matemática: Una publicación del ICM – ESPOL
2010, Vol. 8, No. 1

PROPUESTA DE UN MODELO MATEMÁTICO PARA LA SECUENCIACIÓN ÓPTIMA DE LA COSECHA DE LA CAÑA DE AZÚCAR

Díaz Rosa María[1], **Tisher Irene**[2]

Resumen. *En este artículo se presenta la formulación y desarrollo de un modelo matemático para la secuenciación óptima de la cosecha de la caña de azúcar, que considera el tiempo más temprano y el tiempo más tarde de cosecha (ventana de tiempo). El problema consiste en encontrar una ruta óptima, donde la alzadora visite cada suerte exactamente una vez partiendo de una suerte fija, el tiempo de desplazamiento sea mínimo y cumpla con las restricciones de la ventana de tiempo. Dada la semejanza que existe entre el problema de las alzadoras y el agente viajero, se tomó como base la aplicación que realizó Christos Voudouris [VT99] de la técnica heurística Guided Local Search al problema del agente viajero. Se realizaron los ajustes necesarios a este algoritmo incluyendo ventanas de tiempo y solución inicial factible teniendo en cuenta las restricciones de la ventana. Como resultado, tenemos el algoritmo para el recorrido óptimo de las alzadoras, el cual será utilizado en la programación de la cosecha a mediano plazo (3 meses), corto plazo (1 mes) y cada vez que hayan cambios en la programación (por quemas accidentales, daños de equipos, etc.). Se realizaron las pruebas para validar el modelo, utilizando datos tomados de una simulación y con datos reales obtenidos del ingenio Incauca.*

Palabras Claves: Modelación matemática, optimización combinatorial, heurísticas.

Abstract. *This article presents the design and development of a mathematical model for the optimal sequencing of harvest sugar cane, which considers time earlier and later harvest time (time window). The problem is to find an optimal route, where the fate collator visit exactly once each starting from a fixed fate, travel time is minimal and meet the restrictions of the time window. Given the similarity between the problem of collating and traveling salesman, was made based on the application that made Christos Voudouris [VT99] of Guided Local Search heuristic technique to traveling salesman problem. Necessary adjustments were made to this algorithm including time frame and initial solution feasible considering the constraints of the window. As a result, we have the algorithm for optimal route of lifters, which will be used in scheduling the harvest medium term (3 months), short term (1 month) and every time you have changes in the schedule (for burning accidental, damage of equipment, etc.).. Testing was performed to validate the model using data from simulated and real data obtained Incauca wit.*

Keywords: mathematical modeling, combinatorial optimization, heuristics.

Recibido: Diciembre, 2009
Aceptado: Febrero, 2010

1. INTRODUCCIÓN

Con la globalización de la economía, se empieza un proceso de cambio basado en la liberación de los mercados y la modernización institucional, presentándose elevados niveles de competitividad en el mercado nacional e internacional. Ante este panorama, la agro-industria azucarera colombiana debe introducir herramientas tecnológicas que proporcionen alternativas eficientes para la toma de decisiones, con el fin de mejorar la competitividad en el mercado y poder obtener una alta rentabilidad de sus productos.

La producción de caña de azúcar posee uno de los procesos más complejos pues involucra una serie de actividades que exigen una gran coordinación, un control permanente y una alta eficiencia en su ejecución. De las tres actividades (agrícola, cosecha, fabril) que intervienen en el proceso de producción del azúcar, es en la cosecha donde encontramos más factores no controlables, lo que hace difícil la toma de las decisiones y la realización de una programación eficiente.

Irene Tisher, con un grupo de estudiantes: tres de Ingeniería de Sistemas y uno de matemáticas, se propuso desarrollar una herramienta computacional que sirva de apoyo para la toma de decisiones a nivel estratégico, gerencial y de operación en la cosecha de la caña de azúcar, incluyendo la sistematización y optimización de las operaciones involucradas en la cosecha, las cuales son: aplicación madurante, quema, corte, alce y transporte.

Dentro de este proyecto macro, se encuentran varios subproyectos, uno de ellos es la formulación y desarrollo de un modelo matemático para la secuenciación óptima de la cosecha de la caña de azúcar, el cual corresponde a este artículo.

2. ANÁLISIS Y DESCRIPCIÓN DEL MODELO PROPUESTO

Se realizó un análisis detallado de la información recopilada en la investigación preliminar para identificar en que actividad de la cosecha centrábamos nuestra atención para desarrollar el modelo de optimización.

Inicialmente, se pensaba que la optimización se tenía que realizar en el transporte. El transporte de

[1] *Díaz Palacios Rosa María, Analista de Sistemas, ESPOL; Matemática, Universidad del Valle, Cali – Colombia. (e_mail: rmdiazpala@hotmail.com)*

[2] *Tisher Irene, Matemática; Doctora en Ingeniería Industrial, Universidad Politécnica de Valencia (España), 2000. Área de Trabajo: Bioinformática, simulación computacional y optimización. (e_mail: irene.tischer@correounivalle.edu.co)*

la caña de azúcar desde las suertes hasta los patios en la fábrica es uno de los aspectos más críticos, ya que está sujeto a una serie de parámetros algunos de ellos no controlables (clima, daños mecánicos, retraso por problemas en la vía, etc.) que hacen difícil su programación. Además, el transporte es la actividad que más altos costos tiene dentro del proceso de la cosecha.

