
**CENTRO DE CIENCIAS BÁSICAS**

**DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN**

**OPTIMIZACIÓN INTELIGENTE**

**5° "A"**

# PRÁCTICA 5: RECOCIDO SIMULADO

**Profesor: Aurora Torres Soto**

**Alumno: Joel Alejandro Espinoza Sánchez**

**Fecha de Entrega:** Aguascalientes, Ags.**, 3** de noviembre de 2020

# Práctica 5: Recocido Simulado

**Objetivo:**

Mediante el desarrollo de esta práctica, se implementará el algoritmo de recocido simulado para la solución del problema TSP del grafo mostrado:

**Introducción:**

El algoritmo de recocido simulado es una metaheurística de búsqueda originalmente diseñada para problemas discretos. La característica clave de esta técnica es que permite escapar de óptimos locales mediante la aceptación de soluciones que no siempre mejoran la función objetivo (energía). Como su nombre lo indica, esta técnica está inspirada en la simulación computacional de un proceso metalúrgico conocido como recocido; que se aplica a los materiales para que estos exhiban mejores características mecánicas.

El recocido simulado nace a partir de una fuerte analogía entre los elementos que constituyen un problema de optimización y el proceso de recocido:

- Las soluciones del problema son equivalentes a los estados del sistema físico.
- El costo de una solución es equivalente a la energía de un estado.
- Un parámetro de control juega el papel de la temperatura

Añadiendo al algoritmo anterior un programa de enfriamiento, el algoritmo es capaz de iniciar en una temperatura muy alta e irla disminuyendo a medida que se avanza en el proceso de optimización.

El algoritmo siguiente representa la versión conocida como monotónica, pues la temperatura siempre va en descenso. En este algoritmo se deben manejar los siguientes parámetros:

- $T_0$: Es la temperatura inicial.
- $\alpha$: Es el factor de enfriamiento.
- $\rho$: Es el factor de reducción de iteraciones.
- $K$: Es el número de iteraciones a cierta $T$.

```
Generar i = i0  // (Solución inicial)
T = T0
Inicializar parámetros // (α, ρ, K, k=0)
Mientras ( No se alcance la condición de PARO) hacer
        Mientras  (k < K)  hacer     // k: contador de iteraciones
                Generar j en N(i)
                Si (E(j) – E(i) < 0)      //Minimización
                        i = j
                Si no
                Generar un número r al azar
                    Si (r < exp [(E(i) – E(j))/T])   //  Minimización
                        i = j
                k = k + 1
        Fin del mientras (interno)
T = αT
K = ρK
k = 0
Si T=Tfin   PARO   //Criterio de paro por temperatura
Fin del mientras (externo)
Mostrar i, E(i)
```

Para hacer uso del recocido simulado se deben establecer:

1) Un punto de partida para la búsqueda.
2) Un mecanismo de perturbación.
3) Un programa de enfriamiento.

**Pregunta de Investigación:**

¿De qué manera se pueden adaptar las ideas del recocido simulado para el cálculo eficiente de un buen tour del TSP?

**Predicción:**

Nuevamente creo que la complejidad de este problema requerirá herramientas de modelado y formas de representación de la solución que nos pueda servir para poder llevar a cabo la solución del TSP y la construcción del recocido simulado.

**Materiales:**

| | |
|---|---|
| Una computadora con compilador de C. | Dos hojas de papel |
| Una calculadora. | Lápiz o plumas |

**Método (Variables):**

Dependiente: El resultado que arroje el programa según el recocido simulado encontrado.
Independiente: El algoritmo que uno como alumno se focalizará en elaborar.
Controlada: El procedimiento del recocido simulado que se usará para evaluar.

**Seguridad:**

Realmente no se trabajó en campo, por lo que no se corren riesgos al elaborar el experimento.

**Procedimiento:**

1.- Haciendo uso del lenguaje de programación C, se realizó un programa que implemente el algoritmo del recocido simulado para la solución del Problema del Agente Viajero (TSP).

2.- El programa exhibe las siguientes características:

- La ciudad de inicio del tour es 1.
- El punto de partida será el tour 1, 2, 3, 4, 5, 6, 7, 1.
- Se usará como mecanismo de perturbación, el subviaje inverso.

3.- El programa se probó con algún grafo realizado previamente bajo este algoritmo. Los resultados se procesaron en el presente documento a continuación.

**Obtención y Procesamiento de Datos:**

Primeramente, se realizó el diseño de algoritmo para tener en cuenta todas las estructuras que se van a usar. Este diseño de algoritmo es el siguiente:

Como variables del programa tenemos:

- `T0`: Es la temperatura inicial.
- `Tfin`: Es la temperatura final.
- `α`: Es el factor de enfriamiento.
- `ρ`: Es el factor de reducción de iteraciones.
- `K`: Es el número de iteraciones a cierta `T`.
- `qv`: Es la cantidad de vértices que tendrá el grafo.
- `random1`: Es un número aleatorio.
- `random2`: Es un número aleatorio.
- `random3`: Es un número aleatorio.

Con las variables anteriores se crean las siguientes estructuras:

- `graph`: Una matriz de adyacencia que muestra el costo de viajar por cada vértice:

$$\begin{array}{c} \backslash \\ \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ \dots \\ v_{qv} \end{bmatrix} \end{array} \begin{array}{c} \begin{bmatrix} v_1 & v_2 & v_3 & \dots & v_{qv} \end{bmatrix} \\ \begin{bmatrix} P(v_1,v_1) & P(v_1,v_2) & P(v_1,v_3) & \dots & P(v_1,v_{qv}) \\ P(v_2,v_1) & P(v_2,v_2) & P(v_2,v_3) & \dots & P(v_2,v_{qv}) \\ P(v_3,v_1) & P(v_3,v_2) & P(v_3,v_3) & \dots & P(v_3,v_{qv}) \\ \dots & \dots & \dots & \dots & \dots \\ P(v_{qv},v_1) & P(v_{qv},v_2) & P(v_{qv},v_3) & \dots & P(v_{qv},v_{qv}) \end{bmatrix} \end{array}$$

Donde $v_1$ es el primer vértice, es decir, nos referimos al vértice número $n$ denotándolo como $v_n$. También $P(v_n, v_m)$ es el costo del viaje de ir del vértice $n$ al $m$. Si los costos son simétricos, observaremos que $P(v_n, v_m) = P(v_m, v_n)$.

