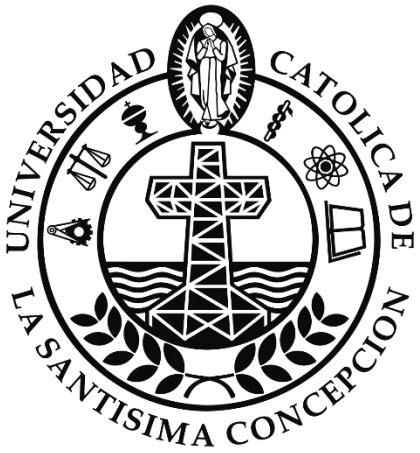
UNIVERSIDAD CATÓLICA DE LA SANTÍSIMA CONCEPCIÓN
Facultad de Ingeniería
Ingeniería Civil Industrial



UN ALGORITMO PARA RESOLVER EL PROBLEMA DE SECUENCIAMIENTO DE VEHÍCULOS EN UNA CADENA DE MONTAJE DE AUTOMÓVILES

Pablo Francisco Campos Echaíz

INFORME DE PROYECTO DE TÍTULO PARA OPTAR EL TÍTULO DE

INGENIERO CIVIL INDUSTRIAL

Profesor Guía: Dr. Cristian Oliva S.M.

Profesor Informante: Dr. Manuel Cepeda J.

Concepción, Septiembre de 2016

Mis agradecimientos van dirigidos a todos y cada uno de los profesores que me guiaron en mi inicio de enseñanza básica y media, a mis profesores de la Universidad Católica de la Santísima Concepción y a la generación 2007 de I.C.I.

De manera especial agradezco a mi Padre y a mi Madre por la confianza, amor y por ser los primeros profesores de mi vida. Gracias a mis hermanos: Cristián, Feña y Pipe por nunca dejar de apoyarme, especialmente a Cristián que fue mi gurú y mentor del proceso. Gracias a tí Massi, mi novia que sin duda ha sido un motor para seguir este sueño adelante (agradezco tu confianza, amor e infinita paciencia). Gracias a mis primos Manolo, Coto, Lalo, Seba, Tomás y Benjamín por hacer de cada broma de "y la tesis ¿Cuándo?" una bandera de lucha para adquirir fortaleza, también a mis tíos Nano, Claudia y Lalo (desde el cielo) y en general a toda la familia Campos Echaíz. Gracias a mis amigos de universidad y muy especialmente a Carla, Mimí, Coni, Nico, Zamo, Colocho, Diego V., Franz, Chelo y Pacheco por la amistad. Gracias Dios.

RESUMEN

En este estudio se formula el Problema de Secuenciamiento de Vehículos proponiendo un único método de resolución. El problema consiste en programar el orden de un lote de vehículos que requieren distintas opciones de configuración, las que son instaladas en distintas estaciones de trabajo y que implica generar una secuencia que minimice la sobrecarga de estaciones. Distintos métodos de resolución fueron estudiados, con el fin de abordar de manera correcta el tipo de problema a enfrentar. El método propuesto consiste en resolver el modelo por medio de un algoritmo de tipo Búsqueda Tabú, que implica seleccionar el valor de los parámetros que ajusten la búsqueda hacia soluciones factibles. Para evaluar la efectividad del método, se utilizó la estandarización que propuso RENAULT en el Desafío ROADEF 2005. Los parámetros de dicho algoritmo fueron calibrados con tres casos que poseen distintos valores de Función Objetivo, posteriormente la evaluación de las instancias será comparada con los resultados publicados por el ranking oficial de la competencia.

Palabras Claves: Problema de Secuenciamiento de Vehículos, Algoritmo de Búsqueda Tabú, Heurísticas, Metaheurísticas.

ABSTRACT

In this study, a Car Sequencing Problem is formulated proposing only one method of solution. The problem is to program the order of a batch of vehicles that require different scheduling options, wich are installed in many workstations and it involve to generate a sequence that minimize the overload of stations. Different solving methods were studied, in order to adress correctly the type of problem to be solved. The proposed method consists in solving the model by a Tabu Search algorithm, it involves select the value of the parameters that fit the search to feasible solutions. To evaluate the effectiveness of this method, the standardization proposed by RENAULT in ROADEF 2005 is used. The parameters of this algorithm were calibrated with three cases with different values of objective function, subsequently the evaluation of the instances will be compared with the results published by the official ranking of the challenge.

Keywords: Car Sequencing Problem, Tabu Search Algorithm, Heuristics, Metaheuristics.

ÍNDICE GENERAL

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1 DESCRIPCIÓN DEL PROBLEMA

1.1 Antecedentes de la Industria Automotriz

Desde los tiempos del empresario Henry Ford y su famoso modelo “Ford T” que los requisitos del producto, y por tanto los requisitos de los sistemas de producción, han cambiado drásticamente. Originalmente las líneas de montaje de vehículos en las industrias automotrices desarrollaban modelos que seguían un patrón estándar (Ev World, 2013).

En el presente, los requerimientos y variaciones de los vehículos confeccionados en una fábrica automotriz han aumentado. “Hoy en día la deslocalización exige que las empresas impulsen una mayor esfuerzo a las fábricas para aumentar su productividad” (ANFAC, 2004). Estos esfuerzos de una forma u otra se relacionan con el secuenciamiento:

- Flexibilización de los procesos productivos;
- Reducción de los tiempos de fabricación;
- Demanda de configuraciones individualizadas.

El desafío para las empresas es entonces, que los aspectos anteriormente mencionados se realicen de forma eficaz; así mayor será la flexibilidad, la reducción de tiempos y la capacidad para atender las demandas personalizadas de los clientes.

La mayoría de las empresas automotrices europeas, como Renault, se basan en un sistema de producción *build-to-order* (construcción a la orden), en lugar a una configuración a inventario (Nguyen et. al., 2005). Este enfoque de la producción tiende a generar un flujo diversificado de vehículos en las plantas de montaje, lo que significa una gran variedad de modelos día a día, debido a que éstos son clasificados por clientes y no

por concesionarios. Ahí en más, las empresas necesitan resolver el problema de saber la distribución de sus estaciones de trabajo y cuánto producir.

1.2 Car Sequencing Problem (CSP)

El Problema de Secuenciamiento de Vehículos (CSP) fue descrito en 1986 por Parello et. al. En [21] y describe la programación de vehículos a lo largo de una línea de ensamblaje, con el fin de instalar distintas opciones para cada tipo de vehículo en particular (Aire Acondicionado, Navegador GPS, Alza Vidrios Eléctricos, entre otros). Estas opciones son instaladas en estaciones de trabajo independientes unas de otras, con el fin de manejar un porcentaje importante de corridas de vehículos por la cadena de montaje; siendo de suma importancia generar un espaciamiento para los vehículos que requieran alguna de las opciones especiales, asegurando que la capacidad de la estación de trabajo no sea excedida.

Esos requerimientos deberán ser formalizados por restricciones de relación $^{p}/_{q}$ y cada opción es asociada a alguna de estas restricciones, definidas a continuación.

1.2.1 Definición de una instancia del problema

Una instancia del CSP es definida por una tupla (V, O, p, q, r), donde:

- $V = \{V_1, V_2, \cdots, V_n\}$ es el conjunto de vehículos que serán producidos;
- $O = \{O_1, O_2, \cdots, O_m\}$ es el conjunto de opciones distintas entre sí;

- $p : O \to \mathbb{N}$ y $q : O \to \mathbb{N}$ definen las restricciones de capacidad asociadas a cada opción $o_i \in O$; esta restricción impone que alguna subsecuencia de q_i vehículos consecutivos en la línea, a lo más p_i de ellos requerirán o_i;

- $r : V \times O \to \{0,1\}$ define la opción de requerimientos, es decir, para cada vehículo $v_j \in V$ y para cada opción $o_i \in O$ se tiene que:

$$r_{ji} = \begin{cases} 1, & si\ la\ opción\ O_i\ se\ instala\ en\ V_j \\ 0, & e.o.c. \end{cases}$$

Notar que dos vehículos diferentes de V pueden requerir las mismas opciones de configuración, es decir, el mismo set de opciones de requerimientos; todos los autos que requieran una misma opción de configuración son agrupados en una misma clase de vehículos. Precisamente, hay k clases distintas de vehículos por lo que V es particionado en k subgrupos, lo que quiere decir que:

$$V = V_1 \cup V_2 \cup \cdots \cup V_k$$

De tal manera que todos los vehículos dentro de un mismo subgrupo V_i requieren una misma opción de configuración.

1.2.2 Solución de una instancia del problema

El CSP tiene como objetivo encontrar una combinación eficiente de los vehículos que pertenecen a V, mediante el secuenciamiento de ellos en la línea de ensamblaje (carrocería, pintura y montaje). Esto debe permitir satisfacer todas las restricciones de capacidad de la planta y tener una secuencia de mínimo costo, que mediante una *Función de Costo* permite evaluar la violación de restricciones.

El problema original propuesto en [21] especificaba que el costo depende de:

- Las opciones afectadas por las restricciones de violación;
- El número de vehículos que exceden las restricciones de capacidad;
- Cuán cerca se han secuenciado esos vehículos.

Sin embargo en la mayoría de los trabajos esta función de costo ha sido simplificada, y fue definida como la suma para cada opción $O_i \in O$ del número de subsecuencias de q_i autos consecutivos, tal que el número de vehículos que requieran la opción O_i en la subsecuencia es mayor que la capacidad p_i de la opción. Existen definiciones distintas del costo de una subsecuencia en [22, 23, 24] donde ésta es definida como el número de vehículos (que no requieren opción) que pueden ser colocados en la secuencia, a fin de que todas las restricciones de capacidad sean enfrentadas.

1.2.3 Complejidades

El estudio de Gent en [11] demostró que la decisión del problema es del tipo NP – Hard mediante una transformación desde el *Problema de Recorrido Hamiltoniano*, mientras que Kis en [16] lo demostró mediante una transformación del problema exacto por un problema de 3 fases.

1.2.4 Búsqueda de espacio y tasas de utilización

La búsqueda de espacio en una instancia (V, O, p, q, r) del CSP está compuesto de todas las permutaciones posibles de los vehículos de V que requieran diferentes opciones de configuración. Por lo tanto, si V es particionado en k clases distintas de vehículos $V_1, V_2, \cdots, V_k$ el número de disposiciones diferentes de los vehículos de V en una secuencia es:

$$\frac{|V|!}{|V_1|! * |V_2|! * \ldots * |V_k|!}$$

La dificultad de una instancia del CSP dependerá del tamaño de su espacio de búsqueda, pero además dependerá de las tasas de utilización de las diferentes opciones, esto fue señalado por Smith en [27]. La tasa de utilización de una opción corresponde a la razón del número de autos que lo requieran, respetando el número máximo de autos en una secuencia que tendría al tiempo que satisfaga su restricción de capacidad. Una tasa de utilización cercana a 1 indica que la demanda es cercana a la capacidad.

1.3 Caso de Estudio

El *Problema de Secuenciamiento de Vehículos* a resolver corresponde al CSP que propuso *Renault* en el desafío ROADEF 2005 (ROADEF: *Sociedad Francesa de Investigación de Operaciones y Análisis de Decisiones*) el cual contenía diversos escenarios para ser analizados por los equipos inscritos en este torneo.

Una instancia del problema está constituida generalmente de una secuencia general de vehículos, un listado con objetivos de optimización, la limitación de lotes de pintura (restricción dura incluida para este problema) y las restricciones de capacidad de las estaciones de trabajo. Cabe señalar que una instancia será independiente de las restantes. A la instancia del CSP descrita en la sección 1.2.1 es necesario agregar el set de colores usado para pintar los vehículos denotado por $C = \{c_1, c_2, \cdots, c_t\}$.

Finalmente el proceso de *Secuenciamiento de Vehículos* puede ser dividido en tres etapas:

1. Identificar los objetivos de optimización a modo de asignar el peso correcto a cada una de las restricciones contenidas en la función objetivo.
2. Calcular la función de costo de la solución inicial y así fijar un valor comparativo para evaluar la mejora o deterioro de la secuencia.
3. Configurar el algoritmo de tipo *Búsqueda Tabú* y utilizarlo para optimizar el valor de la función de costo, mediante una subsecuencia ubicada en el espacio de *Vecindad* de la solución inicial.
4. Procurar que se cumplan las restricciones de tipo *dura* y minimizar el número de violaciones de restricciones de relación.

En la figura 1.1 se muestra un análisis causa – efecto (conocido como Diagrama de Ishikawa) que muestra las principales razones del mal *Secuenciamiento de Vehículos*:

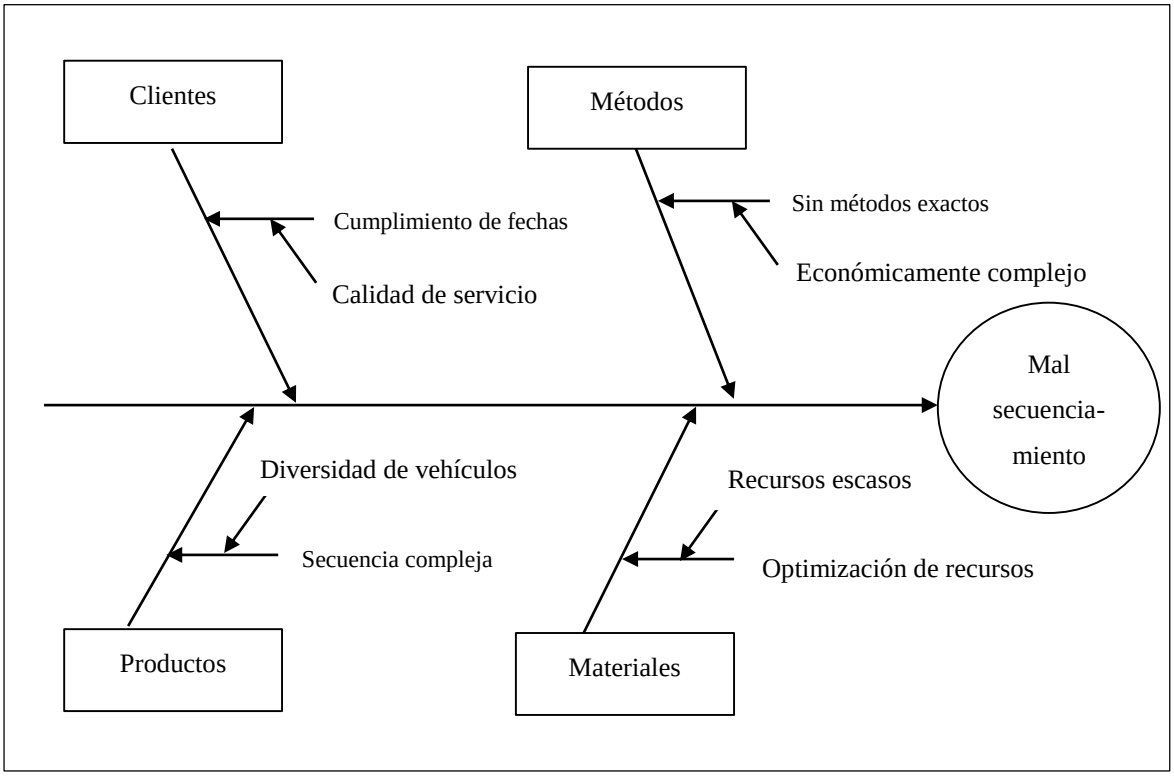


Figura 1.1 Análisis causa - efecto. Fuente: Elaboración Propia

Detalle de las Causas del mal Secuenciamiento de Vehículos

- Cumplimiento de fecha: Existe por parte de *Renault* una política de satisfacción al cliente al aceptar una línea de producción "*build to order*" y esto genera tiempos de ejecución distintos para cada modelo de vehículo; en algunos casos se tiene que reprogramar los trabajos que ingresan en la última cola (generada por la incapacidad de atender todos los pedidos ingresados anteriormente a estos trabajos). La calidad de servicio no puede verse afectada y la misión de la programación es cumplir a cabalidad con todos los requerimientos del cliente, y por tanto agrupar los trabajos para ser entregados oportunamente.

- Sin métodos exactos**:** Este tipo de problemas son considerados de tipo *NP- Hard*, lo que se traduce a buscar alternativas de mejora de secuencias para obtener una programación que se acerque al mejor modelo estudiado; y así evitar complicar los balances económicos de la industria automotriz, pues existe una complejidad al enfocar los esfuerzos por encontrar un mejor software y dejar de lado otro tipo de prioridades competitivas.

- Secuencia compleja: Al existir una gama de productos y opciones para los vehículos, la secuencia inicial tiende a ser complicada de tratar si es que no se posee un método definido inicialmente. La idea es abordar de manera rápida cada día de trabajo para no sobrecargar cada estación de trabajo.

- Optimización de recursos: Cada vehículo posee su propio requerimiento y los recursos que se utilizan para ello son reducidos, estas dos situaciones hace que a veces inevitablemente se sobrepase los límites de recursos (por ejemplo los límites de solvente de pintura), y por tanto la solución tienda a ser infactible.

1.4 Justificación del Problema

El sistema computacional entregará una solución automatizada al *Car Sequencing Problem* que brinda una solución factible y de mejores características en términos de costos. Lo anterior significará una ayuda a la optimización de este tipo de problema *NP – Hard* y aportará en los avances de la investigación y mejora del *Secuenciamiento de Vehículos*. El desarrollo del software se justifica para crear esfuerzos en cuanto al análisis de problemas de optimización y crear parámetros que ayuden a la búsqueda de otro tipo de métodos para resolver el CSP.

1.5 Objetivos

1.5.1 Objetivo General

Proponer un algoritmo basado en la *Búsqueda Tabú*, que permita alcanzar la solución del problema clásico de *Secuenciamiento de Vehículos* para una cadena de montaje de automóviles.

1.5.2 Objetivos Específicos

1. Realizar un análisis de los métodos de resolución del *Problema de Secuenciamiento de Vehículos* vistos en trabajos previos.

2. Identificar los datos de entrada contenidos en cada instancia del problema propuesto por *Renault*.

3. Implementar un algoritmo de tipo *Búsqueda Tabú* que mejore el *Secuenciamiento de Vehículos* y así minimizar la violación de restricciones primarias y secundarias.

4. Analizar y evaluar el desempeño del algoritmo con los resultados obtenidos en ROADEF 2005.

1.6 Delimitación del Problema

Este trabajo plasmará las restricciones y el/los principal(es) criterio(s) recopilados con el usuario para la *Programación de Secuenciamiento de Vehículos*. Posteriormente se implementará el modelo a través de un software usando un algoritmo de tipo *Búsqueda Tabú*. El sistema computacional debe contar con una interfaz gráfica intituiva y de fácil ejecución, que permita ingresar la información necesaria para asignar la posición a cada vehículo. El programa entregará una *Solución Factible* al problema, si es que la hubiera. Ésta debe ser presentada al usuario en un formato apropiado para que la interprete correctamente de acuerdo a sus conocimientos. Es importante mencionar que este trabajo no contempla un análisis de sensibilidad y no permite modificar la solución que entrega el software, y esta estructura puede ser utilizada para la revisión y estudio en algún trabajo a futuro.

2 MARCO TEÓRICO

En el capítulo anterior fue introducido el *Problema de Secuenciamiento de Vehículos* visto de una manera general, basado en la representación hecha del problema de optimización en 1986 por Parello. En éste capítulo se entrega una descripción detallada de la versión del CSP presentada en el Desafío ROADEF 2005.

2.1 Representación del Problema

En 2003, la *Sociedad Francesa de Investigación de Operaciones y Análisis de Decisiones* (ROADEF) organizó una competencia con el *Problema de Secuenciamiento de Vehículos* como protagonista, llamado *ROADEF Challenge*, que fue propuesto y patrocinado por el fabricante de automóviles *Renault*.

El problema propuesto por *Renault* para el desafío ROADEF'2005 difiere del clásico problema discutido en el capítulo 1. Además de las limitaciones de capacidad impuestas por la cadena de montaje, también presenta restricciones de procesamiento por lotes de pintura, y considera dos categorías de restricciones de capacidad que se debe tener en cuenta como prioridad (Nguyen et. al., 2005).

En las siguientes subsecciones se explican las típicas especificaciones que debe tener una planta de automóviles, así como los objetivos que cada estación de planta debe considerar al momento de optimizar.

2.1.1 Planificación y Proceso de Programación

La primera tarea de cada fábrica de automóviles es la asignación de un día de producción para cada vehículo pedido, de acuerdo con las fechas de entrega y la capacidad de la línea de producción. El conjunto de vehículos que se determina en esta etapa no se puede cambiar en otra circunstancia. Entonces, cada fábrica de automóviles tiene que programar el orden en que estos vehículos se pondrán en la línea de producción, la que satisfaga mejor a los requisitos de las estaciones de planta, *taller de pintura* y de *línea de ensamblaje*. En la actualidad estas dos tareas son realizadas por un software que utiliza *Programación Lineal* y *Análisis de Simulación*, respectivamente (Nguyen et al., 2005).

