Capítulo 4

Desarrollo del Algoritmo de Búsqueda en Vecindad en Visual Basic

4.1 Justificación de la utilización de Visual Basic

Los motivos esenciales por los que se ha decidido utilizar Visual Basic son los siguientes:

- Es una herramienta de programación que permite crear aplicaciones propias (programas) para Windows.
- Se pueden crear ventanas, botones, menús y cualquier otro elemento de Windows de una forma fácil e intuitiva.
- La facilidad que presenta para desarrollar aplicaciones complejas en breve tiempo.

Por estos motivos se opta por utilizar el software Visual Basic.

4.2 Entorno de Visual Basic

A continuación se explican de forma resumida, las herramientas de Visual Basic más usadas en el proyecto.

4.2.1 La barra de menús y las barras de herramientas

La barra de menús de Visual Basic resulta similar a la de cualquier otra aplicación de Windows. Debajo de dicha barra aparecen las barras de herramientas, con una serie de botones que permite acceder fácilmente a las opciones más importantes de los menús.

Existen cuatro barras de herramientas: Debug, Edit, Form Editor y Standard. Por defecto sólo aparece la barra Standard. Clicando en menú, en Ver se despliega una lista, donde se encuentra barras de herramientas. Si se coloca sobre ella, se despliega otra lista donde aparecen las cuatro barras de herramientas. A continuación se muestra en la Figura 28 como incluir cualquier barra de menú al proyecto de Visual Basic.

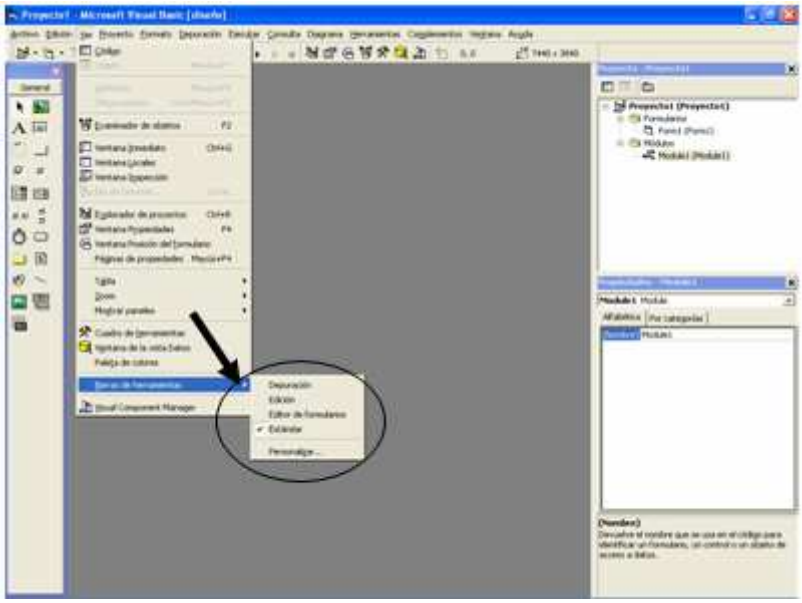


Figura 28. Insertar barras de menús con VB.

4.2.2 Las herramientas (toolbox)

Aparece en una caja de componentes donde se incluyen los controles con los que se puede diseñar la pantalla de la aplicación. Los controles más usados son por ejemplo botones, etiquetas, cajas de textos, etc. Para introducir un control en el formulario basta con clicar en el icono adecuado de la toolbox y colocarlo en el formulario con la posición y el tamaño deseado, clicando y arrastrando con el ratón.

La caja de componentes, toolbox, aparece con unos controles específicos pero existe muchos más, para introducir nuevos componentes se utiliza el comando Components en el menú Project.

A continuación en la Figura 29 se muestra una ventana del software Visual Basic, donde se indica la barra de herramienta.

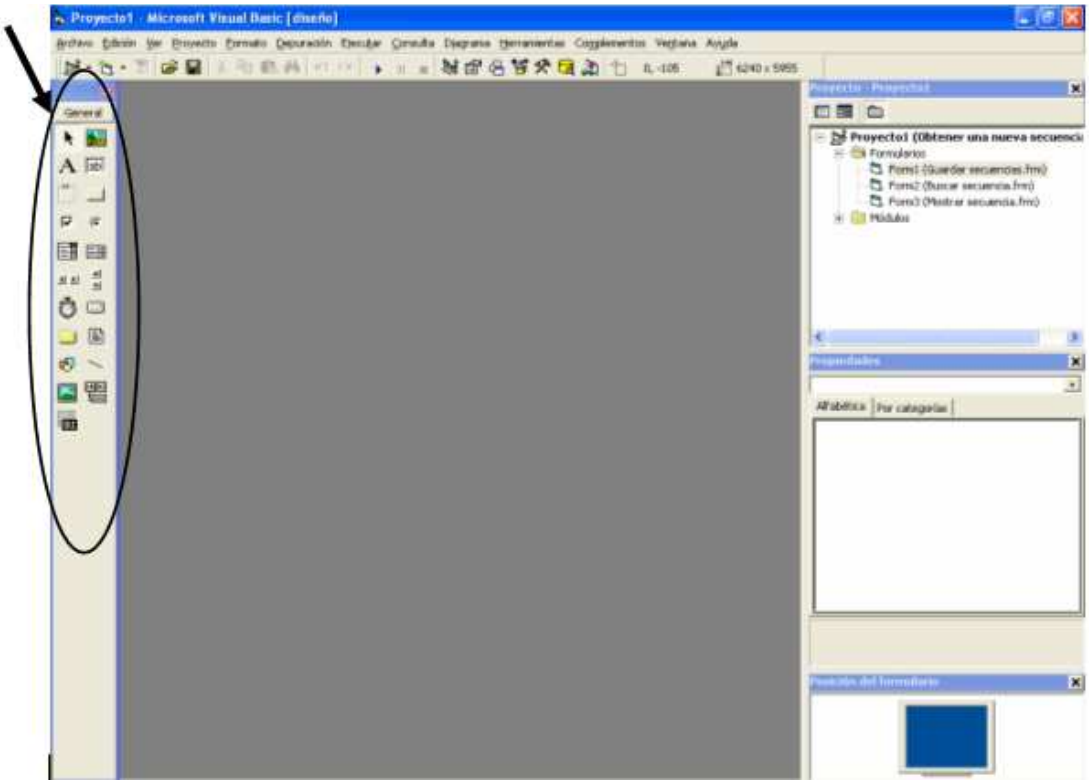


Figura 29. Barra de herramienta de Visual Basic.

4.2.3 Formularios (forms) y módulos

Los formularios son las zonas de la pantalla sobre las que se diseña el programa y sobre las que se sitúan los controles o herramientas de la toolbox. Al ejecutar el programa, el formulario se convertirá en la ventana de la aplicación, donde aparecen los botones de la aplicación, el texto, etc. Pero aunque su estructura es similar a una ventana, posee código de programación y controlará algunos aspectos del formulario, sobre todo en la forma de reaccionar ante las acciones del usuario (eventos).

Para lograr una mejor presentación, en el formulario existe una malla o retícula (grid) que permite alinear los controles manualmente de una forma precisa (evitando tener que introducir coordenadas continuamente) Esta malla sólo será visible en el proceso de diseño del programa; al ejecutarlo no se verá.

Un proyecto realizado en Visual Basic está formado por un conjunto de ficheros o módulos necesarios para que un programa funcione. Dichos módulos pueden ser de varios tipos:

- Aquellos que están asociados a un formulario.
- Los que contiene únicamente líneas de código Basic, denominados módulos estándar.
- Los que definen agrupaciones de código y datos, denominados módulos de clase.

4.2.4 La ventana de proyecto (Project)

Esta ventana, mostrada en la Figura 30, permite acceder a los distintos formularios y módulos que componen el proyecto. Desde ella se puede ver el diseño gráfico de dicho formularios (botón View Object), y también permite editar el código que contienen (botón View Code). Estos botones están situados en la parte superior de la ventana, debajo de la barra de títulos.

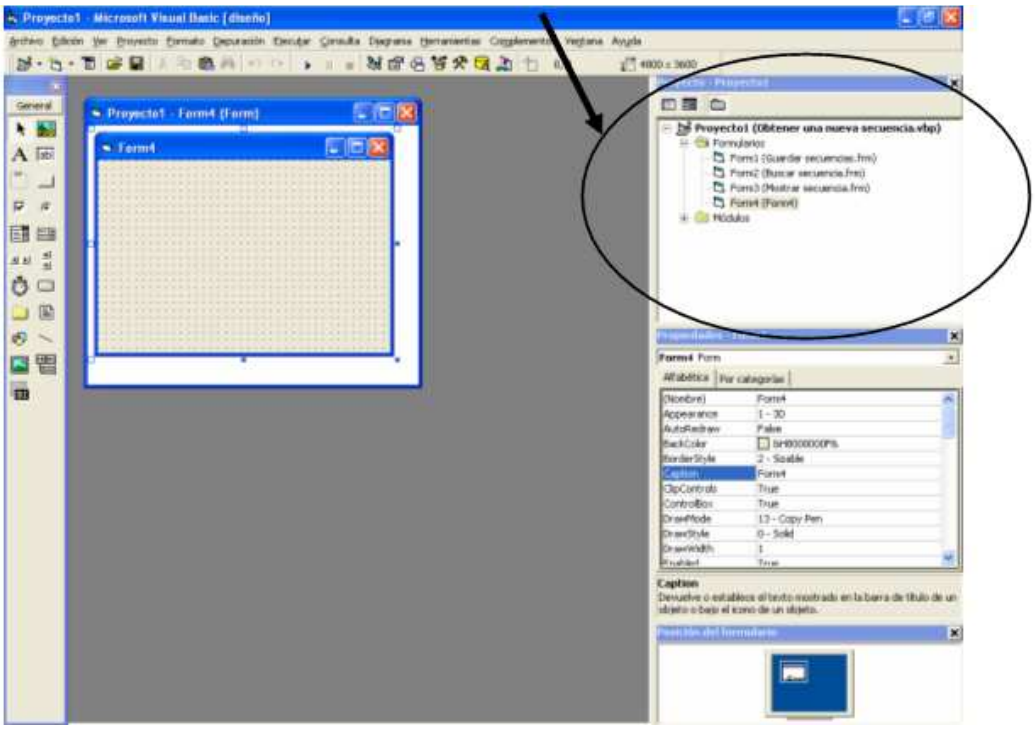


Figura 30. Ventana de proyecto.

