PROPUESTA DE SOLUCIÓN AL PROBLEMA DE LOCALIZACIÓN DE CENTROS DE DISTRIBUCIÓN BASÁNDOSE EN LA META-HEURÍSTICA GRASP

TRABAJO DE GRADO
Presentado como requisito parcial para la obtención del título de
INGENIERO INDUSTRIAL

AUTORES:

ANDRÉS FELIPE APONTE PENAGOS **a.aponte@javeriana.edu.co**
PAULA ALEJANDRA ROSAS CASTRO **rosasp@javeriana.edu.co**

DIRECTOR:
ING. JAIRO RAFAEL MONTOYA TORRES Ph. D
CODIRECTOR:
ING. JUAN PABLO CABALLERO VILLALOBOS

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE PROCESOS PRODUCTIVOS
BOGOTÁ D. C.

MAYO, 2009

TABLA DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

En este trabajo[1] se presenta una metodología para solucionar un problema de localización de instalaciones, de múltiples productos en dos niveles (TUFLP) basada en la meta-heurística GRASP.

El TUFLP es un tipo de problema que se presenta con mucha frecuencia en la actualidad, dentro de las cadenas logísticas donde se necesita decidir qué lugares de almacenamiento, o puntos de servicio abrir o cerrar y cómo debe ser el flujo entre los niveles de la cadena (planta-lugar de almacenamiento, lugar de almacenamiento-clientes) de los diferentes productos a transportar buscando satisfacer toda la demanda y alcanzar los costos mínimos Por esta razón el problema ha sido estudiado desde distintas perspectivas utilizando herramientas que generan soluciones con altos costos computacionales, que no siempre son asequibles para las empresas, o cuyo tiempo de implementación supera al presupuestado para las necesidades del proyecto.

Uno de los principales aportes del trabajo es el uso de la metodología GRASP (procedimientos de búsqueda voraces y aleatorios), la cual no es tan conocida en la literatura como lo son TS (búsqueda Tabú) o ACO (Colonia de hormigas), es muy fácil de implementar y presenta resultados de buena calidad. Adicionalmente no existen evidencias del uso esta metodología para la solución del TUFLP. Los objetivos de este trabajo se detallan a continuación.

1. Objetivos

1.1. Objetivo general

Proponer un procedimiento meta-heurístico basado en el algoritmo GRASP (*Greedy Randomized Adaptive Search Procedure*) para resolver el problema de localización de instalaciones de dos niveles con múltiples productos (TUFLP).

[1] Los resultados de este trabajo se encuentran actualmente en proceso de redacción en la forma de artículo de investigación para ser enviado a una revista indexada (Aponte et al., 2009).

1.2. Objetivos específicos

A. Realizar un estado del arte acerca del problema bajo estudio con el fin de identificar la relevancia de los métodos que se han propuesto para resolverlo.
B. Proponer un algoritmo basado en la meta-heurística GRASP para resolver el problema TUFLP.
C. Analizar la eficiencia y eficacia del método propuesto a través de un estudio experimental con el fin de determinar las condiciones en las cuales el procedimiento tiene buen desempeño.

2. PLAN DE LECTURA

Debido a que el objetivo principal de este trabajo es proponer un procedimiento meta-heurístico (GRASP) para la solución de un problema de localización de instalaciones de dos eslabones con múltiples productos (TUFLP), es necesario, en primera instancia hablar sobre el problema de estudio, y la metodología de solución; y posteriormente realizar la explicación de la implementación de dicha metodología en el problema de estudio, hablar sobre los resultados obtenidos en esta y realizar recomendaciones para futuras investigaciones. Toda esta información es desarrollada a través de cuatro capítulos principales distribuidos de la siguiente forma:

El primer capítulo recopila la información teórica del problema de estudio, primero se describen los problemas de localización y su importancia y posteriormente se explica en mayor detalle en qué consiste el TUFLP y qué se ha realizado para solucionarlo.

El segundo capítulo habla de la metodología de solución implementada (GRASP), el cuál es un método de solución iterativo que consiste en una fase de construcción y una fase de mejora. Primero se hace una breve introducción mencionando algunos problemas donde se ha aplicado anteriormente, cuáles han sido los resultados, en qué consiste y qué parámetros utiliza. Finalmente se explican con detalle cada una de sus fases realizando un paralelo con la elaboración y funcionamiento del algoritmo diseñado para la solución del problema TUFLP siguiendo las pautas de la metodología.

El tercer capítulo hace referencia a los experimentos realizados para probar el rendimiento del algoritmo en la solución del TUFLP, mencionando los objetivos, implementación, representación y análisis de resultados de los experimentos.

Por último, el capitulo 4 presenta las conclusiones generales del trabajo desarrollado teniendo en cuenta los objetivos propuestos inicialmente y los resultados obtenidos en la fase experimental. Así mismo, este capítulo también presenta algunas sugerencias para trabajos futuros.

CAPITULO I
MARCO TEÓRICO

1. INTRODUCCIÓN A LOS PROBLEMAS DE LOCALIZACIÓN

Los problemas de localización parten de la necesidad de encontrar el sitio más conveniente para ubicar instalaciones como: una planta de producción, un centro de distribución, un vertedero de basuras, estaciones de policía, e inclusive una tienda. Dependiendo de la instalación que se quiera ubicar, se consideraran diferentes variables y objetivos en el problema. Por ejemplo, en el caso de una estación de policía, ésta se desea que este cerca de los lugares habitados, ya que entre más cerca este, el sitio ha de ser más seguro y puede dar seguridad a más personas.

Un modelo de localización tiene cuatro características básicas: (1) los clientes que se encuentran localizados en puntos o rutas dependiendo del caso, (2) instalaciones que deben ser localizadas, (3) un espacio en el que los clientes y las instalaciones deben ser localizadas, y (4) una métrica que indique las distancias, costos o tiempos entre los clientes y las instalaciones (ReVelle & Eiselt, 2005). Con estas características se espera que pueda responder a las siguientes cuatro preguntas (Daskin, 1995):

1. ¿Cuántas instalaciones debo utilizar/localizar?
2. ¿Dónde deben ser ubicadas?
3. ¿Qué tamaño deben tener?
4. ¿Cuáles clientes debe atender cada instalación para minimizar los costos o tiempos totales?

Durante muchos años se han estudiado este tipo de problemas, tanto desde el punto de vista teórico, buscando una solución eficiente a los problemas, como desde la práctica, analizando aplicaciones concretas para los sectores productivos público y privado. Inicialmente los matemáticos como Toricelli, Fermat, Silvester, Steiner, resolvieron problemas tales como: hallar el punto del plano tal que la suma de las distancias a tres puntos fijos en el mismo plano sea mínima; hallar el centro del círculo de mínimo radio que encierra a un conjunto de puntos dado; hallar el punto de un polígono fijo tal que sea máximo el radio del círculo con éste como centro y que no contenga a su interior a ninguno

de los puntos de un cierto conjunto P; hallar la forma de interconectar N puntos, de modo que se minimice la suma de las longitudes de los segmentos de conexión.

En el siglo XX, más precisamente a partir de la década de los años 1970, los problemas de localización han sido estudiados de forma continua por investigadores de diversas disciplinas, como la localización de: centros regionales de salud (Albernathy & Hershey, 1972), paraderos de bus (Gleason, 1975), localización de centros de emergencia (Moore & ReVelle, 1982), plantas de generación de energía eléctrica (Cohon et al., 1980), terminales de camiones (Birge & Malyshko, 1985), servicios aéreos (Flynn & Ratick, 1988), órbitas satelitales (Dezner, 1988), pluviómetros (Hogan, 1990), estaciones de inspección de vehículos (Hodgson et al., 1996), plantas de manufactura de procesadores (Cho & Sarrafzadeh, 1994), parqueaderos de vagones de carga (Higgins et al., 1977) (Current, 2002.).

Los modelos básicos más nombrados en la literatura de localización son citados por (Current, 2002.), donde el objetivo principal de todos los modelos es *localizar nuevas instalaciones para optimizar algún objetivo particular.* Los problemas de localización pueden clasificarse en 4 grandes grupos: problemas de cobertura, de centro, de dispersión y de media En las secciones siguiente se explicaran un poco más a fondo mostrando las diferencias que se vislumbran entre estos modelos y los objetivos de cada uno. En la sección 2 se estudiará en detalle el problema bajo estudio en este trabajo.

1.1. PROBLEMAS DE COBERTURA (Set-Covering)

El objetivo principal de este grupo de problemas es el de cubrir una demanda de forma total o parcial. En muchos casos, la distancia o tiempo de respuesta entre los clientes y los puntos que prestan los servicios, es decisiva para la satisfacción del cliente. Por ejemplo, si se incendia un edificio, el tiempo de respuesta de una estación de bomberos es vital, ya que si se demora en llegar es posible que el incendio haya consumido por completo el edificio.

En el ejemplo anterior el tiempo de la respuesta que tiene la estación de bomberos es un parámetro del problema que *Daskin (1995)* llama "estándar de calidad" o "cobertura del servicio". Los problemas de cobertura pueden tratarse de dos maneras:

1. Cobertura total *(set covering models)*: En este tipo de modelos se obtiene como respuesta el número mínimo de instalaciones necesarias para cubrir TODA la demanda, garantizando que cada nodo de demanda sea cubierto al menos por una instalación factible (una instalación es factible para el nodo de demanda *i* si la distancia entre los nodos no supera el parámetro de cobertura).

2. Máxima cobertura *(maximun covering)*: Dado que en el modelo anterior las soluciones tienden a hacerse costosas por el número de instalaciones a localizar, el modelo de máxima cobertura restringe el número de instalaciones que pueden ser localizadas y vuelca la función objetivo sobre la cobertura, maximizándola con un número de instalaciones fijas. Hay que tener en cuenta que en los dos modelos la cantidad demandada de los clientes es igual para todos, lo que en la vida real no sucede.

1.2. PROBLEMAS DE CENTRO (P-center)

En estos modelos a diferencia de los ya mencionados, el parámetro de cobertura que se usa es hallado de forma endógena (en los anteriores era una variable que entraba al problema, *exógena*). Este problema tiene como parámetro de entrada el número de instalaciones que se quieren instalar, y se pide al modelo que minimice la distancia más grande entre una instalación y el nodo de demanda más alejado que le sea asignado. En otras palabras, se está pidiendo que encuentre el parámetro de cobertura mínimo, tal que cubra todas las demandas dado un número finito de instalaciones. Adicionalmente, las soluciones de localización de estos modelos pueden restringirse únicamente a los nodos, lo que se conoce como "*p- center problem vertex"*, o permitir que puedan localizarse tanto en los nodos como en los arcos que los unen, conocido como *"P-center complete".*

1.3. PROBLEMAS DE DISPERCIÓN (P-dispertion)

En los modelos expuestos anteriormente, se asumió una actitud de deseo de cercanía con las instalaciones, tal como en el caso de las estaciones de policía. Pero el caso es completamente inverso cuando se trata, por ejemplo, de un vertedero de basuras, el cual se desea que esté alejado de las zonas habitables. Para que esto se cumpla, lo que el modelo busca es maximizar el mínimo de las distancias. Como se puede observar, éste es el caso completamente inverso al del numeral anterior.

1.4. PROBLEMAS DE MEDIA (P-median)

En los modelos anteriores se habló siempre de la maximización de algún beneficio, en particular de la demanda cubierta por las instalaciones. En este caso se da un giro en la concepción del modelo centrándose en la minimización de los costos. En muchos casos el costo de cubrir la demanda es directamente proporcional a la distancia de las instalaciones, pero en muchos casos no existe una relación lineal, creciendo de forma cóncava o convexa. El objetivo principal del *P-median problem* es el de encontrar la localización de

instalaciones de forma tal que se minimice el costo del servicio (costo asociado a las distancias entre los nodos de demanda y las instalaciones).

2. PROBLEMA DE LOCALZACIÓN TUFLP

El problema de ubicación de las instalaciones con capacidad ilimitada de dos eslabones (TUFLP), (two-echelon uncapacitated facility location problem) es una extensión del problema de producción/distribución de flujo de múltiples productos el cual a su vez, es de los principales problemas de localización existentes.

Este generaliza el problema de ubicación de instalaciones con capacidad ilimitada (UFLP) y el problema de ubicación de instalaciones con múltiples productos (MUFLP). Donde el UFLP (Uncapacited Facility Location Problem) es un problema de localización de instalaciones con capacidad ilimitada el cual tiene el objetivo de determinar que instalaciones se abrirán teniendo en cuenta el balance entre los costos fijos de operación y los costos variables de envío con el fin minimizarlos; y el MUFLP es un problema de localización que tiene en cuenta múltiples productos o múltiples actividades y tiene el objetivo de determinar cuáles tipos de productos o servicios de qué son transportados a la planta. Para algunos autores es considerado como una especie de problema de localización de instalaciones jerarquizado de 2 niveles, donde los costos fijos de localización del nivel inferior son determinados a través de la localización del nivel superior (Klose & Drexl, 2003). Estos problemas son considerados NP-Completos (Cornuejols et al., 1990), lo que significa que no es posible encontrar soluciones óptimas en un tiempo polinomial.Este modelo trabaja entonces múltiples productos y dos niveles de la cadena de abastecimiento, con el fin de localizar centros de distribución sin restricciones de capacidad en un conjunto finito de posibilidades a través de las cuales existe un flujo de productos diferentes entre sí que finalmente llegan a una demanda determinada (Gao & Robinson Jr., 1992).

2.1. IMPORTANCIA DEL PROBLEMA EN LOGÍSTICA

La gestión de la cadena de suministro (CS) es el proceso de planeación, implementación y control de forma efectiva y eficiente del flujo de bienes, servicios, e información, que vienen desde el punto de origen hasta el punto de consumo con el propósito de cumplir los requisitos del cliente (Council of supply Chain management professionals, 2008)

Parte de esta gestión es la planeación para encontrar la mejor configuración de la CS; esta debe ser flexible y ágil para responder rápidamente a los cambios del mercado (Machuca, 1995). Cuando se planea una CS es indispensable tener en cuenta tres factores vitales

(Shen & Daskin, 2005): (1) el costo de localización de una instalación en un lugar determinado; (2) el costo de manutención de los inventarios en ese lugar; y (3) el costo de distribución de los productos desde dicho lugar. Es acá donde las decisiones de localización adquieren un papel estratégico, haciendo que todas las áreas como: planeación, producción, abastecimiento, sistemas de distribución y ruteo, sean consideradas, para encontrar la configuración que se desea.

En general tiende a pensarse que este tipo de decisiones se hace solo cuando una empresa o negocio está siendo concebido, pero en realidad este problema no sólo afecta a los nuevos negocios, ya que pueden tomarse decisiones sobre como expandir una instalación ya existente, añadir nuevas instalaciones en nuevos lugares o trasladarlas a sitios diferentes (Chase & Aquiliano, 2004).