Diseñar y desarrollar un modelo matemático para maximizar la eficiencia del sistema de transporte sería una herramienta que contribuiría a la optimización de los recursos en el proceso de cosecha de la caña de azúcar, como una alternativa para reducir los costos de producción, que es lo que se persigue.

Pero, a medida que se avanzaba en el análisis, nos dimos cuenta que la programación del transporte dependía de la ubicación de los equipos de alce.

Uno de los factores que influyen en la programación de la cosecha está dado por la ubicación del equipo que alza la caña, las alzadoras. El desplazamiento de estos equipos involucra tiempos improductivos que representan costos para la empresa. Además, si no se hace un recorrido eficiente de las alzadoras, la programación de la cosecha no puede alcanzar su objetivo principal que es cubrir los requerimientos de molienda de la fábrica en el menor tiempo posible y con la mejor calidad de caña.

Ahora ya tenemos claro el objetivo: determinar la ruta óptima de las alzadoras. Esta decisión se fundamenta en dos razones. Primero, los equipos de alce deben trabajar al mismo ritmo que trabaja la fábrica; es decir, existe una relación continua entre el alce y la fábrica. Segundo, la programación de aplicación de madurante, quema, corte, y sincronización alce y transporte depende de la programación de los alces a largo plazo (recorrido e itinerario de las alzadoras).

3. PROBLEMA DEL AGENTE VIAJERO

El Agente Viajero (Traveling Salesman Problem, TSP) es uno de los problemas más estudiados en optimización combinatorial. Se define como sigue: Sea G = (V, A) un grafo, donde $V = \{v_1, v_2, \ldots\ldots, v_n\}$ es el conjunto de vértices y $A = \{(v_i, v_j): v_i, v_j \in V, i \neq j\}$ es el conjunto de arcos. Con cada arco (v_i, v_j) está asociado un costo no negativo C_{ij} (el costo puede ser distancia o tiempo de desplazamiento). El TSP consiste en encontrar el ciclo hamiltoniano (incluye todos los vértices) de menor costo pasando a través de cada vértice exactamente una vez. La interpretación más común del TSP es el del vendedor que busca la ruta con menor costo a través de N ciudades, pasando por cada ciudad exactamente una vez y terminando en la ciudad donde inició la ruta. El problema es simétrico si y solo si $C_{ij} = C_{ji}\ \forall_{i,j}$. El problema satisface la desigualdad triangular si y solo si $C_{ij} + C_{ik} \geq C_{jk}\ \forall_{i,j,k}$. Un caso particular es el problema euclidiano, donde los costos corresponden a distancias euclidianas y el problema satisface la desigualdad triangular.

Una variante del TSP es el Agente viajero con ventanas de tiempo (Traveling Salesman with Time Windows – TSPTW). En este caso, cada vértice V_i debe ser visitado dentro de una ventana de tiempo $\left[a_i, b_j\right]$ y cada vértice tiene asociado un tiempo de servicio. Se puede minimizar el tiempo de llegada o el tiempo de desplazamiento sobre la ruta.

El agente viajero pertenece a la clase de problemas de optimización para el cual la función objetivo tiene muchos mínimos locales, por lo que es difícil encontrar el mínimo global. También es un problema de optimización combinatorial difícil (NP- complete problema), no existe un algoritmo en tiempo polinomial que lo resuelva, de ahí la dificultad en encontrar un algoritmo óptimo y eficiente. El tiempo de ejecución del mejor algoritmo conocido para el problema crece en forma exponencial.

Muchos autores han desarrollado algoritmos exactos y heurísticas para resolver el problema, como veremos a continuación. Dumas et al. [DDGS95] describe un algoritmo exacto usando programación dinámica para el TSPTW. Ellos desarrollaron pruebas de eliminación, las cuales mejoran grandemente la técnica de programación dinámica usada en este problema. Las pruebas toman ventaja de las restricciones de la ventana de tiempo para reducir el espacio y el número de transiciones de los estados. El algoritmo fue exitoso al resolver problemas de hasta 200 nodos con ventanas de tiempo pequeñas. Para los problemas de tamaño grande con ventanas de tiempo muy anchas, la ejecución de este algoritmo tiene problemas de memoria.

Gedrau et al. [GHLS98] desarrollaron una heurística de inserción generalizada para el TSPTW. El algoritmo gradualmente construye una ruta, insertando un vértice sobre la ruta actual y ejecutando una reoptimización local. Esto lo hace mientras chequea la factibilidad de la parte restante de la ruta. Cuando una ruta factible es encontrada, se aplican fases de post-optimización basadas en eliminaciones y reinserciones sucesivas de todos los vértices. Las pruebas ejecutadas con esta heurística dan como resultado soluciones cercanas al óptimo dentro de un tiempo computacional razonable. Las pruebas se realizaron con problemas de hasta 100 nodos.