4.2.5 La ventana de propiedades (Properties)

Todos los objetos Visual Baisc tienen unas propiedades que los definen: su nombre (Name), su etiqueta o titulo (Caption), el texto que contiene (Text), su tamaño y posición, su color, si está activo o no (Enabled), etc. En la Figura 31 se muestra parcialmente las propiedades de un formulario. Todas estas propiedades se almacenan dentro de cada control o formulario en forma de estructura.

Para realizar una modificación de las propiedades de un objeto durante el diseño del programa, se activará la ventana de propiedades (con el menú, con el botón de la barra de herramienta o pulsado <F4>). Esta ventana tiene dos lengüetas, que permiten ordenar las propiedades alfabéticas o por categorías. Utilizando la forma que sea más cómoda se localiza con ayuda de la barra de desplazamiento la propiedad que se quiera modificar.

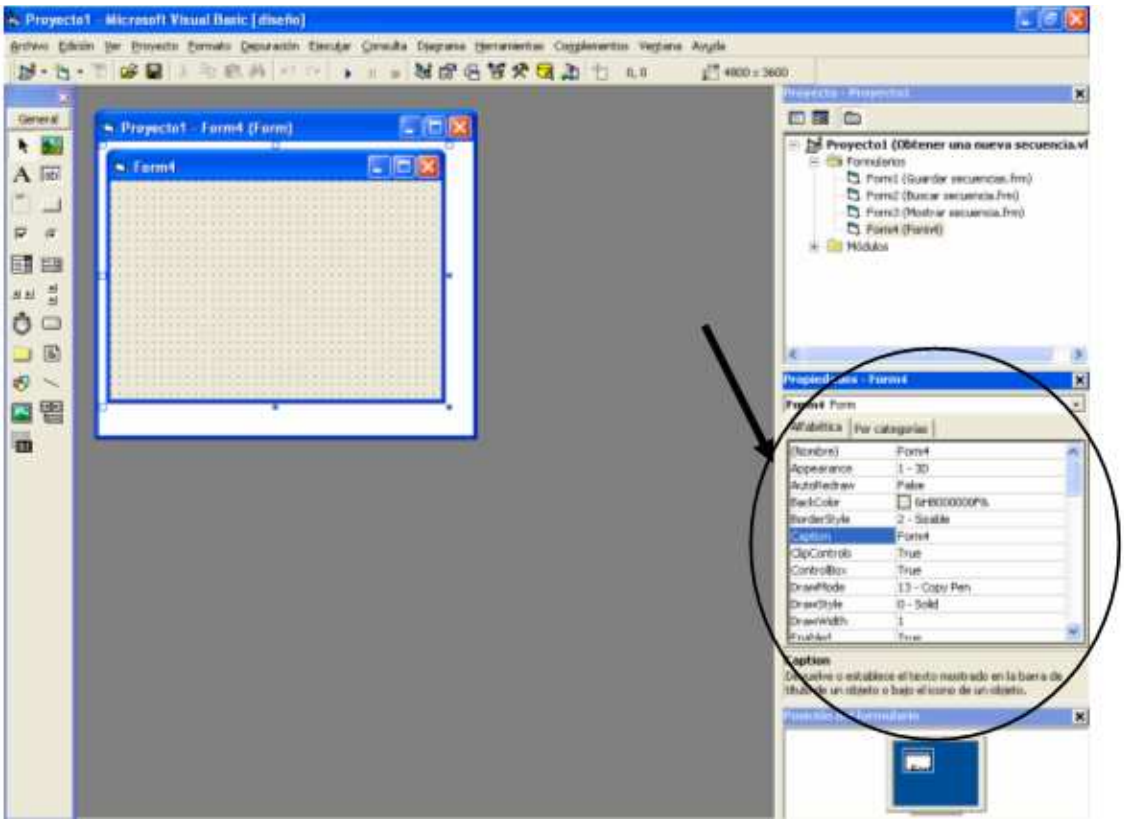


Figura 31. Ventana de propiedades de VB.

4.3 Aplicación del Algoritmo de Búsqueda en Vecindad en Visual Basic

En la aplicación Visual Basic de Microsoft se trabaja en dos modos distintos:

- ***Modo de diseño***: el programador construye interactivamente la aplicación, colocando controles en el formulario, definiendo sus propiedades y desarrollando funciones para gestionar los eventos.
- ***Modo de ejecución***: la aplicación se prueba en modo de ejecución. En ese caso el programador actúa sobre el programa (introduce eventos) y prueba cómo responde el programa.

4.3.1 Modo Diseño del programa

El proyecto está formado por tres formularios: Guardar secuencia, Buscar secuencia y Mostrar secuencia, y un modulo donde se declaran los arrays dinámicos y estáticos, y variables publicas necesarias para llevar a cabo el proyecto.

A continuación se indican las etapas pasos del diseño del programa:

- *Estructurar el programa*: se realiza el esqueleto o base del programa mediante los formularios, botones, etc. que facilita Visual Basic. Y se definen las propiedades y los objetivos de cada elemento del programa.
- *Programación en Visual Basic*: desarrollo de las funciones que gestionan las respuestas del programa ante los distintos eventos del usuario.

Estructura del programa

La estructura del programa se puede dividir en tres etapas que se exponen a continuación:

- Primera etapa: introducir las secuencias, que formarán el conjunto de soluciones factibles, y estudiarlas.

En primer lugar se trabaja sobre la idea de como va ha introducir el usuario las secuencias. Al principio se realizó un formulario, donde el usuario introducía una a una las operaciones que formaban cada estación y a su vez cada secuencia. Pero se llega a la conclusión que era muy laborioso para el usuario y tampoco aportaba grandes ventaja.

Finalmente se decide construir un formulario que permita acceder al fichero, donde el usuario tiene la secuencia. Para ello, se facilita el recorrer el árbol de ficheros y de directorio, localizando interactivamente cada fichero determinado. A partir de los controles FileListBox (para ficheros), el DirListBox (para directorios) y el DriveListBox (para unidades de disco). Los dos primeros son listas, mientras que el tercero es una caja. En la Figura 32 se muestra estos controles en el primer formulario del programa.

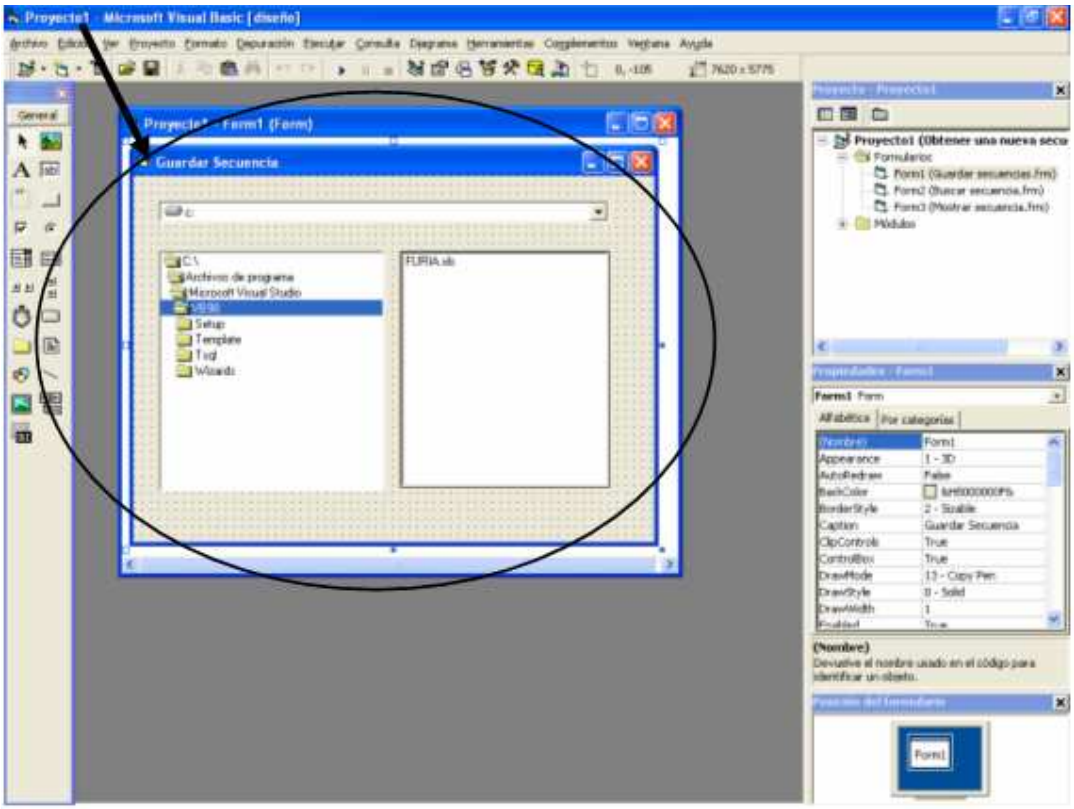


Figura 32. Controles DriveListBox, DirListBox y FileListBox.

Al ejecutar la aplicación se observa que los controles del formulario no se encuentran ligados. Para ligarlos, se hace doble clic sobre drivelistbox y se abre la ventana de código, donde se indica que la ruta del drivelistbox debe ser la misma que la de dirlistbox. Se realiza los mismos pasos pero para ligar las carpetas con los ficheros, es decir, la ruta de dirlistbox debe coincidir con la ruta de filelistbox.