La frecuencia de estas decisiones depende del tipo de instalación y del tipo de empresa. Por ejemplo en empresas como bancos, cadenas de tiendas, restaurantes o empresas hoteleras, la frecuencia con la que se toman estas decisiones es más alta. Otras en cambio, las toman una vez en su historia, pero su incidencia en los costes y la calidad del servicio pueden llegar a ser mucho más relevantes (Sambola et al., 2004).

Debido a que estas decisiones afectan principalmente a los recursos financieros de largo plazo, y a la capacidad competitiva de la empresa (Lacefiel, 2005), se hace indispensable encontrar un balance entre estos dos aspectos con el fin de brindar el mejor servicio al menor costo.

En el TUFLP se encuentran dos eslabones de instalaciones a través de las cuales existe un flujo de productos diferentes entre sí que finalmente llegan a una demanda determinada. Este problema generaliza el problema de ubicación de instalaciones con capacidad ilimitada (UFLP) y el problema de ubicación de instalaciones con múltiples productos (MUFLP). Este modelo trabaja entonces múltiples productos y dos niveles de la CS, con el fin de localizar centros de distribución sin restricciones de capacidad en un conjunto finito de posibilidades (Gao & Robinson Jr., 1992).

Dentro de los modelos que hasta el momento han sido planteados y trabajados por diferentes investigadores, se encuentra un modelo que imita una cadena de suministro, adicionando características, y por ende complejidad, a los modelos básicos, de los cuales se hizo un recuento en la sección 1 de este trabajo.

Cuando se habla de la planeación de la cadena de suministro, es indispensable hablar de proveedores, distribuidores y de los clientes como un conjunto. Pero dentro de la literatura recopilada, que hace referencia al tema de localización de instalaciones, se encontró que los

modelos trabajados hasta el momento, abarcan, en su gran mayoría, solo dos eslabones de la CS (Melo et al., 2008), ya sea de un proveedor a un centro de distribución, o de un centro de distribución al cliente, o simplemente trabajan con un solo tipo de producto, lo que en realidad no es necesariamente cierto, especialmente cuando se habla de una compañía completa y todo su portafolio.

Por las razones anteriormente mencionadas, se considera importante estudiar la formulación de un problema que tenga en cuenta la interacción de varios actores en diferentes niveles, tal como sucede en la cadena de suministro entre proveedores, distribuidores y clientes. Adicionalmente, el modelo debe incluir la distinción de los productos que fluyen entre los actores de la CS con el fin de que se ajuste más a la realidad. Dadas estas dos características el modelo que se desea abordar en este trabajo, cuyo planteamiento lo hace Daskin (1995), es un problema determinístico, que incluye dos niveles de jerarquización, un nivel de localización (centros de distribución), y múltiples productos con el fin de hallar la mejor configuración para la cadena de suministro, de forma tal que se minimicen los costos. Dicho problema es conocido en la literatura como TUFLP.

2.2. DEFINICIÓN DEL MODELO MATEMÁTICO

Según el libro de Daskin (1995) el TUFLP es una extensión del problema de producción/distribución de flujo de múltiples productos y consiste en determinar el número de centros de distribución a ubicar, el sitio donde se ubicarán estos centros de distribución y como deberá funcionar el flujo de producto entre las instalaciones. A continuación se presenta y explica la formulación formal que hace el autor de este problema.

PARÁMETROS

$\boldsymbol{h_i^k}$: Demanda del producto k en el mercado i
$\boldsymbol{f_j}$: Costo de localizar ina instalacion en el sitio candidato i
$\boldsymbol{c_{ijm}^k}$: Costo de llevar una unidad deproducto k de la
planta m al cliente i a traves de la instalación j
$\boldsymbol{S_m^k}$: Capacidad de producir e lproducto k en la planta m
$(asume\ independencea\ de\ producción\ entre\ los\ productos)$
$\boldsymbol{M}$: Un valor muy grande

Variables:

Y_{ijm}^{k} = flujo del producto k desde la planta m hasta el cliente i a traves de j

$$X_j = \begin{cases} 1 & \text{si localiza una instalacion en el sito candidato } j \\ 0 & \text{otro caso} \end{cases}$$

Función Objetivo:

$$Min\ Z = \sum_j f_j X_j + \sum_i \sum_j \sum_m \sum_k c_{ijm}^{k} Y_{ijm}^{k} \qquad (1)$$

Sujeto a:

$$\sum_i \sum_m \sum_k Y_{ijm}^{k} \leq M X_j \qquad \forall\, j \qquad (2)$$

$$\sum_j \sum_m Y_{ijm}^{k} \geq h_i^{k} \qquad \forall\, i,k \qquad (3)$$

$$\sum_i \sum_j Y_{ijm}^{k} \leq S_m^{k} \qquad \forall\, m,k \qquad (4)$$

$$X_j \in \{0,1\} \qquad \forall\, j \qquad (5)$$

$$Y_{ijm}^{k} \geq 0 \qquad \forall\ i,j,m,k \qquad (6)$$

La función objetivo (1) minimiza el costo fijo de localización de los centros de distribución y el costo de transportar una unidad de un producto a través de la red. Es de notar que númerosos criterios de costos pueden ser adicionados en lavariable de costos $\left(c_{mji}^{k}\right)$ tales como el costo de producción y el transporte a través de la red. El costado izquierdo del grupo de restricciones (2) es el flujo a través de un candidato a ser abierto, restringiendo lo a ser exclusivamente mayor a uno solo en el caso que el candidato sea abiero. Las restricciones del siguiente grupo (3) oblian a que toda la demanda de los k productos sea cubierta en su totalidad. Adicionalmente el grupo (4) de restricciones considera las capacidades de producción, cabe aclarar que estas deben ser capaces de cubrir el total de la demanda para poder tener soluciones factibles al problema. Finalmente (5) y (6) son restricciones de no negatividad y variables binarias de decisión respectivamente.

ESQUEMA

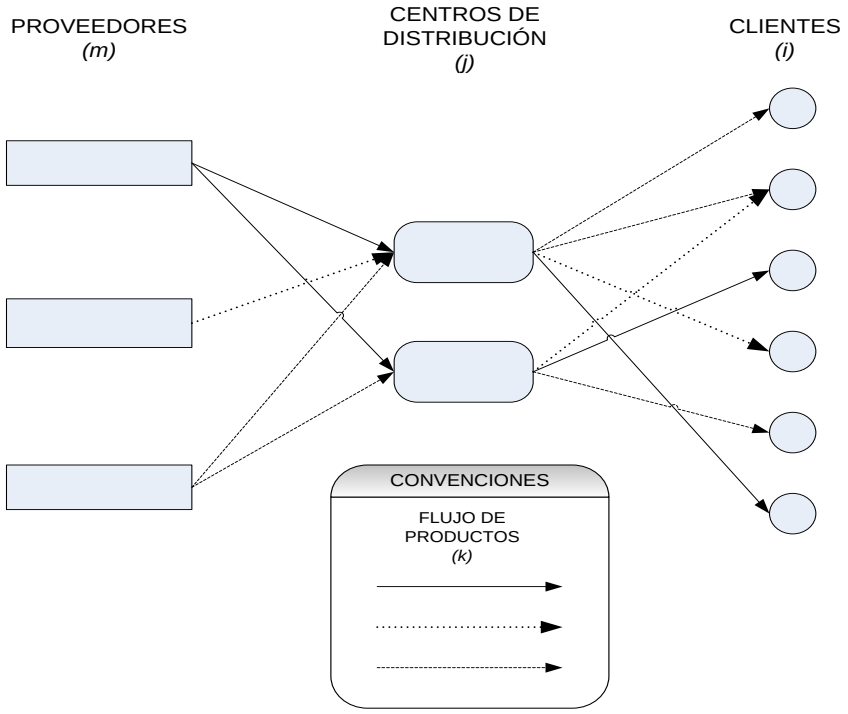

Figura 1: Esquema de representación del problema

2.3. REVISIÓN DE LA LITERATURA: ANTECEDENTES DE SOLUCIÓN DEL TUFLP

Acerca de modelos de localización de sistemas de producción-distribución existe una amplia literatura, Smits(2001) en su articulo realiza una clasificación de los modelos teniendo en cuenta 15 criterios (ver ANEXO 2) establecidos anteriormente por Aikens (1985) y Vidal y Goetschalckx (1996) quienes resumen los diferentes modelos de producción distribución realizados hasta esa fecha, haciendo énfasis en la parte global de la cadena de abastecimiento (Vidal & Goetschalckx, 1997). Zhi-Long Chen (2001) realizan otra clasificación basada en el nivel de decisión, estructura y parámetros de los modelos existentes (Chen, 2004).

Dentro de este contexto, puede verse que casi todos los modelos son formulados con Programación Mixta Entera (MIP) y debido a que las formas exactas para resolver este tipo de problemas en tamaños medianos requieren de grandes esfuerzos computacionales las aproximaciones heurísticas son usadas para resolver los problemas reales. Las principales técnicas utilizadas son modelos de descomposición, jerarquización, relajación de

Lagrangeana, ramificación y acotamiento y algunas meta-heurísticas como: recocido simulado y algoritmos genéticos. (Pirkul & Jayaraman, 1998). El ANEXO 1 muestra un resumen de los principales avances que se han realizado para solucionar problemas de producción/distribución.

2.3.1. TÉCNICAS DE DESCOMPOSICIÓN Y APROXIMACIÓN

Existen dos técnicas clásicas que permiten obtener soluciones exactas al problema llamadas descomposición de Benders y ramificación y acotamiento, las cuales resultan adecuadas y eficientes para resolver problemas de localización de plantas en tamaño pequeño (Miguel Angel Gutiérrez Andrade, 2000). A continuación se describe brevemente este tipo de técnicas

2.3.1.1. DESCOMPOSICIÓN DE BENDERS

Este método de solución consiste en dividir el problema en dos: uno (maestro) que trata el problema original sin algunas restricciones y maneja variables enteras (ubicar los centros de distribución); y otro (auxiliar) que maneja variables continuas (flujo de productos) y utiliza los resultados obtenidos del problema maestro añadiendo restricciones hasta resolver el problema original, lo que permite converger a una solución con pocas iteraciones.

Geoffrion y Graves en 1974, quienes fueron los pioneros en el desarrollo de los problemas de producción-distribución, emplearon un algoritmo basado en Descomposición de Benders para resolver el problema de *multicomodites* en un solo periodo (Geoffrion & Graves, 1974). Este modelo es el que más se acerca al que se pretende desarrollar en este trabajo de grado, se diferencia en que la demanda debe ser satisfecha solamente por un único centro de distribución.

Está técnica de descomposición ha seguido siendo utilizada para resolver problemas relacionados con localización de instalaciones con capacidad limitada, entre los autores se puede mencionar Dogan y Goeteschlackx.

2.3.1.2. RAMIFICACIÓN Y ACOTAMIENTO

Es una técnica de exploración y en la mayor parte de las ocasiones trata de buscar una solución óptima a un problema, consiste en una enumeración en árbol en el cual el espacio de variables enteras se divide de forma sucesiva dando lugar a sub-problemas lineales que se resuelven en cada nodo del árbol. En el nodo inicial las variables enteras se relajan como variables continuas, de tal forma que se les permite tomar valores fraccionarios. Si la

solución de este problema produce de forma natural una solución en la cual todas las variables y toman valores enteros se habría alcanzado la solución, en otro caso se comienza una búsqueda en árbol, que consiste en calcular en cada nodo una cota de los valores de las soluciones situadas por debajo. Si esta cota demuestra que tales soluciones son eventualmente peores que una solución ya encontrada, se abandona la exploración de esa parte del árbol, hasta que la diferencia de cotas inferior y superior estén dentro de una tolerancia o no existan ramas abiertas (José A. Caballero, 2007)

Hindi y Basta en 1994 utilizaron esta técnica para trabajar el mismo modelo de Geoffrion & Graves, añadiendo la restricción de capacidad limitada, calculando los límites inferiores a través de transformaciones estructuradas (Hindi & Basta, 1994).

2.3.1.3. RELAJACIÓN LAGRANGIANA

Consiste en eliminar las restricciones que unen los diferentes dominios de planificación del conjunto de restricciones y añadirlas a la función objetivo. En forma matemática, se añade el término λ $(A\ i\ X - B\ i)$ donde λ es un vector o multiplicador de Lagrange λi. Sin las restricciones que unen los dominios de planificación, el problema queda dividido en N sub-problemas que pueden resolverse individualmente. Dado que las soluciones óptimas dependen de $\lambda\ i$, estos valores se pueden utilizar para coordinar las soluciones de los sub-modelos. Esto se consigue mediante un procedimiento iterativo donde, dado un valor de $\lambda\ i$, se analiza si se viola alguna de las restricciones relajadas que, en caso afirmativo, provoca un ajuste del valor de los parámetros. Si el modelo contiene variables binarias el método iterativo puede presentar dificultades para encontrar soluciones globalmente factibles y se suele proceder a métodos heurísticos para "arreglar" soluciones obtenidas que preserven la factibilidad.

Una aplicación de esta técnica en modelos parecidos al que se está estudiando en este trabajo puede verse en Mazoola y Neebe (Joseph B. Mazzola, 1998), quienes solucionan el problema de localización de instalaciones de múltiples productos con capacidad limitada utilizando relajadores lagrangeanos, los cuales generan cotas inferiores para el algoritmo de Ramificación y Acotamiento, el cual descompone el problema en pequeños sub-problemas UFL (problema de localización de instalación con capacidad infinita) fáciles de resolver.

En 1997 Pirkul & Jayaraman basaron también una heurística en relajadores Langrangeanos (ayudaban a encontrar cotas inferiores en cada iteración) para resolver el modelo llamado (PLANWARE), donde estudian la importancia de la coordinación entre la planeación de la producción y distribución, y el objetivo es determinar además del número centros de distribución, el número de plantas y las cantidades de los envíos necesarias para satisfacer

la demanda minimizando los costos. Otras aplicaciones de esta técnica en problemas de producción-distribución pueden verse en autores como Hasan y Pirkul, Barabroslgu, Yilmaz y Catay (2006), entre otros.

2.3.2. MÉTODOS DE SOLUCIÓN META-HEURÍSTICOS

2.3.2.1. RECOCIDO SIMULADO

Jayaraman y (Ross, 2003) utilizan esta meta-heurística (SA) describen un modelo llamado PLOT, para el diseño de sistemas de distribución, caracterizado por múltiples productos, una planta central, múltiples centros de distribución y centros de *crossdocking*, con el objetivo determinar la mejor configuración de la red de centros de distribución y crossdocking y la cantidad de productos transportados, ellos compararon los resultados obtenidos con los óptimos(utilizando LINGO) y encontraron diferencias menores al 4% ; sin embargo el tiempo de solución oscilaba entre 300 y 400 veces menos.

2.3.2.2. ALGORITMOS GENÉTICOS

Jang Chang y Park (2002) utilizan esta meta-heurística para el diseño de una red de producción/distribución el cual es dividido en 4 módulos (abastecimiento, planeación de la producción y distribución de materiales de proveedores a clientes, administración y manejo de datos). El primer módulo fue resuelto con una heurística utilizando relajación Lagrangeana y el segundo módulo, debido a que la planeación de las actividades ocurre frecuentemente y el tiempo de toma de decisiones es corto, utilizó algoritmos genéticos lo que permite generar buenas teniendo en cuenta la calidad y el tiempo.