Según EV World, las plantas de producción de automóviles se componen de tres áreas principales: la *estación de carrocería*, la *estación de pintura* y la *estación de montaje*. La **estación de carrocería** es donde los robots y los paneles de operadores del metal de soldadura forman la estructura del vehículo. La **estación de pintura** es el lugar donde los vehículos están pintados por robots con pistolas. Por último, la **estación de montaje** está dividido en subestaciones de trabajo más pequeñas, también llamados bahías de trabajo, donde se llevan a cabo diversos procesos, componentes u opciones que se añaden a los vehículos (EV World, 2013).

La figura 2.1 ilustra las tres principales áreas de una planta productora de vehículos:

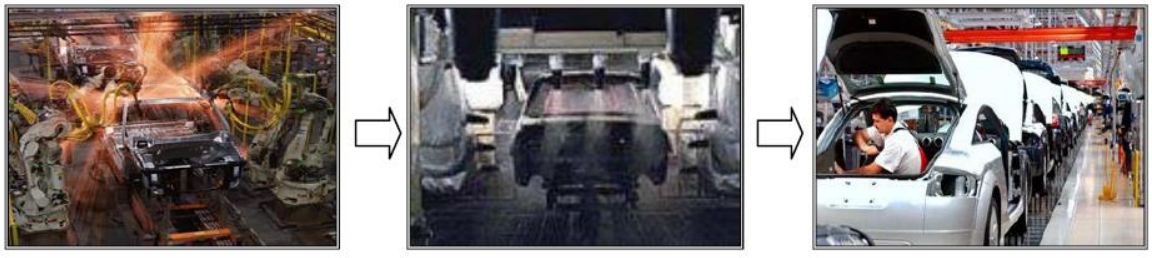


Figura 2.1 Estaciones de Carrocería, Pintura y Montaje. Fuente: www.evworld.com

2.1.2 Requerimientos en Estación de Pintura

El objetivo es *minimizar el número de cambios de color* con el fin de reducir el consumo de solvente usado para lavar las pistolas de pulverización, esto cada vez que se requiera un cambio de color. Significa que para reducir al mínimo el consumo de solvente de pintura, debería ser minimizado el número de cambios de color de pintura como sea posible en la secuencia. Esto se puede lograr mediante la agrupación de los vehículos con el mismo color en lotes y la secuenciación de los lotes, de forma tal que se minimice el número de cambios de pintura. El último vehículo del día anterior de producción tiene que ser tomado en cuenta en la evaluación de los cambios de color, es decir, si el último vehículo producido en el día de producción anterior tiene un color diferente al del primer vehículo de la fecha actual, entonces se debe contar con un cambio de color.

Sin embargo, hay un número máximo de vehículos consecutivos que están autorizados a tener el mismo color, porque las pistolas de pulverización tienen que ser limpiados periódicamente con el fin de garantizar resultados de alta calidad de la pintura. Este límite de lote de pintura es una *restricción dura*, es decir, todas las secuencias viables / soluciones tienen que satisfacer esta restricción. Los vehículos del día de producción anterior no se tienen en cuenta para el control del límite de lote de pintura. Teniendo en cuenta las dos secuencias ilustradas en la Figura 2.2, se verifica que ambas secuencias tienen 2 cambios de color. Sin embargo, si el límite de lote de pintura fuese 3, la secuencia de a) no se considera factible, ya que dispone de 4 vehículos consecutivos con el mismo color.

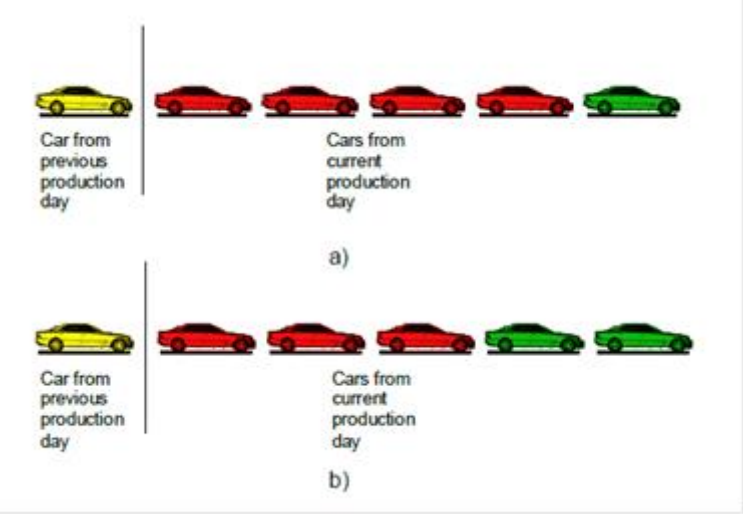


Figura 2.2 Aceptación / Rechazo de secuencia por cambio de color. Fuente: de Olivera, R. (2007)

2.1.3 Requerimientos de Estación de Montaje

El objetivo en la línea de montaje es *suavizar la carga de trabajo* a lo largo de ella, al equilibrar el esfuerzo en las diferentes estaciones de trabajo. En otras palabras, los vehículos que requieren operaciones especiales tienen que ser distribuidos uniformemente a lo largo de la línea de montaje, para evitar la sobrecarga de las estaciones donde estos vehículos se montan. Esto se logra con la definición anteriormente hecha en la relación $^{p}/_{q}$ (Parello et. al. 1986).

En la versión del CSP en ROADEF, hay dos clases de restricciones de relación, *restricciones de relación de prioridad principales* (*HPRC*) y *restricciones de relación de prioridad secundarias* (*LPRC*). La única diferencia entre estas dos clases de relaciones es la carga de trabajo necesaria para montar las componentes. Por ejemplo, los componentes que implican las operaciones más críticas (*HPRC*) son más importantes que otros que causan molestias a pequeña producción (*LPRC*).

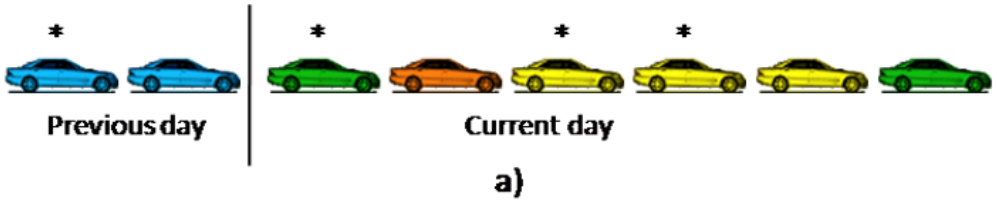
En algunos casos puede no existir secuencia que satisfaga todas las limitaciones de relación. En estos casos se dice que el problema es demasiado restringido. Por lo tanto, el objetivo es *reducir al mínimo el número de violaciones de las restricciones de relación*. Debido al hecho de que las restricciones de relación pueden ser violadas, se clasifican como *restricciones blandas*. Con el fin de obtener una distribución uniforme de los vehículos limitados, se calcula las violaciones de la restricción de relación en las subsecuencias de deslizamiento del día de producción. El propósito es espaciar los vehículos tanto como sea posible, debido a que si existe menor espacio entre los vehículos limitados, mayor es el número de violaciones.

La mejor manera de explicar cómo se calcula las violaciones de la restricción de relación es dando un ejemplo. Considerando una *restricción de relación* de 1/3 y la secuencia de vehículos ilustrada en la figura 2.3 donde los vehículos con un asterisco $(*)$ son los que requerirán la opción especial. Para evaluar esta secuencia hay que dividir la secuencia en grupos de $q = 3$ vehículos, lo que significa que serán 6 subsecuencias a evaluar. Por lo tanto, cada subsecuencia será evaluada por la siguiente fórmula:

$$\boldsymbol{Número\ de\ violaciones\ en\ la\ subsecuencia}^{1} = \left(\begin{array}{c}\boldsymbol{Número\ de\ vehículos\ asociados\ con\ la\ restricción\ de\ razón\ en\ la} \\ \boldsymbol{subsecuencia}\end{array}\right) - (\boldsymbol{Numerador\ de\ la\ restricción\ de\ relación})$$

Mediante la aplicación de esta fórmula para cada una de las subsecuencias, se verifica que no hay violaciones en la segunda y sexta subsecuencia, pero se tiene una violación en las cuatro subsecuencias restantes. Por lo tanto, toda la secuencia se evalúa a 4 violaciones.

[1] Notar que si el número de vehículos asociados con la restricción de razón en la subsecuencia es menor que el numerador de la restricción de razón, el resultado será obviamente cero y por ende no existirán violaciones en una subsecuencia.



Violations	1	0	1	1	1	0
Color Changes	1	1	1	0	0	1

Figura 2.3 Evaluación de restricción de relación y cambios de color. Fuente: de Oliveira, R. (2007)

2.2 Métodos de Resolución del Car Secuencing Problem

Existen muchos enfoques que pueden ser utilizados para resolver el *Problema de Secuenciamiento de Vehículos* (CSP) y encontrar el óptimo o al menos acercarse bajo una aproximación. Estos algoritmos pueden ser categorizados en dos clases: los *exactos* y *aproximados*. Éstos últimos son también llamados *heurísticas / metaheurísticas*. Los algoritmos *exactos* garantizan el encontrar una solución óptima, pero a su vez poseen tiempos de ejecución bastante grandes de acuerdo al tamaño de la instancia; donde una disminución de las instancias es suficiente para encontrar un óptimo. En este caso, la única posibilidad factible de resolución para instancias extensas se obtiene utilizando algoritmos *heurísticos / metaheurísticos*. En otras palabras, se sacrifica la búsqueda de la solución óptima por obtener buenas soluciones en un tiempo más acotado.

A continuación se presentan los distintos trabajos realizados categorizados según tipo (enfoques exactos y aproximaciones heurísticas / metaheurísticas), cada una con sus ventajas y aplicaciones.

2.2.1 Enfoques Exactos

Programación con restricciones

La Programación con Restricciones (*Constraint Programming*) es una herramienta genérica utilizada para resolver problemas que satisfagan restricciones (CSP's), por ejemplo, los problemas modelados por restricciones específicas en soluciones aceptables donde una restricción es una relación entre varias variables incógnitas, cada variable toma un valor de un dominio dado (Tsang, 1993). Para resolver un problema con restricciones con un lenguaje de programación lineal, debe especificarse por medio de variables y restricciones. El proceso se solución es manejado por algoritmos genéricos que son integrados dentro del lenguaje de programación lineal. Esos algoritmos llamados *solucionadores de restricciones* suelen basarse en una exploración sistemática de la búsqueda de espacio, hasta que se encuentre una solución o bien se informe que no se encuentra una solución. Con el fin de reducir el espacio de búsqueda, este enfoque es combinado con técnicas de filtrado que reducen los dominios variables con respecto a algunas consistencias parciales tales como "arco consistencia".

El CSP es un punto de referencia clásico para lenguajes de Programación Lineal, y es el primer problema de "*CSPLib*", la biblioteca de problemas de prueba para solucionador de restricciones en [12]. Esto ha sido usado para ilustrar las capacidades de resolución del lenguaje de PL.

La formulación del CSP usualmente introduce dos diferentes tipos de variables y tres diferentes tipos de restricciones:

- Una *variable de espacio* X_i es asociada con cada posición i en la secuencia de autos. Esta variable corresponde a la clase del i-ésimo auto en la secuencia y su dominio es el set de todas las clases de autos.

- Una *variable de opción* O_i^j que es asociada con cada posición i en la secuencia y cada opción j. Esta variable se le asigna 1 si la opción j debe ser instalada en el vehículo i de la secuencia, y se le asigna 0 en cualquier otro caso, por lo que su dominio es $\{0,1\}$.
- *Restricción de enlace*, especifica el enlace entre el lote y las variables de opción, por ejemplo, $O_i^j = 1$ si y sólo si la opción j debe ser instalada en X_i. Esas restricciones se presentan gracias a elementos de restricciones globales.
- *Restricciones de capacidad*, especifica el control de las capacidades de cada estación para que no sean excedidas. Por ejemplo, para cada opción j y para cada subsecuencia de vehículos q_i, una desigualdad lineal especifica que la suma de las correspondientes variables de opción deben ser menores o iguales a P_i.
- *Restricciones de demanda*, especifica que para cada clase de automóviles, el número de vehículos de esa clase deben ser secuenciados. Esas restricciones son expresadas gracias a la mayoría de restricciones globales (Van Hententyck et. al., 1991).

La mayoría de los trabajos que resuelven el CSP con CP consideran el problema de decisión, y la salida es ya sea una solución que satisfaga todas las restricciones de capacidades, o bien una falla explicando que no existe la solución.

Aplicaciones:

- Bergen et. al. en [1] propuso una aproximación basada en restricciones para un problema particular de *Secuenciamiento de Vehículos* que contiene ambas restricciones del tipo *duras* (que no deben ser violadas y por ende satisfechas) y restricciones del tipo *suaves* (que pueden ser violadas en términos de costo).

- Smith en [27] intentó mejorar el proceso de solución de la programación con restricciones, pues este tipo de programación ha mostrado ser efectiva para resolver instancias fáciles o pequeñas, pero no compite con los enfoques específicos de casos más difíciles o más grandes. Smith Propuso agregar valor a la programación al incluir heurísticas, su idea es programar los vehículos "difíciles" lo más pronto posible y donde la dificultad de un vehículo es definida con respecto a la tasa de utilización de sus opciones.

- Regin y Puget en [26] han introducido restricciones globales de secuenciamiento para imponer límites mínimos y máximos en el número de ocurrencias de valores, incluyendo periodos de unidades de tiempo consecutivas. Ellos propusieron un algoritmo filtrado que es dedicado a estas restricciones de capacidad, y eso hace que explote su semántica global para estrechar más eficientemente las variables de dominios. Este algoritmo filtrado ha sido integrado al solver de *Ilog*, un software de Programación con Restricciones y fue ilustrado en el CSP. Esto permitió al solver *Ilog* resolver las instancias de restricciones factibles más complejas, o para probar la infactibilidad de otras restricciones.

- van Hoeve et. al. [31] propusieron tres nuevos algoritmos filtrados para las restricciones de secuenciamiento globales. Los resultados experimentales sobre las instancias del conjunto de pruebas proporcionado por Lee et. al. en [17] muestran que combinando esos algoritmos al algoritmo filtrado de [26] mejora significativamente el proceso de solución, incluso en algunos casos donde no se resuelven en plazos razonables de tiempo.

Programación Entera

Drexl y Kimms en [9] propusieron un modelo de programación entera para la decisión en el CSP. El modelo es basado en variables binarias C_{ij} asociada con cada clase

de vehículos i y cada posición j, para decidir si el auto en la posición j es de la clase i. Las restricciones lineales aseguran que:

1) Exactamente una clase de vehículo es asignado a cada posición,
2) Todos los autos de cada clase son asignados a una posición,
3) Todas las restricciones de capacidad p_k/q_k son satisfechas para cada opción k.

El CSP es mezclado con un problema de programación de nivel, donde se considera el objetivo de minimizar la suma de desviaciones de los vehículos programados. Por lo tanto el problema considerado es el CSP con restricciones de capacidad complejas y un nivel de objetivo programado. Se propuso un segundo modelo de programación entera, basado en un número exponencial de variables $\{0,1\}$ Y_k tal que $Y_k = 1$ si la secuencia k es seleccionada, donde una secuencia está relacionada a una clase dando las posiciones de los vehículos que pertenezcan a esta clase. Los límites inferior y superior son computarizados a través de un método de generación de columna. Los resultados computacionales son presentados en un set de instancias generadas.

Aplicaciones:

- Gravel et. al. en [14] propusieron un modelo de programación entera para el CSP con restricciones de capacidad suaves. Este modelo asocia la variable C_{ij} ya usada en [9] y una variable binaria Y_{kj} para cada opción k y cada posición j; esto para decidir si la subsecuencia de longitud q_k que comienza en la posición j satisface la restricción de capacidad p_k/q_k. Las restricciones lineales son definidas para asegurar que las variables Y_{kj} son asignadas a uno sí y sólo si la subsecuencia correspondiente viola la restricción de capacidad. El objetivo es minimizar la suma de todas las variables Y_{kj}. Esta formulación de programación entera permitió a los autores encontrar soluciones factibles para todas las instancias del test propuestas

por Lee en [17] en *CSPLib* y cuatro instancias del test propuestas por Smith. Sin embargo, esto puede no probar la optimalidad para las cinco otras instancias.

Método Ad – hoc

Drexel et. al. en [10] propusieron un método de *branch and bound* para resolver el *Secuenciamiento de Vehículos* y el problema de programación de nivel, éstos fueron ya considerados en [9]. Considerando sólo el *Secuenciamiento de Vehículos*, el método es basado en un esquema de ramificación original y basado en el concepto de estado de S*ecuenciamiento de Vehículos* (CS). Un estado de CS está asociado a cada nodo del árbol de *branch and bound*. Además de la secuencia parcial correspondiente al nodo, el estado de CS es caracterizado por una matriz $m_{ij_{0x\{1,\cdots,|V|\}}}$ donde:

$$m_{ij=}\begin{cases} m_{ij} = 0 \; si \; la \; opción \; i \; está \; presente \; en \; la \; posición \; j \\ m_{ij} = 1 \; si \; la \; opción \; i \; puede \; ser \; planificada \; en \; la \; posición \; j \\ m_{ij} = -1 \; si \; la \; opción \; i \; no \; puede \; ser \; planificada \; en \; la \; posición \; j \end{cases}$$

El caso $m_{ij} = -1$ es: ya sea por las decisiones de ramificación o bien porque se está violando alguna restricción.

2.2.2 Aproximaciones Heurísticas

Durante el estudio del *Car Sequencing Problem* se han propuesto diferentes aproximaciones incompletas que dejan de lado la exhaustividad, tratando de encontrar rápidamente soluciones aproximadas de manera oportunista. Por ejemplo, la *Búsqueda Golosa (Greedy Search)*, la *Búsqueda Local*, *Algoritmos Genéticos* y la *Optimización Basada en Colonias de Hormigas (Ant Colony Optimization)*.

Búsqueda Golosa

Dado un CSP, podemos construir una secuencia de una forma codiciosa (golosa), empezando desde una secuencia inicial e iterativamente agregando un nuevo vehículo al final de la secuencia, con respecto a alguna función heurística golosa. Una primera aproximación golosa fue propuesta por Hindi y Ploszajski en [15]. En 2003, Gottlieb et. al. [13] propusieron y compararon seis diferentes heurísticas golosas para el CSP. La mejor heurística realizada, entre las seis heurísticas consideradas está basada en la "*suma dinámica de las tasas de utilización*" es decir, en cada paso iterativo se añade el vehículo que maximice la suma de las opciones requeridas en tasas de utilización, esa tasa se actualiza cada vez que un nuevo vehículo es agregado al final de la secuencia. Gottlieb demostró que este tipo de construcción golosa cuando se combina con una leve cantidad de aleatorización y reinicios múltiples, puede resolver rápidamente todas las instancias del conjunto de pruebas propuestos por Lee [17] en *CSPLib*.

Aproximaciones de la Búsqueda Local

La idea de la *Búsqueda Local* es mejorar una secuencia mediante una exploración local de su "*vecindad*", es decir, el set de secuencias que pueden ser obtenidas a partir de la secuencia actual mediante la realización de una transformación elemental, llamada "*movimiento*". A partir de una secuencia inicial dada, la búsqueda del espacio es explorada de vecindad a vecindad hasta que una secuencia óptima haya sido encontrada o hasta un máximo número de movimientos probados. Muchos enfoques distintos de *Búsqueda Local* fueron propuestos para resolver el CSP, la mayoría de ellos son enfoques genéricos del rendimiento de los cuales se han ilustrado, entre otros problemas, en el P*roblema de Secuenciamiento de Vehículos* en [5, 6, 17, 18, 19, 22]. Sin embargo, algunos enfoques de la B*úsqueda Local* han sido específicamente dedicados a este problema en [13, 23, 25].

Los rendimientos de estos diferentes enfoques de búsqueda local han sido ilustrados en los casos de referencia de *CSPLib* [12]. Todos los enfoques recientes propuestos en [18, 19, 22, 23] han sido muy efectivos en todas esas instancias.

Los enfoques de *Búsqueda Local* para resolver el CSP difieren principalmente con respecto a:

1) La forma de cómo la secuencia inicial es construida,
2) La vecindad considerada en cada movimiento,
3) La "metaheurística" considerada para elegir dentro de la vecindad.

Ant Colony Optimization

La idea básica del ACO en base a [8] es modelar el problema para resolverlo con la búsqueda de un camino de mínimo costo en una gráfica, y usar hormigas artificiales para buscar buenos caminos. El comportamiento de las hormigas está inspirado por las verdaderas hormigas: éstas colocan caminos de feromonas en las componentes de la gráfica y eligen su camino con respecto a probabilidades que dependen de los caminos de feromona que han sido previamente impuestos por la colonia; esos caminos de feromona decrecen progresivamente por la evaporación. Intuitivamente, esta comunicación *Estigmergética*[2] significan objetivos que dan información acerca de la calidad de las componentes de la ruta para atraer hormigas, en las siguientes iteraciones, hacia las correspondientes áreas de la búsqueda de espacio.