Mingozzi et al. [MBR97] presentaron un algoritmo exacto basado en programación dinámica, para el TSPTW y Restricciones de

Precedencia (antes de visitar un vértice i, el viajero debe visitar cada vértice de un conjunto de vértices dados). El algoritmo da buenos resultados con problemas de hasta 120 nodos.

Cristos Voudoris y Edward Tsang (1998) desarrollaron la Heurística Guided Local Search, para resolver el TSP. En comparación con otras heurísticas que existen para el TSP como Tabú Search [VT99], Simulated Annealing [Ski98], ésta es la que encuentra, en menor tiempo y con menor costo computacional, una solución cercana al óptimo.

4. FORMULACIÓN DEL MODELO

El problema de la ruta de las alzadoras con ventanas de tiempo está definido por un grafo dirigido G = (*V*, *A*) donde $V = \{v_1, v_2, \ldots, v_n\}$ es el conjunto de suertes (nodos) y $A = \{(v_i, v_j): v_i, v_j \in V, i \neq j\}$ es el conjunto de arcos dirigidos que representan las carreteras que comunican dichas suertes.

Un tiempo de desplazamiento td_{ij} está asociado a cada arco (v_i, v_j), $i \neq j$ del grafo, éste representa el coste para cada arco. Cada suerte v_i tiene asignado un tiempo de servicio ts_i. La matriz TD= $[td_{ij}]$ que corresponde a los tiempos de desplazamiento es simétrica, $td_{ij} = td_{ji}\ \forall_{i,j} \in S$ (conjunto de suertes). El tiempo de llegada de la alzadora a cada suerte para iniciar el servicio debe estar dentro de un intervalo de *tiempo* $\left[a_i, b_j\right]$ llamado ventana de tiempo. A cada suerte se le asigna una ventana de tiempo, donde a_i es el tiempo más temprano de cosecha y b_j es el tiempo más tarde de cosecha. Tiempo de espera es posible si la alzadora llega a la suerte antes que a_i, no se permite que la alzadora llegue a la suerte después de b_j.

El objetivo es encontrar una ruta óptima, donde la alzadora visite cada suerte exactamente una vez partiendo de una suerte fija, el tiempo de desplazamiento sea mínimo y cumpla las restricciones de la ventana de tiempo. En lugar de tiempo de desplazamiento se puede usar la distancia entre las suertes. La función de costos para el problema de las alzadoras con ventanas de tiempo es la siguiente:

$$G(ruta) = \sum_{i=1}^{N} \sum_{j=1}^{N} td_{ij}, l(v_i, v_j)(ruta)$$

Donde

$$l_{(v_i, v_j)}(ruta) = \begin{cases} 1, (v_i, v_j) \in ruta \\ 0, (v_i, v_j) \in ruta \end{cases}$$

y N es el número de suertes.

5. DELIMITACIÓN DEL MODELO EN LA PROGRAMACIÓN DE LA COSECHA

Dentro del proyecto macro, el modelo está ubicado en la programación de la cosecha a mediano (tres meses) y corto plazo (un mes). En esta parte es donde se maximiza la eficiencia de los alces. En la programación de la cosecha se considera la siguiente información: programación anual de la molienda, dada por fábrica; existencia de caña en campo; capacidad de los recursos: alzadoras, tracto mulas, tractores; clima (pluviosidad, viento); y la programación a largo plazo. De la programación a largo plazo se obtienen las suertes en edad de corte óptima, para el periodo de la programación.

Los ingenios han dividido en zonas el área sembrada en caña de azúcar. A cada zona le corresponde un frente de cosecha (personal de corte) y un alce (equipo y personal de alce). En particular, el alce uno con el frente de cosecha uno le corresponde la zona que se encuentra a menos de 20 Km. del ingenio. El personal correspondiente al alce y frente de cosecha vive cerca o en la zona asignada al alce, esto hace que el alce quede amarrado a la zona.

Por lo expuesto anteriormente, la distribución de las suertes a los alces no se incluye dentro de la optimización, ya que ésta depende de muchos factores, no solo de la distancia de la suerte al ingenio.

El personal del ingenio es el responsable de distribuir, de una forma eficiente, las suertes a cada uno de los alces, con base en los datos históricos, experiencia y capacidad de las alzadoras. Una vez distribuidos los alces, se realiza la optimización de cada alce, para obtener como resultado la ruta óptima de las alzadoras para cada alce.

6. INFORMACIÓN REQUERIDA POR EL MODELO

El modelo necesita como entrada los siguientes datos:

- Suertes en edad de corte de un alce determinado.
  - Identificación de la suerte.
  - Tiempo de servicio: tiempo que se demora la alzadora en alzar la caña.
  - Tiempo de inicio más temprano de corte de la caña en la suerte.
  - Tiempo más tarde de corte de la caña en la suerte.
- Tiempo de desplazamiento entre las suertes.
- Identificación de la suerte donde se encuentra el alce actualmente.