Gen y Syarifm (2005) estudian la integración más eficiente del sistema de producción, distribución e inventarios para múltiples productos y en múltiples periodos de tiempo, utilizando esta técnica junto con controladores de lógica difusa (Fuzzy Logic Controller) los cuales ayudan a auto-regular los parámetros necesarios para implementar GA. Los autores se comparan los resultados obtenidos con el GA estándar y el GA con FCL, mostrando mayor eficiencia en el segundo método.

2.3.3. EL MODELO DE PRODUCCIÓN/DISTRIBUCIÓN

Los modelos de Geoffrion & Graves, Hindi y Basta Pirkul & Jayaraman son los que más se asemejan al problema a tratar, sin embargo existen otros modelos que estudian el problema de producción/distribución, pero poseen características diferentes como demanda estocástica, diferente número de eslabones, único producto, etc. Entre estos puede mencionarse a: (Hodder & Dincer, 1986), (Cohen & Lee, 1988), (Zuo et al., 1991), (Barbarosogl & Ozgur, 1999), (Mohamed, 1999), (Dhaenens-Flipo & Finke, 2001), (Jayaraman & Pirkul, 2001), (Jang et al., 2002), (Yilmaz & Catay, 2006)

(Hodder & Dincer, 1986), consideran aspectos financieros en localización de instalaciones a nivel internacional, utilizando el acercamiento de múltiples factores para transformar un problema cuadrático de gran escala MIP en uno más sencillo (Cohen & Lee, 1988) manejan demandas estocásticas y cuatro etapas (vendedores, proveedores de materias primas a plantas, plantas de producción, centros de distribución y clientes) con el objetivo de determinar el tamaño de las ordenes, minimizando costos, proponiendo una aproximación heurística de jerarquización. (Zuo et al., 1991), realiza una aplicación práctica (múltiples productos y múltiples demandas) en cultivos de producción y distribución de semillas de maíz. (Barbarosogl & Ozgur, 1999), estudian dos eslabones (múltiples productos, 1 planta, múltiples centros de distribución y múltiples clientes,) empleando el método de relajadores Langrangeanos. (Barbarosogl & Ozgur, 1999), desarrolla el modelo de dos eslabones, múltiples productos, múltiples instalaciones y múltiples mercados en diferentes países. (Dhaenens-Flipo & Finke, 2001) manejan tres eslabones, múltiples productos, múltiples plantas, resolviéndolo con CPLEX.

2.4. REVISIÓN DE LA LITERATURA: GRASP

La meta-heurística GRASP (*Greedy Randomized Adaptive Search Procedure*) aparece entre finales de la década de los 80 y principios de la década de los 90, desde entonces ha sido utilizada en gran variedad de problemas computacionalmente complejos, incluyendo aplicaciones de tipo experimental donde, con ayuda de herramientas estadísticas se compara su desempeño con respecto a otras meta-heurísticas o resultados óptimos encontrados, o aplicaciones reales, donde se utiliza para desarrollar mejoras en diferentes industrias. Para este trabajo nos interesaremos más en las aplicaciones de tipo experimental, ya que son éstas las que nos brindan información de la calidad de la solución. A continuación haremos un pequeño recuento de esta trayectoria con el fin de mostrar la validez de su uso en nuestro tema de estudio.

En un principio fue probada en problemas básicos tales como el del agente viajero (Chardaire et al., 2001) y n-reinas (Troncoso & Barriga, 2008). Posteriormente es utilizada para realizar una aproximación al problema de carga de cajas en un contenedor. En este se hacen pruebas comparativas en dos grupos de problemas diferentes: En el primero el tamaño del contenedor es variable y se genera en casi todos los casos espacio suficiente para almacenar todas las cajas que se desean guardar, y en el segundo la cantidad de cajas diferentes a guardar es variable, y se distingue entre problemas con alta o baja disparidad según las diferencias entre los tamaños de las cajas. Las pruebas comparativas utilizaron Meta-heurísticas tales como Búsqueda Tabú, recocido simulado, GRASP y búsqueda iterada obteniendo en general los mejores resultados con GRASP (Oliveira & Moura, 2005)

De este problema se desprenden otros que se ven comúnmente en procesos productivos del acero, el papel, la madera, el vidrio y los textiles entre otros, dichos problemas consisten en maximizar el número de elementos que se pueden cortar de una placa de determinado de material. Al comparar la metodología basada en el algoritmo GRASP se obtuvieron los mejores resultados (Alvarez et al., 2008) superando las soluciones basadas en otras metodologías como Búsqueda Tabú (Burke, 2006), Algoritmos Genéticos (Bortfeld, 2005) y Recocido Simulado (Dowsland, 1993).

Para citar un ejemplo más cercano al que hace referencia el presente trabajo, en un artículo publicado por la revista *INFORMS Journal on Computing* se compara el desempeño de dos meta-heurísticas en dos tipos de instancias diferentes, en el problema de programación y asignación de maquinaria en paralelo. Dicha comparación muestra a GRASP como la más eficiente en los dos tipos de instancias usadas. (Bard & Rojanasoonthon, 2005)

Con respecto a los problemas de localización de instalaciones, GRASP ha sido utilizado, obteniendo muy buenos resultados, como ejemplo puede citarse a (Resende & Werneck, 2002) , quienes estudian el problema de p-median, utilizando diferentes tipos de instancias disponibles en la literatura. Aunque los autores aclaran que el objetivo del estudio no pretende mostrar la metodología basada en GRASP como la mejor, los resultados estadísticos revelan un rendimiento de GRASP superior a la heurística con la que se compara.

Por último se hace referencia una solución hibrida de GRASP con *Scatter Search* a un problema un problema de diseño de redes con múltiples productos básicos o no diferenciados (Alvarez et al., 2005) En este articulo comparan GRASP tradicional y GRASP con memoria, la instancias fueron generadas aleatoriamente y probadas con 5 estrategias diferentes donde GRASP con memoria resulto ser más lento pero con mejor calidad en las respuestas. Los resultados computacionales muestran las ventajas de usar esta metodología

Aunque se trate de mostrar la amplia gama de problemas en los que se ha implementado como solución un algoritmo basado en GRASP, es difícil hacer un recuento minucioso de las aplicaciones que se han hecho, sin embargo es posible encontrar una amplia colección con más de 230 artículos minuciosamente escogidos por uno de los pioneros en el uso de dicha meta-heurística, Mauricio Resende, en http://www.research.att.com/~mgcr/grasp/gannbib/graspbib/index.html.

De las metas-heurísticas hasta el momento conocidas como Búsqueda Tabú, Algoritmos Evolutivos, GRASP y recocido simulado, GRASP es la mejor opción encontrando resultados de muy buena calidad en tiempo record sobre las demás. Esto es un excelente indicio para pensar en esta metodología es ideal para el problema propuesto en el presente trabajo (Delmaire et al., 1999).

CAPITULO II
METODOLOGÍA DE SOLUCIÓN GRASP (GREEDY RANDOMIZED ADAPTIVE SEARCH PROCEDURE)

1. INTRODUCCIÓN A LA METODOLOGÍA

Como se mencionó en el capitulo anterior, las meta-heurísticas son estrategias para el diseño de algoritmos generales que solucionan problemas de alta complejidad, estos algoritmos fusionan métodos de solución de alto nivel y métodos de optimización locales. Siguiendo los pasos marcados por la meta-heurística, se pueden generar soluciones de alta calidad para la toma asistida de decisiones (de localización en nuestro caso de estudio).

Entre las meta-heurísticas más exitosas que aparecieron en los últimos años del siglo pasado se encuentra GRASP (*Greedy Randomized Adaptive Search Procedure)*, un método multi-arranque diseñado para resolver problemas difíciles en optimización combinatoria (Mauricio G. C. Resende, 2003) del que se presenta el seudo código Figura 2: Bloque principal de GRASPFigura 2. Se basa en la premisa de que soluciones iniciales, diversas y de buena calidad, juegan un papel importante en el éxito de métodos locales de búsqueda.

Esta meta-heurística consta de dos fases, La primera una fase constructiva donde se generan soluciones factibles de forma *aleatoria*, basándose en criterios relacionados con el problema con el fin de garantizar su calidad. La segunda fase es una búsqueda local que parte de la solución entregada en la construcción y examina sus vecindades con el fin de mejorarla, incluso hasta encontrar un óptimo local. Este proceso se puede ver en la Figura 2 donde se presenta el seudo código del bloque principal del algoritmo.

La ejecución de cada una de las fases del algoritmo es repetida varias veces, según el criterio de terminación, que usualmente se define como el número de veces que el algoritmo se repite sin encontrar mejora o cuando el valor de la solución se acerca a algún valor determinado.

Rutina $GRASP(MAXIteraciones, Alfa)$
1. $Datos \leftarrow Leer\_datos\ ()$
2. ***for*** $k = 1, \ldots, Max\_Iteraciones$ ***do***
3. $Solución \leftarrow Fase\_Construcción\ (Alfa)$
4. $Solución \leftarrow BusquedaLocal\ (Solucion)$
5. ***Call*** $Actualizar\ (Solución, Mejor\_Solución)$
6. ***end***
7. ***Return*** $Mejor\_Solución$

end GRASP.

Figura 2: Bloque principal de GRASP

Como puede verse, una de las grandes ventajas de esta meta-heurística es su sencilla implementación ya que solo cuenta con dos parámetros, Alfa y Máximo de iteraciones, que corresponden a los valores utilizados para construir la lista de candidatos y para determinar el criterio de parada del algoritmo.

Cada una de las fases y parámetros necesarios para correr al algoritmo se describirán de forma más específica en las secciones siguientes, explicando tanto el objetivo de dichas fases como la forma en que se implementaron para el desarrollo del presente trabajo.

1.1. CONSTRUCCIÓN ALEATORIA DE LA SOLUCIÓN

Este es el primer paso de la meta-heurística, donde se busca generar soluciones completas factibles. Dicha solución se genera desde cero cada vez que se inicia una iteración del bloque principal del algoritmo.

Las soluciones generadas se construyen de manera iterativa añadiendo un elemento en cada paso. Para la selección de estos elementos se utiliza una función de utilidad o *función greedy*, la cual mide la contribución local de cada elemento a la solución que se haya construido hasta ese momento o solución parcial. Con base a esa contribución local se obtiene información de cuáles son los mejores o los peores elementos, y se crea una lista restringida de candidatos (RCL por sus siglas en ingles), con los mejores elementos, donde se le asigna a cada uno un valor de probabilidad para ser escogidos aleatoriamente y posteriormente adicionarlos a la solución parcial.

La construcción de las soluciones comienza por escoger una a una de mayor a menor las demandas representadas con el parámetro h_{ki} en el modelo (véase sección 2.2 Capítulo I).

Si se nota $h_{k'i'}$ como la demanda escogida se obtienen de acá tanto el cliente $i' \epsilon\, i$ como el producto $k' \epsilon\, k$ que conforman las dos dimensiones de este parámetro.

Posteriormente, para evaluar las diferentes posibilidades y la contribución de cada una a la función objetivo, se tienen en cuenta los costos de transporte contenidos en el parámetro c_{mji}^{k} y los costos de localización de las posibles instalaciones dados por el parámetro f_j. Dichas posibilidades o candidatos están definidos como las formas posibles de cumplir la demanda que se escogió anteriormente. Debido a que la demanda está ligada a un cliente y a un producto, se hace necesario evaluar las diferentes combinaciones de plantas e instalaciones, para lo cual se usa una función de utilidad o función greedy obteniendo también como resultado el valor que se asignara a una de las variables Y (véase sección 1.1.1). Puede pensarse hasta este punto que los valores que se asignan a las variables son los mismos de la demanda que se escogió en el principio de la iteración, sin embargo los valores asignados no son siempre iguales a este valor, ya que es posible que alguna de las plantas no tenga la capacidad suficiente para cumplir con la demanda, situación que se explicara en detalle más adelante.

Tras encontrar las contribuciones de cada posibilidad, se construye la RCL en base a un parámetro alfa que determina un *umbral* de aceptación de candidatos. Luego de tener la lista de candidatos completa se le asigna a cada uno una probabilidad según su contribución en la función objetivo, para más tarde escoger uno de forma aleatoria y adicionarlo a la solución.

Cuando un candidato es adicionado a la solución parcial, un valor es asignado a la variable $Y_{m'j'i'}^{k'}$, y se hace necesario actualizar los valores de los parámetros de demanda ($h_{k'i'}$), capacidad ($s_{k'm'}$) y costos fijos de localización ($f_{j'}$) de la instancia que se esté trabajando. Los valores de los parámetros dependerán del valor que tome la variable $Y_{m'j'i'}^{k'}$, siguiendo las reglas que se enumeran a continuación:

1. Si el valor de la variable ($Y_{m'j'i'}^{k'}$) es igual a la capacidad ($s_{k'm'}$), entonces el valor de la capacidad de la planta para producir el producto k' será cero y el nuevo valor de la demanda del mismo producto para el cliente i' será la diferencia entre los dos parámetros $\|s_{k'm'} - h_{k'i'}\|$, siendo este un valor mayor a cero indicará que la demanda no ha sido cubierta por completo, obligando al algoritmo a iterar nuevamente.
2. Si el valor de la variable ($Y_{m'j'i'}^{k'}$) es igual a la demanda ($h_{k'i'}$), entonces el valor de esta será cero, indicando que no habrá más demanda del producto por parte del cliente y el nuevo valor de capacidad será la diferencia entre los dos parámetros $\|s_{k'm'} - h_{k'i'}\|$, siendo este un valor mayor a cero indicará que la planta aun tiene capacidad de producir el producto.

3. Si el valor del costo fijo de instalación ($f_{j'}$) es exclusivamente mayor a cero, este se fijara en cero, caso en el cual significara que la instalación j' fue escogida y que no tendrá un costo de localización la próxima vez que se use.

Finalmente, para terminar la fase constructiva se verifica que la solución hasta el momento construida esta completa utilizando los valores del parámetro de demanda, buscando que $\sum_k \sum_i h_{ki}$ sea igual a cero, lo que indicará que toda la demanda fue cubierta satisfactoriamente tal y como se muestra en la estructura *while* mostrada en el seudo código de la Figura 3

```
Fase_Construccion (Alfa)
1.   Cargar_Valores()
2.   CDatos ← CopiarDatos(Instancia)
3.   ResetSolucion (Solucion)
4.   DemTotal = 1
5.   While ∑k ∑i hki > 0
6.      hk'i' ← EscogerDemanda(CDatos)
7.      Evaluaciones() ← EvaluarCostosIncrementales (k', i', hk'i')
8.      RCL ← CrearRCL(Alfa, Evaluaciones)
9.      m', j', Ym'j'i'k' ← EscogerCandidato(RCL)
10.     Actualice_SolucParcial(k', m', j', i', Ym'j'i'k')
11.  Wend
End Sub
```

Figura 3: Fase de construcción

1.1.1. FUNCIÓN GREEDY

Como se mencionó en la sección anterior, la construcción busca adicionar candidatos a una solución parcial de forma iterativa hasta llegar una solución completa y factible. Con el fin de evaluar siempre con un mismo criterio a los candidatos, y que estos tangan diferentes probabilidades de ser escogidos, la función de utilidad o función greedy mide la contribución local de cada candidato que sea posible adicionar a la solución parcial en cada una de las iteraciones de la construcción.