El siguiente paso es añadir un botón para que el usuario pueda incluir la secuencia localizada con los controles anteriormente mencionados. Por tanto, se introduce un command Button. A este botón mediante la propiedad Caption, se le colocará: “Añadir secuencia”. Cuando el usuario pulse el botón en tiempo de ejecución, el programa debe buscar la secuencia en la ruta indicada, transformarla en una matriz con números, guardarla, calcular su tiempo de ciclo, calcular el número de motocicletas al día fabricado con dicho tiempo de ciclo y almacenar dicho valor.

También se incluirá en el formulario otro command button, cuyo Caption será “Terminado”. Este botón lo pulsará el usuario cuando haya terminado de seleccionar todas las secuencias. Y dará paso al siguiente formulario.

De este modo queda terminado el diseño de este formulario al que cual se le denomina “Guardar secuencias” y queda como se observa en la Figura 33.

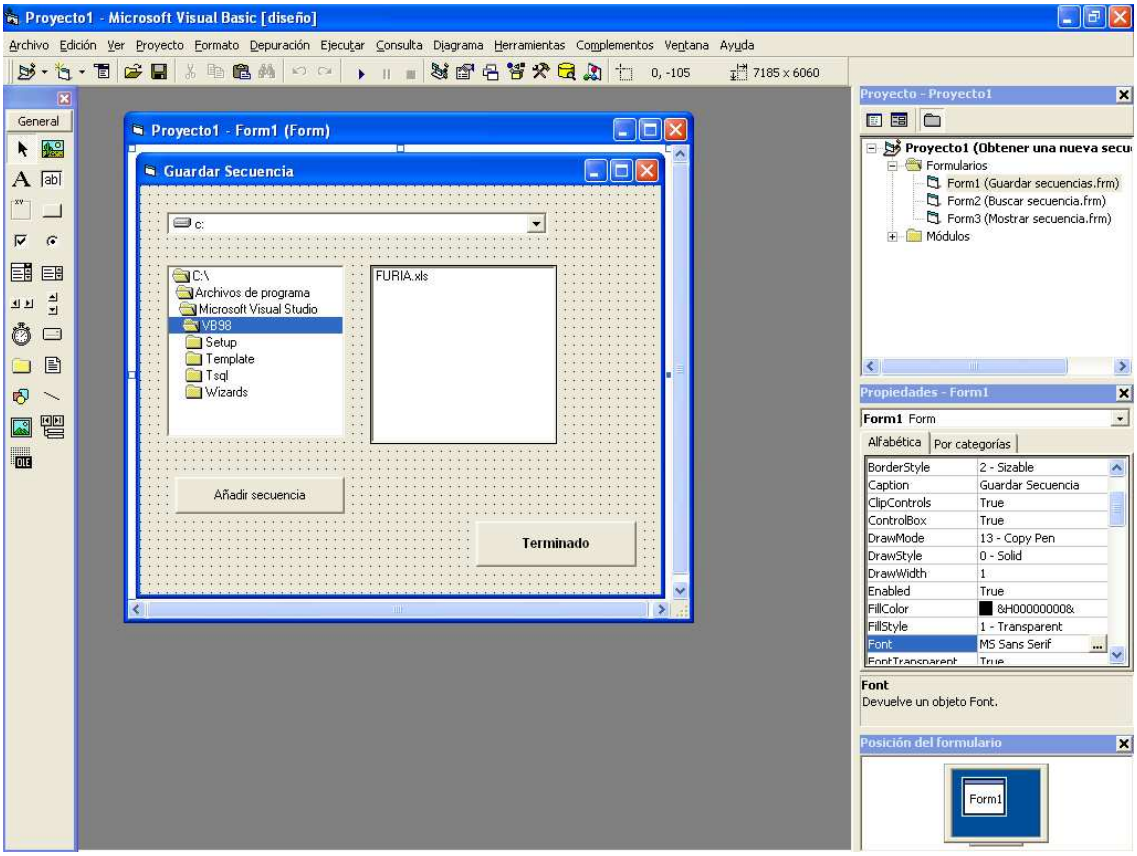


Figura 33. Formulario Guardar secuencias en modo diseño.

- Segunda etapa: buscar una secuencia más óptima.

Para esta segunda etapa se incluye otro formulario al proyecto, donde sólo aparecerá un botón. Cuando el usuario pulse el command button se indicará que se inicie la búsqueda de una secuencia más óptima. A este formulario se le denominará, “Buscar secuencia”. Y al Caption del command button también “Buscar secuencia”.

A continuación se muestra en la Figura 34 este formulario. Se puede observar que se encuentra en modo de diseño, debido a que en el formulario aparece una malla o retícula, que permite alinear los controles que presenta.

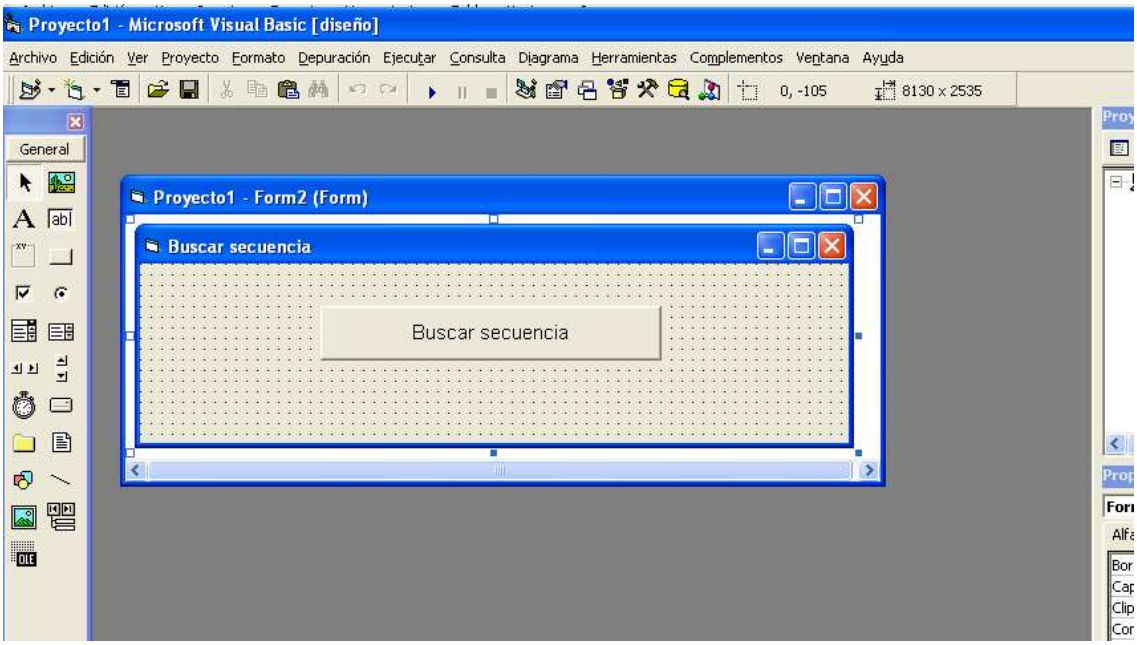


Figura 34. Formulario Buscar secuencia en modo diseño.

- Tercera etapa: comunicar al usuario si se ha encontrado una secuencia más óptima. Y si es así mostrar dicha secuencia por pantalla.

La secuencia más óptima se mostrara en otro formulario al que se denominará, Mostrar secuencia. Y este formulario contiene los siguientes controles: una etiqueta, label, y doces listbox. En el Caption del label se colocara “Nueva secuencia” y los listbox se agruparan en cuatro, que formaran cada estación de la secuencia óptima. Y a su vez estos grupos están formados por tres listbox, cada uno de ello contienen las operaciones que realizan capa operario (operario de la izquierda, ambos operarios y operario de la derecha). Todo esto se refleja en el último formulario que se muestra en la Figura 35.

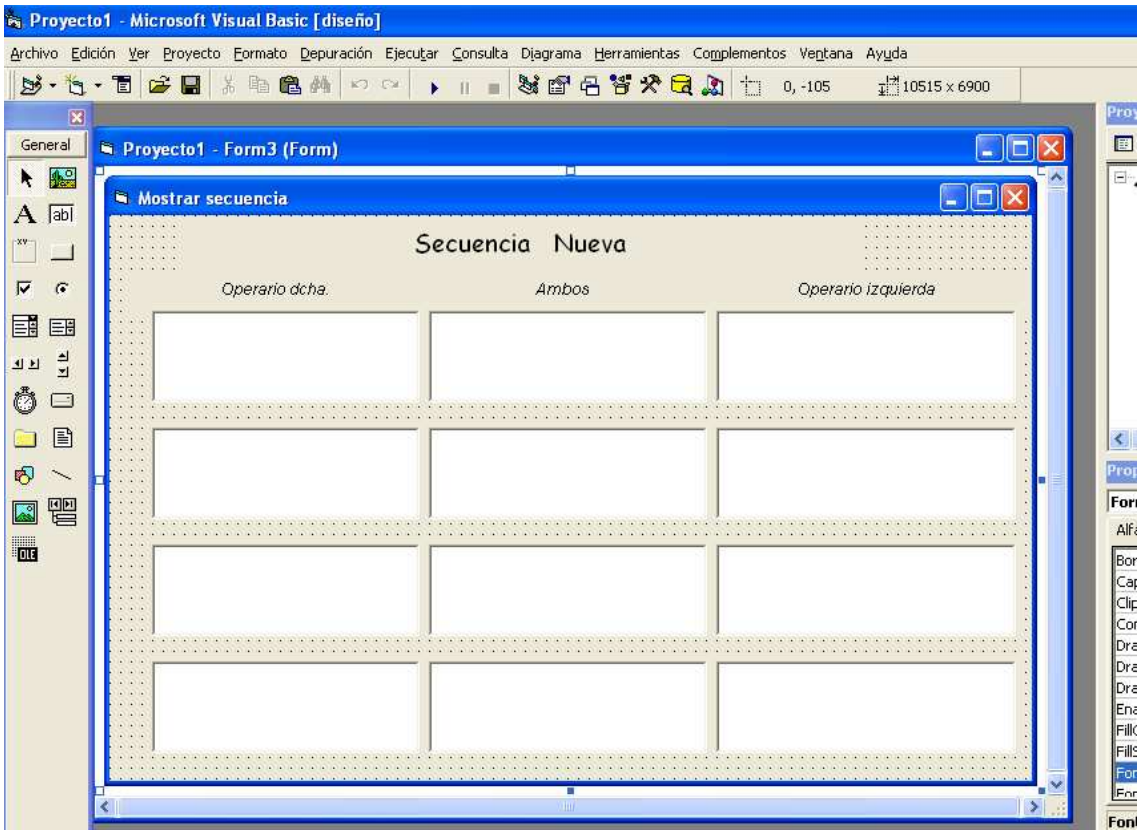


Figura 35. Formulario Nueva Secuencia en modo diseño.