Cada alce cuenta con un equipo de mantenimiento, por lo que no es necesario que las

alzadoras se desplacen al ingenio por mantenimiento. Sólo van al ingenio cuando el daño es grave y no se lo puede arreglar en el campo. Por eso, entre los datos de entrada no se incluye tiempo de desplazamiento por mantenimiento.

7. DESARROLLO DEL MODELO USANDO LA TÉCNICA GLS

Para desarrollar el modelo de optimización, tomaremos como base la aplicación que realizó Christos Voudoris [VT99] de la técnica GLS al problema del agente viajero teniendo en cuenta las siguientes diferencias:

- El algoritmo implementado por Voudouris para el agente viajero es sin ventanas de tiempo. El problema de las alzadoras es con ventanas de tiempo.
- La ruta en el agente viajero es cerrada, es decir el agente debe terminar su ruta en la primera ciudad visitada. La ruta en el problema de las alzadoras no es cerrada.
- Al tener ventanas de tiempo en el problema de las alzadoras, la solución inicial debe cumplir con las restricciones de la ventana. Entonces hay que determinar una estrategia para obtener una solución inicial factible.
- En el algoritmo del agente viajero, la función de costos es entera. En el problema de las alzadoras, la función de costos es real.
- La ruta de las alzadoras debe iniciar en una suerte fija, en cambio en agente viajero puede iniciar su ruta en cualquier ciudad.

7.1 HEURISTICA GUIDED LOCAL SEARCH

GLS es una estrategia de búsqueda general meta – estocástica, utilizada para resolver problemas de satisfacción con restricciones (constraint satisfaction problems) y de optimización. Es una técnica de control diseñada para ayudarle a los algoritmos hill climbing a escapar de un óptimo local y distribuir el esfuerzo de búsqueda. Ha sido aplicada a problemas no triviales incluyendo problemas de optimización artificial[3], problemas de optimización estándar[4] y problemas de la vida real. Los resultados han sido excelentes tanto en términos de velocidad como en calidad de soluciones.

Local Search es la base de muchos métodos heurísticos para problemas de optimización combinatorial, en particular para GLS.

A continuación describimos esta técnica. Toda la información sobre GLS y Local Search fue tomada de [VT99], [Vou97] y [VY95].

Algunas de las aplicaciones de GLS son:

- El problema del Agente Viajero (TSP).
- Problema de asignación de la longitud de la frecuencia de radio.
- Problema de itinerarios para el personal de Telecom British.
- El problema de asignación cuadrático.
- F6, un problema de optimización artificial.
- El problema de la ruta de vehículos.

GLS, fue desarrollada por Christos Voudouris y Edward Tsang y es el resultado de un proyecto de investigación con el fin de extender la arquitectura GENET basada en Redes Neuronales, la cual es aplicable a problemas de satisfacción con restricciones. GLS generaliza algunos de los elementos presentes en GENET, de tal forma que se puede aplicar a problemas de optimización combinatorial en general.

Empezando con GENET, Voudouris y Tsang desarrollaron algoritmos intermedios, tales como el algoritmo Tunneling para concluir con GLS. En contraste con sus predecesores GLS es una técnica de optimización general y compacta.

7.1.1 DESCRIPCIÓN DE LOCAL SEARCH (LS) PARA LA RUTA DE LAS ALZADORAS

A continuación describimos esta técnica. Toda la información sobre GLS y Local Search fue tomada de [VT99], [Vou97] y [VY95].

Para implementar Local Search[5], se necesita un operador de movimiento, el cual transforma una solución s en s'. Este operador debe tener la particularidad de poder definir estructuras de vecindades, para poder aplicar el esquema de búsqueda. Los operadores de movimiento que han dado mejores resultados con el agente viajero son el 2-Opt, 3-Opt[6]. Usaremos en la aplicación 2-Opt, su estructura de vecindad es definida a continuación.

Operador de movimiento 2-Opt: Una solución vecina es obtenida, desde la solución actual, borrando dos arcos, invirtiendo una de las rutas resultantes, reemplazando los dos arcos borrados por otros dos y al final reconectando toda la ruta. La complejidad del peor caso para buscar la vecindad definida por 2-Opt es $O(n^2)$[7]

---

[3] *Problemas que se pueden resolver utilizando inteligencia Artificial.*

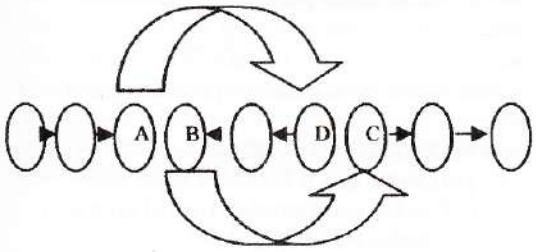
[4] *Problemas que se pueden resolver por métodos de optimización estándar lineales y no lineales.*

[5] *Búsqueda Local, también conocida como Hill-climbing de Vecindades, es una estrategia basada en optimización local. Los algoritmos de búsqueda local constituyen una clase de algoritmos aproximados basados en la exploración de vecinos. Estos presuponen que existe una función de costo y una estructura de vecinos.*

[6] *Operador de movimiento que transforma una solución s en s'. Trabaja con estructura de vecindades.*

[7] *$O(g(n)) = f(n)$: existen constantes positivas c y n_o tal que $0 \le f(n) \le c\, g(n)$ para todo $n \ge n_o$*

FIGURA 1
Propuesta de un modelo matemático para la secuenciación óptima de la cosecha de la caña de azúcar
El operador de movimiento 2-Opt.