Los valores de evaluación encontrados con la función greedy varían en cada iteración del algoritmo. Esta variación da al algoritmo la cualidad voraz, ya que permite que un candidato sea tomado en cuenta o no para la construcción de la solución.

En la implementación del algoritmo GRASP para el presente trabajo, se evalúan todos los posibles candidatos a ser adicionados en la solución tras escoger una demanda en particular, lo que resulta en un arreglo de valores de tamaño $m \times j$. Los valores de cada uno de los espacios del arreglo se llenan con base a los parámetros resultantes de la iteración anterior de la construcción, calculando una evaluación para cada uno de los candidatos como se muestra más adelante en el seudo código Figura 4: Evaluar costos incrementales.

EvaluarCostosIncrementales $(k', i', h_{k'i'})$
1. $Demanda = h_{k'i'}$
2. ***While*** $Evaluaciones()$ NO $esté$ $lleno$
3. $CostoVariable = c_{mji'}^{k'}$
4. $Capacidad = s_{k'm}$
5. $CostoFijo = f_j$
6. ***if*** $Capacidad = 0$
7. ***then***:
8. $Evaluaciones(m, j) = 0$
9. ***else***:
10. $Evaluaciones(m, j) = CostoVariable * Minimo(Capacidad, Demanda) + CostoFijo$
11. ***End If***
12. $Next\ m \times j$
13. ***While end***

End Sub

Figura 4: Evaluar costos incrementales

1.1.2. LISTA DE CANDIDATOS

La lista de candidatos o RCL tiene como objetivo reducir el número de candidatos que es posible escoger para adicionar a la solución que este en construcción, dejando como únicos elementos los candidatos con las mejores evaluaciones.

Según la literatura esta escogencia se puede hacer de dos maneras: la primera es la selección de un parámetro fijo que representaría el número de candidatos que harían parte de la RCL, en este caso la lista tendría en todas las iteraciones un tamaño constante. La segunda forma es utilizar un parámetro $\propto \epsilon[0,1]$ con el que obtendría un valor umbral, el cual servirá de criterio para seleccionar los candidatos de la RCL, dicho umbral será un número contenido en el rango de valores de todas las evaluaciones.

Sub Rutina $ConstruirRCL(Evaluaciones, \alpha)$

1. $umbral = c^{min} + \alpha'^{(c^{max} - c^{min})}$
2. **While** $\sim(fin\ de\ Evaluaciones(\ )\ )$
3. **If** $Evaluaciones\ (c)\ \leq\ umbral$ **then**
4. $RCL\ (a\ ) =\ Evaluaciones(c)$
5. **End if**
6. **While end**
7. **Call** $AsignarProbabilidades(RCL)$
8. **Return** RCL

End $ConstruirRCL$

Figura 5: Construcción de la lista de candidatos

Si se considera un problema de minimización, como el que se presenta en este trabajo, y se utiliza el parámetro Alfa como método de construcción de la RCL, entonces esta estará compuesta por todos los candidatos cuya evaluación sea menor al valor umbral que se halle. Para este efecto es necesario conocer los valores máximo y mínimo de las evaluaciones como se muestra en el seudo código de la Figura 5.

La escogencia apropiada del valor Alfa para la RCL es crítica y relevante para alcanzar un buen balance entre tiempo de cómputo y calidad de las soluciones, debido a que la media y la variancia de la distribución de las soluciones creadas en cada iteración dependen de la naturaleza de la RCL. Como puede verse, la selección de un valor Alfa = 0 resultará en la adición exclusivamente de los candidatos con las mejores evaluaciones en todas las iteraciones, lo que se conoce como un algoritmo totalmente voraz y todas las iteraciones arrojarían el mismo valor. Por el contrario, si se escogiese un valor Alfa=1, el algoritmo no haría ningún tipo de selección haciendo adiciones de candidatos a la solución de forma aleatoria.

Para concluir la construcción de la RCL, es necesario cambiar los valores de las evaluaciones con que se escogieron los candidatos, por un valor de probabilidad inversamente proporcional dichas evaluaciones. La proporcionalidad entre las probabilidades y los valores de evaluación hará que los candidatos con las mejores evaluaciones tengan mayor probabilidad de ser adicionados a la solución en construcción.

Tras tener las probabilidades para cada uno de los candidatos, se genera un número aleatorio ceñido a una distribución uniforme, el cual es usado para escoger uno de los candidatos de la RCL y adicionarlo a la solución en construcción.

1.2. BÚSQUEDA LOCAL

Dado que la fase de construcción no garantiza optimalidad con respecto al entorno en que se está trabajando, se requiere esta fase, la cual explora repetidamente la vecindad de una solución inicial en busca de una mejor solución. Esta búsqueda juega un papel importante ya que sirve para buscar soluciones localmente óptimas en regiones prometedoras del espacio de solución (Mauricio G. C. Resende, 2003)

Primero, se define una estructura de vecindad, tal que cada solución puede ser alcanzada a partir de otra mediante una operación simple denominada movimiento. El proceso iterativo se repite hasta que no se pueda encontrar una solución vecina que mejore el valor de la función objetivo de la solución actual.

En este proyecto se observo que, una vez las instalaciones son seleccionadas (se eliminan las variables binarias del modelo), el problema se convierte en un problema de transporte, el cual es más sencillo de resolver y ya ha sido estudiado anteriormente, por lo que en la literatura ya existen algoritmos de alta eficiencia que lo solucionan. Uno de los más utilizados es el método simplex, el cual es un algoritmo iterativo que parte de un punto extremo de la región de soluciones factibles de un problema lineal y va a otro punto extremo según el incremento de la función objetivo, se detiene hasta encontrar el óptimo local. Este algoritmo es utilizado por diferentes programas computacionales que resuelven problemas de programación lineal como LINGO, EXPRESS, GAMMS, etc. De esta forma utilizando simplex se logra resolver el problema de transporte llegando a encontrar la configuración óptima entre los clientes, productos, plantas e instalaciones elegidas.

Con fines prácticos y debido a que se busca que la metodología pueda ser implementada con facilidad, se usó un programa de licencia libre llamado GLPK (*GNU Linear programming kit*), el cual un conjunto de rutinas escritas en lenguaje de programación ANSI que resuelven problemas lineales y mixtos; entre estas subrutinas se encuentra el método simplex, el algoritmo de punto interior y el algoritmo de ramificación y acotamiento.

En la Figura 6 se muestra el seudo código de la Búsqueda local GRASP utilizado para la implementación en el presente trabajo.

```
Sub Rutina Búsqueda Local ()
1   Datos ordenados  <- OrganizarDatos(cantidad)
2   Solución  <- glpk
End BusquedaLocal
```

Figura 6: Búsqueda local GRASP

En el seudocódigo se observa que para esta fase se usa un software llamado GLPK, que como se menciono anteriormente utiliza el método simplex para solucionar el problema de transporte asociado entre las plantas, productos, instalaciones seleccionadas y clientes. Este algoritmo permite resolver problemas de programación lineal que se encuentran bajo la forma estándar, por lo que es necesario organizar los datos primero de manera que puedan ser leídos por el programa. Este método de búsqueda elige una solución inicial, y se va moviendo en soluciones adyacentes de manera iterativa hasta que no encuentre mejoras. A continuación se explica con más detalle en qué consiste el algoritmo, como se define el vecindario y cuáles son las movidas que realiza

1.2.1. MÉTODO SIMPLEX

Es un procedimiento algebraico basado en la solución de sistemas de ecuaciones (Hillier & Lieberman, 2001). Sus principios son:

- Sólo analiza las soluciones factibles en los vértices (puntos extremos de la región factible)
- Es un algoritmo iterativo que siempre que es posible escoge el origen como punto de inicialización y solo se mueve vértices adyacentes.

Este método se mueve de una solución básica factible (BF) actual a una adyacente mejor mediante la elección de la variable básica entrante y la que sale, después usa la eliminación de Gauss para resolver el sistema de ecuaciones lineales. Cuando la solución actual no tiene una solución BF adyacente que sea mejor, la solución actual es óptima y el algoritmo se detiene.

1.2.2. VECINDARIOS

En una búsqueda local, el vecindario es definido como todas las posibles soluciones que se consideran en cada punto. En este caso el espacio de solución está definido como el conjunto de soluciones básicas factibles (puntos extremos de la región factible) y cada vecindario corresponderá al espacio de soluciones básicas factibles adyacentes (Sharma, 2004). Esto quiere decir que dada una solución básica factible, el conjunto de las posibles soluciones que se considerarán en cada iteración serán solamente aquellas que son adyacentes a esta solución. Como el número de vértices de solución es limitado el algoritmo siempre encuentra una solución

1.2.3. MOVIDAS

La definición de las movidas depende en gran medida de la estructura del problema a resolver como de la función objetivo. En simplex cada movida viene determinada por la tasa de mejoramiento de la función objetivo en cada una de las aristas adyacentes del espacio de solución (Hillier & Lieberman, 2001). .El vecino que se selecciona es el mejor de todos, es decir el que represente el mayor aumento de la función objetivo, cuando ninguno de los vecinos mejora la solución se dice que el algoritmo ha alcanzado el óptimo. Las iteraciones que realiza el algoritmo son las siguientes:

1. Determinar la base B (elegir cuales serán las variables básicas y cuales las no básicas):
   Las variables básicas son aquellas que pertenecen a la solución y las variables no básicas son aquellas que no pertenecen. El algoritmo inicia en el origen (las variables del problema son iguales a 0, y las variables básicas serán iguales a las variables de holgura), cada vez que puede para evitarse cálculos algebraicos

2. Calcular los costos reducidos: Estos representan la variación de la función objetivo aumentar en una unidad la utilización del recurso k.

3. Validar si todos los costos reducidos son negativos, de ser así el algoritmo para, ya que se ha encontrado el óptimo. Entra la variable que tenga el costo reducido positivo más alto

4. Calcular el valor máximo en el cual puede aumentar la variable básica entrante:

   Este valor viene definido por los recursos disponibles, y el cálculo es llamado prueba del cociente mínimo, la cual determina que variable básica llega primero a cero cuando crece la variable entrante.

5. Actualizar las variables básicas y no básicas: Se asignan los valores encontrados y de esta forma se genera una nueva base para continuar con la siguiente iteración.

Como se ve este algoritmo explora todo el espacio de solución, el cual es finito por lo que siempre se encontrará una solución

CAPITULO III
EXPERIMENTACIÓN

1. INTRODUCCIÓN

Un buen experimento debe lograr imparcialidad y validez en las conclusiones y permitir que el experimento sea reproducible (Richard S. Barr, 1995). Los pasos que se siguieron para el desarrollo del experimento fueron los siguientes:

- Definición de objetivos del experimento
- Selección de los indicadores de desempeño y factores a evaluar
- Descripción de la implementación del algoritmo (generación de instancias)
- Implementación del algoritmo
- Análisis y representación gráfica de los resultados

El objetivo principal de la experimentación es medir la eficiencia y eficacia del método de solución propuesto, determinando en qué condiciones tiene buen desempeño. Para lograr una clara evaluación de este objetivo se realizó una lista de características que se consideran importantes en el desempeño de una meta-heurística (Moreno & Melián, 2005):

- **Rapidez:** Produce buenas soluciones en tiempos inferiores a los que se obtienen de otras metodologías, y ajustables a la necesidad del usuario.
- **Exactitud:** Qué tan cercano es el valor de las soluciones obtenidas al valor óptimo (en caso de que se cuente con este valor) ó a valores de soluciones encontrados anteriormente.
- **Eficaz:** La probabilidad de obtener con la meta-heurística soluciones óptimas, o al menos tan buenas como la mejor conocida debe ser alta.
- **Eficiente:** Los resultados que se obtengan con la meta-heurística deben alcanzarse con la menor cantidad de recursos computacionales (tiempo de ejecución y espacio de memoria).
- **Aprovechamiento de los recursos computacionales:** Se refiere al espacio de memoria y demás recursos computacionales que se requieren para su implementación

Teniendo en cuenta las características arriba citadas, se plantearon cuatro preguntas con el fin de guiar el análisis al correcto cumplimiento de los objetivos propuestos en este trabajo. Las dos primeras hacen referencia a la calidad de las soluciones, buscando comparaciones de los valores obtenidos con GRASP y otras soluciones (eficacia), la siguiente pretende medir el consumo de los recursos utilizados para encontrar las soluciones (eficiencia), para finalmente, a partir de las respuestas de las tres primeras preguntas, dar una pauta para la escogencia de la configuración del algoritmo.

- Si es posible encontrar el valor óptimo ¿Qué tan cercana es la solución arrojada por GRASP al valor óptimo?
- Si no se puede tener encontrar el óptimo ¿Qué tan cerca está el valor encontrado con GRASP a alguna cota inferior que se halle?
- ¿Qué tan rápido encuentra buenas soluciones?, ¿Cuánto toma en encontrar la mejor solución?
- ¿En qué casos se encuentran las mejores soluciones?

Como se mencionó anteriormente, la eficiencia del algoritmo debe ser evaluada en términos de la velocidad para encontrar soluciones factibles, lo que se hará midiendo los tiempos de ejecución. Adicionalmente, la exactitud de las soluciones se evaluará por medio del porcentaje de desviación de GRASP con los valores óptimos. Debido a la complejidad del problema, no será posible encontrar los valores óptimos de algunas instancias, caso en el cual se recurrirá a la relajación de algunas restricciones para obtener una cota inferior al problema y usarla para las comparaciones a que haya lugar.

$$\%\boldsymbol{Desviación}(\boldsymbol{Óptimo}) = \frac{\overline{\boldsymbol{GRASP}} - \boldsymbol{FO}(\boldsymbol{Óptimo})}{\boldsymbol{FO}(\boldsymbol{Óptimo})}\ x\ 100\ \% \tag{1}$$

$$\%\boldsymbol{Desviación}\ (\boldsymbol{Relajado}) = \frac{\overline{\boldsymbol{GRASP}} - \boldsymbol{FO}(\boldsymbol{Relajado})}{\boldsymbol{FO}(\boldsymbol{Relajado})}\ x\ 100\ \% \tag{2}$$

Las ecuaciones (1) y (2) presentan el cálculo de los porcentajes de desviación de GRASP, tanto con los valores óptimos como con los de las relajaciones. En dicha figura $\overline{GRASP}$ es el promedio de las soluciones encontradas con el algoritmo propuesto y $FO(Óptimo)$ y $FO(Relajado)$son los valores de la función objetivo para el problema sin relajación y con relajación respectivamente.