[2] De la palabra Estigmergia: Proceso donde las acciones individuales no organizadas sirven como estímulo para las acciones de otros individuos, y como resultado se tiene un grupo de personas que colectivamente se comportan como una entidad única.

Aplicaciones:

- Solnon [28] propuso el primer algoritmo ACO para problemas de satisfacción de restricciones de permutación; la solución de las cuales es una permutación de una tupla de valores dado. El rendimiento de este algoritmo ha sido ilustrada, como en otros problemas, en el CSP. En este algoritmo, las feromonas son colocadas en parejas de autos consecutivos para así enseñar a las prometedoras sub – secuencias de vehículos.

- Gottlieb en [13] probaron y mejoraron el primer algoritmo de ACO mediante heurísticas golosas, y ha sido experimentalmente comparado con el enfoque de *Búsqueda Local* de [25], mostrando que la B*úsqueda Local* es ligeramente inferior al ACO para límites pequeños de tiempo en CPU, mientras que para límites extensos de tiempo ambos avances rinden una solución de calidad comparables.

- Gravel et. al. [14] propusieron otro algoritmo de ACO para el CSP, que integra un procedimiento de *Búsqueda Local* que es usado para probar las soluciones construidas por las hormigas.

2.2.3 Aproximaciones Metaheurísticas

Recocido Simulado (Simulated Annealing)

Es un método heurístico que guarda relación con un campo muy diferente al de la optimización, la termodinámica. En cada iteración una vecindad es generada (una secuencia factible se modifica ligeramente y de forma aleatoria para crear una nueva secuencia que sea también factible). Este vecino es aceptado como una secuencia actual si se considera que tiene baja penalidad. Si por otra parte, este mismo vecino tiene una alta penalización,

esta podría ser aceptada como la actual solución, es decir como una secuencia acorde a una probabilidad relacionada con un parámetro de control denominado *temperatura*. La temperatura, y por lo tanto la probabilidad de que vecinos con alta penalidad sean aceptados, va disminuyendo en cada iteración a más usualmente después de un número de iteraciones. La relación de este método con la termodinámica está en el hecho de que el proceso se asemeja al enfriamiento en el recorrido de metales.

Aplicaciones:

- Briant et. al. en [2] Proponen un algoritmo usando alcances de *Simulated Annealing* tomando en consideración los tres objetivos de optimización en forma decreciente (del más importante al menos importante) y optimiza el método al disminuir en un criterio a la vez. Utiliza una gama de movimientos y se escogen mediante las probabilidades dinámicas que tienen en cuenta la tasa de éxito de cada movimiento para el caso dado. El resultado de este algoritmo es cercano a los que se encontraron en el desafío ROADEF 2005, donde se aprovechó esta base de datos para obtener los resultados.

Algoritmos Genéticos

Este alcance se inspira desde la evolución natural y explora la búsqueda de espacio mediante operadores de selección, cruce y mutación en una población de secuencias. En cada generación, las secuencias elegidas son combinadas mediante operadores de cruce (*cross – over*); como los descendientes creados pueden no satisfacer la restricción de permutación global, son "golosamente" reparados; luego de esto, cada descendiente es "encumbrado" (*hill – climbing*) mediante una función de intercambio (similar a la utilizada en los enfoques de *Búsqueda Lineal*).

Aplicaciones:

- Warwick y Tsang en [32] propusieron un algoritmo genético para resolver el CSP. Los experimentos reportados muestran que este alcance permite resolver instancias sencillas del CSP, con bajos porcentajes de utilización. Sin embargo, con altos porcentajes de utilización, el número de arranques exitosos disminuye considerablemente.
- Cheng et. al. [3] propusieron un algoritmo evolutivo computacional para resolver prácticamente el CSP en *Ford Motor Company*, involucrando restricciones de pintura como en el problema presentado para ROADEF 2005. Los autores propusieron un operador *cross – over* simple, llamado *operador cross – switching*, generando una descendencia mediante la permutación de vehículos que aparecen en una posición elegida al azar en las secuencias padres. Como en [32], los descendientes son reparados mediante operadores de permutación.

Búsqueda Tabú

Dado que el método de resolución del *Problema de Secuenciamiento de Vehículos* se basa en esta metaheurística, será abordado con más detalles a continuación.

La *Búsqueda Tabú* es un tipo de búsqueda por entornos que permite moverse a una solución del entorno aunque no sea mejor que la actual, de este modo se puede escapar de *Óptimos Locales* y continuar la búsqueda de soluciones aún mejores. La forma de evitar viejos *Óptimos Locales* es clasificando un determinado número de los más recientes movimientos como *Movimientos Tabú*, los cuales no serán posibles de repetir durante un determinado horizonte de tiempo. Por lo tanto, en este caso el escape de los *Óptimos Locales* se produce de forma sistemática y no aleatoria.

A continuación se explican algunos aspectos básicos de la *Búsqueda Tabú*:

1) *Solución Inicial* x_0

Dependerá del algoritmo específico que la genera, se puede usar una heurística para proporcionar una buena *Solución Inicial* o se puede partir de una solución obtenida aleatoriamente. Lo usual es utilizar el conocimiento que se tiene de dónde podría haber una buena solución, pero si no se tiene ninguna información puede partirse de cualquier valor y mejorarlo en el proceso de solución.

2) Generación de Vecindad

Un vecino de una configuración x es una configuración x' obtenida a partir de x por medio de un movimiento simple que forma parte de la vecindad $N(x)$. La mayoría de las veces la vecindad $N(x)$ es muy grande, por cuanto implica un elevado esfuerzo de cómputo analizarlo completamente en el proceso de *Búsqueda Local*, por consiguiente se debe reducir el número de vecinos a un conjunto $N^*(x)$ más pequeño que $N(x)$, ilustrado en la figura 2.4. A continuación se mencionan algunas formas de reducir la vecindad:

- Tomar un número determinado de configuraciones seleccionadas aleatoriamente del conjunto $N(x)$.
- Aspiración adicional: Se utiliza algún criterio como puede ser la reducción de la *Función Objetivo* hasta cierto valor. Cuando se encuentra una configuración que cumpla la condición, se evalúan otras configuraciones adicionales. Se deben fijar los números mínimo y máximo de configuraciones a ser analizadas.
- Lista de reducción de candidatos de élite: Se selecciona un conjunto de configuraciones vecinas con atributos particulares, las cuales son llamadas configuraciones de élite.

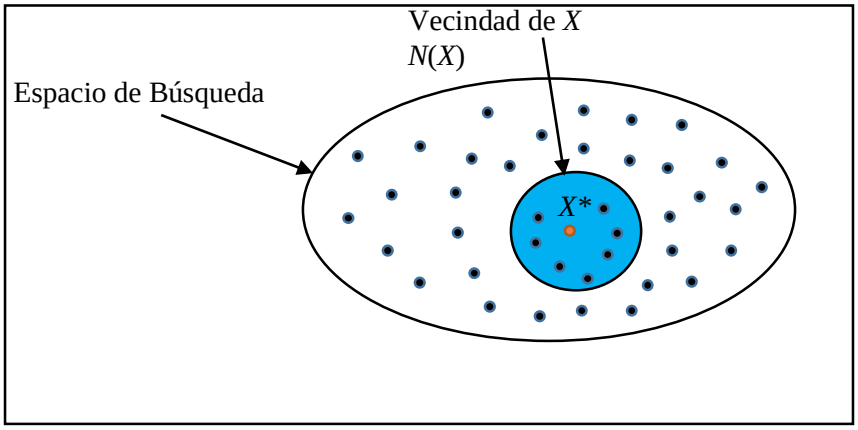


Figura 2.4 Esquema de Vecindad. Fuente: Elaboración Propia

3) Selección de la mejor Vecindad

Se evalúa cada vecino, se determina su *Función Objetivo* y se verifica si cumple las restricciones del problema a resolver para determinar la factibilidad de la configuración vecina. Los vecinos son clasificados en una lista de acuerdo con el valor de la *Función Objetivo* y el proceso selecciona el mejor candidato, técnica conocida como *best improvement rule*.

El primer candidato de la lista (de la mejor *Función Objetivo*) es seleccionado siempre que el movimiento efectuado para pasar de la configuración actual a la configuración vecina no se encuentre prohibido (estado *Tabú*); si es factible este modo de selección, se denomina *selección agresiva*, que imita las características de un algoritmo *goloso*.

Si el mejor vecino de la lista de candidatos está clasificado como *Tabú* por el proceso de optimización, el criterio de aspiración permite que sea seleccionado a pesar de la prohibición, siempre que la configuración o solución que se genere a partir de él, sea la mejor encontrada hasta el momento.

Si en la vecindad generada no existe ninguna configuración que mejore la *Función Objetivo*, entonces se selecciona la mejor de las peores en tanto que no haya sido clasificada como *Tabú* durante el proceso de optimización. Esta estrategia evita que se quede atrapado en *Óptimos Locales*.

4) Actualización de la Estructura Tabú

Búsqueda Tabú usa una estructura que tiene la misma codificación de la configuración x para almacenar la información de los atributos que han cambiado o que han permanecido inalterables en el proceso de búsqueda. Esta información se almacena en la *Memoria de Corto Plazo*, la cual pueden tener un criterio de almacenamiento fijo o variable a lo largo del proceso dependiendo del comportamiento de la búsqueda. Esto es lo que constituye la memoria adaptativa. La memoria usada en *Búsqueda Tabú* puede ser explícita o por atributos. La memoria explícita almacena la información completa de las configuraciones de élite (configuraciones de buena calidad) encontradas durante la búsqueda. La memoria basada en atributos almacena la información de los atributos que cambian al pasar de una configuración a otra, presentando como ventajas la facilidad de almacenamiento, manipulación y verificación. Con esto se evita regresar a configuraciones ya visitadas, lo cual es ventajoso; pero también se podrían dejar de conocer regiones no visitadas que tengan atributos prohibidos de aquellas ya exploradas.

5) Memoria a Largo Plazo

En algunas aplicaciones también se incorpora el concepto de *Memoria a Largo Plazo* (contraria a la *Memoria de Corto Plazo* descrita anteriormente) y sus estrategias asociadas. Esto en general consiste en almacenar soluciones denominadas de élite (*Óptimo Local* de alta calidad), que fueron encontradas en distintas etapas del proceso de búsqueda, para que puedan ser aprovechadas más adelante, como por ejemplo en la etapa de intensificación que

se describirá a continuación. Otra alternativa interesante, consiste en tratar que el manejo de la lista fuerce a que en general las soluciones a analizar contengan estos atributos.

6) Intensificación y Diversificación

En ciertos momentos del proceso de búsqueda puede ser aconsejable centrar la búsqueda en la región actual del espacio de soluciones, si ésta contiene soluciones aceptables. Dicha *Intensificación* se puede conseguir dando prioridad a las soluciones que tienen características en común con la solución actual y penalizando a aquellas que están más alejadas. En otro momento puede ser aconsejable alejarse de la región actual y centrarse en otra región, quizás porque la región actual ya no contiene más soluciones aceptables o porque la otra región parece más prometedora. Esta *Diversificación* se puede conseguir penalizando a las soluciones más parecidas a la solución actual y dando prioridad a aquellas que difieren más. Por lo tanto se puede sustituir la *Función Objetivo* del problema original f, por una nueva *Función Objetivo* f' que incorpore dichas características.

El peso asignado a la *Intensificación* y a la *Diversificación* es variable y será modificado dependiendo de las necesidades que se produzcan durante el proceso de búsqueda.

7) Algoritmo de *Búsqueda Tabú* simple

Procedimiento de Búsqueda Tabú:

Sea $x^* \in \Omega$ la mejor solución encontrada

Sea N_A el nivel de aspiración

$x^* \leftarrow x_0$

WHILE no se cumpla criterio de parada

Seleccionar el conjunto $V \subseteq N(x^*)$ de vecinos de x^*

Sea $x' \in V$ la mejor solución en V

IF el movimiento que generó *x'* no está en lista tabú

Acepte el movimiento haciendo $x^* \leftarrow x'$

Actualice lista tabú e incorpore el movimiento que generó *x'*

Actualizar N_A

IF NOT

IF se cumple el criterio de aspiración

Acepte el movimiento haciendo $x^* \leftarrow x'$

Actualice lista tabú e incorpore el movimiento que generó *x'*

END IF

END IF

END WHILE

Figura 2.5 Procedimiento de Búsqueda Tabú. Fuente: Elaboración Propia

Aplicaciones:

- Zufferey et. al. en [33] proponen un algoritmo de *Búsqueda Tabú* para el CSP basado en la base de datos de *Renault* para ROADEF 2005, donde abordaron cada objetivo de forma separada sin degenerar el valor de la *Función Objetivo*. Las soluciones obtenidas se consideraron mejores que los resultados obtenidos por los participantes de ROADEF 2005.

- Cordeau et. al. en [4] desarrollaron una heurística de *Búsqueda Tabú Iterativa* para solucionar el CSP del desafío ROADEF 2005. El algoritmo inicia bajo una *Búsqueda Tabú* para encontrar la mejor solución alrededor de una acotada vecindad de x y realizando 6 tipos de *perturbaciones* para escapar del *Óptimo Local* vuelve a ejecutar un algoritmo de *Búsqueda Tabú*. Los resultados se asemejan a los obtenidos por los grupos participantes de la competencia del 2005.

3 FORMULACIÓN DEL PROBLEMA

En este proyecto se implementó un algoritmo mediante Microsoft Visual Studio 2005 bajo el lenguaje de programación *Visual Basic .Net* (o por simplicidad: *.Net*). Este algoritmo se implementó mediante una metaheurística de tipo *Búsqueda Tabú* para dar una solución al *Problema de Secuenciamiento de Vehículos* con *Restricciones de Lote de Pintura* asignadas por parte de *Renault*.

Mediante la investigación realizada en los capítulos anteriores, la modelación y formulación implementada es la proporcionada por Cordeau et. al. en [4], esta formulación del problema presentó simplificación de variables importantes en el momento de llevar a cabo la programación. Esta formulación y evaluación va de la mano de las especificaciones impuestas por parte de *Renault*, especificadas por Nguyen en [20].

Se desarrolló esta metaheurística utilizando la técnica de generación de vecindad propuesta en [4]. Esta metaheurística de *Búsqueda Tabú* se desarrolló con la técnica de pivoteo del tipo *best improvement rule* donde en cada iteración de la vecindad se obtiene el mejor movimiento, este mejor movimiento se reflejó en la función objetivo como el mínimo costo obtenido en la iteración.

3.1 Función Objetivo de ROADEF

Debido a que existen diferentes niveles de prioridad entre los objetivos, los organizadores del evento ROADEF 2005 elaboraron una función objetivo del tipo jerárquica con tres términos, que representa a los tres objetivos de optimización: el número de cambios de color, el número de violaciones de restricciones de tipo *HPRC* y el número de violaciones de restricciones de tipo *LPRC*. Como fue revisado anteriormente, las restricciones de tipo *HPRC* tienen una prioridad más alta que las de tipo *LPRC*, lo que significa que existen tres tipos de jerarquías:

- Cambios de Color, HPRC's, LPRC's; (1)
- HPRC's, LPRC's, Cambios de Color; (2)
- HPRC's, Cambios de Color, LPRC's. (3)

En general la función objetivo para este problema se define como:

$$f(s) = Min\ \alpha \cdot C(s) + \beta \cdot \sum_{r \epsilon P} d_r(s) + \gamma \cdot \sum_{r\ \epsilon\ N} d_r(s)$$

Donde:

- $C(s)$ es el número de Cambios de Color requeridos en la secuencia;
- $d_r(s)$ es el número de violaciones de restricciones de relación de tipo $^{p}/_{q}$;
- $P \subseteq R$ y $N \subseteq R$ representan los subsets de restricciones de relación del tipo *HPRC* y *LPRC* respectivamente;

- α, β, γ son los pesos asociados a cada uno de los parámetros.

Importante destacar que los pesos asociados a las restricciones de pintura y relación son variables para cada instancia. Específicamente existen diferentes escenarios propuestos por cada industria automotriz, donde:

3.1.1 Jerarquía tipo (1): Prioridad a la Restricción de Pintura

1. Minimizar el número de cambios de color de la pintura
2. Minimizar el número de violaciones de restricciones de relación con alto nivel de prioridad (*HPRC*)
3. Minimizar el número de violaciones de restricciones de relación con bajo nivel de prioridad (*LPRC*)

$$\boldsymbol{\alpha = 10^6, \beta = 10^3, \gamma = 1}$$

1. Minimizar el número de cambios de color de la pintura
2. Minimizar el número de violaciones de restricciones de relación con alto nivel de prioridad (*HPRC*)

$$\boldsymbol{\alpha = 10^6, \beta = 10^3, \gamma = 0}$$

3.1.2 Jerarquía tipo (2), Prioridad a Restricciones de Relación

1. Minimizar el número de violaciones de restricciones de relación con alto nivel de prioridad (*HPRC*)
2. Minimizar el número de violaciones de restricciones de relación con bajo nivel de prioridad (*LPRC*)
3. Minimizar el número de cambios de color de la pintura

$$\boldsymbol{\alpha = 1, \beta = 10^6, \gamma = 10^3}$$

3.1.3 Jerarquía tipo (3), Prioridad a Restricción de Relación *HPRC*

1. Minimizar el número de violaciones de restricciones de relación con alto nivel de prioridad (*HPRC*)
2. Minimizar el número de cambios de color de la pintura
3. Minimizar el número de violaciones de restricciones de relación con bajo nivel de prioridad (*LPRC*)

$$\boldsymbol{\alpha = 10^3, \beta = 10^6, \gamma = 1}$$

1. Minimizar el número de violaciones de restricciones de relación con alto nivel de prioridad (*HPRC*)
2. Minimizar el número de cambios de color de la pintura

$$\boldsymbol{\alpha = 10^3, \beta = 10^6, \gamma = 0}$$

3.2 Restricciones a Implementar

La especificidad del problema considera la restricción primaria o fuerte que determina la factibilidad de una solución al problema, y las restricciones secundarias o débiles que representan los criterios para evaluar la calidad de una solución.

3.2.1 Restricciones Primarias

1) Los vehículos de la producción de una instancia no pueden ser mezclados con vehículos de otra instancia.

2) Debe existir un espaciado mínimo entre dos vehículos, a fin de que sólo uno a la vez ingrese a la estación correspondiente (montaje en línea).

3) El tamaño de un lote de vehículos que tengan asignado el mismo color no puede exceder el límite de lote de pintura.

4) Sea D el conjunto de vehículos programados para el día actual, se tiene que los vehículos del día de producción $D-1$ ya están programados y no pueden cambiar de posición al entregar una nueva solución.

3.2.2 Restricciones Secundarias

1) Los vehículos tienen una posición disponible para poder ser programados. Existen penalidades por asignar a las subsecuencias que no puedan cumplir las restricciones de relación, por lo tanto la programación será de mejor calidad mientras se tenga la menor cantidad de violación de estas restricciones.

2) Un escenario es considerado que tiene *restricciones de relación con alto nivel de prioridad y fáciles de satisfacer* si la aplicación industrial de *Renault* genera una programación del día de producción sin ninguna violación de las restricciones de relación. Las *restricciones de relación con alto nivel de prioridad y fáciles de satisfacer* serán siempre *factibles.*

3) Un escenario es considerado que tiene *restricciones de relación con alto nivel de prioridad y difíciles de satisfacer* si la aplicación industrial de *Renault* no puede generar una programación del día de producción sin ninguna violación de las restricciones de relación con alto nivel de prioridad. Esto ocurre cuando el número de vehículos restringidos sobrepasa las restricciones de relación: para una instancia, la producción del día contiene el 25% de vehículos restringidos y hay una restricción de relación de alta prioridad $1/5$.

3.3 Datos de entrada

La Función Objetivo presentada en la sección 3.2 se alimenta de tres parámetros impuestos por las restricciones del CSP de Renault. Los datos que serán codificados para posteriormente ser ingresados como la información para la ecuación son los siguientes:

3.3.1 Cantidad de cambios de color en la secuencia

Siendo $C(s)$ la cantidad de cambios de color que requiere la secuencia, lo que significa la limpieza con solventes que en muchos casos tiende a ser limitada. Esta variable comenzará en cero y aumentará conforme se evidencie un requerimiento de color distinto.

El algoritmo evaluará el cambio de color entre el vehículo actual y su antecesor. Siendo V_i un vehículo en la posición i, $C(s)$ aumentará en 1 unidad si c_i es distinto a c_{i+1}, el valor de $C(s)$ no será modificado en el caso que c_i sea idéntico a c_{i+1}.

3.3.2 Violación de restricciones de relación en subsecuencias

En el caso de $d_r(s)$ se evalúa el número de violaciones de Restricciones de tipo Prioritarias $HPRC$ y No Prioritarias $LPRC$, el cual comienza en cero y aumentará en la medida que se encuentre una subsecuencia que no cumpla con las restricciones de los vehículos.