Revisemos cómo trabaja el procedimiento Local Search con 2-Opt aplicado al problema de las alzadoras. Se parte de una solución *S*, la mejor encontrada hasta el momento. LS evalúa todos los intercambios posibles con el arco formado por las suertes A y B, B es adyacente (suerte anterior o siguiente) a A.

A continuación, en un orden estático (1 .. Número de vecinos), se busca por los vecinos de la suerte B para formar el otro arco que necesitamos para el intercambio; C es un vecino de B y D es la suerte adyacente (anterior o siguiente) a B. Ahora con el arco formado por A y B, C y D se evalúa el movimiento para determinar si mejora la función costo y el intercambio cumple con las restricciones de la ventana de tiempo. Si esto se cumple el proceso termina, de lo contrario el proceso continúa con el siguiente vecino de B. Los vecinos de una suerte son todas las suertes (hasta el número total de vecinos de una suerte) ordenadas por el tiempo de desplazamiento en forma ascendente.

Un intercambio cumple con las restricciones de la ventana de tiempo, si para una suerte s_i con ventana $\left[a_i, b_j\right]$ el tiempo de llegada a s_i es menor o igual que el tiempo más tarde de cosecha b_i; donde el tiempo de llegada a s_i = max(tiempo de llegada$_{i-1}$ + tiempo de servicio$_{i-1}$ + tiempo de desplazamiento$_{i-1}$, a_i).

Un factor que limita la eficiencia de LS es el tamaño de la vecindad. Si hay muchos vecinos a considerar, la búsqueda requiere muchos pasos para alcanzar un óptimo local y cada evaluación de la función objetivo requiere una cantidad no trivial de cálculos, entonces la búsqueda puede ser muy costosa. Para solucionar este problema, Chris Voudouris [VT99] propone el método Fast Local Search (FLS) para restringir la vecindad. La intención es ignorar aquellos vecinos que no ayudarán a LS a encontrar una mejor solución, con el fin de mejorar la eficiencia de la búsqueda. A continuación se describe esta técnica.

7.1.2 FAST LOCAL SEARCH (FLS) Y SU APLICACIÓN EN EL PROBLEMA DE LAS ALZADORAS

FLS puede ser combinado con la estructura de vecindades de 2-Opt de la siguiente forma. Para un problema con N suertes, la vecindad es particionada en N subvecindades, una para cada suerte. Una subvecindad es definida por una suerte, la cual contiene todos los intercambios de arcos originados desde uno de los arcos adyacentes a la suerte. En nuestro caso, hay tantas subvecindades como suertes tenga el problema.

Inicialmente, todas las subvecindades son activas. La búsqueda de todas las subvecindades se realiza en un orden estático, desde I al número de suertes. Cuando una subvecindad activa es encontrada, se llama al procedimiento LS 2-Opt para que evalúe todos los intercambios que se pueden realizar a partir de la suerte activa y la suerte adyacente (anterior o posterior) a ella.

Si LS 2-Opt no encuentra un movimiento que mejore la función de costo y sea factible, entonces la subvecindad es desactivada. De lo contrario, se realizan los intercambios para obtener la nueva ruta y todas las suertes involucradas en el movimiento son activadas. El proceso continúa con la próxima subvecindad en el orden estático. Si se realiza una vuelta completa por todas las subvecindades (desde 1 al número de suertes) y no se encuentra un movimiento que mejore la función de costo aumentada, el proceso termina, retornando la ruta encontrada.

7.1.3 GLS APLICADO AL PROBLEMA DE LAS ALZADORAS

En la implementación de GLS, tenemos que definir las siguientes partes:

- Conjunto de Componentes $\mathcal{F}$. Para el agente viajero [VT99], los arcos fueron escogidos como componentes de las soluciones. Nosotros también usaremos arcos, pero arcos dirigidos, ya que para problemas de rutas que involucran tiempos, la dirección es importante. El conjunto $\mathcal{F}$ está formado por todas los posibles arcos dirigidos a_{ij} (i=1,...,NS; j=1, ..., NS; i ≠ j) que pueden aparecer en una ruta.
- Costo de un Componente. El costo de un arco dirigido a_{ij} está dado por el tiempo de desplazamiento td_{ij}. Los tiempos de desplazamiento están ubicados en la matriz TD = [td_{ij}]. Para el problema de las alzadoras la matriz TD es simétrica. Pero puede darse el caso de que esta matriz no sea simétrica (por ejemplo, cuando el terreno no es plano el tiempo de desplazamiento de ida no es igual al tiempo de desplazamiento de regreso), lo que no afecta al algoritmo.