- `tour`: Un vector de dimensión $qv$. Este vector representa el viaje el cual será el punto de partida para realizar la evaluación:

  $$[t_1 \quad t_2 \quad t_3 \quad \dots \quad t_{qv}]$$

  Donde $t_1$ representa el primer vértice del tour, es decir, del que se parte, $t_2$ es el segundo vértice del tour, $t_n$ es el $n$ vértice del tour. Claro es que $t_n$ debe tener una conexión válida con $t_{n+1}$ y en el caso de $t_{qv}$ tenerla con $t_1$.

  A este vector tendremos asociada la variable `ftour` que evaluará el costo total del tour.
- `newTour`: Un vector igual a `tour`. Tendrá asociado también la variable `fnewTour` que evaluará el costo total del tour.

El proceso del algoritmo es el siguiente

1) Se calibra el programa y se insertan los datos debidos.
2) Se evalúa el peso de `tour` y su resultado se guarda en `ftour`.
3) Se calcula el número aleatorio `random1` en el intervalo $[2, qv - 1]$.
4) Se calcula el número aleatorio `random2` en el intervalo $[random1 + 1, qv]$.
5) Construimos la perturbación de tour modificando el `tour` de forma que se invertirán los elementos desde `random1` hasta `random2` de la lista y se guarda en `newTour`.
6) Se verifica si el tour de `newTour` es válido. Si es válido, continuamos, si no es válido, volvemos al paso 3.
7) Se evalúa el peso de `newTour` y su resultado se guarda en `fnewTour`.
8) Para minimizar, si `fnewTour` $<$ `fTour`, entonces ahora, `newTour` se convierte en `tour` y también `fnewTour` se convierte en `ftour`.

   Si `fnewTour` $>$ `fTour`, entonces calculamos el número aleatorio `random3` entre 0 y 1. Ahora, si `random3` $< e^{\frac{fnewTour - ftour}{T}}$ entonces ahora, `newTour` se convierte en `tour` y también `fnewTour` se convierte en `ftour`. De lo contrario, rechazamos `newTour` y dejamos igual `tour`.
9) Repetimos del paso 2 al paso 8 K veces.
10) Multiplicamos $\alpha T$ para obtener una nueva $T$.
11) Multiplicamos $\rho K$ para obtener una nueva $K$.
12) Regresamos a $k = 0$.

Repetimos del paso 2 al paso 12 mientras $T > T_{fin}$.

Este mismo algoritmo se introdujo en el código en C (véase anexo 1)

El mismo programa permite al usuario modificar muchas variables y personalizar incluso la forma en la que se desea ver el procedimiento. Por ejemplo, se puede establecer el modo de acción a cero para ir paso a paso o el modo de acción a uno para realizarlo automáticamente.

Lo primero que pide el programa es calibrar las primeras variables (véase anexo 2 para ver evidencias completas de la ejecución de código):

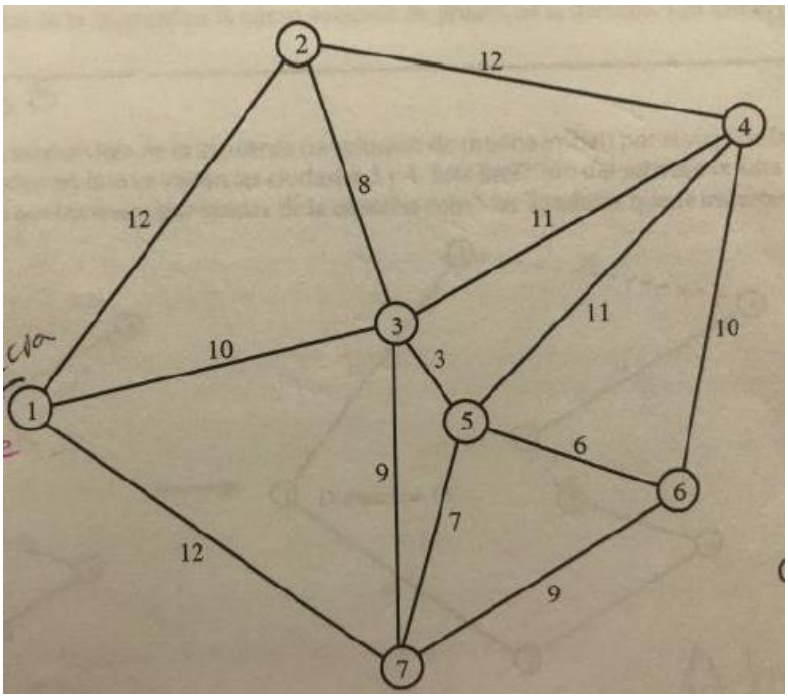
```
============================== RECOCIDO SIMULADO ==============================

--------------------------------
|  Calibración del algoritmo:
|  Seleccione algún número si desea cambiar su valor (o 0 para continuar):
|  0: Listo. Continuar
|  1: T0 (Temperatura inicial): 140.0000
|  2: Tfin (Temperatura final): 10.0000
|  3: alpha (Factor de enfriamiento): 0.5000
|  4: rho (Factor de reducción de iteraciones): 0.5000
|  5: K (Número de iteraciones en T): 30
|  6: qv (Cantidad de vértices que posee el grafo): 7
|  7: autom (Modo de acción): 1
|
```

Una vez que se fijen estas primeras variables, lo siguientes serán las estructuras que se construyen a partir de estas variables, como las matrices y tours que se necesitan dentro de la cantidad de grafos.

```
--------------------------------
|  Calibración del algoritmo:
|  Seleccione algún número si desea cambiar su valor (o 0 para continuar):
|  0: Listo. Continuar
|  1: graph (El grafo a evaluar):
|     [30] [0] [4202664] [0] [4219430] [0] [6487112]
|     [0] [-202942144] [32765] [6479984] [0] [-202966480] [32765]
|     [0] [1071644672] [0] [0] [208] [0] [0]
|     [0] [4200264] [0] [1056964608] [0] [1] [0]
|     [30] [0] [7] [0] [1056964608] [7602296] [1124859904]
|     [4259932] [1092616192] [6881384] [6487392] [0] [6487408] [0]
|     [6487404] [0] [6487400] [0] [6487388] [0] [6487384]
|  2: tour (El viaje inicial):
|     [0] [0] [-203255635] [32765] [30] [0] [204] [0]
|
```

Al principio es normal que la matriz y el tour no tengan sentido, pues no se han ingresado valores a ninguna de las dos estructuras, pero una vez se inicialicen ambas, se verá con sentido el menú anterior.

Se nos pidió analizar el siguiente grafo:




En el programa construido, se puede agregar todo el programa accediendo a la opción de modificar el grafo y escribir la línea siguiente:

```
0 12 10 0 0 0 12 12 0 8 12 0 0 0 10 8 0 11 3 0 9 0 12 11 0 11 10 0
          0 0 3 11 0 6 7 0 0 0 10 6 0 9 12 0 9 0 7 9 0
```

La anterior es la matriz de adyacencia número por número para especificar cada arista.