2. IMPLEMENTACIÓN DEL ALGORITMO

Para realizar los experimentos una vez diseñado el algoritmo, se procedió a generar las instancias, definir el número de replicas y por último, definir las condiciones en que se ejecutarían dichos experimentos.

En la primera parte de esta sección, sección 2.1, se explican los factores técnicos y condiciones que estuvieron relacionados con la programación e implementación del algoritmo. En la segunda parte, sección 2.2, se muestra como fueron generadas las instancias para este problema y que herramientas se usaron con este fin.

2.1. AMBIENTE COMPUTACIONAL

Para evitar variaciones en las variables de salida debido a factores no tenidos en cuenta dentro del diseño factorial, es necesario correr todos los experimentos en una sola maquina, garantizando de esta forma, las menores variaciones que dependan de la configuración de esta.

Las secciones siguientes pretenden dar a conocer dichas condiciones, separándolas en dos grupos diferentes, software, donde se exponen los recursos intangibles utilizados, y hardware, que hace referencia a los componentes físicos de la maquina.

2.1.1. Software

Para la ejecución del algoritmo se usó una maquina Sony que se describirá más adelante en la sección 2.1.2, la cual fue formateada y restaurada a su condición de inicial de fabrica, sin embargo se desactivaron todos los programas o funciones que pudieran interferir con el desempeño de la maquina. Los programas que se desactivaron en particular en la maquina fueron:

- Antivirus: Norton Internet Security 2006
- Tareas Programadas:
    - Microsoft Windows UpDate
    - Microsoft Office UpDate

    - Desfragmentado programado y herramientas del sistema
- Accesos directos a aplicaciones por medio de la barra de tareas.
    - Anti-SpyWare

Luego de esta descripción continuamos con la construcción del algoritmo que se hizo bajo un ambiente de desarrollo de proyectos de Microsoft Visual Basic para Aplicaciones (VBA), el cual se encuentra embebido en el paquete profesional de Microsoft Office. El VBA permite la opción de programar bajo el sistema operativo de Apple, Mac OS, o en el sistema tradicional de su casa matriz, Microsoft Windows, el cual, por cuestiones de compatibilidad y disponibilidad, se usó para el algoritmo del presente trabajo.

Aparte de los recursos propios del VBA, el algoritmo usa un programa de solución lineal de licenciamiento libre llamado desde el código principal, el cual se ejecuta como una aplicación independiente, abriéndose y cerrándose sucesivamente según la necesidad del algoritmo.

	Nombre	**Versión**
Sistema Operativo	Microsoft Windows	XP Media Center Edition - SP2
Entorno de Desarrollo	Visual Basic para Aplicaciones	6.3
Repositorio de datos de las Instancias	MS-Excel	2007
Solucionador Lineal	Glpsol: GNU Linear Programming Kit	4.9.2259.18188

Tabla 1: Recursos de Software

En la Tabla 1 se enumeran los recursos de software usados en este desarrollo, mostrando la versión de cada uno y su nombre comercial. Adicionalmente, en la Tabla 2 se describen los procedimientos que se implementaron, y que dieron forma al algoritmo GRASP, cuyo código fuente puede ser consultado en el Anexo 3. Es de notar que el desempeño de algoritmo no depende únicamente del software usado sino también de los de los recursos de hardware disponibles (i.e. los resultados se obtendrán en menor tiempo en una maquina con un procesador más rápido).

Función/Procedimiento	Descripción	Valor devuelto
Sub main()	Esta función es la que se encarga de automatizar el proceso de ejecución del algoritmo leyendo de una tabla en MS-Excel tanto los parámetros como la instancia que se necesitaba correr	
Private Sub Cargar_Valores()	Según la instancia que se esté corriendo esta función lee el tamaño. Estos valores se usan a lo largo de todo el algoritmo.	
Sub GRASP(Alfa, Max_iteraciones)	Este procedimiento es el procedimiento principal del algoritmo, y es el que controla los ciclos de entrada y salida de a las fases de la meta heurística.	
Private Sub actualizar(**ByRef** iteracion)	Mantiene actualizada la solución con el mejor valor encontrado.	
Private Sub Fase_Construccion()	Esta rutina genera una solución factible al problema en la hoja de solución parcial	
Private Sub CopiarDatos()	Hace la copia de la hoja de parámetros en la hoja CDatos	
Private Sub ResetSolucion(**ByRef** VAFO)	Inicializa la hoja de solución para comenzar de cero una nueva fase de construcción	
Private Sub EscogerDemanda(**ByRef** cliente, **ByRef** producto, **ByRef** demanda, **ByRef** DemTotal)	Escoge la demanda más alta de la hoja CDatos por la que el algoritmo adiciona un candidato a la solución en construcción	
Private Sub EvaluarCostosIncrementales(cliente, producto, demanda, **ByRef** Evaluaciones)	Evaluar recibe y modifica un vector de tamaño m*j que contiene los costos totales. Hay que tener en cuenta que si no se tiene capacidad de producción disponible no se multiplica toda la demanda sino por lo que sea posible transportar según por la capacidad disponible.	
Private Function CrearRCL(Evaluaciones)	En esta función se genera la lista de candidatos del algoritmo. Con base a esta lista son seleccionados los candidatos que se adicionan a la solución	Matriz de tres renglones.
Private Function MinSinCeros(Evaluaciones)	Busca el mínimo valor del vector de evaluaciones sin tener en cuenta los datos = 0	mínimo
Private Sub ArreglarValores(**ByRef** RCL, MINI, MA)	Asigna al mayor valor un número que sea el menor de todos los de la lista.	
Private Function SumarValores(**ByRef** RCL)	Hace la suma de los valores de los candidatos en la RCL con el fin asignarles una probabilidad.	Suma de valores
Private Sub EscogerCandidato(RCL, **ByRef** planta, **ByRef** instalacion)	Genera un número aleatorio bajo una distribución uniforme y escoge un candidato de la RCL.	
Private Sub Actualice_SolucParcial(cliente, producto, planta, instalacion)	Actualiza la solución parcial con el candidato que se haya escogido de la RCL	
Private Sub CREAR_DATOS()	Genera un archivo de texto donde se almacenan los datos que lee el GLPsol	
Private Sub LLamaGLP()	Hace la llamada externa del GLPsol y espera a que este genere la solución en el directorio indicado	
Private Function LeerSolucion()	Lee el valor objetivo de la solución generada	Solución final

Tabla 2: Resumen de los procedimientos implementados

2.1.2. Hardware

Con el fin de describir de forma detallada la maquina utilizada para correr el algoritmo se cita la Tabla 3. En esta tabla se exponen los componentes principales del hardware y las direcciones donde se ubican los controladores. Cabe aclarar que todos los controladores que administran el sistema son proveídos por el fabricante, Sony Corporation para este caso.

Componente	Descripción
OS Name	Microsoft Windows XP Professional
Version	5.1.2600 Service Pack 3 Build 2600
OS Manufacturer	Microsoft Corporation
System Manufacturer	Sony Corporation
System Model	VGN-FE680G
System Type	X86-based PC
Processor	x86 Family 6 Model 14 Stepping 8 GenuineIntel ~1829 Mhz
BIOS Version/Date	Phoenix Technologies LTD R0130J3, 5/11/2006
SMBIOS Version	2.40
Windows Directory	C:\WINDOWS
System Directory	C:\WINDOWS\system32
Boot Device	\Device\HarddiskVolume2
Locale	United States
Hardware Abstraction Layer	Version = "5.1.2600.5512 (xpsp.080413-2111)"
Total Physical Memory	2,048.00 MB
Available Physical Memory	1.17 GB
Total Virtual Memory	2.00 GB
Available Virtual Memory	1.96 GB
Page File Space	3.84 GB
Page File	C:\pagefile.sys

Tabla 3: Referencia de hardware generada por el sistema

2.2. GENERACIÓN DE INSTANCIAS

Una vez diseñado y construido el algoritmo, se procedió a generar las instancias, proceso que se basó en los criterios usados en (Marín, 2007) para definir el tamaño de las instancias, y los valores de los costos de localización. Sin embargo el artículo citado genera instancias para un problema con un solo producto, por lo que se optó por agregar instancias con tres (3) y diez (10) productos con el fin de acomodarlo a los requerimientos del problema planteado, generando en total dieciocho (18) instancias como se muestra en la tabla 4.

INSTANCIA	PRODUCTOS	PLANTAS	INSTALACIONES	CLIENTES	No.
	k	m	j	i	VARIABLES
I 1	1	10	16	50	8000
I 2	1	10	25	50	12500
I 3	1	10	50	50	25000
I 4	1	20	16	50	16000
I 5	1	20	25	50	25000
I 6	1	20	50	50	50000
I 7	3	10	16	50	24000
I 8	3	10	25	50	37500
I 9	3	10	50	50	75000
I 10	3	20	16	50	48000
I 11	3	20	25	50	75000
I 12	3	20	50	50	150000
I 13	10	10	16	50	80000
I 14	10	10	25	50	125000
I 15	10	10	50	50	250000
I 16	10	20	16	50	160000
I 17	10	20	25	50	250000
I 18	10	20	50	50	500000

Tabla 4: Tamaños de las instancias

Para definir los valores de los parámetros, se partió de la premisa de que las capacidades de producción deben ser al menos iguales a las demandas para cada uno de los productos, de lo contrario según las restricciones 3 y 4 de la sección titulada DEFINICIÓN DEL MODELO MATEMÁTICO, no sería posible solucionar el problema.

Adicionalmente los costos de localización deben ser comparativamente más altos a los costos de transporte por unidad transportada (ver Tabla 5: Generación de parámetros).

PARÁMETRO	DESCRIPCIÓN	COMENTARIO	FORMULACIÓN
$\boldsymbol{f_i^j}$	Valores que siguen una distribución uniforme entre el rango de 7500 a 25000.	Tomado de (Marín, 2007)	$\boldsymbol{f_i^j} \sim U(7500,25000)$
$\boldsymbol{c_{ijm}^k}$	Valores que siguen una distribución uniforme entre el rango de 25 a 150.	Este rango se da con el fin de garantizar que sean comparablemente más bajos que el costo de abrir una instalación.	$\mathrm{c}_{\boldsymbol{i}\mathrm{jm}}^{\mathrm{k}} \sim U(25,150)$
$\boldsymbol{h_i^k}$	Valores que siguen una distribución uniforme entre 0 y 200.		$\boldsymbol{h_i^k} \sim U(0,200)$
$\boldsymbol{S_m^k}$	Se generan valores uniformemente distribuidos	Se usan los valores uniformes para generar valores consistentes con el problema y poderlo solucionar.	$S'^k_m \sim U(0,1)$ $\rightarrow \frac{S'^k_m}{\sum_m S'^k_m} \times h_i^k = S_m^k$

Tabla 5: Generación de parámetros

Adicionalmente, el algoritmo GRASP requiere la definición de dos parámetros: Alfa y el número máximo de iteraciones sin encontrar mejoramiento de la solución. Para el primero se tomaron valores de 0, 0.1 y 0.5, mientras que para el segundo se tomaron valores de 10 y 50 iteraciones. Un valor de Alfa=0 indica que no hay aleatoriedad en el desempeño de GRASP (i.e. se comporta como un algoritmo greedy, totalmente voraz). Por lo tanto, sólo se tienen 5 combinaciones de estos parámetros.

FACTORES	No. NIVELES	NIVELES
Productos (k)	3	1,3,10
Plantas (m)	2	10,20
Instalaciones (j)	3	16,25,50
Clientes (i)	1	50
Alfa	3	0, 0.1, 0.5
MaxIteraciones	2	10, 50

Tabla 6: Factores y niveles del diseño

En la Tabla 6 se resumen los diferentes factores y niveles del diseño experimental. Es de notar que se tienen 18 instancias, que sometidas a los diferentes valores de los parámetros del algoritmo, se obtienen 90 experimentos (18×5). Debido al comportamiento aleatorio de GRASP, se realizaron 5 réplicas para cada experimento, dando un total de 450 corridas del algoritmo.

3. RESULTADOS

3.1. PRESENTACIÓN DE LA SOLUCIÓN

El algoritmo GRASP propuesto obtiene las cantidades de cada producto que deben ser transportadas entre los nodos de la red. La solución arrojada muestra un resumen donde se exponen el valor encontrado, el tipo de resultado (i.e. si es óptimo o no lo es), el número de filas que representan el número de variables y el numero de columnas para notar el número de restricciones. Posteriormente se encuentran dos tablas, la primera contiene los valores de restricción del problema y las cantidades usadas de cada recurso para llegar a la solución, la siguiente tabla contiene los valores encontrados para cada una de las variables. Adicionalmente, ambas tablas exponen valores marginales útiles para hacer análisis de sensibilidad de cada solución.

A manera de ejemplo, en el anexo 4 se presenta la impresión de una solución dada por el algoritmo. La solución particular que se expone en dicho anexo es una de las

encontradas para la instancia $I\ 1$ conformada por un producto, diez plantas, dieciséis instalaciones y cincuenta clientes, representada como $1 \times 10 \times 16 \times 50$.

Para hacer clara la representación del resultado arrojado por GRASP se presenta la Figura 7, donde se ubican rectángulos en el lado izquierdo para representar las plantas, en el centro las instalaciones, y círculos en el lado derecho que representen los clientes. Además se representa el flujo de productos en la red por medio de líneas, la cuales finalizan con una flecha. Los rombos se usan para mostrar a través de cual instalación debe llevarse el producto. Por último, el valor que tomara la variable dentro del modelo se escribe sobre las líneas, tanto donde inicia como donde finaliza.

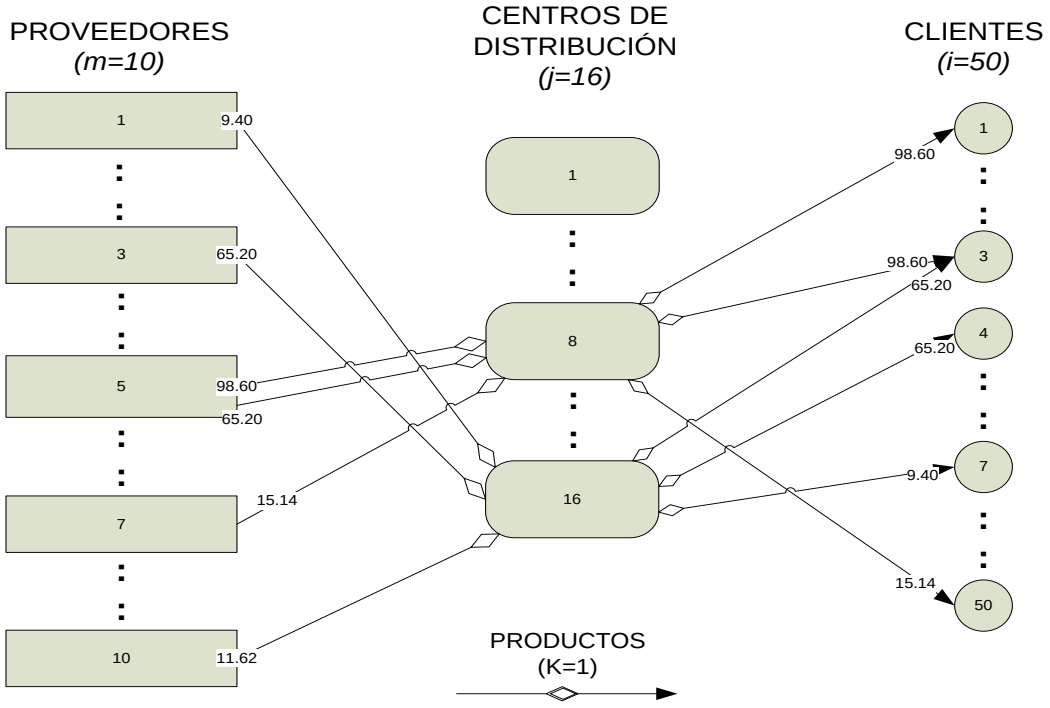

Figura 7: Representación grafica para las variables 107, 403, 651, 653, 1000 y 1454 de la solución de la instancia I 1

Por ejemplo, una de las variables representadas es la $Y^{1}_{1,16,7}$, marcada dentro de la solución con el número 107 (ver Tabla 7: Variables representadas en el esquema), esta

tiene en la solución un valor de 9.40346 unidades y puede distinguirse en la Figura 7 por ser la única que sale de la planta 1 y la única que llega al cliente 7.