- Penalización de los componentes. Cada arco a_{ij}, que conecta la suerte i con la suerte j, tiene asignado una penalización p_{ij}. Los arcos penalizados son ubicados en una matriz de penalizaciones simétrica $P = [p_{ij}]$.
- Función de costo aumentada. Las penalizaciones se combinan con la función de costo del problema para formar la función de costo aumentada, que puede ser definida usando la matriz de tiempo de desplazamiento auxiliar. $TD' = TD + \lambda * P = [td_{ij} + \lambda\, p_{ij}]$ Local Search debe usar TD´ en lugar de TD para evaluar los movimientos.
- Función de utilidad: El criterio para seleccionar los arcos a ser penalizados en un mínimo local es de acuerdo a la función utilidad

$$\acute{u}til(ruta, a_{ij}) = \mathrm{Ind}_{aij}(ruta) * \left(\frac{td_{ij}}{(1 + p_{ij})} \right) \quad (1)$$

Donde:

$$\mathrm{Ind}_{aij}(ruta) = \begin{cases} 1, a_{ij} \in ruta \\ 0, a_{ij} \in ruta \end{cases}$$

A continuación presentamos el algoritmo para GLS.

```
Procedure GLS (S0, λ,Ind,NS)
Begin
/* Inicializa las penalizaciones en cero y active las
subvecindades */
For i  1 until NS do
  begin
       biti ← V,
      for j ← 1 until j ≤ i do
         pij ← 0;
end
k ← 1;  sk ← s0;  Maxutil ← 0;
while CriterioParada do
begin
    hΠ  ∑Ind i,i+1  (sk) * tdi+1 +  λ , * ∑Ind i,i+1
        (sk) * pi,i+1;
Sk+1 = FLS2-Opt(sk, h, bit, NS);
K ← k+l;
for i ← 1 until NS do
begin
    utilidad ← Indi,i+1 (sk) * (tdi,i+1 / pi,i+1 + 1)
    if (Utilidad > Maxutil) then
        Maxutil ← utilidad;
    if (utilidad = Maxutil)
      begin
         pi,i+1 ← pi,i+1 + 1;
         biti,i+1  ← 1;
      end
end
  end
  s* ←  mejor solución encontrada con respecto
a
   la función de costo g;
  return s*;
 end
```

Los parámetros de entrada del procedimiento GLS son:

- s_0: solución Inicial Factible
- λ : parámetro de GLS
- Ind : función característica para el arco $a_{i,i+1}$
- NS : número de Suertes

7.1.4 Cómo funciona GLS con el problema de las Alzadoras

GLS inicia con una solución inicial factible y las penalizaciones p_{ij} inicializadas en cero. Llama al procedimiento FLS2-Opt para que encuentre un mínimo local. Usando la función de utilidad (1), selecciona los arcos que pertenecen la mínimo local para penalizarlos ($p_{ij} = p_{ij} + 1$). Las subvecindades (suertes) que se encuentran al final de los arcos penalizados son activadas. Se vuelve a llamar a FLS2-Opt para que encuentre un nuevo mínimo local a partir del mínimo local anterior. GLS incrementa las penalizaciones cada vez que un mínimo local es encontrado o cuando FLS2-Opt no puede escapar desde un mínimo local actual. El proceso termina cuando se cumple la condición de parada.

El criterio para la condición de parada es:

. Número de iteraciones > Límite de iteraciones o

. Tiempo de Ejecución > Tiempo Límite de Ejecución o

. Movimientos seguidos que no mejoran la función objetivo > Límite en los movimientos que no mejoren la función objetivo.

7.1.5 Solución Inicial Factible

Se construye una ruta de acuerdo al orden en que son ingresadas las suertes. Se obtiene una ruta inicial a partir de esta, ordenando las suertes de acuerdo a la ventana de tiempo [a,b]. Primero de ordenar por el tiempo más tarde de cosecha (b) y luego por el tiempo más temprano de cosecha (a). Los dos criterios son ordenados en forma ascendente. Esta solución inicial es transformada en una solución factible, minimizando la desviación de la ventana.

La desviación de la ventana de tiempo correspondiente a la suerte i es:

$$Desviación_i = \begin{cases} 1, \text{Tiempo de llegada}_i - b_i, \\ \text{si el tiempo de llegada a la suerte i} > b \\ 0, \text{en caso contrario} \end{cases}$$

La función objetivo del problema es

$$\text{Min} \sum_{i=1}^{NS} desviación_i$$

Donde NS el número de suertes.
Para encontrar la solución factible, se utilizó el mismo algoritmo que encuentra la ruta óptima, GLS con FLS2-Opt. Con la diferencia, que cuando encuentra un movimiento que mejora la función de costo (en el procedimiento LS2-Opt), se lo acepta sólo si este movimiento disminuye la desviación de la ventana.