Para Restricciones Prioritarias y No Prioritarias la forma de evaluar estas violaciones será como la ecuación ilustrada en la sección 2.1.3; es decir que el número de violaciones se cuantifica al identificar la cantidad de vehículos que se asocian con la restricción de relación, y restándole a ese valor el numerador de la restricción de relación. El valor de $d_r(s)$ cambia según el subconjunto de vehículos con restricciones de relación a nominar, donde $P \subseteq R$ y $N \subseteq R$ representan los subsets de restricciones del tipo *HPRC* y *LPRC* respectivamente.

4 ALGORITMO DE TIPO BÚSQUEDA TABÚ

En este capítulo se explican los distintos elementos que componen el algoritmo *Metaheurístico* de tipo *Búsqueda Tabú* que se propone para solucionar el CSP que se plantea en este trabajo.

4.1 Elementos input del algoritmo

El elemento de entrada para el algoritmo de *Búsqueda Tabú* es la tupla compuesta por (V, O, p, q, r). En términos prácticos el algoritmo interpreta los archivos de tipo *".txt"* y los codifica para obtener resultados; estas instancias contienen los siguientes elementos:

4.1.1 Objetivos de Optimización

La estructura de este archivo muestra en su primera línea `rank;objective name`; por ejemplo, si el algoritmo encuentra una línea de tipo `2;high_priority_level_and_easy_to_satisfy_ratio_constraints;` interpretará que la Restricción de Relación de tipo *HPRC,* que tiene la particularidad de ser fácil de satisfacer, será la segunda restricción que debe ser optimizada en esta instancia.

En la figura 4.1 se muestra un diagrama resumen del proceso de asignación de prioridades de optimización del algoritmo presentado en este capítulo:

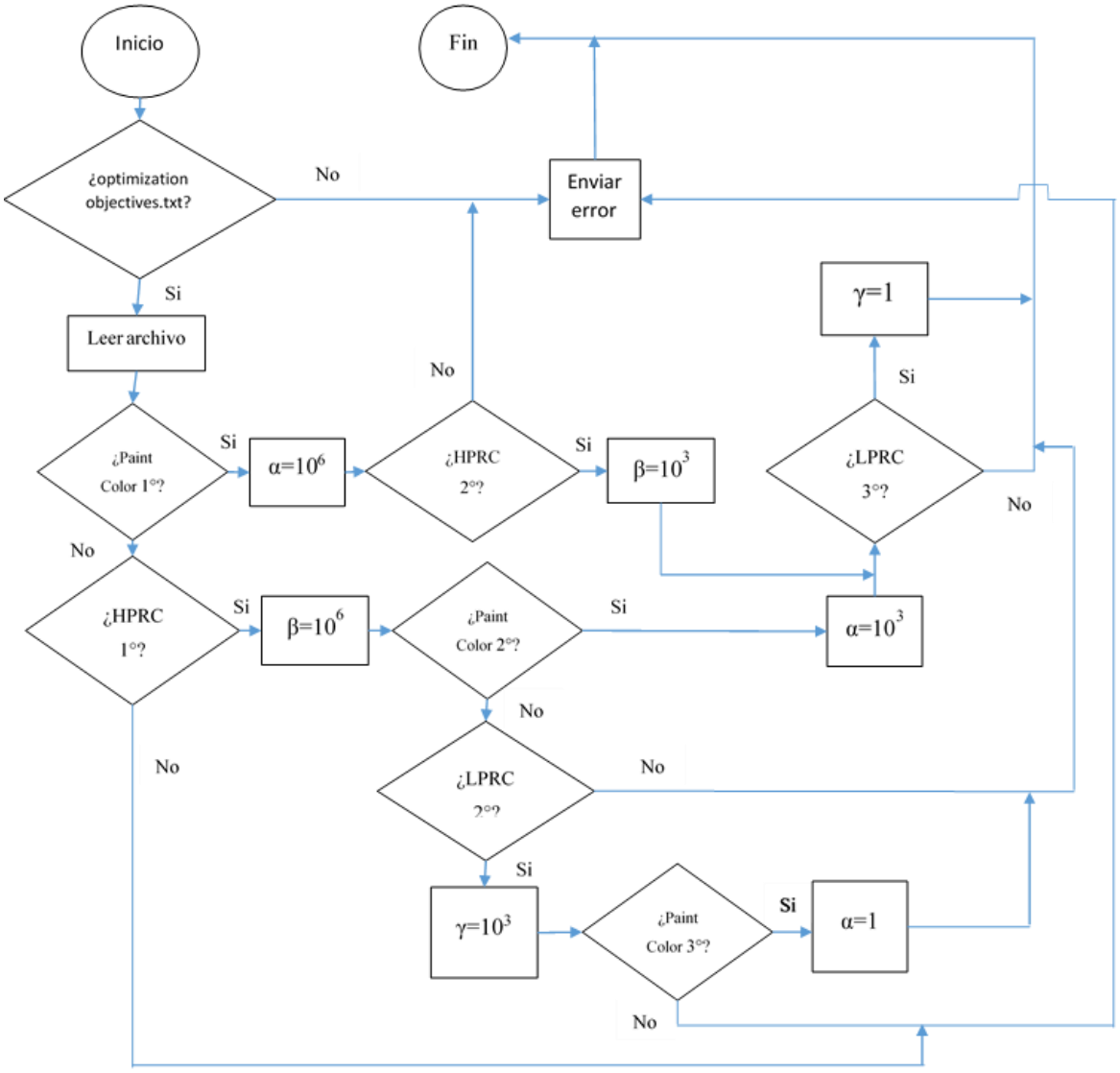


Figura 4.1 Diagrama de Flujo para definición de prioridades de optimización. Fuente: Elaboración Propia

4.1.2 Límite de Lotes de Pintura

El archivo de nombre *paint_batch_limit.txt* muestra en su primera línea `limitation;` lo que significa que al leer una única línea que contenga `450;` interpretará que en la secuencia no pueden existir más de 450 cambios de color ya que inevitablemente la solución será infactible.

En la figura 4.2 se muestra un diagrama resumen del proceso de conteo de cambios de color de pintura que realiza el algoritmo:

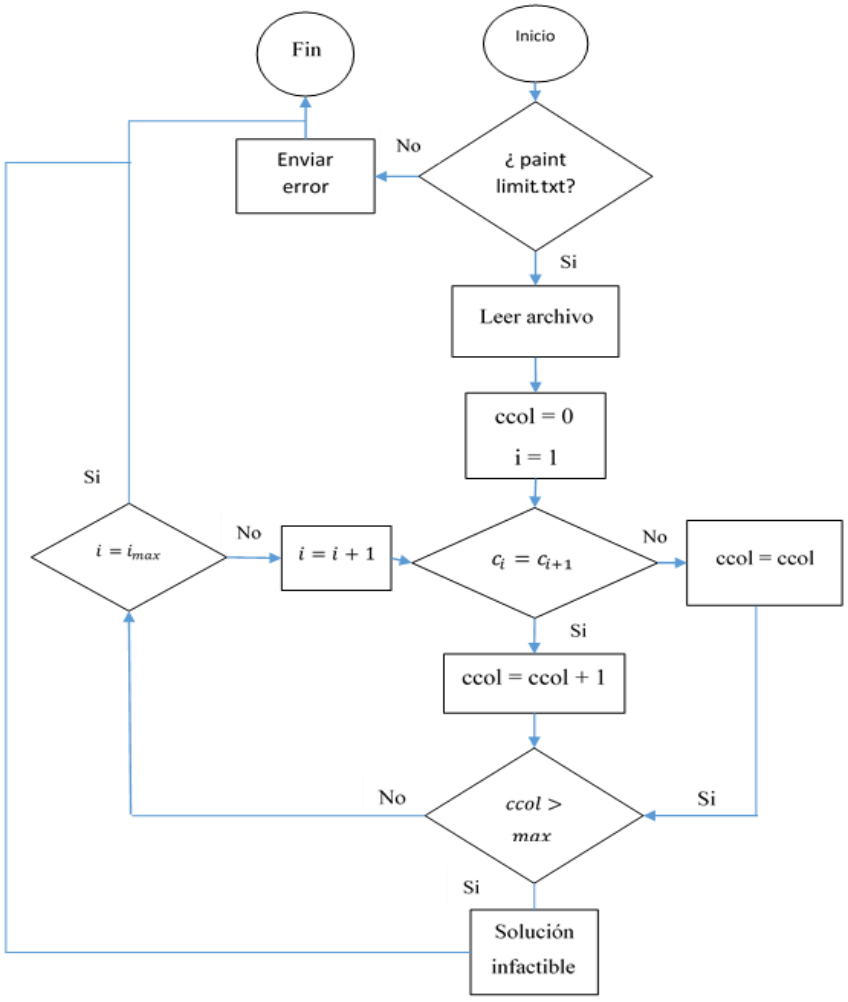


Figura 4.2 Diagrama de Flujo para conteo de cambios de color. Fuente: Elaboración Propia

4.1.3 Restricciones de Relación

El archivo *ratios.txt* contiene en su primera línea los identificadores `Ratio;Prio;Ident;` lo que implica que al leer líneas como `1/3;1;HPRC2;` y `1/10;0;LPRC3;` el algoritmo entenderá que la restricción HPRC número 2 con prioridad tiene como valor 1/3 (entre dos vehículos que requieren esta configuración habrá un espaciado de al menos 2 vehículos que no requieran la configuración), mientras que para la segunda línea se entenderá que la restricción LPRC número 3 sin prioridad tiene como valor 1/10 (entre dos vehículos que requieren la configuración habrá un espaciado de al menos 9 vehículos que no requieran la configuración).

A continuación se presenta en la tabla 4.3 el diagrama resumen del proceso de asignación de valores para *HPRC* y *LPRC* que serán utilizados para ser evaluados en el listado de vehículos:

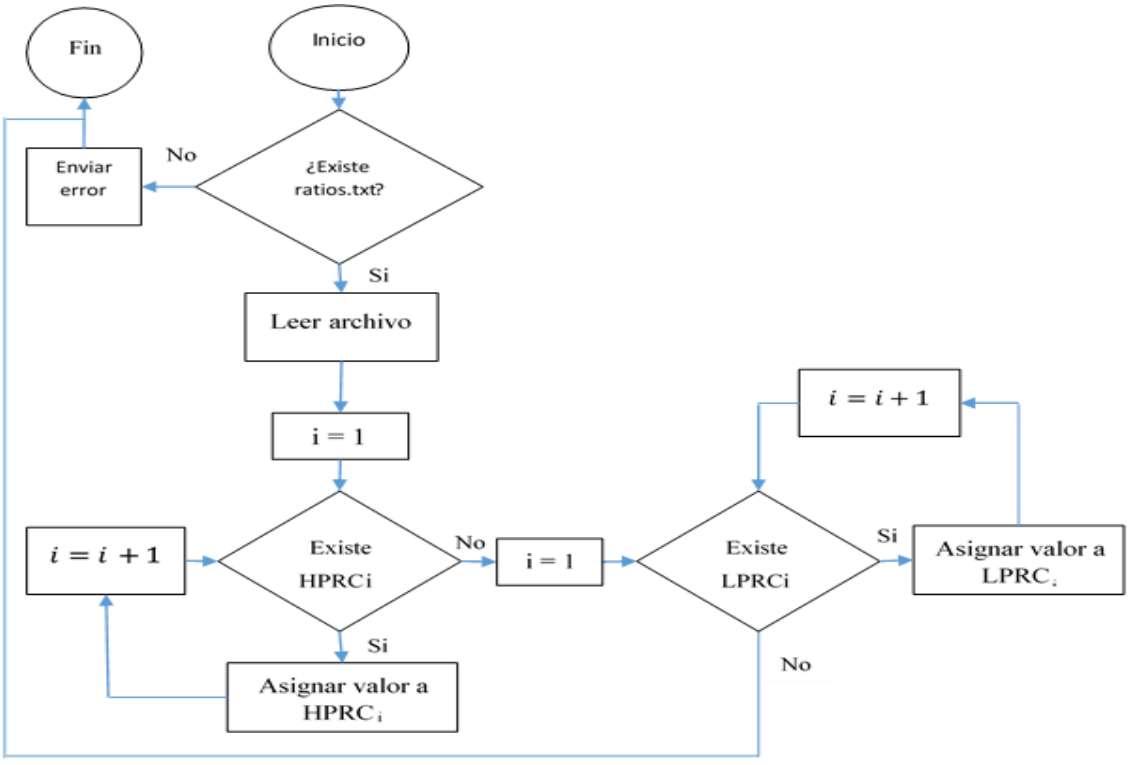


Figura 4.3 Diagrama de Flujo para valores de restricciones de relación. Fuente: Elaboración Propia

4.1.4 Secuencia Inicial de Vehículos

Dado el concepto del problema, el archivo de texto *Vehicles.txt* es el más importante puesto que sus datos son complementados con los respectivos archivos de optimización, restricciones y límite de pintura. El archivo que el algoritmo tiene que interpretar es posible explicarlo con el siguiente ejemplo de una subsecuencia en la tabla 4.1-1:

Date	SeqRank	Ident	Paint Color	HPRC1	HPRC2	LPRC1	LPRC2	LPRC3
2003 38 3	497	022033830010	1	0	0	0	0	0
2003 38 3	498	022033830154	1	1	0	0	0	0
2003 38 3	499	022033830167	1	1	0	0	0	0
2003 38 4	1	022033830049	3	1	0	0	1	0
2003 38 4	2	022033840384	3	1	1	0	0	1
2003 38 4	3	022033840247	3	1	1	0	1	0

Tabla 4.1 Ejemplo de un archivo “vehicles.txt”. Fuente: Elaboración Propia

Los primeros tres vehículos son de la programación del día anterior, lo que significa que la programación es realizada para el día 18 de Septiembre de ese año. En la segunda columna se muestra el ranking de llegada de los pedidos, mientras que la tercera columna corresponde al número único que identifica a ese vehículo. En esta subsecuencia se necesita un cambio de color puesto que luego de programar el tercer vehículo se debe ocupar la pintura número 3 que se mantendrá constante durante los tres vehículos restantes. Finalmente para cada restricción de relación se tiene numeración del tipo {0,1} e implica que si un vehículo requiere esa configuración tendrá valor 1, y tendrá 0 cuando no sea requerida esa configuración.

4.2 Relajación de la Función Objetivo

Las soluciones infactibles son permitidas a lo largo de la búsqueda al relajar la restricción de pintura y al penalizar estas violaciones en la *Función Objetivo*. La *Función Objetivo* $f(s)$ es reemplazada con una función $g(s) = f(s) + \zeta \cdot t(s)$, donde $t(s)$ es el *número de violaciones de la restricción de pintura* en s; ζ es inicialmente equivalente a 1 y autoajustado en el transcurso de la búsqueda para permitir un mix de soluciones factibles e infactibles. En cada iteración el valor de ζ es multiplicado por 2 si la solución en ese instante es infactible, y dividido por 2 en otro caso.

4.3 Construcción de la Solución Inicial

Dado que el algoritmo permite soluciones intermedias infactibles (es decir, soluciones que violen la *restricción de lotes de pintura*), este puede ser iniciado con cualquier permutación de los elementos de V. En lo práctico se obtiene como solución inicial s_0 los vehículos en el orden en el cual fueron definidos en el input del problema.

4.4 Generación de Vecindad

La *Vecindad* de un *Evento* consiste en generar un número determinado de posibles movimientos de éste dentro de la estructura, haciendo pequeños cambios a la configuración actual para minimizar el valor de la *Función Objetivo*.

Puesto que la totalidad de la *Vecindad* de un *Evento* es extensa, para hacer más eficiente este proceso del algoritmo se escoge una cierta cantidad de vecinos. Para movimientos de *tipo 1* se escogen TV[3] par de posiciones (i, j) para generar la vecindad, mientras que para movimientos de *tipo 2* se eligen sólo TV par de posiciones (i, j) que generen una mejora en la solución actual.

El algoritmo debe alternar entre los movimientos siguientes para crear la *Vecindad* $N(s)$:

- *Movimiento tipo 1*: Insertar en la posición j el vehículo v_i actualmente asignado a la posición $i \neq j$;
- *Movimiento tipo 2*: Intercambiar los vehículos v_i y v_j actualmente asignados a las posiciones i y $j \neq i$.

En intercambios de *tipo 1*, v_{i-1} y v_{i+1} serán consecutivos luego de la inserción (salvo que $i = n$ donde v_{i-1} se convierte en el último vehículo en la secuencia), y v_i es insertado entre v_j y v_{j+1} (ver figura 4.4.1). En intercambios de *tipo 2*, v_i y v_j son intercambiados y todos los demás vehículos permanecen en la misma posición (ver figura 4.4.2).

[3] Es el tamaño de vecindad, el cual corresponde al número máximo de combinaciones o permutaciones que se generan a partir de un determinado *Evento* en cada iteración y se evalúa completamente.

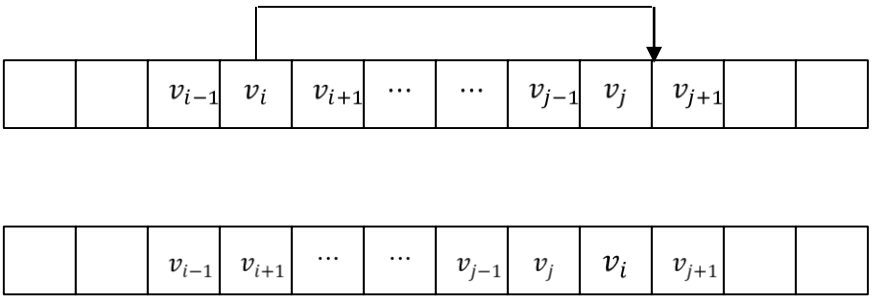


Figura 4.4 Movimiento de Tipo 1. Fuente: Elaboración Propia

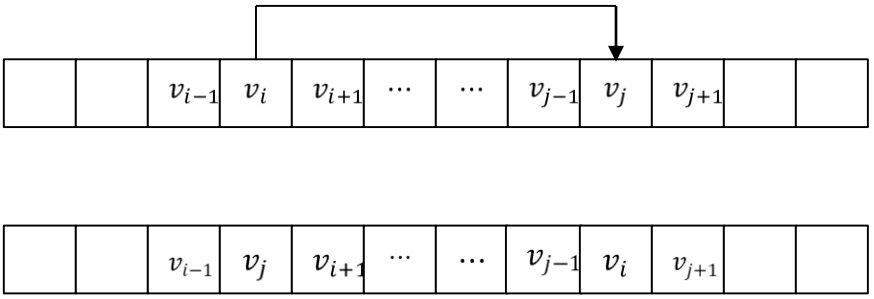


Figura 4.5 Movimiento de Tipo 2. Fuente: Elaboración Propia

4.5 Memoria de Corto Plazo

En el algoritmo de tipo *Búsqueda Tabú* se implementa una estructura de *Memoria a Corto Plazo* que trata de evitar regresar a configuraciones que han sido visitadas, y así no quedar atrapado en un *Óptimo Local*. La estructura almacena los últimos movimientos

realizados por el algoritmo en la *Lista Tabú* de tamaño experimental TL[4], la que se ajusta en la sección 5.3.

Por cada iteración se ingresa un nuevo movimiento a la *Lista Tabú* y a su vez se descuenta en una unidad la duración en la lista para cada movimiento, cuando éste llega a cero se elimina el movimiento de la *Lista Tabú*.

4.6 Criterio de Aspiración

El criterio aplicado es el *Criterio de Aspiración* basado en el *Atributo*, es decir si el movimiento está en la *Lista Tabú* pero genera una mejor solución que la encontrada hasta el momento, el cambio puede ser realizado de todas formas.

4.7 Intensificación

Una vez que se han hecho NIPI[5] iteraciones y la solución no mejora, se recupera la mejor solución obtenida hasta el momento y se intensifica la búsqueda en esa zona. La *Intensificación* guarda directa relación con la *Memoria a Largo Plazo*.

[4] Es la cantidad de iteraciones que se mantiene un movimiento en la *Lista Tabú* y que está dado por un valor aleatorio entre t_{min} *y* t_{max} *iteraciones, donde ambos son un porcentaje del número total de eventos del problema.*

[5] Es el Número de Iteraciones Previo a Intensificar.

4.8 Diversificación

El procedimiento de búsqueda utiliza un mecanismo de diversificación continua para expandir la exploración del espacio de búsqueda. Cuando un movimiento *tipo 1* o *tipo 2* dé como resultado un deterioro de la solución actual, el valor de $g(s)$ se incrementa artificialmente a $g(s) = g(s) + h(s) + p(s)$, donde $h(s)$ es el término de diversificación que penaliza movimientos realizados con frecuencia, y $p(s)$ fomenta la creación de lotes extensos de vehículos que requieren el mismo color, siempre satisfaciendo el límite de lote de pintura l. El resultado de esta adaptación es que si se alcanza un *Óptimo Local*, la búsqueda favorecerá movimientos que no han sido realizados y los cuales son propensos a cambiar las características de la solución.

Dado un movimiento del vehículo v_i a la posición j, el valor de $h(s)$ es equivalente a $\sqrt{n}$ multiplicada por el número de veces que un vehículo v_i ha sido movido desde su posición actual hacia otra posición.

