduling Problem Considering The Employee work Preferences". Annals of Operations Research. Vol. 128, pp. 135-158.

YAMADA T., NAKANO R. (1999). Job-shop scheduling. In P. J. Fleming A. M.S. Zalzala, editor, Genetic Algorithms in Engineering Systems, chapter 7, pp. 134-160.

CAPÍTULO 8: ANEXOS.

8.1. ENTRADA DE DATOS AL PROGRAMA

N_Colaboradores: 10

Necesidad:

L: 1	1	2	2	3	3	6	6	6	6	6	6	6
	6	6	6	6	6	6	6	6	6	6	1	1
	1	1	1	1	1	1	1	1	1	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	
V: 1	1	1	1	2	2	8	8	8	8	8	8	8
	8	8	6	4	4	6	6	6	6	6	2	2
	2	2	1	1	1	1	1	1	1	1	1	1
	1	1	1	1	1	1	1	1	1	1	1	
S: 1	1	1	1	2	2	8	8	8	8	8	8	8
	8	8	6	4	4	6	6	6	6	6	2	2
	2	2	1	1	1	1	1	1	1	1	1	1
	1	1	1	1	1	1	1	1	1	1	1	
D: 1	1	1	1	2	2	6	6	6	6	8	8	8
	8	8	8	2	2	2	2	5	5	5	5	3
	3	2	2	2	2	1	1	1	1	1	1	1
	1	1	1	1	1	1	1	1	1	1	1	

Restricciones:

RT1: 27

RT5: 3

RT8: 19

Contrato:

CT1:45

CT2:45

CT3:45

CT4:45

CT5:45

CT6:45

CT7:30

CT8:30

CT9:30

CT10:20