De cada uno de los nodos que representa una planta sale al menos un flujo de producto y cada uno de los clientes recibe al menos un flujo de producto, sin embargo no todas las de las instalaciones tienen un flujo asociado, lo que significa que dichas instalaciones no han de ser abiertas. Adicionalmente es posible encontrar flujos que compartan el mismo origen y destino entre plantas e instalaciones, lo que se da en caso que una planta supla la demanda de dos clientes diferentes a través de la misma instalación, como se ve con las variables 651 y 653 de la Tabla 7: Variables representadas en el esquema.

No. de variable	Producto	Planta	Instalación	Cliente	Valor
107	1	1	16	7	9.40346
403	1	3	16	3	65.2076
651	1	5	8	1	98.6099
653	1	5	8	3	8.48284
1000	1	7	8	50	15.1428
1454	1	10	16	4	11.6277

Tabla 7: Variables representadas en el esquema

3.2. ANÁLISIS DE RESULTADOS

Para poder analizar comparativamente los resultados obtenidos en los 450 experimentos mencionados en la sección anterior, se resolvió el problema de dos formas. La primera busca encontrar una cota inferior al problema utilizando un método de solución lineal (Simplex), donde se relajaron las restricciones de las variables binarias para que estas pudieran tomar valores no enteros. Por otro lado, en la segunda se buscaba encontrar el óptimo para cada una de las instancias propuestas, para lo cual se utilizó un algoritmo de ramificación y acotamiento. En ambos casos, tanto para la relajación como para los óptimos, se corrieron los algoritmos una vez para cada instancia, midiendo de forma independiente tanto el valor de la función objetivo como el tiempo de ejecución de cada uno de los métodos.

En las Tabla 8 a Tabla 12 se presenta un resumen de los valores encontrados para cada una de las cinco combinaciones de los niveles de los factores Alfa y MaxIteraciones. Los datos se organizaron en dos partes: La primera expone los valores obtenidos para la función objetivo del problema, mostrando de izquierda a derecha los valores de la cota inferior, los valores óptimos y por último los promedios de GRASP. La segunda parte expone los tiempos de ejecución de cada uno de los métodos organizados de la misma manera, adicionando para GRASP el tiempo mínimo, promedio y máximo.

Dentro de los datos contenidos en las tablas citadas anteriormente se encuentran algunos vacios correspondientes a los valores óptimos para las instancias $I\,12, I\,15, I\,16, I\,17$ e $I\,18$, de las cuales no fue posible obtenerlos por su gran tamaño.

Además, puede notarse que los valores óptimos para las instancias $I\,9$ e $I\,14$ están marcados con un asterisco, indicando que son los mejores valores que se obtuvieron por el algoritmo de ramificación y acotamiento tras ser detenido voluntariamente después de extensas horas de ejecución sin llegar a un resultado. Debido a esto, es posible encontrar algunos valores de la función objetivo hallados con GRASP menores a los óptimos registrados para dichas instancias.

	FUNCIÓN OBJETIVO			TIEMPO DE EJECUCIÓN				
INSTANCIA	RELAJADO	ÓPTIMO	GRASP	RELAJADO	ÓPTIMO	Mín GRASP	Prom GRASP	Máx GRASP
I 1	$ 142,044.46	$ 185,872.18	$ 224,283.87	0:00:02	0:00:17	0:00:22	0:00:22	0:00:23
I 2	$ 141,864.66	$ 190,206.15	$ 220,878.32	0:00:02	0:01:43	0:00:33	0:00:33	0:00:33
I 3	$ 128,788.86	$ 181,114.06	$ 209,349.57	0:00:05	1:37:10	0:00:44	0:00:44	0:00:45
I 4	$ 130,773.55	$ 164,937.19	$ 179,886.54	0:00:03	0:00:36	0:00:44	0:00:44	0:00:44
I 5	$ 140,462.45	$ 180,055.83	$ 202,773.32	0:00:05	0:03:37	0:00:55	0:00:55	0:00:55
I 6	$ 129,999.38	$ 165,879.30	$ 194,086.37	0:00:10	0:55:01	0:01:28	0:01:28	0:01:28
I 7	$ 404,484.29	$ 476,764.00	$ 616,239.26	0:00:05	0:05:44	0:00:55	0:00:55	0:00:56
I 8	$ 387,554.01	$ 460,894.17	$ 506,590.37	0:00:08	1:06:30	0:01:17	0:01:17	0:01:17
I 9	$ 367,388.28	$ 460,372.56 *	$ 501,383.55	0:00:16	15:53:12	0:02:12	0:02:12	0:02:12
I 10	$ 381,479.78	$ 448,804.18	$ 524,434.17	0:00:10	0:08:57	0:01:39	0:01:39	0:01:39
I 11	$ 350,585.83	$ 405,942.96	$ 485,357.57	0:00:17	0:25:04	0:02:12	0:02:13	0:02:14
I 12	$ 382,779.06		$ 538,269.25	0:00:31		0:04:02	0:04:02	0:04:02
I 13	$ 1,275,849.78	$ 1,421,960.93	$ 1,967,808.66	0:00:21	2:59:00	0:02:56	0:02:58	0:03:01
I 14	$ 1,300,700.04	$ 1,447,932.92 *	$ 1,976,027.17	0:00:35	16:04:31	0:04:02	0:04:02	0:04:02
I 15	$ 1,247,152.62		$ 1,627,896.81	0:01:07		0:06:47	0:06:50	0:06:53
I 16	$ 1,271,199.67		$ 1,698,794.44	0:00:48		0:05:19	0:05:19	0:05:20
I 17	$ 1,229,591.87		$ 1,632,482.68	0:01:15		0:07:20	0:07:20	0:07:20
I 18	$ 1,287,390.98		$ 1,743,450.52	0:02:36		0:13:05	0:13:08	0:13:10
PROMEDIO	**$ 594,449.42**	**$ 343,929.80**	**$ 836,110.69**	**0:00:29**	**2:11:11**	**0:00:22**	**0:03:09**	**0:13:10**

Tabla 8: Resumen Alfa 0.

*Valor encontrado después de detener voluntariamente el algoritmo.

	FUNCIÓN OBJETIVO			TIEMPO DE EJECUCIÓN				
INSTANCIA	**RELAJADO**	**ÓPTIMO**	**GRASP**	**RELAJADO**	**ÓPTIMO**	**Mín GRASP**	**Prom GRASP**	**Máx GRASP**
I 1	$ 142,044.46	$ 185,872.18	$ 188,471.69	0:00:02	0:00:17	0:00:36	0:00:49	0:01:00
I 2	$ 141,864.66	$ 190,206.15	$ 190,509.48	0:00:02	0:01:43	0:00:45	0:01:05	0:01:42
I 3	$ 128,788.86	$ 181,114.06	$ 183,101.38	0:00:05	1:37:10	0:01:00	0:01:36	0:02:10
I 4	$ 130,773.55	$ 164,937.19	$ 166,729.58	0:00:03	0:00:36	0:00:48	0:01:15	0:01:32
I 5	$ 140,462.45	$ 180,055.83	$ 180,055.83	0:00:05	0:03:37	0:01:00	0:01:16	0:01:35
I 6	$ 129,999.38	$ 165,879.30	$ 166,839.40	0:00:10	0:55:01	0:01:45	0:02:27	0:03:12
I 7	$ 404,484.29	$ 476,764.00	$ 501,520.65	0:00:05	0:05:44	0:01:36	0:01:46	0:02:00
I 8	$ 387,554.01	$ 460,894.17	$ 483,805.75	0:00:08	1:06:30	0:01:27	0:02:09	0:02:55
I 9	$ 367,388.28	$ 460,372.56 *	$ 459,323.30	0:00:16	15:53:12	0:02:24	0:03:41	0:06:00
I 10	$ 381,479.78	$ 448,804.18	$ 479,862.37	0:00:10	0:08:57	0:01:40	0:02:06	0:02:53
I 11	$ 350,585.83	$ 405,942.96	$ 420,752.71	0:00:17	0:25:04	0:02:36	0:03:41	0:04:46
I 12	$ 382,779.06		$ 460,775.99	0:00:31		0:04:02	0:06:14	0:09:55
I 13	$ 1,275,849.78	$ 1,421,960.93	$ 1,549,876.48	0:00:21	2:59:00	0:03:50	0:05:53	0:08:02
I 14	$ 1,300,700.04	$ 1,447,932.92 *	$ 1,537,935.36	0:00:35	16:04:31	0:04:42	0:07:53	0:10:38
I 15	$ 1,247,152.62		$ 1,484,916.21	0:01:07		0:06:59	0:09:29	0:14:04
I 16	$ 1,271,199.67		$ 1,498,045.48	0:00:48		0:05:55	0:06:43	0:07:30
I 17	$ 1,229,591.87		$ 1,442,289.58	0:01:15		0:07:50	0:08:13	0:09:10
I 18	$ 1,287,390.98		$ 1,507,009.89	0:02:36		0:15:52	0:19:53	0:28:23
PROMEDIO	**$ 594,449.42**	**$ 343,929.80**	**$ 716,767.84**	**0:00:29**	**2:11:11**	**0:00:36**	**0:04:47**	**0:28:23**

Tabla 9: Resumen Alfa 0.1 y MaxIteraciones 10

*Valor encontrado después de detener voluntariamente el algoritmo.

	FUNCIÓN OBJETIVO			TIEMPO DE EJECUCIÓN				
INSTANCIA	**RELAJADO**	**ÓPTIMO**	**GRASP**	**RELAJADO**	**ÓPTIMO**	**Mín GRASP**	**Prom GRASP**	**Máx GRASP**
I 1	$ 142,044.46	$ 185,872.18	$ 222,239.72	0:00:02	0:00:17	0:01:10	0:01:17	0:01:34
I 2	$ 141,864.66	$ 190,206.15	$ 247,381.77	0:00:02	0:01:43	0:01:15	0:01:32	0:02:12
I 3	$ 128,788.86	$ 181,114.06	$ 262,729.36	0:00:05	1:37:10	0:01:24	0:02:10	0:02:54
I 4	$ 130,773.55	$ 164,937.19	$ 204,700.32	0:00:03	0:00:36	0:01:16	0:01:37	0:02:25
I 5	$ 140,462.45	$ 180,055.83	$ 228,556.02	0:00:05	0:03:37	0:01:30	0:02:28	0:04:14
I 6	$ 129,999.38	$ 165,879.30	$ 274,219.89	0:00:10	0:55:01	0:02:56	0:03:48	0:05:17
I 7	$ 404,484.29	$ 476,764.00	$ 509,435.58	0:00:05	0:05:44	0:01:53	0:02:57	0:03:59
I 8	$ 387,554.01	$ 460,894.17	$ 513,785.96	0:00:08	1:06:30	0:02:08	0:02:22	0:02:41
I 9	$ 367,388.28	$ 460,372.56 *	$ 604,049.41	0:00:16	15:53:12	0:04:49	0:06:01	0:07:02
I 10	$ 381,479.78	$ 448,804.18	$ 480,995.98	0:00:10	0:08:57	0:02:43	0:04:30	0:07:13
I 11	$ 350,585.83	$ 405,942.96	$ 475,803.94	0:00:17	0:25:04	0:03:28	0:05:43	0:07:26
I 12	$ 382,779.06		$ 612,497.56	0:00:31		0:06:15	0:07:55	0:10:17
I 13	$ 1,275,849.78	$ 1,421,960.93	$ 1,444,612.61	0:00:21	2:59:00	0:05:05	0:10:25	0:19:21
I 14	$ 1,300,700.04	$ 1,447,932.92 *	$ 1,506,270.47	0:00:35	16:04:31	0:08:18	0:09:41	0:12:34
I 15	$ 1,247,152.62		$ 1,593,398.40	0:01:07		0:13:32	0:22:56	0:33:11
I 16	$ 1,271,199.67		$ 1,415,444.88	0:00:48		0:10:16	0:18:01	0:37:03
I 17	$ 1,229,591.87		$ 1,444,683.12	0:01:15		0:18:56	0:30:15	0:55:33
I 18	$ 1,287,390.98		$ 1,653,569.36	0:02:36		0:30:22	0:50:18	1:08:43
PROMEDIO	**$ 594,449.42**	**$ 343,929.80**	**$ 760,798.57**	**0:00:29**	**2:11:11**	**0:01:10**	**0:10:13**	**1:08:43**

Tabla 10: Resumen Alfa 0.1 y MaxIteraciones 50

*Valor encontrado después de detener voluntariamente el algoritmo.