De la misma forma, se puede agregar el punto de partida accediendo a la opción de modificar el punto de partida y escribir la línea siguiente:

```
1 2 3 4 5 6 7 1
```

De esta forma, el menú ahora cobra sentido:

```
--------------------------------
|  Calibración del algoritmo:
|  Seleccione algún número si desea cambiar su valor (o 0 para continuar):
|  0: Listo. Continuar
|  1: graph (El grafo a evaluar):
|     [0] [12] [10] [0] [0] [0] [12]
|     [12] [0] [8] [12] [0] [0] [0]
|     [10] [8] [0] [11] [3] [0] [9]
|     [0] [12] [11] [0] [11] [10] [0]
|     [0] [0] [3] [11] [0] [6] [7]
|     [0] [0] [0] [10] [6] [0] [9]
|     [12] [0] [9] [0] [7] [9] [0]
|  2: tour (El viaje inicial):
|     [1] [2] [3] [4] [5] [6] [7] [1]
|
```

Ahora el programa está configurado y listo para ejecutarse, si previamente se calibró el modo de acción en 0, entonces el programa esperará la tecla Enter para seguir adelante. Este modo está pensado para pocas iteraciones y un poco más para el momento en el que se estaba desarrollando y verificar que estaba haciéndolo correctamente

Si el programa se ejecuta en el modo de acción en 1, sólo queda dar un Enter y el proceso estará completamente reportado en la consola:

```
Evaluamos con los valores:
    T = 35.0000
    K = 7

  El tour [1, 2, 3, 4, 6, 5, 7, 1] genera un costo de: 66
  El nuevo tour [1, 2, 3, 4, 5, 6, 7, 1] genera un costo de: 69
    No hemos escogido aún (fnewTour >= ftour)
      Hemos descartado newTour por probabilidad de Boltzmann (r3 = 0.9200 >= 0.9179)

  El tour [1, 2, 3, 4, 6, 5, 7, 1] genera un costo de: 66
  El nuevo tour [1, 2, 3, 4, 5, 6, 7, 1] genera un costo de: 69
    No hemos escogido aún (fnewTour >= ftour)
      Hemos escogido newTour por probabilidad de Boltzmann (r3 = 0.6300 < 0.9179)

  El tour [1, 2, 3, 4, 5, 6, 7, 1] genera un costo de: 69
  El nuevo tour [1, 2, 3, 4, 6, 5, 7, 1] genera un costo de: 66
    Hemos escogido newTour (fnewTour < ftour)

  El tour [1, 2, 3, 4, 6, 5, 7, 1] genera un costo de: 66
  El nuevo tour [1, 2, 3, 4, 5, 6, 7, 1] genera un costo de: 69
    No hemos escogido aún (fnewTour >= ftour)
      Hemos escogido newTour por probabilidad de Boltzmann (r3 = 0.5000 < 0.9179)

  El tour [1, 2, 3, 4, 5, 6, 7, 1] genera un costo de: 69
  El nuevo tour [1, 2, 3, 4, 6, 5, 7, 1] genera un costo de: 66
    Hemos escogido newTour (fnewTour < ftour)

  El tour [1, 2, 3, 4, 6, 5, 7, 1] genera un costo de: 66
  El nuevo tour [1, 7, 5, 6, 4, 3, 2, 1] genera un costo de: 66
    No hemos escogido aún (fnewTour >= ftour)
      Hemos escogido newTour por probabilidad de Boltzmann (r3 = 0.1900 < 1.0000)

  El tour [1, 7, 5, 6, 4, 3, 2, 1] genera un costo de: 66
  El nuevo tour [1, 7, 5, 6, 4, 2, 3, 1] genera un costo de: 65
    Hemos escogido newTour (fnewTour < ftour)
```

```
Evaluamos con los valores:
    T = 17.5000
    K = 3

  El tour [1, 7, 5, 6, 4, 2, 3, 1] genera un costo de: 65
  El nuevo tour [1, 7, 5, 6, 4, 3, 2, 1] genera un costo de: 66
    No hemos escogido aún (fnewTour >= ftour)
      Hemos escogido newTour por probabilidad de Boltzmann (r3 = 0.8400 < 0.9445)

  El tour [1, 7, 5, 6, 4, 3, 2, 1] genera un costo de: 66
  El nuevo tour [1, 2, 3, 4, 6, 5, 7, 1] genera un costo de: 66
    No hemos escogido aún (fnewTour >= ftour)
      Hemos escogido newTour por probabilidad de Boltzmann (r3 = 0.5600 < 1.0000)

  El tour [1, 2, 3, 4, 6, 5, 7, 1] genera un costo de: 66
  El nuevo tour [1, 2, 3, 4, 5, 6, 7, 1] genera un costo de: 69
    No hemos escogido aún (fnewTour >= ftour)
      Hemos escogido newTour por probabilidad de Boltzmann (r3 = 0.6600 < 0.8425)
```

Al final también se reporta el mejor viaje encontrado a partir de este análisis:

```
El tour final fue [1, 3, 7, 6, 5, 4, 2, 1] que genera un costo de: 69
```

Observemos las siguientes configuraciones:

```
--------------------------------
|  Calibración del algoritmo:
|  Seleccione algún número si desea cambiar su valor (o 0 para continuar):
|  0: Listo. Continuar
|  1: T0 (Temperatura inicial): 200.0000
|  2: Tfin (Temperatura final): 3.0000
|  3: alpha (Factor de enfriamiento): 0.9000
|  4: rho (Factor de reducción de iteraciones): 0.9500
|  5: K (Número de iteraciones en T): 100
|  6: qv (Cantidad de vértices que posee el grafo): 7
|  7: autom (Modo de acción): 1
|  0
```