El término $h(s)$ es utilizado para direccionar la búsqueda hacia las regiones menos exploradas del espacio de soluciones cuando se alcanza un *Óptimo Local*.

Por otra parte, el término $p(s)$ es introducido para fomentar la creación de grupos largos de vehículos del mismo color. Si v_i y v_j requieren el mismo color, pero v_i pertenece a un grupo más pequeño de vehículos consecutivos del mismo color que el grupo de v_j, entonces $p(s) = 0$. Lo mismo sucede si v_i y v_{j+1} tienen el mismo color, pero v_i pertenece a un grupo más pequeño de vehículos consecutivos del mismo color que el grupo de v_{j+1}. En ambos casos, al mover v_i desde su posición actual a la posición j conducirá a la creación de un grupo extenso de vehículos del mismo color. Para todos los demás casos, se tiene que $p(s) = \alpha \cdot \psi$, donde α es el coeficiente asociado al número de cambios de color en la función objetivo, y $\psi = 0{,}1$ por denominación del autor.

Cabe destacar que este término es considerado sólo cuando es teóricamente posible reducir el número de grupos de vehículos de un color dado. Este no siempre es el caso, ya que el número de grupos en la solución actual puede ser igual al número más bajo posible de grupos.

4.9 Criterio de Parada

El algoritmo de *Búsqueda Tabú* se detiene cuando ya no existe mejora de la *Función Objetivo* o cuando se alcanza un tiempo T[6], parámetro que puede ser ingresado por el usuario.

[6] Es el tiempo de ejecución del software.

4.10 Diagrama Resumen de Mejora de la Solución Inicial del Algoritmo

En la figura 4.6 se muestra un diagrama resumen del proceso de mejora de la *Solución Inicial* del algoritmo presentado en este capítulo:

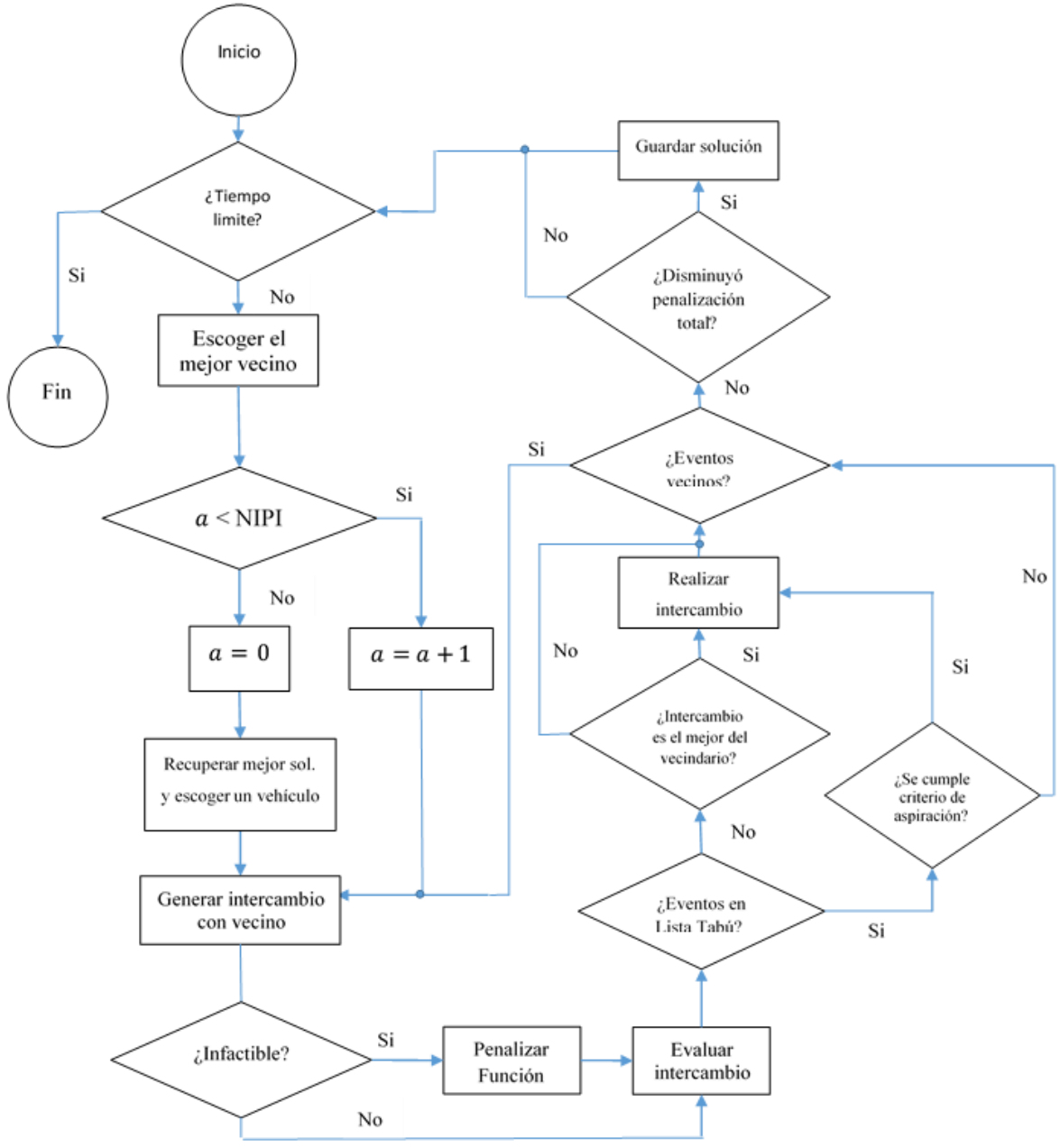


Figura 4.6 Diagrama de Flujo de mejora de solución inicial. Fuente: Elaboración Propia

5 EVALUACIÓN DEL ALGORITMO

En este capítulo se describen los casos de prueba, la interfaz utilizada, los recursos empleados para realizar las pruebas y la evaluación del desempeño del algoritmo propuesto en el capítulo anterior. Posteriormente se detallan e interpretan los resultados de las evaluaciones y se fijan los parámetros del algoritmo para que alcance un mayor rendimiento.

5.1 Interfaz y Recursos Empleados

El algoritmo de Tipo *Búsqueda Tabú* utiliza estructuras de datos para representar el problema, por lo que fue desarrollado en "*.net*", ya que este lenguaje tiende a ser dinámico, rápido y potente para manejar estas estructuras al poder crear nuestras propias funciones y métodos constructores; además los proyectos tienen la capacidad de ser ejecutados donde se estime conveniente. En cuanto a la interfaz, ésta fue diseñada en Visual Basic 6.0 y los resultados almacenados en una planilla Excel.

Las pruebas se realizaron en un computador con procesador AMD de 1.8 GHz, con 1 GB de memoria RAM y sistema operativo Windows XP.

5.2 Casos de Prueba

Para evaluar la calidad que posee el algoritmo desarrollado en este trabajo, calibrar los parámetros y realizar todas las evaluaciones pertinentes, se utilizaron las instancias *022_3_4_EP_RAF_ENP*, *022_3_4_RAF_EP_ENP* y *024_38_3_EP_ENP_RAF* contenidas en *Set A* (Primer caso de estudio en ROADEF 2005). En la tabla 5.1 se detallan las instancias en términos de: Objetivos de Optimización, Restricciones de Relación, Restricción en Lotes de Pintura y Cantidad de Vehículos a Procesar:

	Instancia		
	022_3_4_EP_RAF_ENP	***022_3_4_RAF_EP_ENP***	***024_38_3_EP_ENP_RAF***
Objetivos de Optimización	1: high priority level and easy to satisfy ratio constraints 2: paint color batches 3: low priority level ratio constraints	1: paint color batches 2: high priority level and easy to satisfy ratio constraints 3: low priority level ratio constraints	1: high priority level and easy to satisfy ratio constraints 2: low priority level ratio constraints 3: paint color batches
Límite subsecuencias (veh)	450	450	10
Ratios (p/q)	HPRC1: 5/6; Prioridad: 1 HPRC2: 1/3; Prioridad: 1 HPRC3: 1/4; Prioridad: 1 LPRC1: 10/15; Prioridad 0 LPRC2: 1/6; Prioridad 0 LPRC3: 1/10; Prioridad 0 LPRC4: 1/3; Prioridad 0 LPRC5: 1/6; Prioridad 0 LPRC6: 1/3; Prioridad 0	HPRC1: 5/6; Prioridad: 1 HPRC2: 1/3; Prioridad: 1 HPRC3: 1/4; Prioridad: 1 LPRC1: 10/15; Prioridad 0 LPRC2: 1/6; Prioridad 0 LPRC3: 1/10; Prioridad 0 LPRC4: 1/3; Prioridad 0 LPRC5: 1/6; Prioridad 0 LPRC6: 1/3; Prioridad 0	HPRC1: 2/3; Prioridad: 1 HPRC2: 1/15; Prioridad: 1 HPRC3: 1/4; Prioridad: 1 HPRC4: 1/6; Prioridad: 1 HPRC5: 1/5; Prioridad: 1 LPRC1: 1/10; Prioridad 0 LPRC2: 1/3; Prioridad 0 LPRC3: 1/6; Prioridad 0 LPRC4: 1/3; Prioridad 0 LPRC5: 1/6; Prioridad 0 LPRC6: 1/8; Prioridad 0 LPRC6: 1/3; Prioridad 0 LPRC6: 1/15; Prioridad 0
Tamaño de la secuencia (veh)	15 de día D-1 484 de día D	14 de día D-1 486 de día D	14 de día D-1 1261 de día D

Tabla 5.1 Casos de prueba y calibración. Fuente: Elaboración Propia

5.3 Procedimiento de Evaluación

Existe una etapa experimental que permite ajustar los parámetros del algoritmo para que alcance un mayor rendimiento, para lo cual se fijan valores que parezcan razonables, y luego se realizan pruebas sistemáticas variando uno solo de ellos por vez. Los valores a los cuales se van a fijar los parámetros para realizar las pruebas se han obtenido producto del ensayo durante la etapa de programación y también de la Revisión Bibliográfica revisada durante el Anteproyecto de Título**.**

Los valores de los parámetros que se van a fijar y posteriormente a calibrar se presentan en la tabla 5.2:

Parámetro	**Valor**
T	60 segundos
TV	0,1 x E
NIPI	1000
TL	aleatorio entre Tmin y Tmax, con valores 0,01 * E y 0,03 * E respectivamente

Tabla 5.2 Valores de los parámetros. Fuente: Elaboración Propia

Una vez calibrados los parámetros, se compara la calidad de las soluciones entregadas por el algoritmo mediante el emparejamiento de los valores de la *Función Objetivo* conseguidas por éste con las obtenidas por la *Solución Inicial*.

5.4 Calibración de Parámetros

Para ajustar cada uno de los parámetros del algoritmo se realizan 10 pruebas por cada valor distinto que toma cada uno de ellos y luego se calculan sus promedios. En la tabla 5.3 se muestran los resultados de los promedios de la F.O.[7] para cada uno de los parámetros. Notar que algunas entradas se repiten para facilitar la lectura de variaciones de parámetros con sus correspondientes resultados. En el anexo 1.1 se detallan los resultados de las pruebas.

Tmin - Tmax	NIPI	TV	T	Tipo I	Tipo II	Tipo III
0,01*E - 0,03*E	1000	0,1*E	60	11.371.430	5.169.541	57.696
0,1*E - 0,3*E	1000	0,1*E	60	11.626.967	6.105.250	67.961
0,05*E - 0,2*E	1000	0,1*E	60	11.666.335	5.567.980	58.271
0,01*E - 0,03*E	**1000**	0,1*E	60	11.371.430	5.169.541	57.696
0,01*E - 0,03*E	**2000**	0,1*E	60	11.611.427	6.302.999	61.281
0,01*E - 0,03*E	1000	**0,05*E**	60	11.516.125	5.682.535	58.535
0,01*E - 0,03*E	1000	**0,1*E**	60	11.371.430	5.169.541	57.696
0,01*E - 0,03*E	1000	**0,15*E**	60	11.680.848	5.531.962	59.615

Tabla 5.3 Promedios de F.O. para cada parámetro. Fuente: Elaboración Propia

.

[7] Función Objetivo del algoritmo que es utilizada para evaluar la calidad de una solución y poder guiar la búsqueda en el proceso de mejora de la solución inicial.

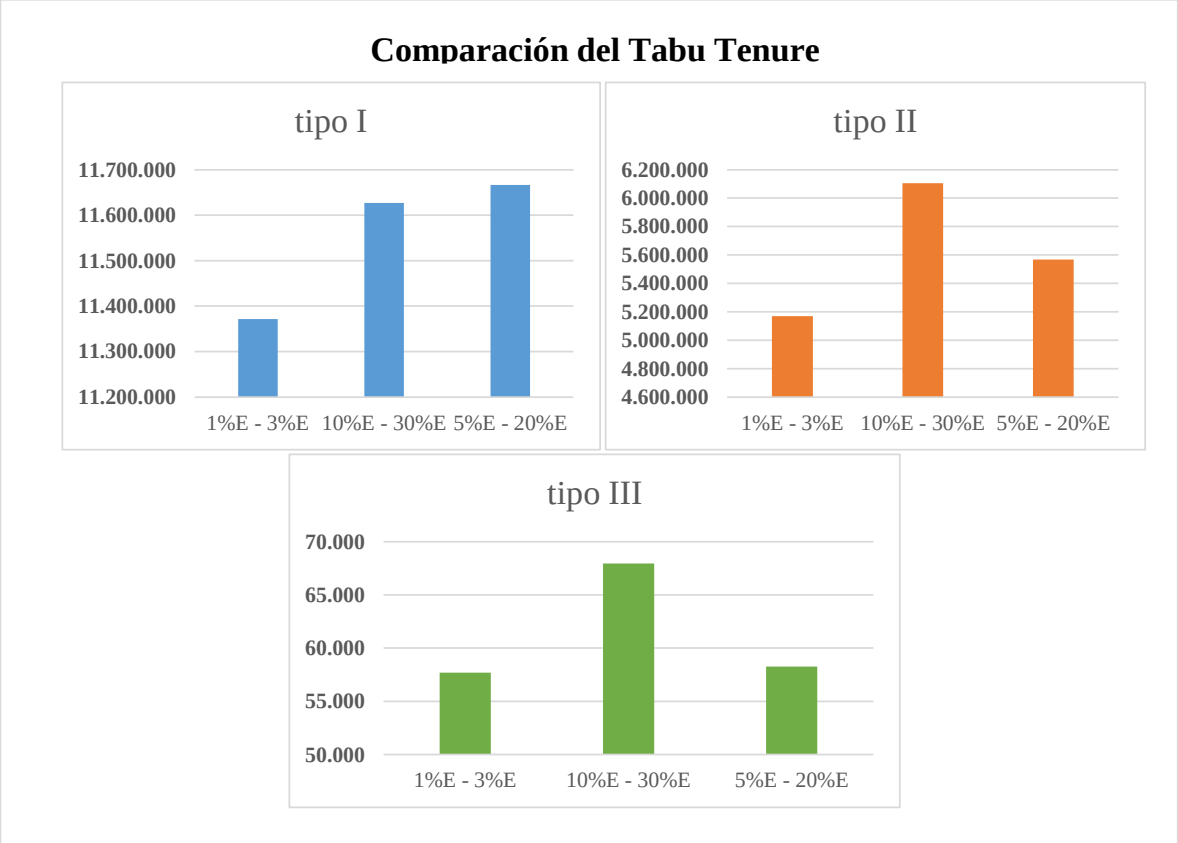


Figura 5.1 Comparación del Tabú Tenure. Fuente: Elaboración Propia

Al comparar los resultados del Tabú Tenure en los tres casos de prueba, se puede observar que la Función Objetivo presenta una mejora cuando este valor está entre un 1% y 3% del número total de eventos.

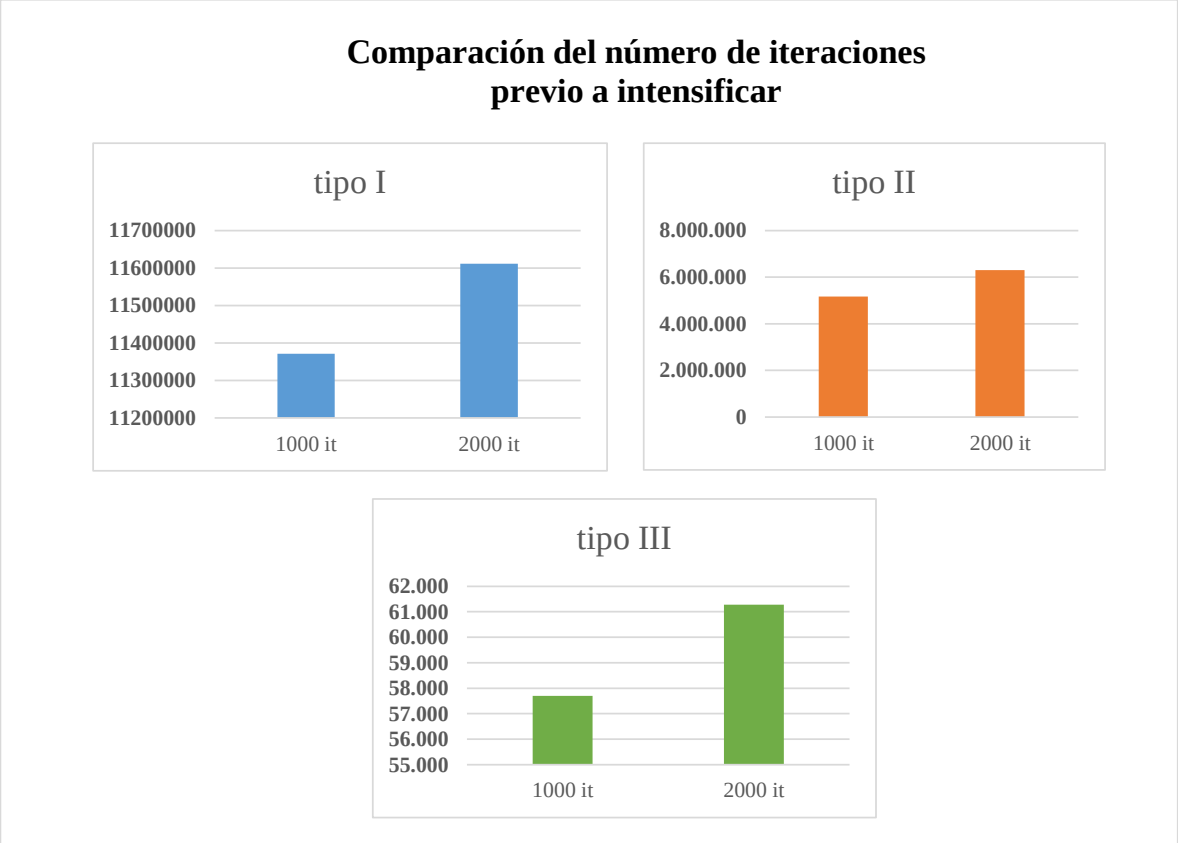


Figura 5.2 Comparación del número de iteraciones previo a intensificar. Fuente: Elaboración Propia

Para los tres casos se puede observar que la Función Objetivo mejora al reducir el número del NIPI de 2000 a 1000, viéndose mejor reflejado en el tercer caso.