	FUNCIÓN OBJETIVO			TIEMPO DE EJECUCIÓN				
INSTANCIA	RELAJADO	ÓPTIMO	GRASP	RELAJADO	ÓPTIMO	Mín GRASP	Prom GRASP	Máx GRASP
I 1	$ 142,044.46	$ 185,872.18	$ 186,679.81	0:00:02	0:00:17	0:04:35	0:06:30	0:10:40
I 2	$ 141,864.66	$ 190,206.15	$ 190,209.28	0:00:02	0:01:43	0:04:31	0:05:40	0:06:55
I 3	$ 128,788.86	$ 181,114.06	$ 182,632.59	0:00:05	1:37:10	0:06:11	0:09:15	0:16:41
I 4	$ 130,773.55	$ 164,937.19	$ 164,937.19	0:00:03	0:00:36	0:05:12	0:07:25	0:10:18
I 5	$ 140,462.45	$ 180,055.83	$ 180,055.83	0:00:05	0:03:37	0:05:57	0:06:59	0:07:49
I 6	$ 129,999.38	$ 165,879.30	$ 166,409.46	0:00:10	0:55:01	0:09:24	0:12:37	0:17:10
I 7	$ 404,484.29	$ 476,764.00	$ 493,821.77	0:00:05	0:05:44	0:06:48	0:08:22	0:13:00
I 8	$ 387,554.01	$ 460,894.17	$ 474,609.35	0:00:08	1:06:30	0:10:09	0:13:00	0:15:22
I 9	$ 367,388.28	$ 460,372.56 *	$ 454,173.92	0:00:16	15:53:12	0:11:54	0:14:51	0:19:22
I 10	$ 381,479.78	$ 448,804.18	$ 469,692.16	0:00:10	0:08:57	0:09:25	0:16:04	0:20:12
I 11	$ 350,585.83	$ 405,942.96	$ 417,371.40	0:00:17	0:25:04	0:13:45	0:15:22	0:17:44
I 12	$ 382,779.06		$ 453,341.76	0:00:31		0:22:26	0:37:34	1:00:07
I 13	$ 1,275,849.78	$ 1,421,960.93	$ 1,521,633.34	0:00:21	2:59:00	0:16:21	0:23:29	0:34:21
I 14	$ 1,300,700.04	$ 1,447,932.92 *	$ 1,525,767.19	0:00:35	16:04:31	0:20:01	0:28:21	0:35:11
I 15	$ 1,247,152.62		$ 1,466,802.78	0:01:07		0:46:34	1:07:29	1:28:28
I 16	$ 1,271,199.67		$ 1,490,972.31	0:00:48		0:27:18	0:31:28	0:36:15
I 17	$ 1,229,591.87		$ 1,411,068.68	0:01:15		0:37:05	0:52:37	1:40:34
I 18	$ 1,287,390.98		$ 1,504,265.43	0:02:36		1:05:19	1:18:21	1:32:38
PROMEDIO	**$ 594,449.42**	**$ 343,929.80**	**$ 708,580.24**	**0:00:29**	**2:11:11**	**0:04:31**	**0:24:11**	**1:40:34**

Tabla 11: Resumen Alfa 0.5 y MaxIteraciones 10

*Valor encontrado después de detener voluntariamente el algoritmo.

	FUNCIÓN OBJETIVO			TIEMPO DE EJECUCIÓN				
INSTANCIA	RELAJADO	ÓPTIMO	GRASP	RELAJADO	ÓPTIMO	Mín GRASP	Prom GRASP	Máx GRASP
I 1	$ 142,044.46	$ 185,872.18	$ 215,872.32	0:00:02	0:00:17	0:03:18	0:03:50	0:04:33
I 2	$ 141,864.66	$ 190,206.15	$ 238,713.18	0:00:02	0:01:43	0:03:25	0:05:19	0:07:12
I 3	$ 128,788.86	$ 181,114.06	$ 250,910.52	0:00:05	1:37:10	0:04:36	0:07:59	0:13:09
I 4	$ 130,773.55	$ 164,937.19	$ 193,816.57	0:00:03	0:00:36	0:04:15	0:05:17	0:06:27
I 5	$ 140,462.45	$ 180,055.83	$ 215,872.52	0:00:05	0:03:37	0:08:25	0:11:05	0:14:33
I 6	$ 129,999.38	$ 165,879.30	$ 261,654.55	0:00:10	0:55:01	0:09:20	0:12:54	0:16:45
I 7	$ 404,484.29	$ 476,764.00	$ 508,487.43	0:00:05	0:05:44	0:06:54	0:12:54	0:21:58
I 8	$ 387,554.01	$ 460,894.17	$ 507,556.04	0:00:08	1:06:30	0:10:51	0:15:15	0:22:56
I 9	$ 367,388.28	$ 460,372.56 *	$ 584,245.63	0:00:16	15:53:12	0:14:56	0:24:11	0:43:49
I 10	$ 381,479.78	$ 448,804.18	$ 470,912.62	0:00:10	0:08:57	0:12:23	0:19:23	0:25:31
I 11	$ 350,585.83	$ 405,942.96	$ 456,448.49	0:00:17	0:25:04	0:23:38	0:28:58	0:35:04
I 12	$ 382,779.06		$ 597,463.82	0:00:31		0:33:58	0:47:47	0:59:00
I 13	$ 1,275,849.78	$ 1,421,960.93	$ 1,439,277.22	0:00:21	2:59:00	0:36:03	0:51:51	1:19:34
I 14	$ 1,300,700.04	$ 1,447,932.92 *	$ 1,495,051.89	0:00:35	16:04:31	0:44:11	0:57:20	1:10:15
I 15	$ 1,247,152.62		$ 1,575,265.36	0:01:07		1:07:28	1:29:15	2:03:35
I 16	$ 1,271,199.67		$ 1,407,054.56	0:00:48		0:49:50	1:45:31	3:10:32
I 17	$ 1,229,591.87		$ 1,431,664.63	0:01:15		1:26:02	3:03:29	5:30:54
I 18	$ 1,287,390.98		$ 1,655,668.92	0:02:36		2:39:38	4:16:53	6:07:40
PROMEDIO	**$ 594,449.42**	**$ 343,929.80**	**$ 750,329.79**	**0:00:29**	**2:11:11**	**0:03:18**	**0:52:11**	**6:07:40**

Tabla 12: Resumen Alfa 0.5 y MaxIteraciones 50

*Valor encontrado después de detener voluntariamente el algoritmo.

Resultados Alfa contra Función Objetivo Promedio de GRASP

Con el fin de mostrar la pertinencia de usar únicamente valores menores a Alfa=0.5 se incluyen las Figura 8 a Figura 13, que presentan el comportamiento del valor promedio de la función objetivo con respecto al valor del parámetro Alfa para cada instancia. En estas gráficas se busca observar el comportamiento de la variable de salida del algoritmo respecto a los diferentes valores de Alfa. A pesar que las escalas de la variable dependiente (valor de la función objetivo promedio de GRASP) son diferentes en todas las gráficas, el análisis sobre estas se hace teniendo en cuenta únicamente la forma y tendencia de las líneas, razón por la que no será sesgado.

Como puede observarse, existe una clara concavidad positiva en 14 de las 18 instancias, lo cual corresponde perfectamente a un problema de minimización, teniendo su punto más bajo en valores siempre cercanos a Alfa=0.1. Aunque las instancias $I\ 10, I\ 13, I\ 14$ e $I\ 16$ no muestran la misma claridad en la concavidad, es en ese mismo punto donde sus valores de pendiente cambian abruptamente buscando conseguir la concavidad positiva del resto.

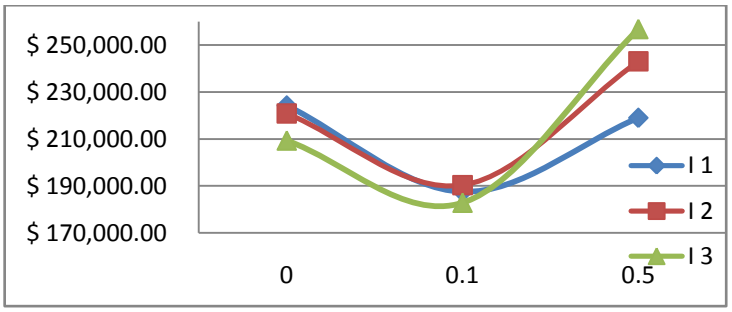

Figura 8: Analisis parametro Alfa con I 1 a I 3

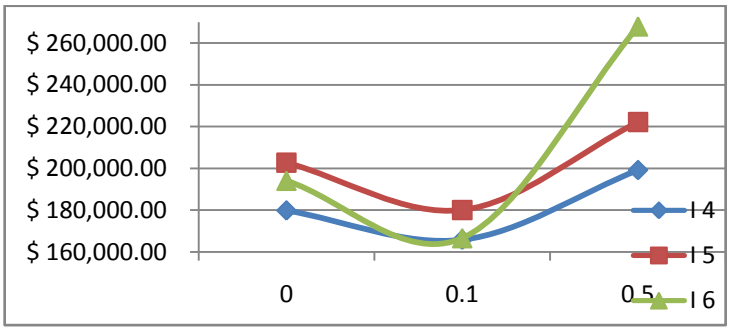

Figura 9: Análisis parámetro Alfa con I 4 a I 6

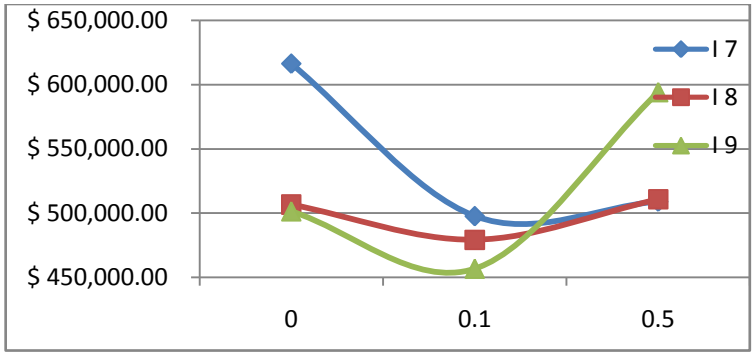

Figura 10: Análisis parámetro Alfa I 7 a I 9

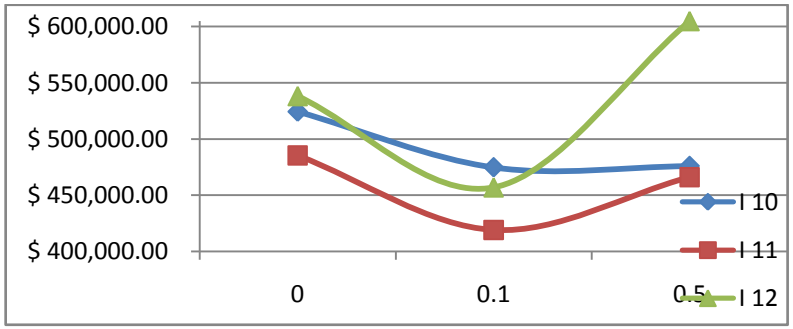

Figura 11: Análisis parámetro Alfa con I 10 a I 12

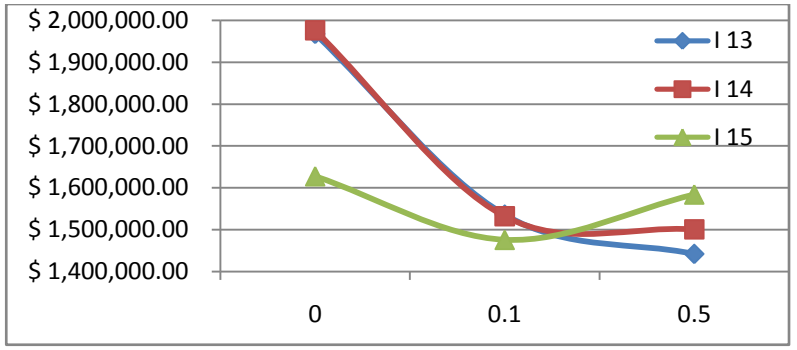

Figura 12: Análisis parámetro Alfa con I 13 a I 15

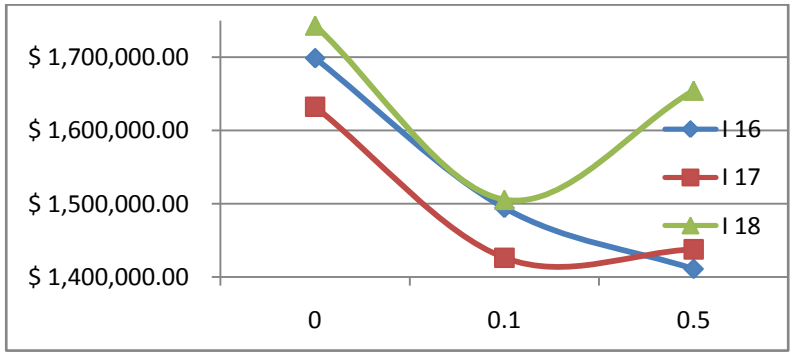

Figura 13: Análisis parámetro Alfa con I 16 a I 18

Desviaciones de GRASP con los Valores Óptimos Encontrados

Luego de analizar el parámetro Alfa por sí solo, se presenta la Figura 14, que expone de forma visual las desviaciones del valor promedio de la función objetivo para las cinco replicas encontradas con GRASP y los valores óptimos en cada una de las instancias propuestas, datos que están registrados en la Tabla 13. En esta tabla se pueden apreciar algunos vacios que representan los valores óptimos que no se pudieron hallar. Para las instancias marcadas con un asterisco no fue posible encontrar una solución óptima, sin embargo se ejecutó el algoritmo de ramificación y acotamiento, deteniéndolo voluntariamente después de extensas horas de ejecución y registrando el mejor valor hasta ese momento encontrado. Por lo tanto la Figura 14 y su tabla de valores únicamente muestran las instancias en las que fue posible obtener los valores óptimos incluyendo las instancias marcadas con asteriscos.

Puede observarse en la figura citada arriba, que en general las mejores soluciones son siempre encontradas con un valor de Alfa=0.1, obteniendo valores de desviación promedio de menos del 1% en el 41% de las veces (excluyendo los valores de los datos con asteriscos), sin embargo cabe notar que el 100% de estos casos son para instancias con un solo producto. Adicionalmente puede notarse que la línea que pertenece a MaxIteraciones=50 y Alfa=0.1 está más cercana a los valores óptimos que la que pertenece a MaxIteraciones=10 y Alfa=0.1, lo que se puede comprobar haciendo referencia a la Tabla 13 donde el 66% de los valores inferiores a 1% pertenecen a dicha linea. Por otro lado, en la misma gráfica citada anteriormente se refleja que los porcentajes de desviación de los valores de Alfa=0.5 se hacen pequeños para las instancias más grandes, no obstante se encuentran únicamente cuatro (4) valores validos (valores para los cuales se obtuvo el óptimo), inferiores al 10% para instancias mayores a la I 8.

Es de notar, que mientras en las primeras instancias ninguna desviación de Alfa=0.5 es mejor que las de Alfa=0.1, para las instancias más grandes puede verse un acercamiento de los valores de desviación de los dos tratamientos, encontrando incluso mejores valores para Alfa=0.5 como en el caso de la instancia I 13.