7.1.6 El parámetro λ

El único parámetro de GLS, que requiere calibración es el parámetro λ. En [VT99], se encuentra una expresión para calcular λ para el caso del agente viajero.

$$\lambda = \alpha(g(\text{mínimo local}) / NS) \quad (2)$$

Donde g(mínimo local) es el costo del primer mínimo local antes que las penalizaciones son aplicadas y NS es el número de suertes. Se introduce el parámetro α, el cual depende del problema. En la aplicación del agente viajero, valores de α entre 0.125 y 0.5 dieron buenos resultados. Para el problema de las alzadoras con ventanas de tiempo, la expresión (2) no nos dio el mejor valor de λ. Por esta razón, se decidió ajustar λ para la situación específica del ingenio donde se realizaron las pruebas con datos reales.

8. VALIDACIÓN DEL MODELO

Las pruebas para validar el modelo de optimización para el problema de las alzadoras se realizó en dos etapas:

- Pruebas realizadas con el algoritmo con ventanas de tiempo, con datos tomados desde una simulación.
- Pruebas realizadas con el algoritmo con ventanas de tiempo, con datos reales tomados del Ingenio Incauca.

Los experimentos se realizaron en una computadora Pentium (166 Mhz) con el algoritmo desarrollado en Turbo C++ versión 3.0 (Borland International, Inc)

8.1 RESULTADOS DE LAS PRUEBAS CON DATOS REALES

Del informe mensual de cosecha del mes de junio, se tomaron las suertes del alce 1 y del alce 7 y se calcularon los datos de entrada al modelo. Se incluyeron las quemas accidentales de ese mes para poder comparar la ruta que genera el algoritmo con la ruta que siguieron los alces en la realidad. El valor exacto de λ usado en las pruebas fue ajustado manualmente, ejecutando el algoritmo varias veces con valores distintos de λ.

La mejor solución encontrada para los dos alces fue con un valor de λ= 7. Como los dos alces tomados para las pruebas son tan distintos (por su capacidad, distancia al ingenio, distancia entre suertes), podemos suponer que el valor de λ encontrado sirve para todos los alces y que corresponde a la situación específica del ingenio. En la implementación del algoritmo, se puede verificar que en realidad este valor de λ se ajusta a todos los alces. El criterio de parada fue 10.000 iteraciones, en la tabla 1, se encuentran los resultados para el alce 1 y 7. Las filas contienen el nombre del problema, el número total de suertes, el tiempo de desplazamiento, tiempo de finalizar el servicio en la última suerte de la ruta, tiempo de espera, tiempo que se demoró GLS en encontrar la mejor solución, tiempo de desplazamiento de la ruta que siguieron los alces en la realidad, tiempo de finalizar el servicio en la última suerte de la ruta real.

TABLA I
Propuesta de un modelo matemático para la secuenciación óptima de la cosecha de la caña de azúcar
Resultados de GLS obtenidos con datos reales tomados del Ingenio Incauca

	N° de suertes	Tiempo Desplaz. (días)	Tiempo Final (días)	Tiempo Espera (días)	Tiempo CPU (seg.)	Tiempo Desplaz. Real	Tiempo Final Real
Alce 1	41	3.43	27.93	0	53.69	6	31.69
Alce 7	51	7.08	29.4	0.33	57.86	9	31.36

Los resultados presentados en la tabla I reflejan las ventajas del algoritmo de optimización. Para los dos alces tenemos que el tiempo de desplazamiento es menor a los tiempos de desplazamiento que siguieron los alces en la realidad. También el tiempo final donde están incluidos los tiempos de desplazamiento, tiempo de servicio y tiempo de espera, es menor que el real. Para el caso del alce 1 termina su recorrido 3,7 días antes, lo que para el ingenio representa poder aumentar la capacidad del alce y cosechar más suertes en un mes, poder decidir en un momento dado que el alce no trabaje por la noche lo que reduce costos. Para los problemas del alce 1 y 7, no se tiene la solución óptima, ya que el algoritmo exacto no resuelve problemas con ventanas de tiempo anchas, para estos casos tiene limitaciones de memoria.

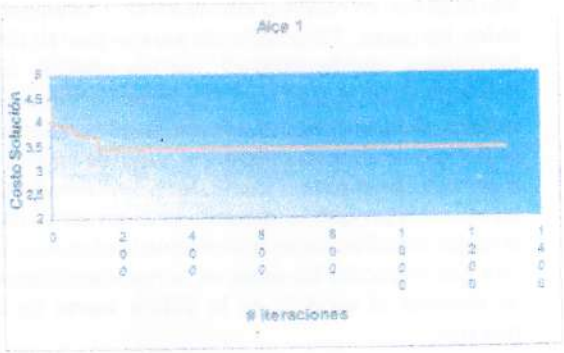
En la figura 1, se muestran las fluctuaciones del tiempo de desplazamiento durante las 1.300 iteraciones de GLS con FLS-2Opt con los datos del alce 1. Como podemos apreciar en la figura, después del descenso inicial tan abrupto en el costo, el algoritmo visita soluciones con costos menores. Intuitivamente esto nos dice que converge al óptimo. También es de anotar que con solo 76 iteraciones ya se tiene una buena solución

con un costo de 3.69. El comportamiento con el alce 7 es similar.