Programación en Visual Basic

Una vez realizado el esqueleto del programa, el siguiente paso es definir las funciones que realizan los objetivos de cada control de cada formulario.

Esta última etapa se divide en tres fases, que están asociadas a cada formulario:

Formulario Guardar secuencia:

Este será el formulario por el que comience la aplicación, por tanto, se tiene que indicar al proyecto. Para ello se entra en Proyecto, en Propiedades Proyecto y se abre una ventana, la cual se observa en la Figura 36, en pestaña General se indica el formulario que comienza la ejecución del programa.

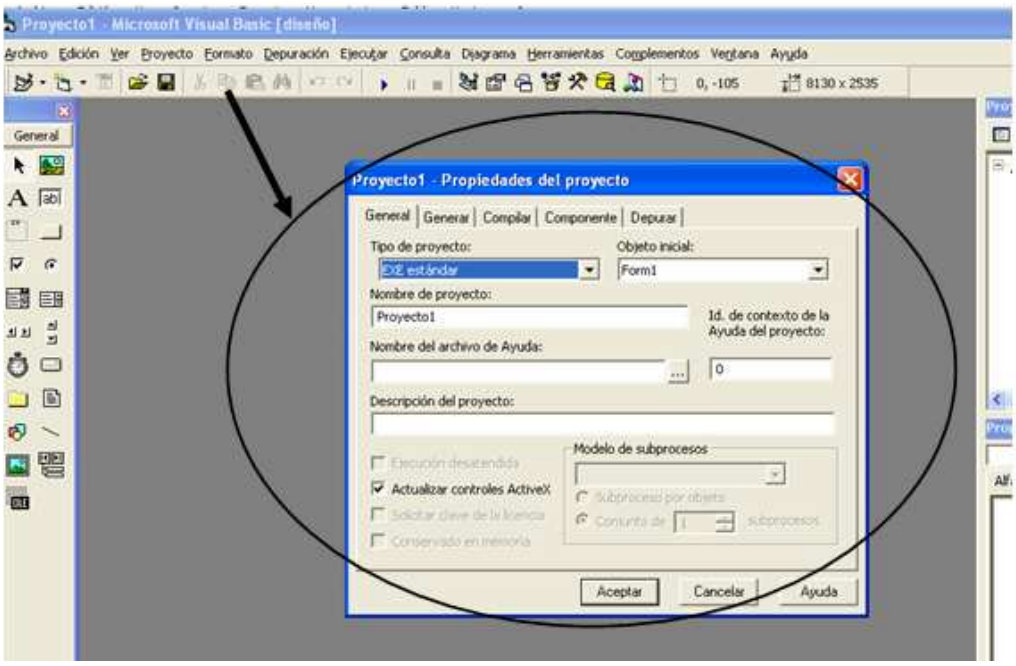


Figura 36. Propiedades de Proyecto.

Antes de comenzar la descripción de las funciones de cada elemento. Se debe de saber que cada operación tiene asignado un número que la identifica, y que sirve para la asignación de predecesores de una operación determinada.

Se comienza por el command button, “Añadir secuencia” que se observa en la Figura 37, el cual tiene los siguientes objetivos:

- Leer la secuencia de la ruta indicada por el usuario.
- Busca el número asociado a cada una de las operaciones que forma la secuencia y las guarda en una matriz denominada secuencia. Esta matriz será un array público para que se pueda acceder a él desde cualquier punto del proyecto. Ya que posteriormente, se seleccionará una secuencia y se trabajara con ella. En ese momento se necesita crear un modulo para declarar dicho array publico.
- Calcular el tiempo de ciclo de dicha secuencia y el número de motocicletas al día que se fábrica con dicho tiempo.
- Almacenar el número de motocicletas fabricados al día, en otro array publico.

Y el command button, “Terminado” que también se puede observar en la Figura 37, realiza los siguientes pasos:

- Calcula cual es el número de motocicletas mayor, alcanzado por las secuencias introducidas por el usuario, y lo almacena en una variable denominada mejor.
- Aplica un peso a cada secuencia, según el número de motocicletas fabricada al día.
- Indica, mediante un mensaje, al usuario cual es la mejor secuencia y el número de motocicletas al día que fábrica.
- Muestra el siguiente formulario, y cierra este.

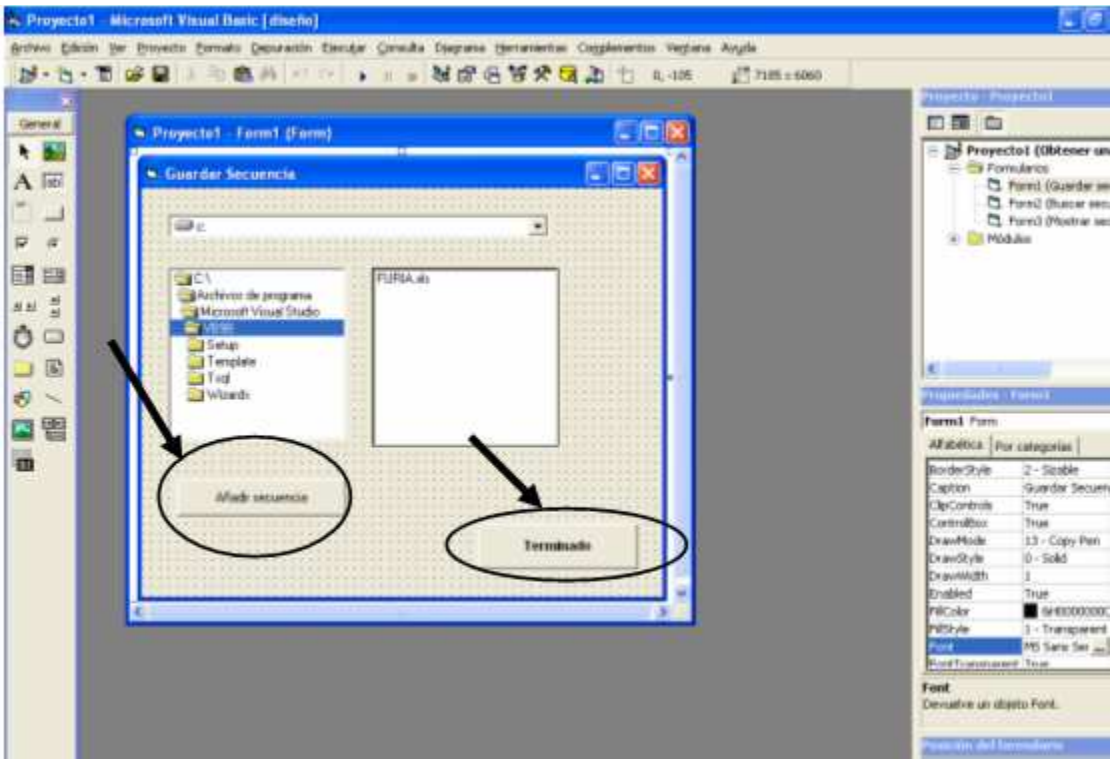


Figura 37. Formulario Guardar secuencia.

Formulario Buscar secuencia:

En este formulario sólo existe un command button, “Buscar secuencia” que se observa en la Figura 38. Las funciones que desempeñará el botón en tiempo de ejecución son los siguientes:

- A partir de los pesos de cada secuencia, se selecciona una secuencia por el método de la ruleta.
- Se realiza movimientos de vecindad en la secuencia seleccionada. El movimiento de vecindad (intercambio de dos operaciones) se realiza en los pasos siguientes:
  - Intercambiar dos operaciones. Se realizará N_w veces dentro de la estación cuello botella y otras N_k en toda la secuencia.
  - Si el movimiento de vecindad es admisible se calcula el tiempo de ciclo y el número de motocicletas. Y se compara con el mejor resultado (variable mejor). Sino se vuelva al paso primero.
  - Si el número de motocicletas es mayor a mejor, por tanto, sustituye al padre en el conjunto de soluciones factibles y se trabaja ahora con la secuencia obtenida, una secuencia vecina. Sino vuelve al primer paso.
- Al finalizar las iteraciones, se pregunta si se ha obtenido una nueva secuencia. Si no es así, se indica al usuario que no se ha encontrado una secuencia mejor. Si no indica al usuario el número de motocicletas de la nueva secuencia y muestra el formulario siguiente.