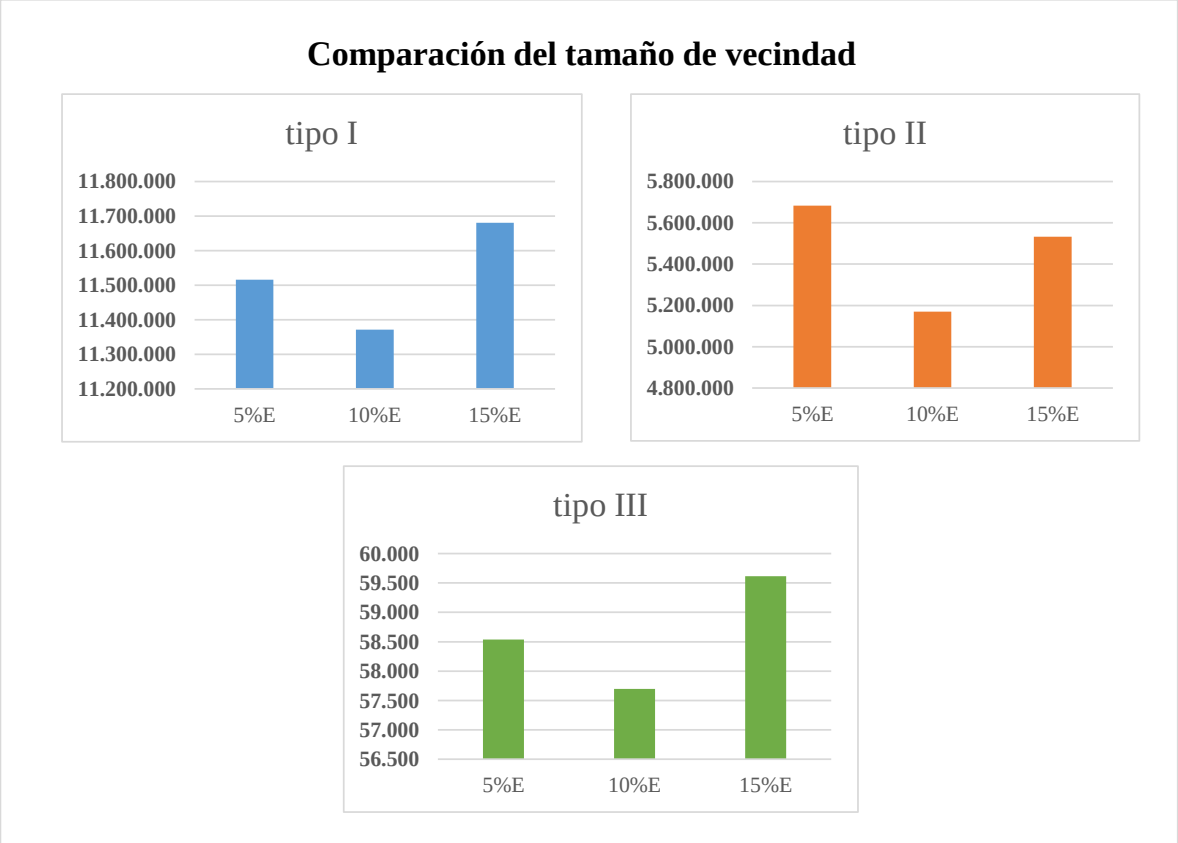


Figura 5.3 Comparación del tamaño de vecindad. Fuente: Elaboración Propia

Al comparar los resultados de la Función Objetivo se observan variaciones en los tres casos, y la disminución se encuentra al considerar un tamaño de vecindad del 10% del total de eventos.

Una vez calibrado el TV, NIPI y TL se hacen pruebas usando estos parámetros ajustados, para determinar el tiempo de ejecución óptimo para el correcto funcionamiento del algoritmo y así obtener soluciones de buena calidad, para ello se corre 10 veces el programa durante 30, 60 y 90 segundos y luego se calculan sus promedios. En la tabla XX se muestran los resultados de los promedios de la *Función Objetivo* para los tiempos de ejecución anteriormente mencionados, mientras que los resultados de dichas pruebas se encuentran detallados en el anexo 1.2.

Tmin - Tmax	NIPI	TV	T	Tipo I	Tipo II	Tipo III
0,01*E - 0,03*E	1000	0,1*E	**30**	11.454.525	5.322.686	52.726
0,01*E - 0,03*E	1000	0,1*E	**60**	11.371.430	5.169.541	57.696
0,01*E - 0,03*E	1000	0,1*E	**90**	11.316.939	5.076.763	50.882

Tabla 5.4 Resultados de los promedios de la F.O. según tiempos de ejecución. Fuente: Elaboración Propia

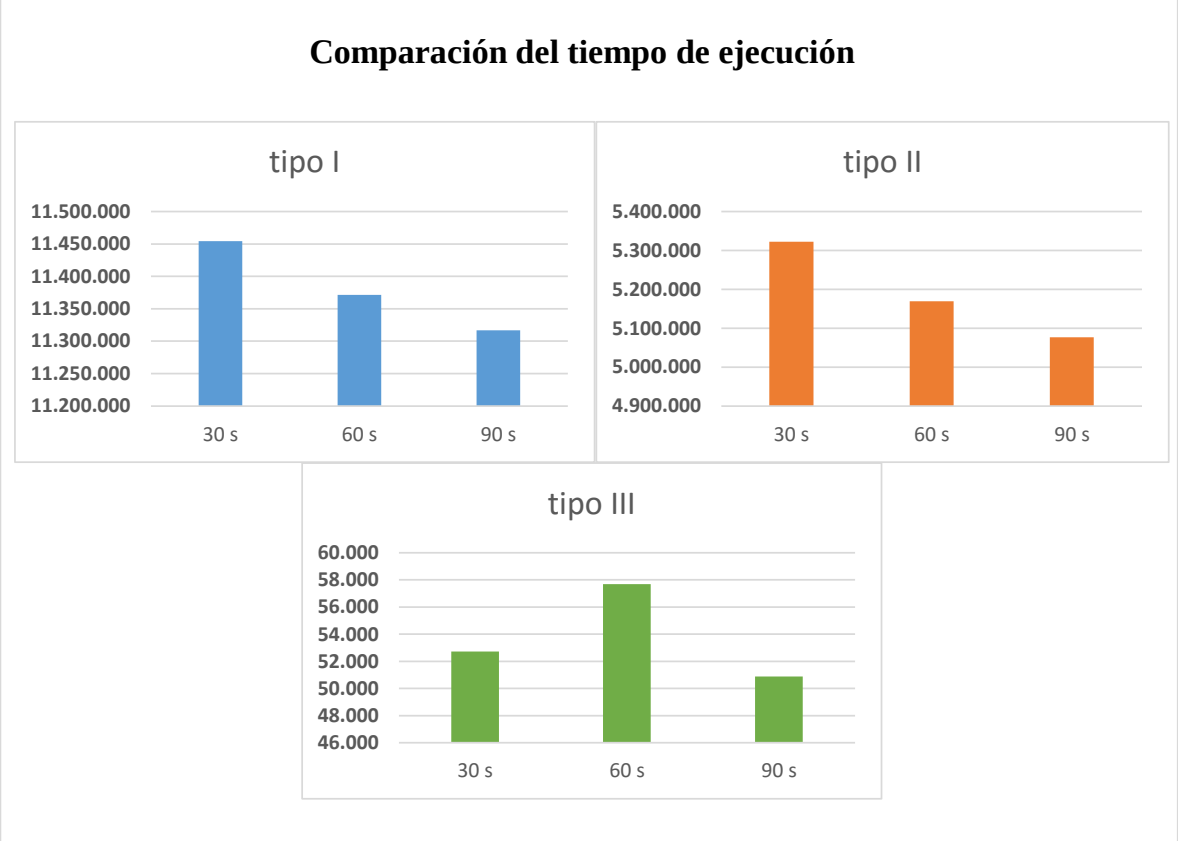


Figura 5.4 Comparación de los tiempos de ejecución. Fuente: Elaboración Propia

Al comparar los resultados de la *Función Objetivo* para los distintos tiempos de ejecución, se puede observar que para el primer y segundo caso existe una tendencia a disminuir el valor de la F.O. a medida que el tiempo de ejecución aumente. Para el tercer caso no se puede asegurar lo mismo ya que la diferencia es mínima para 30 y 90 segundos de ejecución, pues no se aprecia la misma tendencia que los casos anteriores. Basándose en el valor mínimo que ha sido obtenido para los tres casos se usará un tiempo de ejecución de

90 segundos, donde el software podrá explorar de mejor manera el espacio de soluciones y esto no afectará el funcionamiento del algoritmo.

Una vez calibrado el tiempo de ejecución se procederá a correr el algoritmo para cada una de las instancias. En particular se calculará el promedio de F.O.f.[8] para 10 corridas de las instancias del *Set A* y 5 corridas de las instancias del *Set X*, para finalmente hacer un contraste de los mejores resultados obtenidos durante el transcurso del *Desafío ROADEF 2005*.

5.5 Cálculo del puntaje final (Caso ROADEF 2005)

Para fines comparativos se implementa una puntuación especial para cada instancia y así tener un control sobre la eficacia del algoritmo que se ha utilizado.

Para una instancia *X* será necesario obtener los siguientes parámetros:

- *Peor*: Es el valor más alto que tomará la *Función Objetivo* durante el transcurso del experimento.
- *Mejor*: Es el valor mínimo de la *Función Objetivo* registrado durante el experimento.
- *Resultado*: Similar a F.O.f., es el resultado que se obtiene al finalizar el algoritmo.

La *puntuación* se calcula dividiendo la *desviación* del *rango,* la *desviación* es la diferencia entre el valor que entrega el algoritmo y la peor solución encontrada durante el transcurso del experimento, mientras que el rango es la diferencia entre el mejor y el peor valor que se registra en un experimento:

[8] Valor final de la función objetivo obtenida por el algoritmo.

$$Puntaje = \frac{(Peor - Resultado)}{(Peor - Mejor)}$$

El puntaje se moverá en un rango de $[0,1]$, y significa que mientras se tenga un puntaje más cercano a 1 significará que el algoritmo habrá sido capaz de captar una solución eficiente y quedarse con ella.

Para cada instancia se contrastarán los valores que se obtuvieron mediante el algoritmo con los valores del trabajo realizado por Cordeau et. al. en [4], cuya Metaheurística de *Iterated Tabu Search* quedó posicionada en el tercer lugar del ranking *ROADEF 2005*.

5.5.1 Instancias Set A

En la tabla 5.5 se da a conocer los resultados que obtuvieron Cordeau et. al. para las partidas del algoritmo en el *Set A*, consiguiendo un puntaje de 15,4036 de 16 posibles:

Instance	Worst	Best	Result	Score
022-3-4-EP-RAF-ENP	112.001	31.001	31.001	1
022-3-4-RAF-EP-ENP	11.048.003	11.039.001	11.039.002	0,9999
024-38-3-EP-ENP-RAF	71.118.491	4.000.302	4.017.352	0,9997
024-38-3-EP-RAF-ENP	63.376.290	4.249.083	4.293.209	0,9993
024-38-5-EP-ENP-RAF	82.165.438	4.034.309	4.070.379	0,9995
024-38-5-EP-RAF-ENP	75.477.143	4.280.079	4.322.099	0,9994
025-38-1-EP-ENP-RAF	1.704.363	9.972	156.774	0,9644
025-38-1-EP-RAF-ENP	320.753	231.134	234.603	0,9613
039-38-4-EP-RAF-ch1	84.248.000	13.129.000	13.149.000	0,9997
039-38-4-RAF-EP-ch1	68.344.000	68.155.000	68.253.000	0,4815
048-39-1-EP-ENP-RAF	27.135.374	6.129	78.308	0,9994
048-39-1-EP-RAF-ENP	27.202.853	174.612	188.737	0,9995
064-38-2-EP-RAF-ENP-ch1	240.741	112.759	112.759	1
064-38-2-EP-RAF-ENP-ch2	54.014	34.051	34.051	1
064-38-2-RAF-EP-ENP-ch1	64.499.822	63.423.782	63.423.782	1
064-38-2-RAF-EP-ENP-ch2	27.376.087	27.367.052	27.367.052	1
		Total		15,4036

Tabla 5.5 Puntaje de Set A – Iterated Tabu Search. Fuente: Cordeau (2008)

Cabe destacar que el resultado obtenido en el *Set A* en [4] tiende a acercarse a conseguir el puntaje ideal, siendo en la instancia *039-38-4-RAF-EP-ch1* donde el algoritmo pudo haber quedado atrapado en un óptimo local y por tanto disminuyó el puntaje total de la instancia. Todas las demás instancias obtuvieron un puntaje sobre 0,96 y pudieron alcanzar un resultado cercano a la mejor solución obtenida durante el transcurso de la búsqueda.

A continuación se exponen los resultados obtenidos por el algoritmo de *Búsqueda Tabú* para las instancias del *Set A*, para luego poder ser analizados con los resultados mostrados en la tabla anterior:

Instancia	Peor	Mejor	Resultado	Puntaje
022-3-4-EP-RAF-ENP	3.939.022	31.218	53.174	0,9944
022-3-4-RAF-EP-ENP	12.024.807	11.042.486	11.624.106	0,4079
024-38-3-EP-ENP-RAF	81.977.840	4.027.125	5.849.772	0,9766
024-38-3-EP-RAF-ENP	79.352.228	4.336.521	5.890.245	0,9793
024-38-5-EP-ENP-RAF	107.154.723	4.139.736	4.866.384	0,9929
024-38-5-EP-RAF-ENP	101.476.572	4.497.232	5.619.638	0,9884
025-38-1-EP-ENP-RAF	2.657.604	10.144	12.071	0,9993
025-38-1-EP-RAF-ENP	626.607	243.532	275.156	0,9174
039-38-4-EP-RAF-ch1	153.150.000	13.542.000	15.844.900	0,9835
039-38-4-RAF-EP-ch1	87.773.000	68.405.000	70.989.700	0,8665
048-39-1-EP-ENP-RAF	72.240.907	7.104	9.296	0,99997
048-39-1-EP-RAF-ENP	69.914.182	180.623	344.683	0,9976
064-38-2-EP-RAF-ENP-ch1	3.515.527	109.264	110.119	0,9997
064-38-2-EP-RAF-ENP-ch2	30.555.745	39.965	1.511.995	0,9518
064-38-2-RAF-EP-ENP-ch1	70.724.345	64.349.523	66.857.236	0,6066
064-38-2-RAF-EP-ENP-ch2	44.216.083	30.860.492	31.679.963	0,9386
		Total		14,6007

Tabla 5.6 Puntaje de Set A – Algoritmo de Búsqueda Tabú. Fuente: Elaboración Propia

De los resultados expuestos en la tabla 5.6 se puede destacar que la búsqueda pudo mantener la cercanía entre el mejor resultado explorado y el resultado final del algoritmo en 6 instancias (puntaje sobre 99%); sin embargo en la instancia *022-3-4-RAF-EP-ENP* se obtuvo el puntaje más bajo (40,79%) debido a la diferencia entre el mejor valor explorado y el resultado final del algoritmo, que se debe principalmente al aumento del número de violaciones de restricción del tipo HPRC (aproximadamente 620 violaciones).

Lo anterior implica que la puntuación del algoritmo esté 0,80 puntos por debajo de los resultados obtenidos por Cordeau et. al. en [4], pero sin quitarle capacidad al software propuesto en este proyecto.

5.5.2 Instancias Set X

En la tabla 5.7 se exponen los resultados de las instancias del *Set X* obtenidos por la Metaheurística propuesta por Cordeau et. al. en [4], la que fue usada en ROADEF 2005 para conformar el ranking final de los participantes de la competencia. El puntaje alcanzado fue 18,8112 de 19 puntos posibles:

Instance	Worst	Best	Result	Score
022-RAF-EP-ENP-S49-J2	12.202.004	12.002.003	12.002.003	1
023-EP-RAF-ENP-S49-J2	314.044	192.466	206.073	0,8881
024-EP-RAF-ENP-S49-J2	49.844.834	337.006	400.415	0,9987
025-EP-ENP-RAF-S49-J1	16.950.509	160.408	160.828	1
028-CH1-EP-ENP-RAF-S50-J	338.608.118	36.341.495	36.386.296	0,9999
028-CH2-EP-ENP-RAF-S51-J	9.403	3	3	1
029-EP-RAF-ENP-S49-J5	181.024	110.298	111.242	0,9867
034-VP-EP-RAF-ENP-S51-J1	14.553	55.995	59.060	0,9658
034-VU-EP-RAF-ENP-S51-J1	12.103.052	8.087.036	8.095.256	0,998
035-CH1-RAF-EP-S50-J4	6.010.000	5.010.000	5.010.000	1
035-CH2-RAF-EP-S50-J4	7.056.000	6.056.000	6.056.000	1
039-CH1-EP-RAF-ENP-S49-J	88.237	69.239	69.311	0,9962
039-CH3-EP-RAF-ENP-S49-J	285.101	231.030	231.671	0,9882
048-CH1-EP-RAF-ENP-S50-J	2.410.707	197.006	197.825	0,9996
048-CH2-EP-RAF-ENP-S49-J	34.951.538	31.077.916	31.096.860	0,9951
064-CH1-EP-RAF-ENP-S49-J	67.263.120	61.187.230	61.218.289	0,9949
064-CH2-EP-RAF-ENP-S49-J	72.400	37.000	37.000	1
655-CH1-EP-RAF-ENP-S51-J	53.000	30.000	30.000	1
655-CH2-EP-RAF-ENP-S52-J	306.032.000	153.034.000	153.034.000	1
		Total		18,8112

Tabla 5.7 Puntaje de Set X – Iterated Tabu Search. Fuente: Cordeau (2008)

En este caso los resultados finales fueron logrados haciendo 5 repeticiones por cada instancia, y el resultado de cada instancia es el promedio de 5 soluciones entregadas por la Metaheurística, y en este caso el puntaje obtenido fue muy cercano al puntaje ideal del set de instancias, pudiendo recoger soluciones aceptables y cercanas al mejor valor captado durante los experimentos. Cabe mencionar que esta mejora se obtuvo ya que los participantes pudieron utilizar las instancias del *Set A* y *Set B* para calibrar cada uno de los parámetros utilizados en sus programaciones.

En la tabla 5.8 se exponen los resultados que se obtuvieron con el algoritmo de *Búsqueda Tabú* para las instancias del *Set X*:

Instancia	Peor	Mejor	Resultado	Puntaje
022-RAF-EP-ENP-S49-J2	14.926.905	13.192.851	13.277.597	0,9511
023-EP-RAF-ENP-S49-J2	266.420.772	327.869	1.601.476	0,9952
024-EP-RAF-ENP-S49-J2	75.118.207	457.858	1.918.055	0,9804
025-EP-ENP-RAF-S49-J1	755.251.008	327.799	2.374.882	0,9973
028-CH1-EP-ENP-RAF-S50-J	136.215.384	36.613.235	38.277.802	0,9833
028-CH2-EP-ENP-RAF-S51-J	3.045.062	139	769	0,9998
029-EP-RAF-ENP-S49-J5	508.736	116.958	135.884	0,9517
034-VP-EP-RAF-ENP-S51-J1	176.622	59.252	63.961	0,9599
034-VU-EP-RAF-ENP-S51-J1	12.910.358	8.114.622	8.963.057	0,8231
035-CH1-RAF-EP-S50-J4	6.680.000	5.147.000	6.424.000	0,1670
035-CH2-RAF-EP-S50-J4	7.985.000	6.167.000	6.519.000	0,8064
039-CH1-EP-RAF-ENP-S49-J	95.074	69.664	70.816	0,9547
039-CH3-EP-RAF-ENP-S49-J	323.585	232.828	247.148	0,8422
048-CH1-EP-RAF-ENP-S50-J	4.620.735	197.422	200.583	0,9993
048-CH2-EP-RAF-ENP-S49-J	37.957.564	31.581.154	33.819.856	0,6489
064-CH1-EP-RAF-ENP-S49-J	68.197.767	62.071.208	63.714.166	0,7318
064-CH2-EP-RAF-ENP-S49-J	80.000	38.000	40.000	0,9524
655-CH1-EP-RAF-ENP-S51-J	56.000	31.000	33.000	0,9200
655-CH2-EP-RAF-ENP-S52-J	318.070.000	154.149.000	169.436.800	0,9067
		Total		16,5712

Tabla 5.8 Puntaje de Set A – Algoritmo de Búsqueda Tabú. Fuente: Elaboración Propia

De acuerdo a los resultados mostrados en la tabla 5.8 es posible mencionar que el rendimiento del algoritmo propuesto se puede considerar aceptable, ya que de las 19 instancias puestas a disposición para ser analizadas, existen 4 instancias con puntaje sobre 0,99 así como 7 instancias que tienen un puntaje de no más de 0,9. La instancia *035-CH1-RAF-EP-S50-J4* y su puntaje obtenido (rendimiento del 16,7%) tiene directa influencia del distanciamiento entre el puntaje obtenido por Cordeau et. al. en [4] y el algoritmo que ha sido propuesto.

5.5.3 Ranking Final del Algoritmo

Una vez analizado las puntuaciones obtenidas en las instancias del *Set A Set A* y *Set X* se puede considerar la valoración del algoritmo frente a los participantes de ROADEF 2005, considerando para nuestro caso el Puntaje Final obtenido en las instancias del *Set X*. La información mencionada se encuentra en la tabla 5.9:

Equipo	Ranking Final	Puntaje
Nouioua et al	1	18,9935
Ribeiro et al	2	18,8896
Naddef et al	3	17,9173
Bloemen	4	17,9465
Kuipers	5	17,8340
Gavranovic	6	16,8067
Cordeau et al	7	18,6665
Stützle et al	8	16,9372
Gendreau et al	9	18,1654
Pawlak et al	10	16,3762
Gravel et al	11	17,6204
Benoist et al	12	16,4190
Jaszkiewicz et al	13	16,8937
Montemanni	14	12,6513
Bonizzoni et al	15	11,8235
Caseau	16	13,8650
Zufferey et al	17	13,7882
Klau et al	18	8,7213
Cordeau et al	----	18,8112
Tabu Search	----	**16,5712**

Tabla 5.9 Comparación ROADEF 2005 – Tabú Search. Fuente: Cordeau (2008)

Las posiciones de cada equipo fueron ordenadas de acuerdo a la cantidad de soluciones infactibles que pudieron haber obtenido en las instancias, por lo que si se considera sólo el Puntaje Final obtenido, el algoritmo de *Búsqueda Tabú* quedará posicionado de la siguiente manera:

Equipo	**Puntaje**
Nouioua et al	18,9935
Ribeiro et al	18,8896
Cordeau et al	18,8112
Cordeau et al	18,6665
Gendreau et al	18,1654
Bloemen	17,9465
Naddef et al	17,9173
Kuipers	17,8340
Gravel et al	17,6204
Stützle et al	16,9372
Jaszkiewicz et al	16,8937
Gavranovic	16,8067
Tabu Search	**16,5712**
Benoist et al	16,4190
Pawlak et al	16,3762
Caseau	13,8650
Zufferey et al	13,7882
Montemanni	12,6513
Bonizzoni et al	11,8235
Klau et al	8,7213

Tabla 5.10 Posición de Tabú Search en Ranking ROADEF 2005. Fuente: Elaboración Propia

De acuerdo a lo mostrado en la tabla 5.9 se puede inferir que la puntuación que obtuvo el algoritmo *Tabú Search* se acercó al valor promedio de participantes (incluida la Metahurística de [4]) en 0,2863[9] unidades, esto utilizando un tiempo de 90 segundos para ejecutar la búsqueda de solución.