FIGURA 1

Propuesta de un modelo matemático para la secuenciación óptima de la cosecha de la caña de azúcar

El tiempo de desplazamiento de las soluciones generadas en las primeras 1.300 iteraciones de GLS con FLS-20pt con datos del alce 1



9. CONCLUSIONES Y RECOMENDACIONES

- En este trabajo, se logró formular y desarrollar un modelo matemático para la secuenciación óptima de la cosecha de la caña de azúcar. El modelo incluye ventanas de tiempo (tiempo de inicio más temprano y más tarde de cosecha). La función objetivo es minimizar el tiempo de desplazamiento.
- Para desarrollar el modelo, se utilizó la técnica heurística GLS con FLS – 2Opt., la cual es superior a otras heurísticas tales como Simulated Annealing y Tabu Search. Esta técnica es efectiva y eficiente, da soluciones muy cercanas al óptimo en un tiempo computacional razonable. Es fácil de implementar y sólo necesita el ajuste de un solo parámetro.
- El algoritmo desarrollado es aplicable a muchas áreas como por ejemplo secuenciación de la producción, itinerarios, rutas, etc.
- Existen varios problemas abiertos para futuras investigaciones. Un problema es la convergencia del algoritmo, la técnica GLS es muy reciente y no existen estudios sobre su convergencia.

Otra investigación interesante es el ajuste automático del parámetro λ.

- La secuenciación óptima de los alces forma parte de un Sistema de soporte para la toma de decisiones a nivel estratégico, gerencial y de operación en la cosecha de la caña de azúcar.
- La contribución de este trabajo para los Ingenios del Valle del Cauca es la reducción de los costos de producción en la cosecha de la caña de azúcar. Al tener una secuenciación óptima de la cosecha se pueden disminuir horas extras, tiempo de espera, tiempos de parada de la fábrica. También se puede aumentar la capacidad de cada alce y cosechar más suertes en un mes.
- Para la aplicación del algoritmo en un ingenio se deben ajustar los parámetros de entrada al modelo. De la exactitud de estos parámetros depende el buen funcionamiento del algoritmo.

REFERENCIAS BIBLIOGRÁFICAS Y ELECTRÓNICAS

[1]. **ARBELAEZ, J., MELÓ A.** (1998). *"Propuesta para el mejoramiento de los tiempos de alce y transporte de caña de azúcar en el área de cosecha del Ingenio Providencia S.A."* Cali. Trabajo de grado (Ingeniero Industrial). Universidad Javeriana. Facultad de Ingeniería.

[2]. **DE BACKER, B. et al.** (1997). *Solving Vehicle Routing Problems using constraint programming and Metaheuristcs.* Journal of Heuristics, Vol. 21, No.1.

[3]. **CORMEN, T., LEISERSON, C., RIVEST R.** (1990). *Introduction to Algorithms.* Estados Unidos: Mit Press.

[4]. **DUMAS, Et al.** (1995). *An Optimal Algorithm for the Traveling Salesman Problem with Time Windows.* Operations Research, Vol. 43, No. 2 (March-April).

[5]. **FOULDS, L.** (1984). *Combinatoria! Optimization for Undergraduates.* New York: Springer - Verlag.

[6]. **GENDRAU, M., et al.** *"A Generalized Insertion Heuristic for the Traveling Salesman Problem with Time Windows".* Operations Research, Vol. 43, No. 3 (May-June 1998).

[7]. **HOLSTEIN, D., MOSCATO P.** (1998). *"Memetic Algorithms using Guided Local Search: a case study".* (Comunicación personal).

[8]. **MINGOZZI, A., BIANDO L., RICCIARDELLI S.** (1997). *"Dynamic Programming Strategies for the Traveling Salesman Problem with Time Window and Precedence Constraints".* Operations Research, Vol. 45, No. 3 (May-June)

[9]. **MONTGOMERY, D., PECK E.** (1982). *"Introduction to Linear Regression Analysis".* Estados Unidos.

[10]. **SHAFFER, C.** (1997). *"Introduction to Data Structures y Algorithm Analysis".* Estados Unidos. Prentice Hall.

[11]. **SKIENA S.** (1998). *"The Algorithm Design Manual".* New York: Springer Verlag.

[12]. **UZURIAGA, V.** (1998). *"Comprobación probabilística de resultados de programas para problemas de optimización en Teoría de Grafos". Cali.* Tesis (Magister en Matemáticas). Universidad del Valle, Facultad de Ciencias, Área de Matemáticas.

[13]. **VOUDOURIS, C.** (1997). *"Guided Local Search for combinatorial optimization problems".* Colchester. Ph.D. Thesis. University of Essex, Department of Computer Science.

[14]. **VOUDOURIS, C., TSANG E.** (1999). *"Guided Local Search and its application to the traveling salesman problem".* Europan Journal of Operations Research, Vol. 113.

[15]. **VOUDOURIS C., TSANG E.** (1995). *"Guided Local Search, Technical Report CSM-247".* University of Essex, Department of Computer Science. Colchester.

[16]. **WALPOLE R., MYERS R.** (1992). *"Probabilidad y Estadística".* Cuarta Edición: Mc Graw Hill.