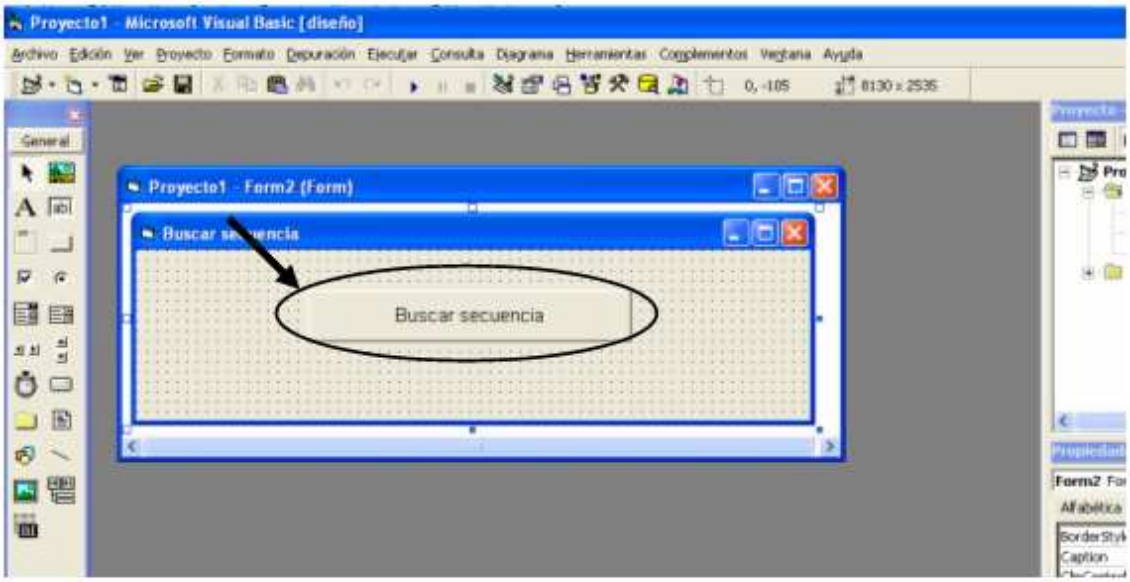


Figura 38. Command button, buscar secuencia.

Formulario Nueva secuencia:

En este formulario solo se muestra la secuencia vecina. Este formulario aparece en la Figura 35, que se mostró anteriormente.

Definición de las funciones usadas en VB

Una vez explicado los objetivos de cada elemento, se expone las cinco funciones más importantes usadas en el programa:

- Función oper_est
- Función crear_matriz
- Función tiemp_ciclo
- Función guardar_sec
- Función movimt_vecind

Función oper_est

Prototipo: void oper_est (f As Integer, oper As Integer, desde As Integer, hasta As Integer).

- *Parámetros de entrada*: número de fila donde comienza a leer la hoja Excel, el número de operaciones de la estación, numero de la fila del comienzo de la estación y el número de fila del final de la estación.
- *Parámetros de salida*: void (un entero).

Definición: el modulo de memoria de corto plazo llama a esta función para que asignar a la variable oper el numero de operaciones que hay en la estación.

Función crear_matriz

Prototipo: void crear_matr (oper As Integer, desde1 As Integer, hasta As Integer, filas As Integer, data As Integer)

- *Parámetros de entrada*: el número de filas de la matriz a crear, el numero de la fila donde comienza a leer la hoja Excel, el número hasta donde lee, puntero que recorre la hoja Excel y puntero que recorre la matriz.

- *Parámetros de salida*: void (ninguno)

Definición: el modulo de memoria de corto plazo llama a esta función para crear una matriz con numero que identifica a cada operación de la secuencia.

Función tiemp_ciclo

Prototipo: void tiemp_ciclo (e As Integer)

- *Parámetros de entrada*: el número de la estación.

- *Parámetros de salida*: void (ninguno)

Definición: el modulo de memoria de corto plazo llama a esta función para calcular el tiempo de la estación indicada, guardándolo en un vector tiemp_estac.

En esta función se llaman a varias funciones que se describe a continuación:

Función buc_tiemp_oper

Prototipo: date busc_tiemp_oper (f As Integer, i As Integer, j As Integer).

- *Parámetros de entrada*: puntero que recorre la hoja Excel donde busca el tiempo de la operación, el numero de la fila de la matriz y el numero de la columna de la matriz.

- *Parámetros de salida*: date (una variable de tiempo)

Definición: el modulo de memoria de corto plazo llama a esta función para calcular el tiempo de la operación que esta leyendo en la matriz de número.

Función buc_pred

Prototipo: integer busc_pred (i As Integer, j As Integer, hasta As Integer)

- *Parámetros de entrada*: el número de la fila y el número de la columna de la matriz, y hasta que fila de la matriz tiene que buscar.

- *Parámetros de salida*: integer (una variable entera)

Definición: el modulo de memoria de corto plazo llama a esta función para calcular la posición más alejada de sus predecesores.

Función guardar_sec

Prototipo: void guard_sec (c As Integer, h As Integer, data As Integer, e As Integer, contador As Integer).

- *Parámetros de entrada*: el número de filas de la matriz de número, puntero para las filas de matriz de número, puntero para las columnas de la matriz de

número, el número de la estación de la secuencia y la posición que ocupa dicha secuencia en la matriz secuencia.

- *Parámetros de salida*: void (ninguno)

Definición: el modulo de memoria de largo plazo llama a esta función para guardar la matriz de numero en una matriz donde se almacenan todas las secuencias.

Función movimt_vecind

Prototipo: void movimt_vecind (v1 As Double)

- *Parámetros de entrada*: el peso de la secuencia seleccionada.
- *Parámetros de salida*: void (ninguno)

Definición: el modulo de memoria de largo plazo llama a esta función para intercambiar, un numero determinado de veces, dos operaciones de la secuencia seccionada, y comparar si esta secuencia vecina es mejor que la seleccionada. Si es así la guardara.

En esta función se llaman a varias funciones que se describe a continuación:

Función pos_cuello_botell

Prototipo: integer posc_cuell_botell (h As Integer, v As date).

- *Parámetros de entrada*: la posición de la secuencia en la matriz secuencia y el vector tiemp_est.
- *Parámetros de salida*: integer (una variable entera), posición del cuello de botella.

Definición: el modulo de memoria de corto plazo llama a esta función para calcular cual es la estación cuello botella de la secuencia seleccionada.

Función compr_operario

Prototipo: void compr_operario (oper As Integer, c As Integer)

- *Parámetros de entrada*: el numero de la operación intercambiada y el numero de la columna donde esta colocada, esto indica de forma indirecta el operario que la realiza.

- *Parámetros de salida*: void (ninguno)

Definición: el modulo de memoria de corto plazo llama a esta función para comprobar que los cambios realizados cumplen las precedencias de obligación de la operación.

Función motos_sec

Prototipo: Integer busc_pred (i As Integer, j As Integer, hasta As Integer)

- *Parámetros de entrada*: el número de la fila y el número de la columna de la matriz, y hasta que fila de la matriz tiene que buscar.

- *Parámetros de salida*: Integer (una variable entera)

Definición: el modulo de memoria de corto plazo llama a esta función para calcular el numero de motocicletas que fabricara la secuencia vecina.

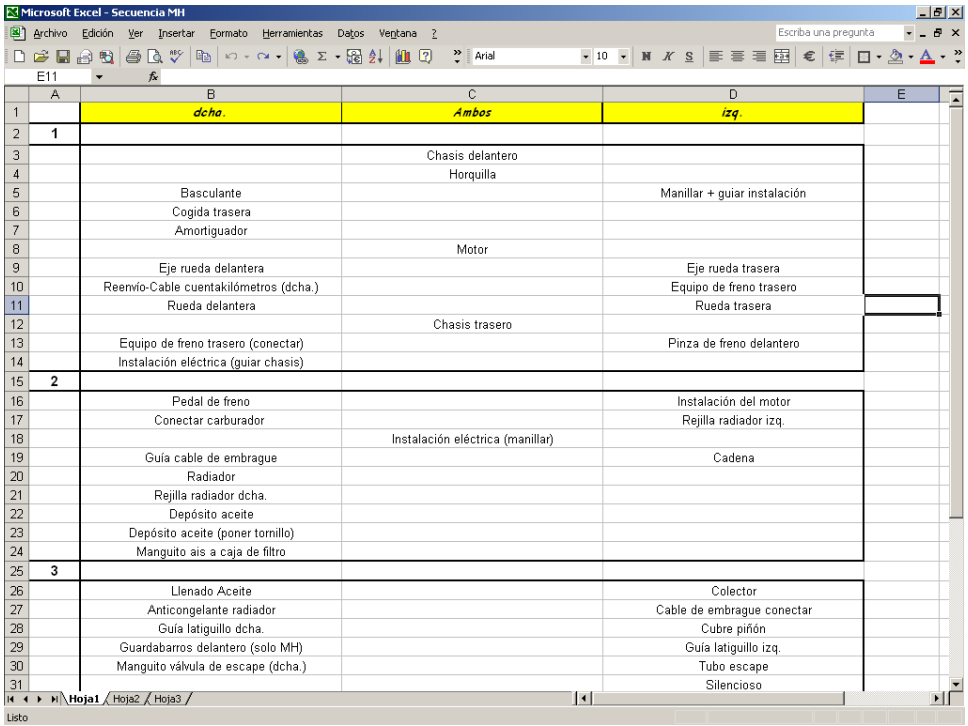
4.3.2 Modo de ejecución

En primer lugar se describe como debe estar construida la hoja Excel, que contiene la secuencia que proporciona el usuario.

Estructura de la hoja Excel:

- La primera fila de la hoja se deja para colocar un encabezado.
- En la primera columna de la hoja se coloca el número de la estación, dejando el resto de la fila vacía. Y a partir de esa fila se comienza a colocar las operaciones de la estación ya enumerada.
- Según el operario que realice la operación se colocara en la segunda (si es operario derecha), tercera (si es una operación conjunta) o cuarta (si es operario de la izquierda) columna.
- Se repite esta estructura para cada estación.