[9] Puntaje promedio: 16,2849 por lo tanto; 16,2849 – 16,5712 = 0,2863

Respecto a la eficiencia de este algoritmo es pertinente señalar que la búsqueda puede perfeccionarse mediante la combinación de los parámetros T, TV, NIPI y TL y así encontrar un equilibrio para direccionar el programa al mejor espacio de soluciones.

CONCLUSIONES Y RECOMENDACIONES

En este trabajo se ha presentado un *Problema de Secuenciamiento de Vehículos* propuesto por RENAULT, considerado como un problema *NP – Hard* de difícil programación al contener gran cantidad de variables, datos y restricciones. En definitiva, para abordar el problema, el *CSP* implicó el estudio de los métodos de resolución para ser caracterizado y modelado; luego se direccionó la investigación hacia las metaheurísticas, proponiendo un método de resolución mediante un algoritmo del tipo *Búsqueda Tabú*, el cual fue capaz de encontrar soluciones que minimizaran la Función de Costo en las instancias del *Set A*. Este lote de instancias fue utilizado en ROADEF 2005 para calibrar los métodos utilizados por los participantes, y en particular para este trabajo fueron utilizadas para la calibración de los parámetros a utilizar durante la búsqueda. Al obtener soluciones en tiempos razonables se pudo comprobar que el programa propuesto direcciona su búsqueda hacia regiones aceptables para resolver el *CSP* propuesto por RENAULT. Las instancias del *Set X* tuvieron un desempeño aceptable al situarse con una puntuación cercana al promedio de los participantes del Desafío ROADEF 2005, confirmando que la *Búsqueda Tabú* tiene un buen desempeño en problemas de compleja resolución.

Si bien el algoritmo desarrollado es para el problema que propuso RENAULT, es importante destacar que puede utilizarse bases de datos de otro origen para utilizar este programa, siempre que las instancias cumplan con las estructuras utilizadas en este trabajo.

Finalmente, en trabajos futuros se pueden agregar al software nuevos métodos de profundización del espacio de soluciones, como nuevos métodos de perturbación de la secuencia inicial y subsecuencias de vehículos con requerimientos similares, en particular:

1) Estudiar otros tipos de métodos de generación de vecindad que permitan acercarse a otros óptimos y escapar de soluciones que puedan estar penalizadas.

2) Realizar análisis de sensibilidad a los nuevos parámetros, a fin de seleccionar el mejor valor para así explorar regiones nuevas y otros candidatos a ser buenas soluciones.

3) En la puntuación para posicionar cada trabajo en el ranking referencial del Desafío ROADEF 2005, considerar la cantidad de soluciones infactibles, ya que también es un punto importante a la hora de filtrar los candidatos a mejor programación.

REFERENCIAS

[1] Bergen, M.E.; van Beek, P.; Carchrae, T. (2001). Constraint-based vehicle assembly line sequencing. *Proceedings of the 14th Canadian Conference on Artificial Intelligence, Vol. 27(3)*, 88-99.

[2] Briant, O.; Naddef, D.; Mounié, G. (2008). Greedy approach and multi-criteria simulated annealing for the car sequencing problem. *European Journal of Operational Research, 191*, 993-1003.

[3] Cheng, J.; Lu, Y.; Puskorius, S.; Bergeon, S.; Xiao, J. (1999). Vehicle sequencing based on evolutionary computation. *Evolutionary Computation (CEC 99), Vol. 2*, 1207-1214. IEEE.

[4] Cordeau, J.-F.; Laporte, G.; Pasin, F. (2008). Iterated tabu search for the car sequencing problem. *European Journal of Operational Research, 191*, 945-956.

[5] Davenport, A.; Tsang, E.; Wang, C.; Zhu, K. (1994). Genet: A connectionist architecture for solving constraint satisfaction problems by iterative improvement. *Proceedings of AAAI'94*, 325-330.

[6] Davenport, A.J.; Tsang, E.P.K. (1999). Solving constraint satisfaction sequencing problems by iterative repair. *Proceedings of the First International Conference on the Practical Applications of Constraint Technologies and Logic Programming (PACLP)*, 345-357.

[7] de Oliveira, R.J. (2007). *Solving the Car Sequencing Problem from a Multiobjective Perspective.* Instituto Superior Técnico - Universidade Técnica de Lisboa.

[8] Dorigo, M.; Stützle, T. (2005). Ant Colony Optimization. *MIT Press*.

[9] Drexl, A.; Kimms, A. (2001). Sequencing jit mixed-model assembly lines under station load and part-usage constraints. *Management Science, 47(3)*, 480-491.

[10] Drexl, A.; Kimms, A.; Matthiessen, L. (2006). Algorithms for the car sequencing and the level scheduling problem. *Journal of Scheduling, 9(2)*.

[11] Gent, I., P. (1998). Two results on car-sequencing problems. *Tecnical report*. Obtenido de http://www.apes.cs.strath.ac.uk/apesreports.html

[12] Gent, I.P; Walsh, T. (1999). Csplib: A benchmark library for constraints. *Technical report, APES-09*. Obtenido de http://www.csplib.org

[13] Gottlieb, J.; Puchta, M.; Solnon, C. (2003). A study of greedy, local search and ant colony optimization approaches for car sequencing problems. *Applications of evolutionary computing, Vol. 2611 of LNCS*, 246-257.

[14] Gravel, M.; Gagné, C.; Price, W.L. (2005). Review and comparison of three methods for the solution of the car-sequencing problem. *Journal of the Operational Research Society, 56(11)*, 1287-1295.

[15] Hindi, K.S.; Ploszajski, M.G. (1994). Formulation and solution of a selection and sequencing problem in car manufacture. *Computers and Industrial Engineering, 26(1)*, 203-211.

[16] Kis, T. (2004). On the complexity of the car sequencing problem. *Operations Research Letters*, 32: 331-335.

[17] Lee, J.H.M.; Leung, H.F.; Won, H.W. (1998). Performance of a comprehensive and efficient constraint library using local search. *11th Australian JCAI, LNAI*.

[18] Michel, L.; Van Hentenryck, P. (2002). A constraint-based architecture for local search. *OOPSLA'02: Proceedings of the 17th ACM SIGPLAN Conference on Oriented Programming, Systems, Languajes, and Applications*, págs. 83-100. ACM Press.

[19] Neveu, B.; Trombettoni, G.; Glover, F. (2004). Id walk: A candidate list strategy with a simple diversification device. *Proceedings of CP'2004, Vol. 2358 of LNCS*, 423-437.

[20] Nguyen, A. (2005). Challenge ROADEF'2005 Car Sequencing Problem. Obtenido de Renault - alain.nguyen@renault.com

[21] Parello, B., D.; Kabat, W., C.; Wos, L. (1986). Job-shop scheduling using automated reasoning: A case study of the car sequencing problem. *Journal of Automated Reasoning*, 2: 1-42.

[22] Perron, L.; Shaw, P.; Furnon, V. (2004). Propagation guided large neighborhood search. *Proceedings of Principles and Practice of Constraint Programming (CP'2004), Vol. 3258 of LNCS*, 468-481.

[23] Perron, L; Shaw P. (2004). Combining forces to solve the car sequencing problem. *Proceedings of CP-AI-OR'2004, vol. 3011 of LNCS*, 225-239.

[24] Ploszajski, K. S., & Hindi, M. G. (1994). Formulation and solution of a selection and sequencing problem in car manufacture. *Computers and Industrial Engineering, 26(1)*, 203-211.

[25] Puchta, M.; Gottlieb, J. (2002). Solving car sequencing problems by local optimization. *Evo Workshops, Vol. 2056 of LNCS*, 132-142.

[26] Regin, J.-C.; Puget, J.-F. (1997). A filtering algorithm for global sequencing constraints. *In CP97, Vol. 1330 of LNCS*, 32-46.

[27] Smith, B. (1996). Suceed-first or fail-first: A case study in variable and value ordering heuristics. *Third Conference on the Practical Applications of Constraint Technology PACT'97*, 321-330.

[28] Solnon, C. (2000). Solving permutation constraint satisfaction problems with artificial ants. *Proceedings of ECAI'2000, IOS Press, Amsterdam, The Netherlands*, 118-122.

[29] Tsang, E.P.K. (1993). Foundations of Constraint Satisfaction. *Academic Press, London, UK*.

[30] Van Hententyck, P.; Saraswat, V.; Deville, Y. (1991). The cardinality operator: A new logical connective and its application to constraint logic programming. *In ICLP-91*.

[31] van Hoeve, W.-J.; Pesant, G.; Rosseau, L.-M.; Sabharwal, A. (2006). Revisiting the sequence constraint. *12th Internacional Conference on Principles and Practice of Constraint Programming (CP'2006)*, 620-634.

[32] Warwick, T.; Tsang, E. (1995). Tackling car sequencing problems using a genetic algorithm. *Journal of Evolutionary Computation - MIT Press, 3(3)*, 267-298.

[33] Zufferey, N.; Studer, M.; Silver, E.A. (2006). Tabu search for a car sequencing problem. *American Association for Artificial Intelligence*. Obtenido de http://www.aaai.org

ANEXOS

Anexo 1. Pruebas y Resultados

En este anexo se presentan los resultados de las pruebas hechas en el proceso de calibración de parámetros y los resultados finales obtenidos por el algoritmo con los parámetros ajustados.

Anexo 1.1. Resultados de la Calibración de Parámetros

Aquí se muestra la tabla de pruebas necesarias para calibrar cada uno de los parámetros del algoritmo.

Tmin - Tmax	NIPI	TV	T	Tipo I	Tipo II	Tipo III
0,01*E - 0,03*E	1000	0,1*E	60	11.454.807	4.115.888	81.989
0,01*E - 0,03*E	1000	0,1*E	60	11.635.417	5.943.305	56.270
0,01*E - 0,03*E	1000	0,1*E	60	11.109.824	6.972.509	69.376
0,01*E - 0,03*E	1000	0,1*E	60	11.621.064	5.072.893	74.020
0,01*E - 0,03*E	1000	0,1*E	60	11.056.704	4.297.738	47.947
0,01*E - 0,03*E	1000	0,1*E	60	11.471.224	5.236.060	53.689
0,01*E - 0,03*E	1000	0,1*E	60	11.657.874	4.119.802	36.719
0,01*E - 0,03*E	1000	0,1*E	60	11.152.694	5.512.395	52.851
0,01*E - 0,03*E	1000	0,1*E	60	11.057.682	4.717.696	42.476
0,01*E - 0,03*E	1000	0,1*E	60	11.497.014	5.707.123	61.623
Promedio				**11.371.430**	**5.169.541**	**57.696**

Tmin - Tmax	NIPI	TV	T	Tipo I	Tipo II	Tipo III
0,1*E - 0,3*E	1000	0,1*E	60	11.154.198	7.398.976	74.767
0,1*E - 0,3*E	1000	0,1*E	60	11.905.926	6.281.214	65.403
0,1*E - 0,3*E	1000	0,1*E	60	11.877.642	7.143.584	65.733
0,1*E - 0,3*E	1000	0,1*E	60	11.710.743	4.579.969	70.722
0,1*E - 0,3*E	1000	0,1*E	60	11.789.224	5.900.279	37.175
0,1*E - 0,3*E	1000	0,1*E	60	12.005.510	5.212.335	82.226
0,1*E - 0,3*E	1000	0,1*E	60	11.216.054	7.433.094	58.743
0,1*E - 0,3*E	1000	0,1*E	60	11.241.112	6.338.751	75.938
0,1*E - 0,3*E	1000	0,1*E	60	11.789.987	4.366.618	82.668
0,1*E - 0,3*E	1000	0,1*E	60	11.579.271	6.397.679	66.232
Promedio				**11.626.967**	**6.105.250**	**67.961**

Tmin - Tmax	NIPI	TV	T	Tipo I	Tipo II	Tipo III
0,05*E - 0,2*E	1000	0,1*E	60	11.077.151	4.258.836	52.151
0,05*E - 0,2*E	1000	0,1*E	60	11.812.268	6.518.664	55.630
0,05*E - 0,2*E	1000	0,1*E	60	11.220.720	6.870.204	33.111
0,05*E - 0,2*E	1000	0,1*E	60	11.920.765	4.663.816	70.719
0,05*E - 0,2*E	1000	0,1*E	60	11.951.331	4.632.386	73.891
0,05*E - 0,2*E	1000	0,1*E	60	11.302.356	7.427.430	36.843
0,05*E - 0,2*E	1000	0,1*E	60	11.839.981	6.660.664	66.552
0,05*E - 0,2*E	1000	0,1*E	60	11.864.646	4.535.025	81.903
0,05*E - 0,2*E	1000	0,1*E	60	11.973.816	4.360.658	52.773
0,05*E - 0,2*E	1000	0,1*E	60	11.700.318	5.752.119	59.132
Promedio				**11.666.335**	**5.567.980**	**58.271**

Tmin - Tmax	NIPI	TV	T	Tipo I	Tipo II	Tipo III
0,01*E - 0,03*E	**2000**	0,1*E	60	11.822.617	7.106.778	82.936
0,01*E - 0,03*E	**2000**	0,1*E	60	11.177.590	5.688.512	63.766
0,01*E - 0,03*E	**2000**	0,1*E	60	12.023.941	4.548.311	69.294
0,01*E - 0,03*E	**2000**	0,1*E	60	11.990.301	4.827.617	55.469
0,01*E - 0,03*E	**2000**	0,1*E	60	11.869.572	6.461.700	31.234
0,01*E - 0,03*E	**2000**	0,1*E	60	11.320.674	7.372.524	77.265
0,01*E - 0,03*E	**2000**	0,1*E	60	11.554.786	6.668.605	43.469
0,01*E - 0,03*E	**2000**	0,1*E	60	11.665.199	6.720.105	33.108
0,01*E - 0,03*E	**2000**	0,1*E	60	11.273.345	6.500.309	80.969
0,01*E - 0,03*E	**2000**	0,1*E	60	11.416.243	7.135.526	75.299
Promedio				**11.611.427**	**6.302.999**	**61.281**

Tmin - Tmax	NIPI	TV	T	Tipo I	Tipo II	Tipo III
0,01*E - 0,03*E	1000	**0,05*E**	60	11.102.165	4.670.622	75.318
0,01*E - 0,03*E	1000	**0,05*E**	60	11.909.832	7.472.268	49.220
0,01*E - 0,03*E	1000	**0,05*E**	60	11.323.700	6.168.409	80.674
0,01*E - 0,03*E	1000	**0,05*E**	60	11.980.062	7.004.689	63.799
0,01*E - 0,03*E	1000	**0,05*E**	60	11.271.104	6.908.815	50.252
0,01*E - 0,03*E	1000	**0,05*E**	60	11.997.122	4.607.143	34.254
0,01*E - 0,03*E	1000	**0,05*E**	60	11.674.959	4.111.204	38.903
0,01*E - 0,03*E	1000	**0,05*E**	60	11.597.214	5.533.305	72.572
0,01*E - 0,03*E	1000	**0,05*E**	60	11.190.104	5.485.658	77.669
0,01*E - 0,03*E	1000	**0,05*E**	60	11.114.991	4.863.238	42.687
Promedio				**11.516.125**	**5.682.535**	**58.535**

Tmin - Tmax	NIPI	TV	T	Tipo I	Tipo II	Tipo III
0,01*E - 0,03*E	1000	**0,15*E**	60	11.980.204	6.020.667	68.245
0,01*E - 0,03*E	1000	**0,15*E**	60	12.010.091	5.228.510	31.032
0,01*E - 0,03*E	1000	**0,15*E**	60	11.913.329	4.638.112	67.481
0,01*E - 0,03*E	1000	**0,15*E**	60	11.583.590	6.282.031	37.995
0,01*E - 0,03*E	1000	**0,15*E**	60	11.364.787	6.495.653	71.872
0,01*E - 0,03*E	1000	**0,15*E**	60	11.321.284	4.601.298	71.562
0,01*E - 0,03*E	1000	**0,15*E**	60	11.753.455	4.887.330	52.338
0,01*E - 0,03*E	1000	**0,15*E**	60	11.739.696	4.853.511	56.481
0,01*E - 0,03*E	1000	**0,15*E**	60	11.669.725	6.380.941	67.077
0,01*E - 0,03*E	1000	**0,15*E**	60	11.472.315	5.931.565	72.062
Promedio				**11.680.848**	**5.531.962**	**59.615**

Anexo 1.2. Resultados Tiempos de ejecución

En la siguiente tabla se presentan los resultados del algoritmo con los parámetros ajustados y con distintos tiempos de ejecución. Notar que para mejor lectura se repiten los resultados que se obtuvieron con 60 segundos de ejecución:

Tmin - Tmax	NIPI	TV	T	Tipo I	Tipo II	Tipo III
0,01*E - 0,03*E	1000	0,1*E	**30**	11.692.713	4.600.198	73.736
0,01*E - 0,03*E	1000	0,1*E	**30**	11.058.302	5.088.849	39.446
0,01*E - 0,03*E	1000	0,1*E	**30**	11.064.707	4.477.833	32.079
0,01*E - 0,03*E	1000	0,1*E	**30**	11.391.867	4.765.314	73.080
0,01*E - 0,03*E	1000	0,1*E	**30**	11.563.042	7.407.468	59.558
0,01*E - 0,03*E	1000	0,1*E	**30**	11.400.542	4.984.459	33.262
0,01*E - 0,03*E	1000	0,1*E	**30**	11.900.539	5.084.614	36.122
0,01*E - 0,03*E	1000	0,1*E	**30**	11.715.202	4.609.503	48.107
0,01*E - 0,03*E	1000	0,1*E	**30**	11.323.735	6.307.931	74.525
0,01*E - 0,03*E	1000	0,1*E	**30**	11.434.596	5.900.692	57.345
Promedio				**11.454.525**	**5.322.686**	**52.726**

Tmin - Tmax	NIPI	TV	T	Tipo I	Tipo II	Tipo III
0,01*E - 0,03*E	1000	0,1*E	**60**	11.454.807	4.115.888	81.989
0,01*E - 0,03*E	1000	0,1*E	**60**	11.635.417	5.943.305	56.270
0,01*E - 0,03*E	1000	0,1*E	**60**	11.109.824	6.972.509	69.376
0,01*E - 0,03*E	1000	0,1*E	**60**	11.621.064	5.072.893	74.020
0,01*E - 0,03*E	1000	0,1*E	**60**	11.056.704	4.297.738	47.947
0,01*E - 0,03*E	1000	0,1*E	**60**	11.471.224	5.236.060	53.689
0,01*E - 0,03*E	1000	0,1*E	**60**	11.657.874	4.119.802	36.719
0,01*E - 0,03*E	1000	0,1*E	**60**	11.152.694	5.512.395	52.851
0,01*E - 0,03*E	1000	0,1*E	**60**	11.057.682	4.717.696	42.476
0,01*E - 0,03*E	1000	0,1*E	**60**	11.497.014	5.707.123	61.623
Promedio				**11.371.430**	**5.169.541**	**57.696**

Tmin - Tmax	NIPI	TV	T	Tipo I	Tipo II	Tipo III
0,01*E - 0,03*E	1000	0,1*E	**90**	11.570.821	5.849.491	77.034
0,01*E - 0,03*E	1000	0,1*E	**90**	11.431.625	4.138.564	65.185
0,01*E - 0,03*E	1000	0,1*E	**90**	11.209.303	4.626.742	57.602
0,01*E - 0,03*E	1000	0,1*E	**90**	11.734.221	6.532.137	57.403
0,01*E - 0,03*E	1000	0,1*E	**90**	11.065.431	4.486.543	31.339
0,01*E - 0,03*E	1000	0,1*E	**90**	11.084.527	5.101.351	76.673
0,01*E - 0,03*E	1000	0,1*E	**90**	11.100.554	4.340.997	39.237
0,01*E - 0,03*E	1000	0,1*E	**90**	11.552.632	4.604.379	36.877
0,01*E - 0,03*E	1000	0,1*E	**90**	11.189.623	5.013.067	31.299
0,01*E - 0,03*E	1000	0,1*E	**90**	11.230.655	6.074.355	36.168
Promedio				**11.316.939**	**5.076.763**	**50.882**

Anexo 2. Ejemplo de Solución

Este anexo contiene los resultados para la instancia *022_3_4_EP_RAF_ENP* en el cual el valor de la *Función Objetivo* es 53.174. Los resultados que se muestran en la tabla son los mismos que se exportan a Excel. Por simplicidad en este apartado no se muestra la secuencia final del programa, sin embargo más adelante se expondrá el funcionamiento del programa.