Con las anteriores, se obtuvo el resultado final que se muestra a continuación:

```
Evaluamos con los valores:
    T = 3.2846
    K = 6

  El tour [1, 3, 2, 4, 5, 6, 7, 1] genera un costo de: 68
  El nuevo tour [1, 3, 2, 4, 6, 5, 7, 1] genera un costo de: 65
    Hemos escogido newTour (fnewTour < ftour)

  El tour [1, 3, 2, 4, 6, 5, 7, 1] genera un costo de: 65
  El nuevo tour [1, 3, 2, 4, 5, 6, 7, 1] genera un costo de: 68
    No hemos escogido aún (fnewTour >= ftour)
      Hemos descartado newTour por probabilidad de Boltzmann (r3 = 0.5600 >= 0.4012)

  El tour [1, 3, 2, 4, 6, 5, 7, 1] genera un costo de: 65
  El nuevo tour [1, 3, 2, 4, 5, 6, 7, 1] genera un costo de: 68
    No hemos escogido aún (fnewTour >= ftour)
      Hemos escogido newTour por probabilidad de Boltzmann (r3 = 0.2600 < 0.4012)

  El tour [1, 3, 2, 4, 5, 6, 7, 1] genera un costo de: 68
  El nuevo tour [1, 3, 7, 6, 5, 4, 2, 1] genera un costo de: 69
    No hemos escogido aún (fnewTour >= ftour)
      Hemos escogido newTour por probabilidad de Boltzmann (r3 = 0.2300 < 0.7375)

  El tour [1, 3, 7, 6, 5, 4, 2, 1] genera un costo de: 69
  El nuevo tour [1, 3, 2, 4, 5, 6, 7, 1] genera un costo de: 68
    Hemos escogido newTour (fnewTour < ftour)

  El tour [1, 3, 2, 4, 5, 6, 7, 1] genera un costo de: 68
  El nuevo tour [1, 3, 2, 4, 6, 5, 7, 1] genera un costo de: 65
    Hemos escogido newTour (fnewTour < ftour)



El tour final fue [1, 3, 2, 4, 6, 5, 7, 1] que genera un costo de: 65
```

## Conclusiones:

Considero muy importante haber realizado primeramente el modelado que se plantea aquí previa a la programación del algoritmo, pues esto nos permite visualizar de mejor manera el algoritmo antes de empezar a realizar las estructuras en el programa. Este procedimiento realizado durante el algoritmo genético y el recocido simulado nos permite programar algoritmos de mayor dificultad sin batallar mucho.

Así también podemos concluir que el Recocido Simulado es un algoritmo muy útil para optimizar problemas encontrando buenas soluciones, aunque puede que en esta ocasión no sea tan fácil de ver, puesto que realmente su optimización encuentra buenas soluciones muy cercanas entre ellas mismas.

## Referencias:


Purcell, E. (2007). *Cálculo.* Londres: Pearson Education.

Talbi, E. (2009). *Metaheuristics from design to implementation.* New Jersey: John Wiley & Sons Publication.

Torres, A. (2020). *Apuntes: Optimización Inteligente.* 5° ICI. México: Universidad Autónoma de Aguascalientes.