Finalmente la hoja queda de la forma que se muestra en la Figura 39:



	A	B	C	D	E
1		*dcha.*	*Ambos*	*izq.*	
2	1				
3			Chasis delantero		
4			Horquilla		
5		Basculante		Manillar + guiar instalación	
6		Cogida trasera			
7		Amortiguador			
8			Motor		
9		Eje rueda delantera		Eje rueda trasera	
10		Reenvío-Cable cuentakilómetros (dcha.)		Equipo de freno trasero	
11		Rueda delantera		Rueda trasera	
12			Chasis trasero		
13		Equipo de freno trasero (conectar)		Pinza de freno delantero	
14		Instalación eléctrica (guiar chasis)			
15	2				
16		Pedal de freno		Instalación del motor	
17		Conectar carburador		Rejilla radiador izq.	
18			Instalación eléctrica (manillar)		
19		Guía cable de embrague		Cadena	
20		Radiador			
21		Rejilla radiador dcha.			
22		Depósito aceite			
23		Depósito aceite (poner tornillo)			
24		Manguito ais a caja de filtro			
25	3				
26		Llenado Aceite		Colector	
27		Anticongelante radiador		Cable de embrague conectar	
28		Guía latiguillo dcha.		Cubre piñón	
29		Guardabarros delantero (solo MH)		Guía latiguillo izq.	
30		Manguito válvula de escape (dcha.)		Tubo escape	
31				Silencioso	

Figura 39. Estructura de la hoja Excel.

El programa comienza, como ya se había mencionado, con el formulario Guardar secuencia. Al ejecutar el programa aparece dicho formulario, a continuación el usuario señala la ruta de la secuencia que quiere añadir, como se puede observar en la Figura 40 , y le da al botón Añadir secuencia. En unos segundos sale en pantalla una ventana que comunica al usuario que la secuencia ya ha sido guardada, como la se puede observar en la Figura 41. Este proceso se repita tantas veces como secuencia (soluciones factibles) quiera incluir el usuario.

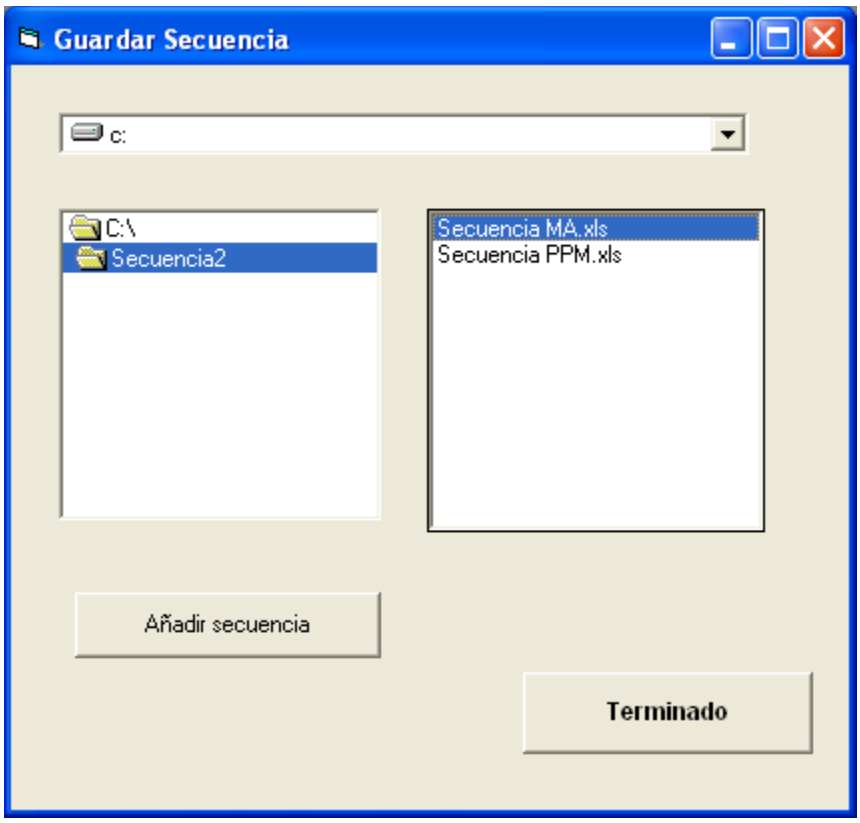


Figura 40. Ruta de la secuencias.

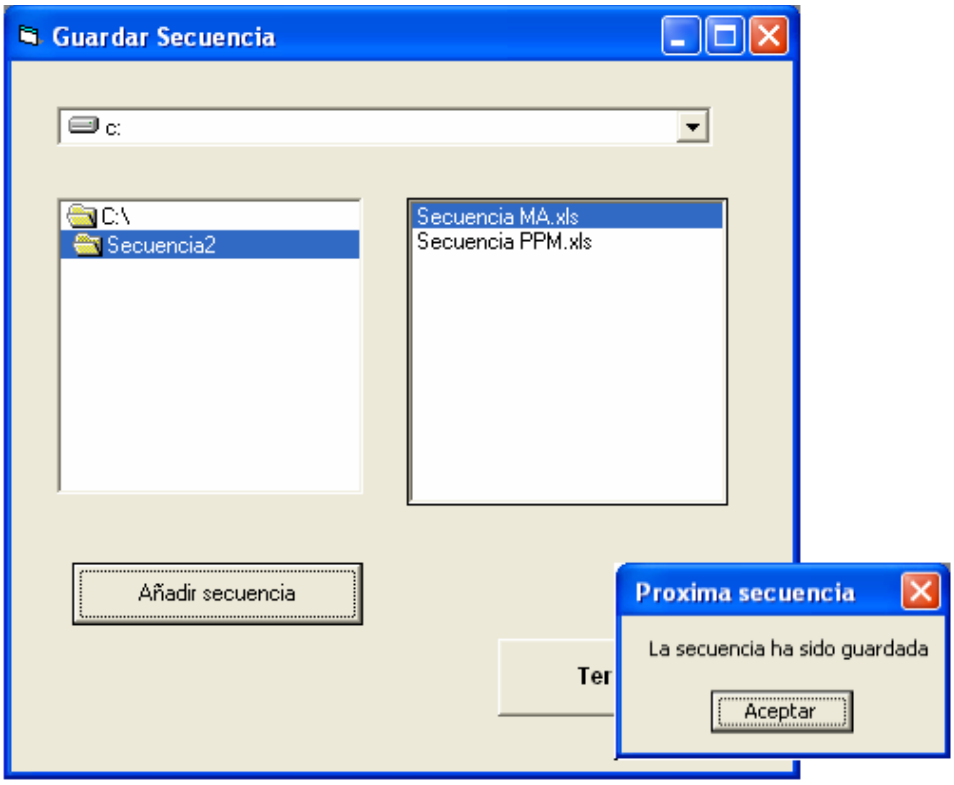


Figura 41. Ventana de mensaje "Secuencia guardada".

El programador en el modo de ejecución es donde comprueba si el programa responde a las funciones programadas. Por tanto, coloca un listbox, para visualizar el tiempo de cada estación de cada secuencia. Se muestra en la Figura 42 los tiempos de cada estación para la secuencia del escenario MA, pero este listbox no será visible para el usuario como se observa en la Figura 41.

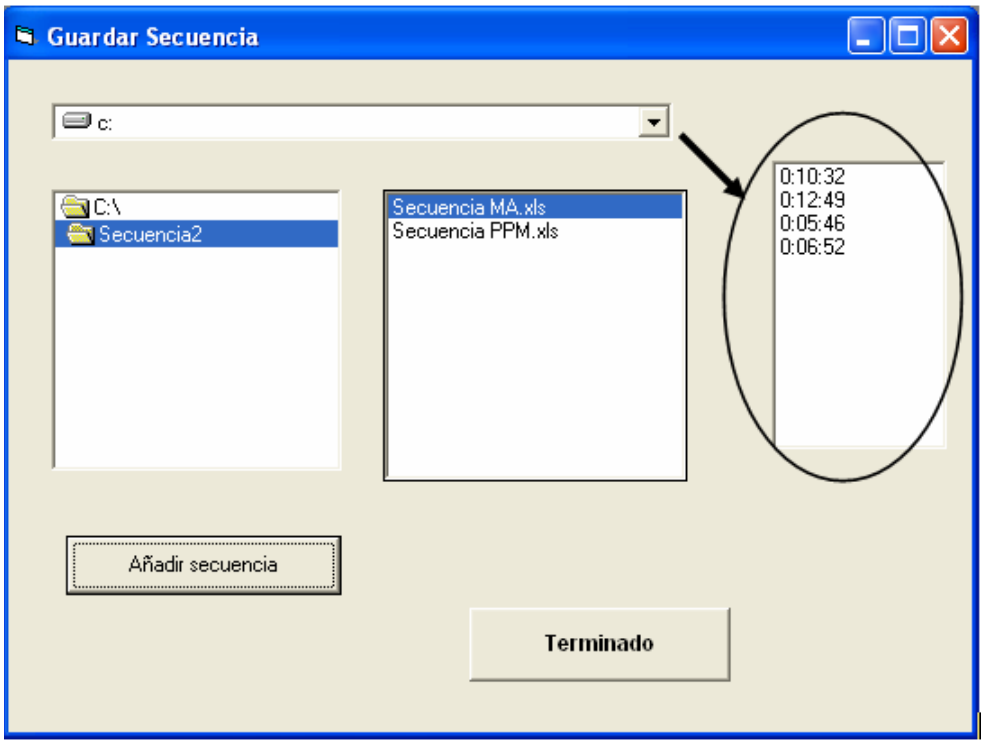


Figura 42. Tiempos de las estaciones.

Cuando se incluye todas las secuencias el usuario pulsa el botón terminar y en pantalla aparece una ventana que indica cual es la mejor secuencia y cual es el número de motocicletas que fábrica en un día. Se muestra a continuación en la Figura 43.