Instancia	F.O.i.	Peor	Mejor	Resultado
022_3_4_EP_RAF_ENP	2.071.002	2.580.116	41.815	33.636
	2.071.002	3.939.022	48.294	77.905
	2.071.002	2.943.326	64.691	31.752
	2.071.002	3.517.490	61.248	83.852
	2.071.002	3.684.169	39.800	59.395
	2.071.002	3.780.260	75.620	39.112
	2.071.002	3.582.264	31.218	38.722
	2.071.002	3.469.871	72.038	83.277
	2.071.002	2.504.741	54.374	48.019
	2.071.002	3.384.073	36.960	36.070
		3.939.022	31.218	53.174
		Promedio		

Anexo 3. Programa Car Sequencing Problem

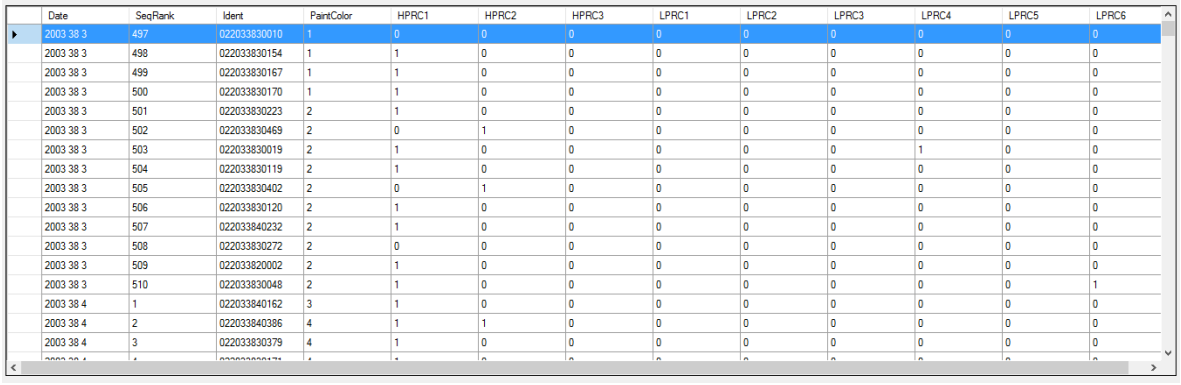
Este anexo contiene el funcionamiento del software resultado de la programación del algoritmo bajo el lenguaje *.net* y escrito en *Microsoft Visual Basic 6.0*. Se detallarán cada uno de los parámetros que deben ser ingresados:

Pantalla de Inicio

- Menú *Inicio*: Muestra la opción que carga una instancia en la pantalla principal. Se selecciona la carpeta completa que contiene los archivos "*.txt"* para ser codificados por el programa.
- Menú *Ayuda*: Muestra la información del propietario del programa vinculado a este Proyecto de Título.
- Casilla *TL*: El usuario puede definir un valor entre $[0,100]$ para el tamaño de *Lista Tabú* para la instancia.

- Casilla *NIPI*: En este espacio el usuario puede definir cuántas iteraciones hará el programa antes que ejecute el proceso de *Intensificación*
- Casilla *TV*: En esta casilla el usuario puede elegir el porcentaje del total de eventos de la instancia será el elegido como tamaño de la vecindad, valor que oscila entre [0,100].

Una vez que el usuario cargue la instancia se desplegarán todos los registros del archivo contenido en *vehicles.txt*:



Date	SeqRank	Ident	PaintColor	HPRC1	HPRC2	HPRC3	LPRC1	LPRC2	LPRC3	LPRC4	LPRC5	LPRC6
2003 38 3	497	022033830010	1	0	0	0	0	0	0	0	0	0
2003 38 3	498	022033830154	1	1	0	0	0	0	0	0	0	0
2003 38 3	499	022033830167	1	1	0	0	0	0	0	0	0	0
2003 38 3	500	022033830170	1	1	0	0	0	0	0	0	0	0
2003 38 3	501	022033830223	2	1	0	0	0	0	0	0	0	0
2003 38 3	502	022033830469	2	0	1	0	0	0	0	0	0	0
2003 38 3	503	022033830019	2	1	0	0	0	0	0	1	0	0
2003 38 3	504	022033830119	2	1	0	0	0	0	0	0	0	0
2003 38 3	505	022033830402	2	0	1	0	0	0	0	0	0	0
2003 38 3	506	022033830120	2	1	0	0	0	0	0	0	0	0
2003 38 3	507	022033840232	2	1	0	0	0	0	0	0	0	0
2003 38 3	508	022033830272	2	0	0	0	0	0	0	0	0	0
2003 38 3	509	022033820002	2	1	0	0	0	0	0	0	0	0
2003 38 3	510	022033830048	2	1	0	0	0	0	0	0	0	1
2003 38 4	1	022033840162	3	1	0	0	0	0	0	0	0	0
2003 38 4	2	022033840386	4	1	1	0	0	0	0	0	0	0
2003 38 4	3	022033830379	4	1	0	0	0	0	0	0	0	0

Al hacer click en el botón *Procesar* el programa procede a ejecutar el algoritmo para finalmente entregar un mensaje emergente con los valores *Peor*, *Mejor* y *Resultado Final* de la *Búsqueda Tabú*. Finalmente el usuario puede exportar los datos finales y el secuenciamiento final a Excel haciendo click en el botón *Exportar a Excel*, con el fin de ser analizado según se requiera.

Anexo 4. Solución Instancias Set A

En este anexo se muestran los resultados de las instancias del *Set A* obtenidos por el Software ejecutado en 10 oportunidades, siendo el resultado final un promedio de estos experimentos:

Instancia	Peor	Mejor	Resultado
022_3_4_EP_RAF_ENP	2.580.116	41.815	33.636
	3.939.022	48.294	77.905
	2.943.326	64.691	31.752
	3.517.490	61.248	83.852
	3.684.169	39.800	59.395
	3.780.260	75.620	39.112
	3.582.264	31.218	38.722
	3.469.871	72.038	83.277
	2.504.741	54.374	48.019
	3.384.073	36.960	36.070
	3.939.022	31.218	53.174
	Promedio		

Instancia	Peor	Mejor	Resultado
022_3_4_RAF_EP_ENP	12.000.505	11.078.306	12.009.737
	11.956.675	11.051.025	11.982.501
	12.024.807	11.274.242	11.305.903
	11.956.087	11.042.486	11.304.215
	11.719.832	11.407.574	11.110.552
	11.847.372	11.094.787	11.905.458
	11.774.977	11.398.636	11.775.148
	11.932.459	11.061.065	11.831.553
	11.899.114	11.372.521	11.890.248
	11.949.716	11.191.981	11.125.740
	12.024.807	11.042.486	11.624.106
	Promedio		

Instancia	Peor	Mejor	Resultado
024_38_3_EP_ENP_RAF	75.580.703	4.219.537	4.544.175
	80.744.014	6.137.518	7.360.761
	75.675.552	5.913.150	7.136.419
	71.300.975	5.642.367	7.266.631
	72.197.879	4.027.125	4.036.744
	81.977.840	4.432.792	4.697.879
	80.968.040	4.530.505	6.718.663
	75.685.615	4.673.661	5.641.753
	71.934.399	5.413.407	5.560.070
	72.387.914	4.262.315	5.534.629
	81.977.840	4.027.125	5.849.772
	Promedio		

Instancia	Peor	Mejor	Resultado
024_38_3_EP_RAF_ENP	73.720.600	4.845.725	6.645.596
	78.847.364	5.065.296	4.374.087
	77.011.617	5.006.554	5.642.598
	75.930.356	4.603.558	5.434.419
	77.458.426	4.609.624	6.579.727
	79.079.168	4.513.317	7.006.019
	79.352.228	4.439.008	4.620.109
	77.916.299	4.882.464	6.135.847
	74.799.222	4.984.272	5.450.792
	79.348.283	4.336.521	7.013.260
	79.352.228	4.336.521	5.890.245
	Promedio		

Instancia	Peor	Mejor	Resultado
024_38_5_EP_ENP_RAF	106.993.529	4.764.650	5.019.297
	106.795.784	4.534.904	4.812.002
	106.833.051	4.927.015	5.006.711
	107.045.052	4.139.736	4.808.741
	106.999.158	4.180.604	4.292.326
	106.325.275	4.269.394	5.039.237
	107.154.723	4.164.528	4.998.893
	106.970.167	4.689.068	4.952.432
	107.023.273	4.873.908	4.973.341
	106.907.483	4.497.832	4.760.861
	107.154.723	4.139.736	4.866.384
	Promedio		

Instancia	Peor	Mejor	Resultado
024_38_5_EP_RAF_ENP	99.342.548	5.636.572	5.927.632
	99.219.006	5.355.462	6.092.227
	100.435.357	5.274.529	6.053.093
	101.476.572	4.497.232	5.141.827
	99.177.940	4.689.748	5.115.091
	99.258.002	5.171.882	5.780.431
	99.790.206	4.626.915	5.434.438
	100.216.762	5.432.672	5.470.644
	98.904.728	4.871.517	4.999.965
	100.077.907	6.175.189	6.181.033
	101.476.572	4.497.232	5.619.638
	Promedio		

Instancia	Peor	Mejor	Resultado
025_38_1_EP_ENP_RAF	2.059.273	11.974	12.461
	2.489.307	12.245	12.322
	2.657.604	10.144	12.459
	1.875.579	11.487	11.558
	2.114.744	10.569	11.607
	2.238.681	10.472	11.727
	1.927.935	10.613	12.290
	1.745.472	10.992	11.668
	2.609.550	11.800	12.367
	1.710.569	11.012	12.252
	2.657.604	10.144	12.071
	Promedio		

Instancia	Peor	Mejor	Resultado
025_38_1_EP_RAF_ENP	610.278	274.041	304.915
	617.555	313.583	314.911
	555.315	290.320	258.686
	552.048	316.983	247.950
	518.112	255.698	261.501
	626.607	261.308	285.040
	461.883	255.274	250.300
	490.243	243.532	305.501
	569.285	252.651	265.362
	564.245	303.851	257.393
	626.607	243.532	275.156
	Promedio		

Instancia	Peor	Mejor	Resultado
039_38_4_EP_RAF_ch1	123.230.000	13.542.000	16.139.000
	150.746.000	13.878.000	16.481.000
	142.212.000	16.620.000	16.868.000
	130.455.000	13.890.000	14.585.000
	146.408.000	13.555.000	14.043.000
	152.541.000	15.375.000	16.742.000
	144.248.000	15.120.000	16.860.000
	128.109.000	15.332.000	16.133.000
	153.150.000	13.860.000	15.436.000
	147.474.000	14.909.000	15.162.000
	153.150.000	13.542.000	15.844.900
	Promedio		

Instancia	Peor	Mejor	Resultado
039_38_4_RAF_EP_ch1	87.262.000	69.004.000	69.517.000
	81.778.000	68.405.000	72.164.000
	77.475.000	71.337.000	72.220.000
	76.615.000	70.359.000	71.555.000
	83.588.000	68.850.000	70.320.000
	87.773.000	70.486.000	70.512.000
	80.153.000	70.756.000	70.850.000
	86.375.000	70.369.000	71.062.000
	80.706.000	68.865.000	70.677.000
	84.008.000	70.635.000	71.020.000
	87.773.000	68.405.000	70.989.700
	Promedio		

Instancia	Peor	Mejor	Resultado
048_39_1_EP_ENP_RAF	64.951.100	17.550	10.767
	61.058.244	8.482	7.560
	72.240.907	17.141	11.765
	67.910.014	16.950	9.550
	59.808.751	15.892	8.933
	70.791.054	12.762	6.215
	67.727.999	7.104	6.204
	62.008.596	12.265	8.643
	71.535.343	8.314	7.497
	64.347.274	18.474	15.823
	72.240.907	7.104	9.296
	Promedio		

Instancia	Peor	Mejor	Resultado
064_38_2_EP_RAF_ENP_ch1	3.265.782	111.040	110.096
	2.912.891	110.179	109.527
	2.945.070	109.264	109.220
	3.083.747	111.171	110.808
	2.227.010	109.341	109.301
	2.427.342	110.925	110.377
	3.186.322	110.455	110.289
	3.315.043	112.087	109.209
	2.553.458	112.396	110.122
	3.515.527	112.689	112.238
	3.515.527	109.264	110.119
	Promedio		

Instancia	Peor	Mejor	Resultado
064_38_2_EP_RAF_ENP_ch2	28.869.521	1.926.842	1.997.565
	29.769.413	976.778	1.114.767
	29.067.466	289.290	1.300.842
	29.592.788	2.032.023	2.041.672
	28.919.305	1.079.087	1.557.619
	30.555.745	659.390	1.037.091
	28.926.875	688.936	1.291.415
	29.413.781	1.094.610	1.450.796
	29.782.826	39.965	1.352.694
	28.084.233	1.697.221	1.975.489
	30.555.745	39.965	1.511.995
	Promedio		

Instancia	Peor	Mejor	Resultado
064_38_2_RAF_EP_ENP_ch1	70.073.300	64.411.555	65.920.201
	70.724.345	64.443.868	67.179.167
	69.535.001	67.621.935	67.734.722
	69.661.267	65.131.660	67.035.967
	70.671.617	67.401.196	67.556.048
	68.365.460	66.570.438	66.850.064
	68.584.691	64.349.523	65.380.844
	69.361.053	67.038.027	68.118.083
	68.998.354	64.517.495	65.522.179
	69.791.836	67.226.707	67.275.080
	70.724.345	64.349.523	66.857.236
	Promedio		

Instancia	Peor	Mejor	Resultado
064_38_2_RAF_EP_ENP_ch2	43.610.100	36.748.791	34.315.907
	44.216.083	37.914.818	30.035.153
	44.165.001	37.423.597	30.614.558
	43.752.693	30.860.492	30.689.970
	43.917.290	31.023.535	29.998.547
	42.411.941	39.848.262	33.941.417
	44.129.588	32.820.092	30.022.472
	42.658.491	34.857.470	31.459.541
	43.275.484	36.473.163	35.874.978
	43.814.869	31.367.907	29.847.082
	44.216.083	30.860.492	31.679.963
	Promedio		

Anexo 5. Solución Instancias Set X

En este anexo se muestran los resultados de las instancias del *Set X* obtenidos por el Software ejecutado en 5 oportunidades, igualando la cantidad de experimentos utilizados por los participantes en ROADEF 2005. El resultado final será por tanto el promedio de estos experimentos:

Instancia	Peor	Mejor	Resultado
022_RAF_EP_ENP_S49_J2	14.669.174	13.297.410	13.048.960
	14.483.176	13.839.699	13.756.836
	14.912.260	13.376.012	13.358.271
	14.926.905	13.307.248	13.049.416
	14.378.796	13.192.851	13.174.500
	14.926.905	13.192.851	13.277.597
	Promedio		

Instancia	Peor	Mejor	Resultado
023_EP_RAF_ENP_S49_J2	251.773.417	494.333	794.281
	251.433.946	2.145.061	2.583.764
	266.420.772	1.158.664	1.177.164
	262.363.341	327.869	805.929
	260.372.123	2.118.470	2.646.242
	266.420.772	327.869	1.601.476
	Promedio		

Instancia	Peor	Mejor	Resultado
024_EP_RAF_ENP_S49_J2	73.101.965	1.414.073	2.055.032
	74.773.984	1.076.698	1.942.358
	72.013.807	1.776.860	1.842.573
	72.832.704	457.858	2.067.923
	75.118.207	1.539.544	1.682.388
	75.118.207	457.858	1.918.055
	Promedio		

Instancia	Peor	Mejor	Resultado
025_EP_ENP_RAF_S49_J1	722.865.519	2.985.447	3.065.701
	722.347.104	327.799	1.284.456
	755.251.008	2.327.620	3.086.779
	745.212.138	737.521	2.513.239
	719.199.308	1.475.635	1.924.236
	755.251.008	327.799	2.374.882
	Promedio		

Instancia	Peor	Mejor	Resultado
028_CH2_EP_ENP_RAF_S51_J1	3.045.062	635	730
	3.043.223	139	391
	3.044.373	909	953
	3.043.876	777	1.001
	3.044.577	751	772
	3.045.062	139	769
	Promedio		

Instancia	Peor	Mejor	Resultado
028_CH2_EP_ENP_RAF_S51_J1	395.979	139.301	143.356
	479.387	120.452	142.555
	273.997	134.567	136.425
	449.060	133.678	136.621
	508.736	116.958	120.464
	508.736	116.958	135.884
	Promedio		

Instancia	Peor	Mejor	Resultado
034_VP_EP_RAF_ENP_S51_J1_J2_J3	172.421	61.014	63.989
	152.295	59.252	64.079
	168.721	61.530	63.808
	176.622	60.523	64.250
	154.861	62.027	63.680
	176.622	59.252	63.961
	Promedio		

Instancia	Peor	Mejor	Resultado
034_VU_EP_RAF_ENP_S51_J1_J2_J3	12.629.892	10.811.842	9.533.096
	12.910.358	9.837.894	8.602.925
	11.912.291	10.631.709	9.879.402
	11.198.867	8.114.622	8.110.687
	11.363.563	8.801.027	8.689.174
	12.910.358	8.114.622	8.963.057
	Promedio		

Instancia	Peor	Mejor	Resultado
035_CH1_RAF_EP_S50_J4	6.252.000	5.739.000	6.111.000
	6.594.000	5.147.000	6.521.000
	6.391.000	5.806.000	6.611.000
	6.304.000	5.477.000	6.476.000
	6.680.000	5.840.000	6.401.000
	6.680.000	5.147.000	6.424.000
	Promedio		

Instancia	Peor	Mejor	Resultado
035_CH2_RAF_EP_S50_J4	7.491.000	6.971.000	6.771.000
	7.985.000	6.655.000	6.494.000
	7.718.000	6.354.000	6.244.000
	7.153.000	6.589.000	6.358.000
	7.203.000	6.167.000	6.728.000
	7.985.000	6.167.000	6.519.000
	Promedio		

Instancia	Peor	Mejor	Resultado
039_CH1_EP_RAF_ENP_S49_J1	90.746	69.664	70.030
	91.603	71.109	71.195
	92.896	70.600	70.814
	95.074	71.209	71.212
	90.514	70.436	70.829
	95.074	69.664	70.816
	Promedio		

Instancia	Peor	Mejor	Resultado
039_CH3_EP_RAF_ENP_S49_J1	294.155	232.828	236.235
	302.113	238.826	253.013
	312.102	238.514	242.665
	323.585	233.173	251.581
	314.625	235.171	252.247
	323.585	232.828	247.148
	Promedio		

Instancia	Peor	Mejor	Resultado
048_CH1_EP_RAF_ENP_S50_J4	4.267.944	200.023	202.023
	3.888.420	197.556	197.909
	3.457.269	199.695	200.920
	4.620.735	200.630	201.546
	3.112.258	197.422	200.515
	4.620.735	197.422	200.583
	Promedio		

Instancia	Peor	Mejor	Resultado
048_CH2_EP_RAF_ENP_S49_J5	35.728.833	31.812.042	32.806.927
	37.957.564	34.972.759	35.119.722
	37.437.571	31.581.154	31.706.012
	36.077.338	32.194.183	34.566.427
	35.929.403	32.516.331	34.900.193
	37.957.564	31.581.154	33.819.856
	Promedio		

Instancia	Peor	Mejor	Resultado
064_CH1_EP_RAF_ENP_S49_J1	67.432.166	63.370.615	63.612.707
	67.813.784	62.854.626	63.691.485
	68.145.638	64.167.791	64.170.256
	68.197.767	62.071.208	63.298.742
	68.062.165	62.308.431	63.797.639
	68.197.767	62.071.208	63.714.166
	Promedio		

Instancia	Peor	Mejor	Resultado
064_CH2_EP_RAF_ENP_S49_J4	80.000	39.000	41.000
	76.000	39.000	41.000
	78.000	38.000	39.000
	73.000	40.000	40.000
	74.000	39.000	39.000
	80.000	38.000	40.000
	Promedio		

Instancia	Peor	Mejor	Resultado
655_CH1_EP_RAF_ENP_S51_J2_J3_J4	54.000	31.000	31.000
	53.000	32.000	32.000
	55.000	34.000	34.000
	55.000	32.000	34.000
	56.000	33.000	34.000
	56.000	31.000	33.000
	Promedio		

Instancia	Peor	Mejor	Resultado
655_CH2_EP_RAF_ENP_S52_J1_J2_S01_J1	308.965.000	159.170.000	164.124.000
	306.521.000	163.494.000	174.895.000
	312.221.000	154.149.000	168.535.000
	318.070.000	155.947.000	167.763.000
	313.959.000	155.604.000	171.867.000
	318.070.000	154.149.000	169.436.800
	Promedio		