**Anexos:**

## Anexo 1: Código del programa en lenguaje C:

```
/*
    Universidad Autónoma de Aguascalientes

            Centro de Ciencias Básicas
  Departamento de Ciencias de la Computación
            Optimización Inteligente

                        5° "A"

          Práctica 5: Recocido Simulado

             Doctora Aurora Torres Soto

   Alumno: Joel Alejandro Espinoza Sánchez

  Fecha de Entrega: 3 de noviembre del 2020

Descripción:
*/
//Cargamos las librerías
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>
#include <time.h>
#include <math.h>

void setValues1(float alpha, int autom, int K, int qv, float rho,
float T, float Tfin, float *alphaP, int *automP, int *KP, int
*qvP, float *rhoP, float *TP, float *TfinP);
void setValues2(int qv, int graph[qv][qv], int tour[qv + 1]);
int checkTour(int qv, int graph[qv][qv],int tour[qv + 1]);
void evaluateTour(int qv, int graph[qv][qv],int tour[qv + 1], int
ftour, int *ftourP);
void backToTour(int qv, int tour[qv + 1],int newTour[qv + 1]);

main()
{
     setlocale(LC_ALL,"");
     srand(time(NULL));
```

```
	//1. Declaramos las variables que usaremos
	/*
		alpha: Factor de enfriamiento
		autom: Modo de acción, donde 0 será Manual y 1 
Automático, que:
			Si autom = 0 ejecutará todo el proceso 
rápidamente
			Si autom = 1 ejecutará el proceso de uno en uno
		k: Iterador sobre K
		K: Número de iteraciones a cierta T
		qv: Cantidad de vértices
		r1: Un número aleatorio
		r2: Un número aleatorio
		r3: Un número aleatorio
		rho: Factor de reducción de iteraciones
		T: Temperatura actual
		Tfin: Temperatura final
	*/
	int autom, check, i, j, k, K, qv, r1, r2, r3a, repeat;
	float alpha, r3, rho, T, Tfin;

	printf("============================== RECOCIDO SIMULADO 
==============================\n");

	do
	{
		alpha = 0.5;
		autom = 1;
		k = 0;
		K = 30;
		qv = 7;
		rho = 0.5;
		repeat = 0;
		T = 140;
		Tfin = 10;

		//2. Calibramos los primeros valores del programa

	setValues1(alpha,autom,K,qv,rho,T,Tfin,&alpha,&autom,&K,&qv,&
rho,&T,&Tfin);
```

```
          //3. Declaramos más variables
          /*
               graph: La matriz de adyacencia del grafo que
analizaremos
               tour: Un vector de dimensión qv que representa el
viaje i
               ftour: La energío de tour
               newTour: Un vector igual a tour que representa el
viaje J
               fnewTour: La energío de newTour
          */
          int graph[qv][qv], tour[qv + 1], ftour, newTour[qv + 1],
fnewTour;
          setValues2(qv,graph,tour);
          getchar();

          //4. Comenzamos el Recocido
          do
          {
               printf("\n");
               printf("\n");
               printf("Evaluamos con los valores:\n");
               printf("    T = %.4f\n",T);
               printf("    K = %.d\n",K);
               do
               {
                    //5. Evaluamos el costo de tour
                    evaluateTour(qv,graph,tour,ftour,&ftour);

                    printf("El tour [%d",tour[0]);
                    for(i = 1; i <= qv; i++)
                    {
                         printf(", %d",tour[i]);
                    }
                    printf("] genera un costo de: %d\n",ftour);
                    if(autom == 0)
                    {
                         getchar();
                    }

                    //6. Perturbaremos tour
                    do
```

```
                    {
                         r1 = 2 + (rand() % ((qv - 1) - 2 + 1));
                         r2 = (r1 + 1) + (rand() % (qv - (r1 + 1)
+ 1));

                         //Construimos newTour con un subviaje
inverso aleatorio de tour
                         for(i = 0; i < r1; i++)
                         {
                              newTour[i] = tour[i];
                         }

                         j = 0;
                         for(i = r1 - 1; i < r2; i++)
                         {
                              newTour[i] = tour[r2 - 1 - j];
                              j++;
                         }

                         for(i = r2; i < qv + 1; i++)
                         {
                              newTour[i] = tour[i];
                         }

                         //No pasaremos a la siguiente fase hasta
verificar que newTour sea un tour válido
                         check = checkTour(qv,graph,newTour);
                    }
                    while(check == 0);

                    //7. Evaluamos newTour

     evaluateTour(qv,graph,newTour,fnewTour,&fnewTour);

//                  printf("r1 = %d\n",r1);
//                  printf("r2 = %d\n",r2);
                    printf("El nuevo tour [%d",newTour[0]);
                    for(i = 1; i <= qv; i++)
                    {
                         printf(", %d",newTour[i]);
                    }
```

```
                        printf("] genera un costo de:
%d\n",fnewTour);
                        if(autom == 0)
                        {
                              getchar();
                        }

                        //8. Evaluamos para minimizar (Para maximizar
cambiamos el signo del condicional < por >)
                        if(fnewTour < ftour)
                        {
                              printf("Hemos escogido newTour (fnewTour
< ftour)\n");
                              if(autom == 0)
                              {
                                    getchar();
                              }
                              backToTour(qv,tour,newTour);
                              ftour = fnewTour;
                        }
                        else
                        {
                              printf("No hemos escogido aún (fnewTour
>= ftour)\n");
                              if(autom == 0)
                              {
                                    getchar();
                              }

                              r3a = rand() % 100;
                              r3 = ((float)r3a)/100;
                              if(r3 < (exp((ftour - fnewTour)/T)))
                              {
                                    printf("Hemos escogido newTour por
probabilidad de Boltzmann (r3 = %.4f < %.4f)\n",r3,exp((ftour -
fnewTour)/T));
                                    if(autom == 0)
                                    {
                                          getchar();
                                    }

                                    backToTour(qv,tour,newTour);
```

```
                                    ftour = fnewTour;
                               }
                               else
                               {
                                    printf("Hemos descartado newTour
por probabilidad de Boltzmann (r3 = %.4f >= %.4f)\n",r3,exp((ftour
- fnewTour)/T));
                                    if(autom == 0)
                                    {
                                         getchar();
                                    }
                               }
                          }

                          k++;
                     }
                     while(k < K);

                     T = alpha*T;
                     K = rho*K;
                     k = 0;
               }
               while(T > Tfin);

               printf("\n");
               printf("\n");
               printf("\n");
               printf("El tour final fue [%d",tour[0]);
               for(i = 1; i <= qv; i++)
               {
                    printf(", %d",tour[i]);
               }
               printf("] que genera un costo de: %d\n",ftour);

               printf("\n");
               printf("\n");
               printf("\n");
               printf("¿Desea repetir el código?\n");
               printf("0. No\n");
               printf("1. Sí\n");