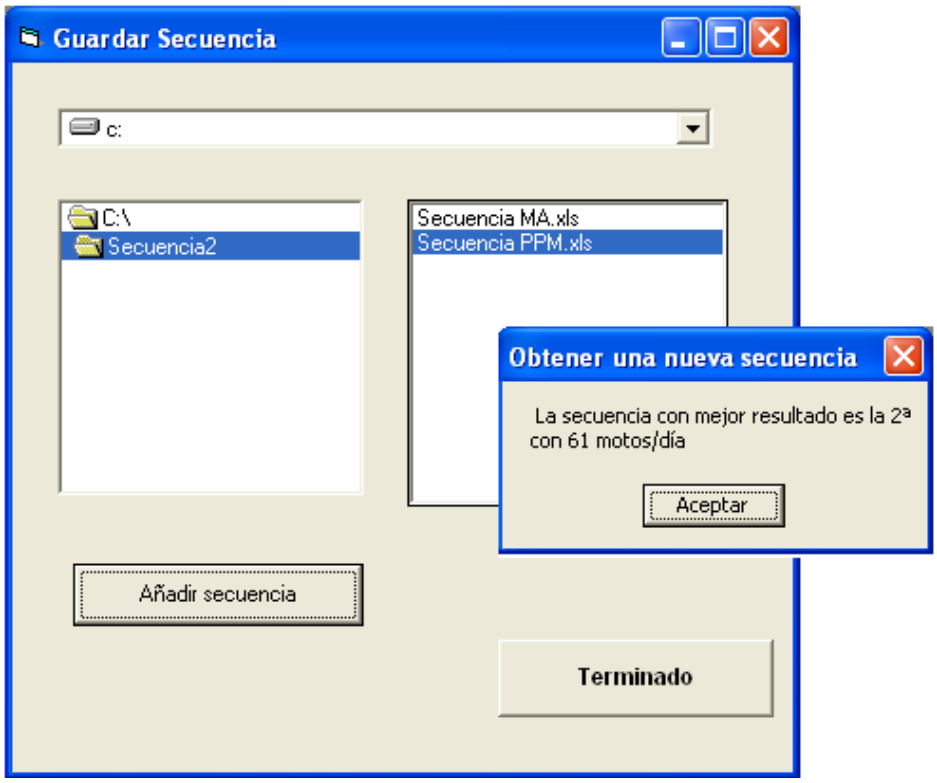


Figura 43. Ventana de mensaje al pulsar terminar.

Cuando el usuario le da a aceptar a la ventana de mensaje aparece el siguiente formulario, Buscar secuencia. En la Figura 44 se puede observar un único botón, buscar secuencia, como ya se había indicado anteriormente. Cuando el operario pulsa, se selecciona una secuencia entre las introducidas, mediante el método de la ruleta, y comienza los movimientos de vecindad en dicha secuencia. El único detalle que le diferencia con el modo de diseño es la retícula del formulario, que ya no aparece.

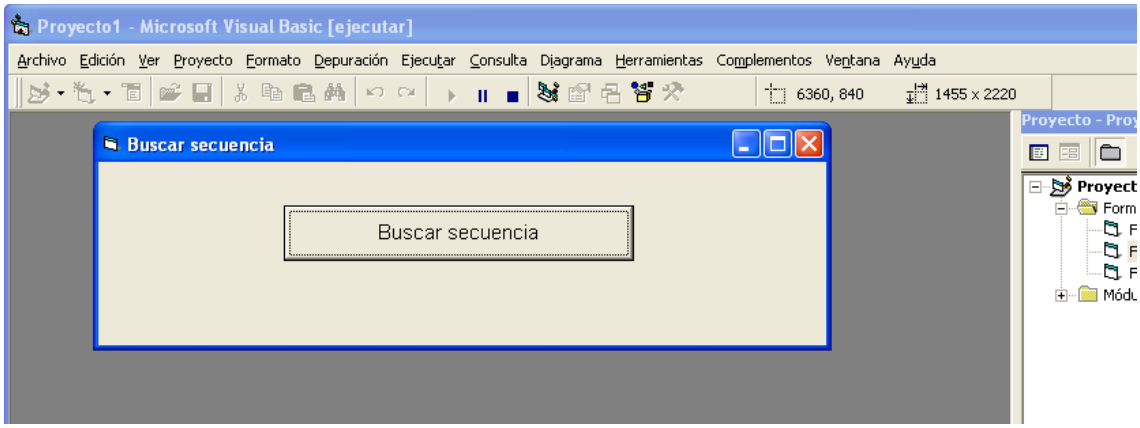


Figura 44. Formulario Buscar secuencia en modo de ejecución.

Si el programa no encuentra una secuencia mejor. Aparece una ventana mensaje que lo indica, como se observa en la Figura 45.

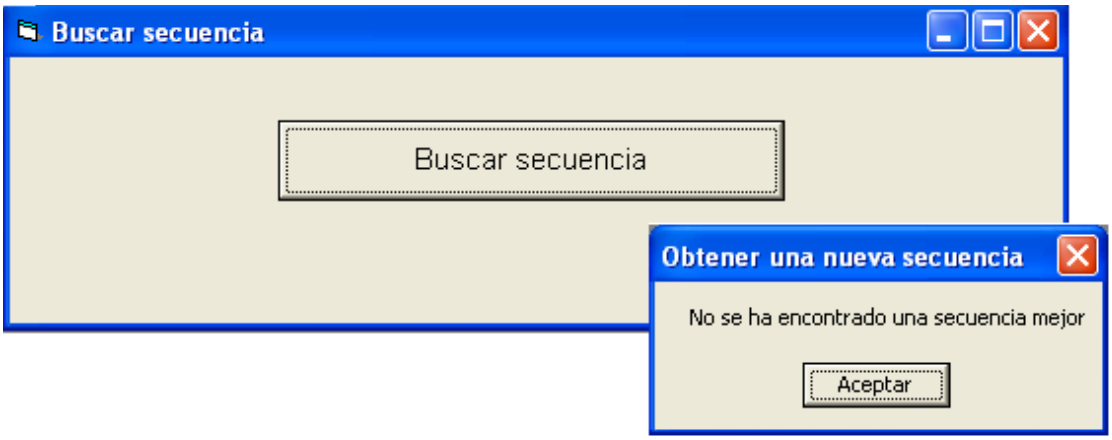


Figura 45. Ventana de mensaje "No se ha encontrado una mejor secuencia"

Si no es así, muestra un mensaje con el número de motocicletas de la nueva secuencia, como en la Figura 46.

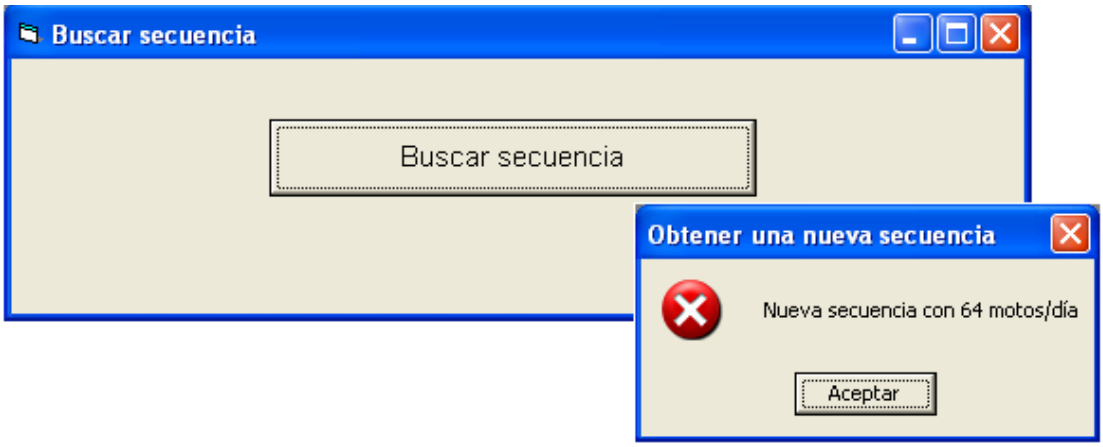


Figura 46. Ventana de mensaje "Nueva secuencia"

Y se pasa al formulario mostrar secuencia y muestra la secuencia encontrada. Se muestra en la Figura 47 una de las soluciones obtenidas a lo largo de la ejecución del programa.

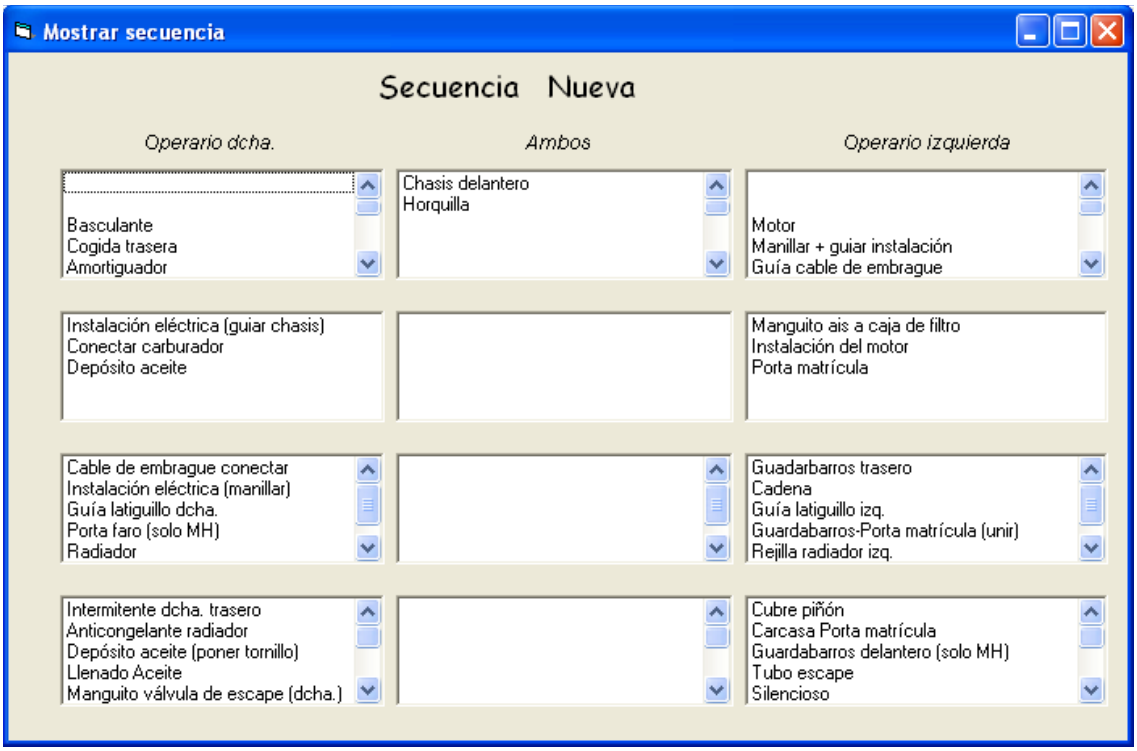


Figura 47. Formulario mostrar secuencia.

En este formulario esta formado, como se observa en la Figura 47, por una matriz de listbox, donde cada fila forma una estación y cada columna el operario que realiza la operación.

4.4 Rendimiento del Algoritmo de Búsqueda en Vecindad (ABV)

Este estudio parte de la secuencia seleccionada por el programa, propuesta PPM. El objetivo perseguido es obtener una mejor secuencia, propuesta por el algoritmo de búsqueda en vecindad (ABV).

Para la búsqueda de la nueva solución se distinguen dos tipos de vecindad: vecindad obtenida por intercambios entre operaciones en toda la línea de montaje y vecindad conseguida por intercambios entre operaciones de la estación cuello botella.

A continuación se expone el estudio de cada vecindad para conocer los resultados que aportan.

Vecindad obtenida por intercambios entre operaciones en toda la línea de montaje.

Para el análisis de esta vecindad se realizan los siguientes pasos:

- Se realizan 60 iteraciones, para probar que funciona correctamente el algoritmo.
- A continuación se realizan 200 iteraciones, donde no se consigue una nueva solución, es decir, no se obtiene una solución mejora.
- Después se aumento a 500 iteraciones, y tampoco se obtuvo una nueva solución.
- Por ultimo se realizo 1000 iteraciones y se obtuvo una solución con 62 motocicletas / día.

A continuación se presenta, en la Figura 48, una grafica con el tiempo de ejecución del programa frente al número de iteraciones realizadas.

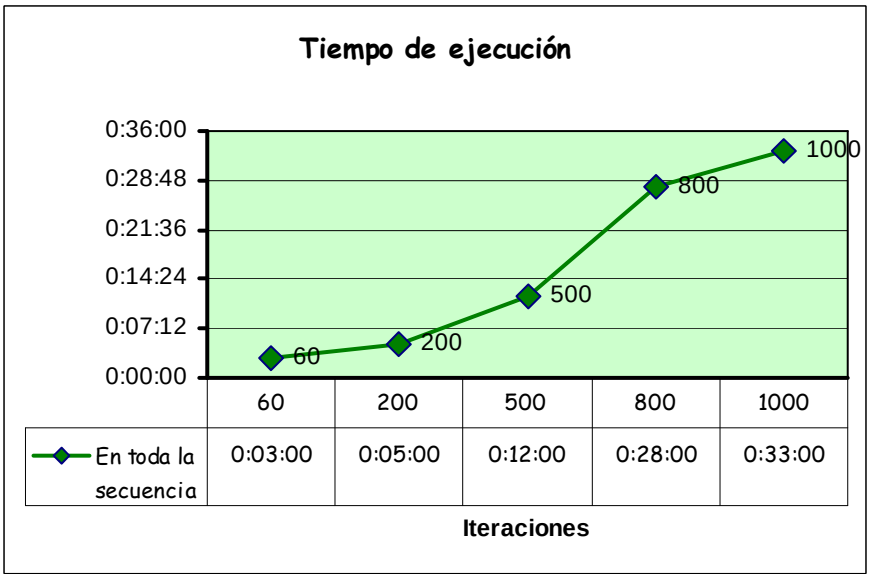


Figura 48. Tiempo de ejecución de ABV en toda la línea de montaje.

En la Figura 48 se observa que el tiempo de ejecución no sigue una función lineal. Esto es razonable, debido a que el algoritmo usado es un proceso aleatorio. Con lo cual todos los intercambios realizados no recorren todo el algoritmo. Sino que mucho serán rechazados por no cumplir el chequeo de admisibilidad.

Después de este estudio, se observa que estas iteraciones no aumenta en gran medida el rendimiento del ABV. Ya que no consigue obtener una nueva solución (secuencia mejorada) hasta las 1000 iteraciones.

Vecindad conseguida por intercambios en la estación cuello botella.

El procedimiento seguido para el análisis de esta vecindad es el siguiente:

- Se realizan 60 iteraciones y no se consiguen una mejora, solución nueva.
- Se vuelve a iterar con 130 y se obtiene una nueva solución que aumenta la tasa de producción a 64 motocicletas / día.
- Después se realiza con 400 y 800 iteraciones alcanzando una nueva solución con una tasa de producción de 66 motocicletas / día.
- Y por ultimo se itera con 1000 iteraciones, logrado de nuevo la misma solución con 66 motocicletas / día.

A continuación se presenta, en la Figura 49, un gráfico con el tiempo de ejecución frente a las iteraciones realizadas dentro de la estación cuello botella:

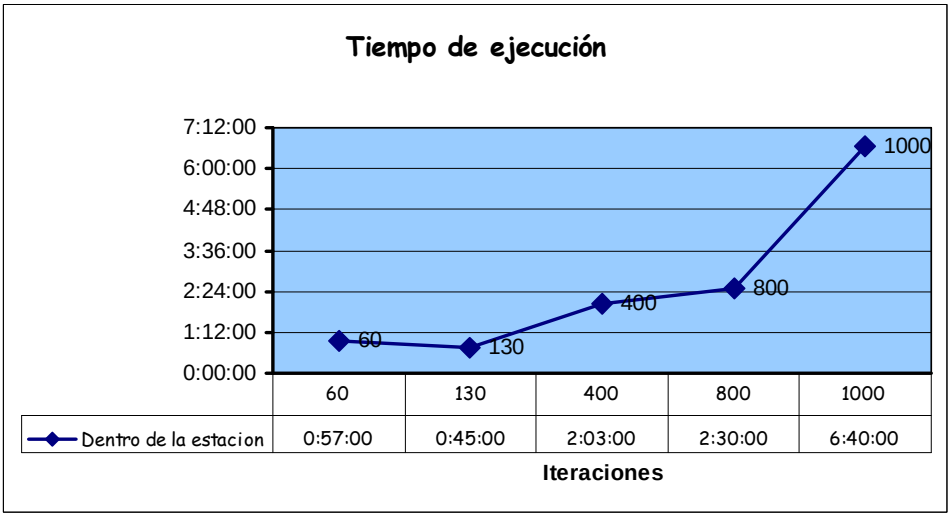


Figura 49. Tiempo de ejecución del ABV dentro de la estación cuello botella.

En la Figura 49, se observa que se obtiene tiempos de ejecución mayores con número de iteraciones menores. Como antes se indico, esto es debido a la aleatoriedad del algoritmo.

Se observa, en la Figura 50, otro gráfico que representa el número de motocicletas al día frente a las iteraciones realizadas. Este gráfico puede hacer suponer que el número optimo de iteraciones esta alrededor de las 800. Pero no se puede olvidar que este algoritmo realiza intercambios de manera aleatoria, esto implica que puede que se realicen 200 iteraciones y se consiga el máximo alcanzado con las 800. Por tanto, se considera realizar unas 1000 iteraciones, para asegurar que se alcance un óptimo.

También ha de mencionarse que el programa presenta la posibilidad de realizar varias búsquedas, cuando no se alcanza ninguna nueva solución.

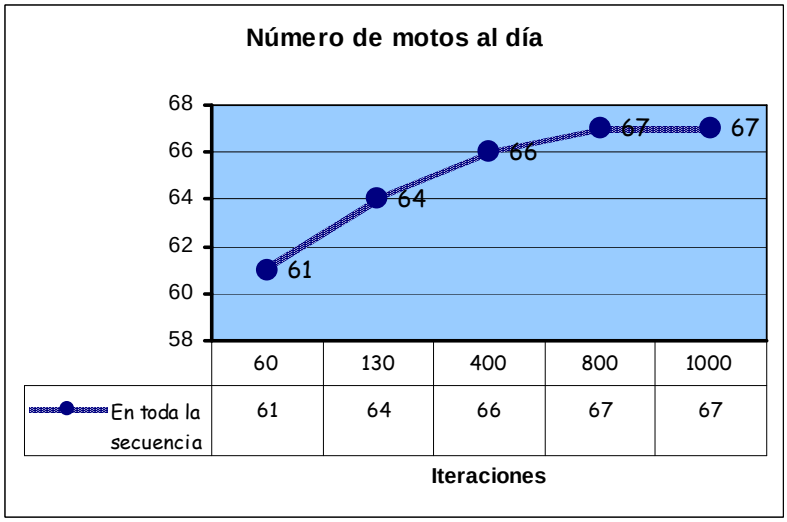


Figura 50. Numero de motocicletas al día.

Con los gráficos de la Figura 49 y la Figura 50 se puede detectar que aunque las transformaciones dentro de las estación cuello botella son más lenta, es más rentable y eficaz dados los resultados obtenidos. Esto implica que las iteraciones realizadas en la vecindad dentro de la estación cuello botella provocan un gran aumento en el rendimiento del ABV.