               scanf("%d",&repeat);
          }
```

```
	while(repeat == 1);

	getchar();
}

void setValues1(float alpha, int autom, int K, int qv, float rho,
float T, float Tfin, float *alphaP, int *automP, int *KP, int
*qvP, float *rhoP, float *TP, float *TfinP)
{
	int aux, done = 0;
	*alphaP = alpha;
	*automP = autom;
	*KP = K;
	*qvP = qv;
	*rhoP = rho;
	*TP = T;
	*TfinP = Tfin;

	do
	{
		printf("\n");
		printf("--------------------------------\n");
		printf("|  Calibración del algoritmo:\n");
		printf("|  Seleccione algún número si desea cambiar su
valor (o 0 para continuar):\n");
		printf("|  0: Listo. Continuar\n");
		printf("|  1: T0 (Temperatura inicial): %.4f\n",T);
		printf("|  2: Tfin (Temperatura final): %.4f\n",Tfin);
		printf("|  3: alpha (Factor de enfriamiento):
%.4f\n",alpha);
		printf("|  4: rho (Factor de reducción de iteraciones):
%.4f\n",rho);
		printf("|  5: K (Número de iteraciones en T): %d\n",K);
		printf("|  6: qv (Cantidad de vértices que posee el
grafo): %d\n",qv);
		printf("|  7: autom (Modo de acción): %d\n",autom);
		printf("|  ");
		scanf("%d",&aux);

		switch (aux)
		{
			case 0:
```

```
                    {
                         //Se continúa con el programa
                         done = 1;
                         break;
                    }
               case 1:
                    {
                         //Se modifica T

                         printf("|     Inserte un nuevo valor
para T0 (Antiguo valor para T0: T0 = %.4f)\n",T);
                         printf("|  ");
                         scanf("%f",&T);
                         *TP = T;

                         break;
                    }
               case 2:
                    {
                         //Se modifica Tfin

                         printf("|     Inserte un nuevo valor
para Tfin (Antiguo valor para Tfin: Tfin = %.4f)\n",Tfin);
                         printf("|  ");
                         scanf("%f",&Tfin);
                         *TfinP = Tfin;

                         break;
                    }
               case 3:
                    {
                         //Se modifica alpha

                         printf("|     Inserte un nuevo valor
para alpha (Antiguo valor para alpha: alpha = %.4f)\n",alpha);
                         printf("|  ");
                         scanf("%f",&alpha);
                         *alphaP = alpha;

                         break;
                    }
               case 4:
```

```
                    {
                         //Se modifica rho

                         printf("|     Inserte un nuevo valor
para rho (Antiguo valor para rho: rho = %.4f)\n",rho);
                         printf("|  ");
                         scanf("%f",&rho);
                         *rhoP = rho;

                         break;
                    }
               case 5:
                    {
                         //Se modifica K

                         printf("|     Inserte un nuevo valor
para K (Antiguo valor para K: K = %d)\n",K);
                         printf("|  ");
                         scanf("%d",&K);
                         *KP = K;

                         break;
                    }
               case 6:
                    {
                         //Se modifica qv

                         printf("|     Inserte un nuevo valor
para qv (Antiguo valor para qv: qv = %d)\n",qv);
                         printf("|  ");
                         scanf("%d",&qv);
                         *qvP = qv;

                         break;
                    }
               case 7:
                    {
                         //Se modifica autom

                         printf("|     Inserte un nuevo valor
para autom (Antiguo valor para autom: autom = %d)\n",autom);
                         printf("|  ");
```

```
                        scanf("%d",&autom);
                        *automP = autom;

                        break;
                    }
                default:
                    {
                        //Valor no válido

                        printf("|     Ha insertado un número
inválido\n");

                        break;
                    }
            }
        }
        while(done == 0);

        return;
}

void setValues2(int qv, int graph[qv][qv], int tour[qv + 1])
{
        int aux, done = 0,i,j;
        do
        {
            printf("\n");
            printf("--------------------------------\n");
            printf("|  Calibración del algoritmo:\n");
            printf("|  Seleccione algún número si desea cambiar su
valor (o 0 para continuar):\n");
            printf("|  0: Listo. Continuar\n");
            printf("|  1: graph (El grafo a evaluar):\n");
            for(i = 0; i < qv; i++)
            {
                printf("|     ");
                for(j = 0; j < qv; j++)
                {
                    printf("[%d] ",graph[i][j]);
                }
                printf("\n");
            }
```

```
            printf("|  2: tour (El viaje inicial):\n");
            printf("|     ");
            for(i = 0; i <= qv; i++)
            {
                  printf("[%d] ",tour[i]);
            }
            printf("\n");
            printf("|  ");
            scanf("%d",&aux);

            switch (aux)
            {
                  case 0:
                        {
                              //Se continúa con el programa
                              done = 1;
                              break;
                        }
                  case 1:
                        {
                              //Se modifica graph

                              for(i = 0; i < qv; i++)
                              {
                                    for(j = 0; j < qv; j++)
                                    {
                                          printf("|     Inserte el peso
de la arista que une al vértice %d con el vértice %d (0 para
indicar que no existe tal unión)\n",i + 1, j + 1);
                                          printf("|  ");
                                          scanf("%d",&graph[i][j]);
                                    }
                              }

                              break;
                        }
                  case 2:
                        {
                              //Se modifica tour

                              for(i = 0; i <= qv; i++)
                              {
```

```
                                    printf("|     Inserte el vértice
que se visitará en la parada %d\n",i);
                                    printf("|  ");
                                    scanf("%d",&tour[i]);
                               }

                               break;
                          }
                     default:
                          {
                               //Valor no válido

                               printf("|     Ha insertado un número
inválido\n");

                               break;
                          }
               }
          }
          while(done == 0);

          return;
}

int checkTour(int qv, int graph[qv][qv],int tour[qv + 1])
{
          int i;
          for(i = 0; i < qv; i++)
          {
               if(graph[tour[i] - 1][tour[i + 1] - 1] == 0)
               {
                    return 0;
               }
          }

          return 1;
}

void evaluateTour(int qv, int graph[qv][qv],int tour[qv + 1], int
ftour, int *ftourP)
{
          int i;
```

```
	*ftourP = ftour;

	ftour = 0;
	for(i = 0; i < qv; i++)
	{
		ftour = ftour + graph[tour[i] - 1][tour[i + 1] - 1];
	}

	*ftourP = ftour;
	return;
}

void backToTour(int qv, int tour[qv + 1],int newTour[qv + 1])
{
	int i;
	for(i = 0; i <=qv; i++)
	{
		tour[i] = newTour[i];
	}
	return;
}
```

Anexo 2: Impresión de pantalla completa al ejecutar el programa

```
============================== RECOCIDO SIMULADO ==============================

--------------------------------
|  Calibración del algoritmo:
|  Seleccione algún número si desea cambiar su valor (o 0 para continuar):
|  0: Listo. Continuar
|  1: T0 (Temperatura inicial): 140.0000
|  2: Tfin (Temperatura final): 10.0000
|  3: alpha (Factor de enfriamiento): 0.5000
|  4: rho (Factor de reducción de iteraciones): 0.5000
|  5: K (Número de iteraciones en T): 30
|  6: qv (Cantidad de vértices que posee el grafo): 7
|  7: autom (Modo de acción): 1
|
```

```
--------------------------------
|  Calibración del algoritmo:
|  Seleccione algún número si desea cambiar su valor (o 0 para continuar):
|  0: Listo. Continuar
|  1: graph (El grafo a evaluar):
|     [30] [0] [4202664] [0] [4219430] [0] [6487112]
|     [0] [-202942144] [32765] [6479984] [0] [-202966480] [32765]
|     [0] [1071644672] [0] [0] [208] [0] [0]
|     [0] [4200264] [0] [1056964608] [0] [1] [0]
|     [30] [0] [7] [0] [1056964608] [7602296] [1124859904]
|     [4259932] [1092616192] [6881384] [6487392] [0] [6487408] [0]
|     [6487404] [0] [6487400] [0] [6487388] [0] [6487384]
|  2: tour (El viaje inicial):
|     [0] [0] [-203255635] [32765] [30] [0] [204] [0]
|
```

```
--------------------------------
|  Calibración del algoritmo:
|  Seleccione algún número si desea cambiar su valor (o 0 para continuar):
|  0: Listo. Continuar
|  1: graph (El grafo a evaluar):
|     [0] [12] [10] [0] [0] [0] [12]
|     [12] [0] [8] [12] [0] [0] [0]
|     [10] [8] [0] [11] [3] [0] [9]
|     [0] [12] [11] [0] [11] [10] [0]
|     [0] [0] [3] [11] [0] [6] [7]
|     [0] [0] [0] [10] [6] [0] [9]
|     [12] [0] [9] [0] [7] [9] [0]
|  2: tour (El viaje inicial):
|     [1] [2] [3] [4] [5] [6] [7] [1]
|
```

```
Evaluamos con los valores:
    T = 140.0000
    K = 30
El tour [1, 2, 3, 4, 5, 6, 7, 1] genera un costo de: 69
El nuevo tour [1, 7, 6, 5, 4, 3, 2, 1] genera un costo de: 69
No hemos escogido aún (fnewTour >= ftour)
Hemos escogido newTour por probabilidad de Boltzmann (r3 = 0.6400 < 1.0000)
El tour [1, 7, 6, 5, 4, 3, 2, 1] genera un costo de: 69
El nuevo tour [1, 7, 6, 4, 5, 3, 2, 1] genera un costo de: 65
Hemos escogido newTour (fnewTour < ftour)
El tour [1, 7, 6, 4, 5, 3, 2, 1] genera un costo de: 65
El nuevo tour [1, 7, 6, 5, 4, 3, 2, 1] genera un costo de: 69
No hemos escogido aún (fnewTour >= ftour)
Hemos escogido newTour por probabilidad de Boltzmann (r3 = 0.8500 < 0.9718)
El tour [1, 7, 6, 5, 4, 3, 2, 1] genera un costo de: 69
El nuevo tour [1, 7, 6, 5, 4, 2, 3, 1] genera un costo de: 68
Hemos escogido newTour (fnewTour < ftour)
El tour [1, 7, 6, 5, 4, 2, 3, 1] genera un costo de: 68
El nuevo tour [1, 3, 2, 4, 5, 6, 7, 1] genera un costo de: 68
No hemos escogido aún (fnewTour >= ftour)
Hemos escogido newTour por probabilidad de Boltzmann (r3 = 0.4900 < 1.0000)
El tour [1, 3, 2, 4, 5, 6, 7, 1] genera un costo de: 68
El nuevo tour [1, 7, 6, 5, 4, 2, 3, 1] genera un costo de: 68
No hemos escogido aún (fnewTour >= ftour)
Hemos escogido newTour por probabilidad de Boltzmann (r3 = 0.7700 < 1.0000)
El tour [1, 7, 6, 5, 4, 2, 3, 1] genera un costo de: 68
El nuevo tour [1, 7, 6, 5, 4, 3, 2, 1] genera un costo de: 69
No hemos escogido aún (fnewTour >= ftour)
Hemos escogido newTour por probabilidad de Boltzmann (r3 = 0.4900 < 0.9929)
El tour [1, 7, 6, 5, 4, 3, 2, 1] genera un costo de: 69
El nuevo tour [1, 7, 6, 4, 5, 3, 2, 1] genera un costo de: 65
Hemos escogido newTour (fnewTour < ftour)
El tour [1, 7, 6, 4, 5, 3, 2, 1] genera un costo de: 65
El nuevo tour [1, 7, 6, 5, 4, 3, 2, 1] genera un costo de: 69
No hemos escogido aún (fnewTour >= ftour)
Hemos escogido newTour por probabilidad de Boltzmann (r3 = 0.3500 < 0.9718)
El tour [1, 7, 6, 5, 4, 3, 2, 1] genera un costo de: 69
El nuevo tour [1, 7, 6, 5, 3, 4, 2, 1] genera un costo de: 65
Hemos escogido newTour (fnewTour < ftour)
El tour [1, 7, 6, 5, 3, 4, 2, 1] genera un costo de: 65
El nuevo tour [1, 7, 6, 5, 4, 3, 2, 1] genera un costo de: 69
No hemos escogido aún (fnewTour >= ftour)
Hemos escogido newTour por probabilidad de Boltzmann (r3 = 0.4100 < 0.9718)
El tour [1, 7, 6, 5, 4, 3, 2, 1] genera un costo de: 69
El nuevo tour [1, 7, 6, 5, 4, 2, 3, 1] genera un costo de: 68
Hemos escogido newTour (fnewTour < ftour)
El tour [1, 7, 6, 5, 4, 2, 3, 1] genera un costo de: 68
El nuevo tour [1, 7, 6, 5, 4, 3, 2, 1] genera un costo de: 69
No hemos escogido aún (fnewTour >= ftour)
Hemos escogido newTour por probabilidad de Boltzmann (r3 = 0.3800 < 0.9929)
El tour [1, 7, 6, 5, 4, 3, 2, 1] genera un costo de: 69
El nuevo tour [1, 7, 6, 5, 4, 2, 3, 1] genera un costo de: 68
Hemos escogido newTour (fnewTour < ftour)
El tour [1, 7, 6, 5, 4, 2, 3, 1] genera un costo de: 68
El nuevo tour [1, 7, 6, 5, 4, 3, 2, 1] genera un costo de: 69
No hemos escogido aún (fnewTour >= ftour)
Hemos escogido newTour por probabilidad de Boltzmann (r3 = 0.5700 < 0.9929)
El tour [1, 7, 6, 5, 4, 3, 2, 1] genera un costo de: 69
El nuevo tour [1, 7, 6, 5, 4, 2, 3, 1] genera un costo de: 68
```

```
Evaluamos con los valores:
    T = 35.0000
    K = 7
El tour [1, 7, 5, 6, 4, 2, 3, 1] genera un costo de: 65
El nuevo tour [1, 7, 5, 6, 4, 3, 2, 1] genera un costo de: 66
No hemos escogido aún (fnewTour >= ftour)
Hemos escogido newTour por probabilidad de Boltzmann (r3 = 0.6400 < 0.9718)
El tour [1, 7, 5, 6, 4, 3, 2, 1] genera un costo de: 66
El nuevo tour [1, 7, 5, 6, 4, 2, 3, 1] genera un costo de: 65
Hemos escogido newTour (fnewTour < ftour)
El tour [1, 7, 5, 6, 4, 2, 3, 1] genera un costo de: 65
El nuevo tour [1, 7, 6, 5, 4, 2, 3, 1] genera un costo de: 68
No hemos escogido aún (fnewTour >= ftour)
Hemos escogido newTour por probabilidad de Boltzmann (r3 = 0.1000 < 0.9179)
El tour [1, 7, 6, 5, 4, 2, 3, 1] genera un costo de: 68
El nuevo tour [1, 7, 5, 6, 4, 2, 3, 1] genera un costo de: 65
Hemos escogido newTour (fnewTour < ftour)
El tour [1, 7, 5, 6, 4, 2, 3, 1] genera un costo de: 65
El nuevo tour [1, 7, 5, 6, 4, 3, 2, 1] genera un costo de: 66
No hemos escogido aún (fnewTour >= ftour)
Hemos escogido newTour por probabilidad de Boltzmann (r3 = 0.7300 < 0.9718)
El tour [1, 7, 5, 6, 4, 3, 2, 1] genera un costo de: 66
El nuevo tour [1, 7, 5, 6, 4, 2, 3, 1] genera un costo de: 65
Hemos escogido newTour (fnewTour < ftour)
El tour [1, 7, 5, 6, 4, 2, 3, 1] genera un costo de: 65
El nuevo tour [1, 7, 5, 6, 4, 3, 2, 1] genera un costo de: 66
No hemos escogido aún (fnewTour >= ftour)
Hemos escogido newTour por probabilidad de Boltzmann (r3 = 0.2100 < 0.9718)


Evaluamos con los valores:
    T = 17.5000
    K = 3
El tour [1, 7, 5, 6, 4, 3, 2, 1] genera un costo de: 66
El nuevo tour [1, 7, 6, 5, 4, 3, 2, 1] genera un costo de: 69
No hemos escogido aún (fnewTour >= ftour)
Hemos escogido newTour por probabilidad de Boltzmann (r3 = 0.5000 < 0.8425)
El tour [1, 7, 6, 5, 4, 3, 2, 1] genera un costo de: 69
El nuevo tour [1, 7, 6, 5, 4, 2, 3, 1] genera un costo de: 68
Hemos escogido newTour (fnewTour < ftour)
El tour [1, 7, 6, 5, 4, 2, 3, 1] genera un costo de: 68
El nuevo tour [1, 7, 6, 5, 4, 3, 2, 1] genera un costo de: 69
No hemos escogido aún (fnewTour >= ftour)
Hemos escogido newTour por probabilidad de Boltzmann (r3 = 0.3700 < 0.9445)



El tour final fue [1, 7, 6, 5, 4, 3, 2, 1] que genera un costo de: 69



¿Desea repetir el código?
0. No
1. Sí
```

# Anexo 3: Algunas capturas del código en el IDE

```
/*
    Universidad Autónoma de Aguascalientes

        Centro de Ciencias Básicas
  Departamento de Ciencias de la Computación
          Optimización Inteligente

                5° "A"

        Práctica 5: Recocido Simulado

        Doctora Aurora Torres Soto

   Alumno: Joel Alejandro Espinoza Sánchez

  Fecha de Entrega: 3 de noviembre del 2020

Descripción:
*/
//Cargamos las librerías
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>
#include <time.h>
#include <math.h>

void setValues1(float alpha, int autom, int K, int qv, float rho, float T, float Tfin, float *alphaP, int *automP, int *KP, int *qvP, float
void setValues2(int qv, int graph[qv][qv], int tour[qv + 1]);
int checkTour(int qv, int graph[qv][qv],int tour[qv + 1]);
void evaluateTour(int qv, int graph[qv][qv],int tour[qv + 1], int ftour, int *ftourP);
void backToTour(int qv, int tour[qv + 1],int newTour[qv + 1]);
```

```
main()
{
    setlocale(LC_ALL,"");
    srand(time(NULL));

    //1. Declaramos las variables que usaremos
    /*
        alpha: Factor de enfriamiento
        autom: Modo de acción, donde 0 será Manual y 1 Automático, que:
                Si autom = 0 ejecutará todo el proceso rápidamente
                Si autom = 1 ejecutará el proceso de uno en uno
        k: Iterador sobre K
        K: Número de iteraciones a cierta T
        qv: Cantidad de vértices
        r1: Un número aleatorio
        r2: Un número aleatorio
        r3: Un número aleatorio
        rho: Factor de reducción de iteraciones
        T: Temperatura actual
        Tfin: Temperatura final
    */
    int autom, check, i, j, k, K, qv, r1, r2, r3a, repeat;
    float alpha, r3, rho, T, Tfin;

    printf("============================== RECOCIDO SIMULADO ==============================\n");

    do
    {
        alpha = 0.5;
        autom = 1;
        k = 0;
        K = 30;
        qv = 7;
        rho = 0.5;
        repeat = 0;
        T = 140;
        Tfin = 10;
```

```c
//2. Calibramos los primeros valores del programa
setValues1(alpha,autom,K,qv,rho,T,Tfin,&alpha,&autom,&K,&qv,&rho,&T,&Tfin);

//3. Declaramos más variables
/*
    graph: La matriz de adyacencia del grafo que analizaremos
    tour: Un vector de dimensión qv que representa el viaje i
    ftour: La energío de tour
    newTour: Un vector igual a tour que representa el viaje J
    fnewTour: La energío de newTour
*/
int graph[qv][qv], tour[qv + 1], ftour, newTour[qv + 1], fnewTour;
setValues2(qv,graph,tour);
getchar();

//4. Comenzamos el Recocido
do
{
    printf("\n");
    printf("\n");
    printf("Evaluamos con los valores:\n");
    printf("    T = %.4f\n",T);
    printf("    K = %.d\n",K);
    do
    {
        //5. Evaluamos el costo de tour
        evaluateTour(qv,graph,tour,ftour,&ftour);

        printf("El tour [%d",tour[0]);
        for(i = 1; i <= qv; i++)
        {
            printf(", %d",tour[i]);
        }
        printf("] genera un costo de: %d\n",ftour);
```

```
//6. Perturbaremos tour
do
{
    r1 = 2 + (rand() % ((qv - 1) - 2 + 1));
    r2 = (r1 + 1) + (rand() % (qv - (r1 + 1) + 1));

    //Construimos newTour con un subviaje inverso aleatorio de tour
    for(i = 0; i < r1; i++)
    {
        newTour[i] = tour[i];
    }

    j = 0;
    for(i = r1 - 1; i < r2; i++)
    {
        newTour[i] = tour[r2 - 1 - j];
        j++;
    }

    for(i = r2; i < qv + 1; i++)
    {
        newTour[i] = tour[i];
    }

    //No pasaremos a la siguiente fase hasta verificar que newTour sea un tour válido
    check = checkTour(qv,graph,newTour);
}
while(check == 0);
```

```
//7. Evaluamos newTour
evaluateTour(qv,graph,newTour,fnewTour,&fnewTour);

printf("r1 = %d\n",r1);
printf("r2 = %d\n",r2);
printf("El nuevo tour [%d",newTour[0]);
for(i = 1; i <= qv; i++)
{
    printf(", %d",newTour[i]);
}
printf("] genera un costo de: %d\n",fnewTour);
if(autom == 0)
{
    getchar();
}

//8. Evaluamos para minimizar (Para maximizar cambiamos el signo del condicional < por >)
if(fnewTour < ftour)
{
    printf("Hemos escogido newTour (fnewTour < ftour)\n");
    if(autom == 0)
    {
        getchar();
    }
    backToTour(qv,tour,newTour);
    ftour = fnewTour;
}
```

```
    else
    {
        printf("No hemos escogido aún (fnewTour >= ftour)\n");
        if(autom == 0)
        {
            getchar();
        }

        r3a = rand() % 100;
        r3 = ((float)r3a)/100;
        if(r3 < (exp((ftour - fnewTour)/T)))
        {
            printf("Hemos escogido newTour por probabilidad de Boltzmann (r3 = %.4f < %.4f)\n",r3,exp((ftour - fnewTour)/T));
            if(autom == 0)
            {
                getchar();
            }

            backToTour(qv,tour,newTour);
            ftour = fnewTour;
        }
        else
        {
            printf("Hemos descartado newTour por probabilidad de Boltzmann (r3 = %.4f >= %.4f)\n",r3,exp((ftour - fnewTour)/T));
            if(autom == 0)
            {
                getchar();
            }
        }
    }

    k++;
}
while(k < K);
```

```
            T = alpha*T;
            K = rho*K;
            k = 0;
        }
        while(T > Tfin);

        printf("\n");
        printf("\n");
        printf("\n");
        printf("El tour final fue [%d",tour[0]);
        for(i = 1; i <= qv; i++)
        {
            printf(", %d",tour[i]);
        }
        printf("] que genera un costo de: %d\n",ftour);

        printf("\n");
        printf("\n");
        printf("\n");
        printf("¿Desea repetir el código?\n");
        printf("0. No\n");
        printf("1. Sí\n");
        scanf("%d",&repeat);
    }
    while(repeat == 1);

    getchar();
}
```