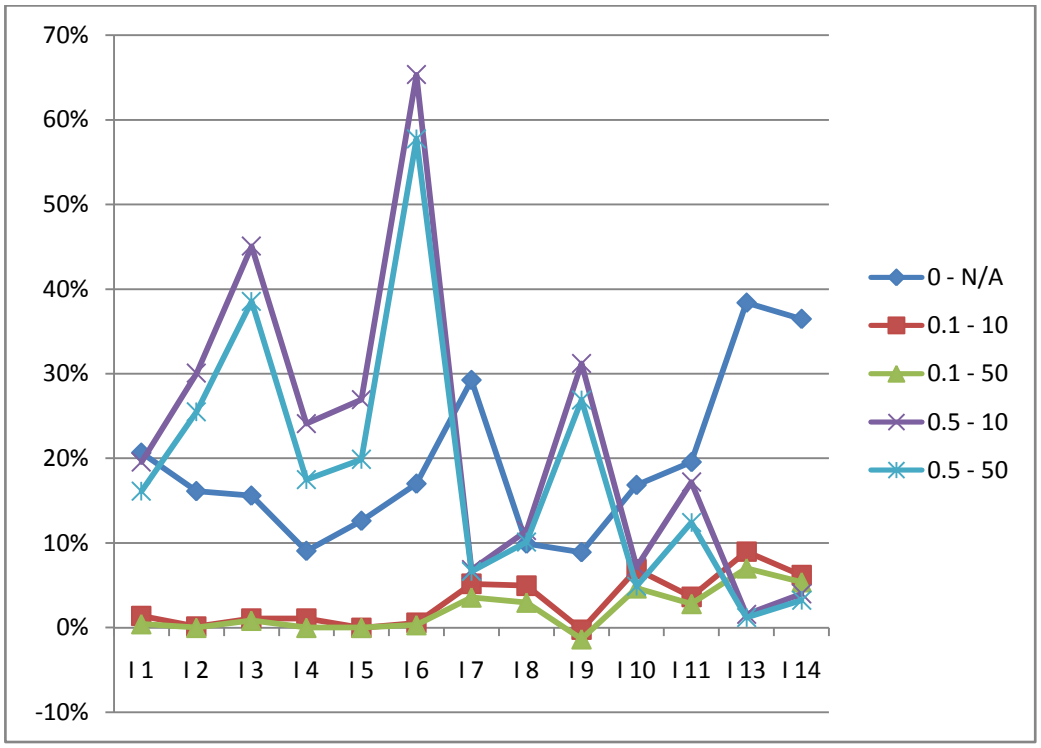

Figura 14: Desviaciones con respecto a los óptimos

	0	0.1		0.5		<-Alfa
INSTANCIA	**N/A**	**10**	**50**	**10**	**50**	<-MaxIteraciones
I 1	20.67%	1.40%	0.43%	19.57%	16.14%	
I 2	16.13%	0.16%	0.00%	30.06%	25.50%	
I 3	15.59%	1.10%	0.84%	45.06%	38.54%	
I 4	9.06%	1.09%	0.00%	24.11%	17.51%	
I 5	12.62%	0.00%	0.00%	26.94%	19.89%	
I 6	17.00%	0.58%	0.32%	65.31%	57.74%	
I 7	29.25%	5.19%	3.58%	6.85%	6.65%	
I 8	9.91%	4.97%	2.98%	11.48%	10.12%	
I 9*	8.91%	-0.23%	-1.35%	31.21%	26.91%	
I 10	16.85%	6.92%	4.65%	7.17%	4.93%	
I 11	19.56%	3.65%	2.82%	17.21%	12.44%	
I 12						
I 13	38.39%	9.00%	7.01%	1.59%	1.22%	
I 14*	36.47%	6.22%	5.38%	4.03%	3.25%	
I 15						
I 16						
I 17						
I 18						

Tabla 13: Desviaciones con respecto a los óptimos

Marcadas con asterisco las instancias donde se detuvo la ejecución del algoritmo sin llegar al resultado óptimo

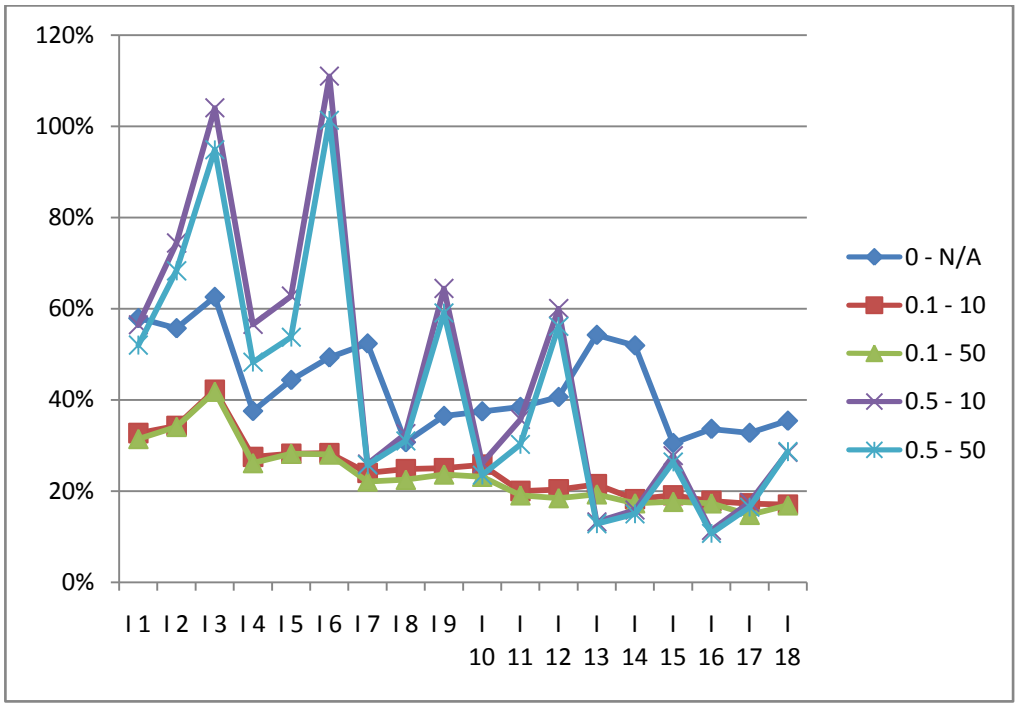

Tabla 14: Desviaciones con respecto a las relajaciones

	0	0.1		0.5		<-Alfa
INSTANCIA	**N/A**	**10**	**50**	**10**	**50**	<-MaxIteraciones
I 1	57.90%	32.68%	31.42%	56.46%	51.98%	
I 2	55.70%	34.29%	34.08%	74.38%	68.27%	
I 3	62.55%	42.17%	41.81%	104.00%	94.82%	
I 4	37.56%	27.49%	26.12%	56.53%	48.21%	
I 5	44.36%	28.19%	28.19%	62.72%	53.69%	
I 6	49.30%	28.34%	28.01%	110.94%	101.27%	
I 7	52.35%	23.99%	22.09%	25.95%	25.71%	
I 8	30.71%	24.84%	22.46%	32.57%	30.96%	
I 9	36.47%	25.02%	23.62%	64.42%	59.03%	
I 10	37.47%	25.79%	23.12%	26.09%	23.44%	
I 11	38.44%	20.01%	19.05%	35.72%	30.20%	
I 12	40.62%	20.38%	18.43%	60.01%	56.09%	
I 13	54.24%	21.48%	19.26%	13.23% *	12.81% *	
I 14	51.92%	18.24%	17.30%	15.80%	14.94%	
I 15	30.53%	19.06%	17.61%	27.76%	26.31%	
I 16	33.64%	17.85%	17.29%	11.35% *	10.69% *	
I 17	32.77%	17.30%	14.76%	17.49%	16.43%	
I 18	35.43%	17.06%	16.85%	28.44%	28.61%	

Figura 15: Desviaciones con los valores relajados

Marcados con asterisco los valores mínimos

Para valores de Alfa=0.5 se presentan los mayores porcentajes de desviación, especialmente en las instancias I3, I6 e I9, las cuales tienen el mayor número de instalaciones (j=50). Pero debido a la falta de valores válidos para hacer conjeturas acerca de los resultados para las instancias grandes, es necesario recurrir a la Tabla 14 y a la Figura 15, donde se muestran las desviaciones respecto a los valores de solución encontrados con la relajación de las variables binarias.

Desviaciones de GRASP con los Valores Relajados

En las Desviaciones con los valores relajados puede verse un comportamiento parecido de las líneas al estudiado en las Desviaciones con respecto a los óptimos, sin embargo en la nueva gráfica se muestran todas las instancias debido a la obtención de todos los valores de relajación.

Con ayuda de la Tabla 14 es posible notar que el porcentaje de desviación promedio respecto a los valores de las relajaciones, son más bajos para el 77.7% de las Instancias utilizando Alfa=0.1, es de notar que 12 de las 15 instancias donde esto sucede son las que tienen 1 y 3 productos siendo estas las que menos variables tienen. Adicionalmente se puede inferir de la gráfica que los valores que en promedio más se acercan a los valores de las relajaciones de todo el conjunto de datos están dados para Alfa=0.5 en las instancias $I\,13$ e $I\,16$ (marcados con un asterisco en la Tabla 13: Desviaciones con respecto a los óptimos)

Análisis de Tiempos de Ejecución

Con el fin de analizar el consumo de recursos de GRASP se presentan las Figura 16 y Figura 17, las cuales muestran los valores de tiempo de ejecución tanto de GRASP como del algoritmo usado para hallar los valores óptimos para cada una de las instancias. Es necesario aclarar que debido a que no fue posible encontrar los valores óptimos para todos los casos, la Figura 17 muestra únicamente las instancias donde estos valores se hayan obtenido.

Como puede verse en las gráficas citadas, a medida que el número de variables aumenta, los tiempos de ejecución crecen de manera exponencial, tanto para GRASP como para los valores óptimos. Es de notar, que el crecimiento de los tiempos de estos últimos aumenta mucho más rápido cuando el número de variables binarias aumenta, razón por la cual, se grafican en grupos que separen las instancias por número de variables continuas y número de variables binarias, mostrando diferentes líneas en la misma grafica.

A partir de estas gráficas, podemos ver que aunque el tiempo de ejecución de GRASP también aumenta exponencialmente con el numero de variables, este no presenta, a diferencia del tiempo de ejecución para los valores óptimos, picos en las instancias cuyo factor relacionado con el número de instalaciones sea fijado en el más alto de sus niveles (i.e. cuando el número de instalaciones j=50, que correspondería al número de variables binarias del problema). Adicionalmente, puede verse para las instancias con j=50 que GRASP llega a soluciones factibles usando entre el 0.2% y el 1.4% de tiempo que usa el algoritmo óptimo dependiendo de los valores de los parámetros Alfa y MaxIteraciones (véanse Tabla 8 a Tabla 12).

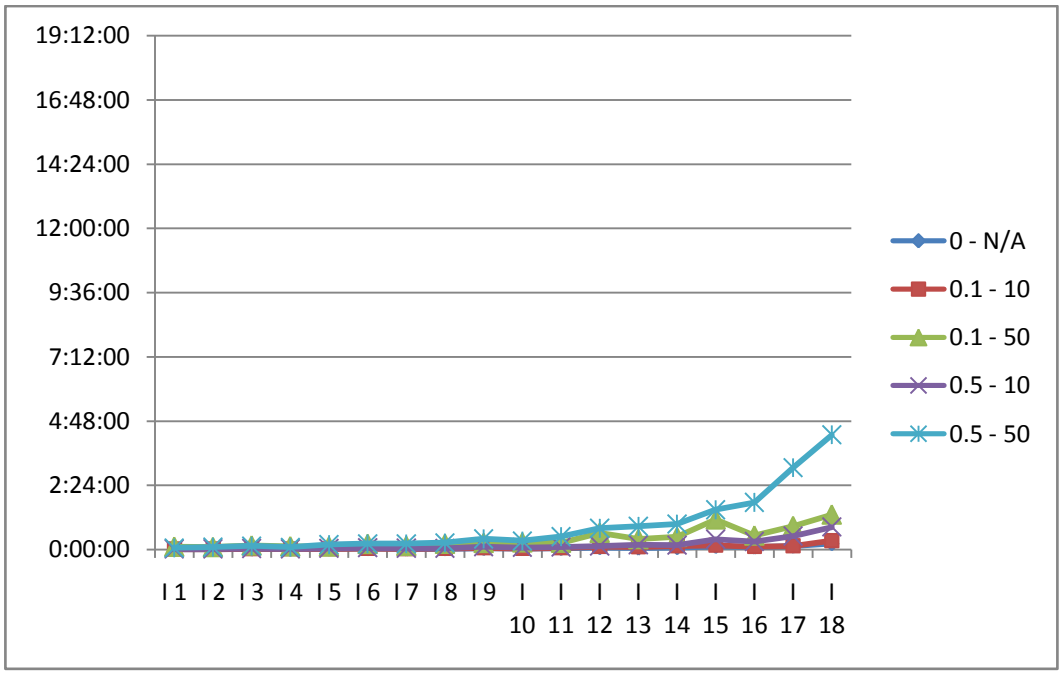

Figura 16 Tiempo de ejecución de GRASP para los diferentes parámetros

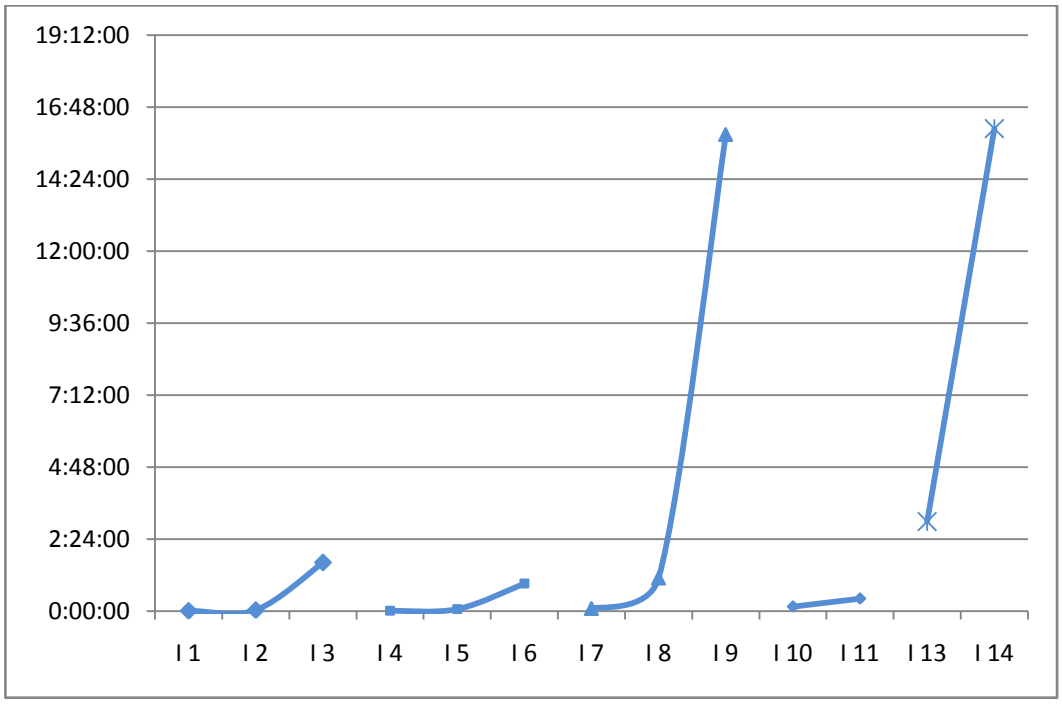

Figura 17: Tiempo de ejecución para los valores óptimos

Comparaciones de Tiempo Contra las Desviaciones

Para hacer comparaciones entre los tiempos de ejecución de GRASP con las desviaciones respecto a los valores óptimos, se presenta la Figura 18. En esta figura se muestra una nube de puntos donde cada uno representa una desviación promedio para cada instancia con diferentes combinaciones de parámetros (Alfa y MaxIteraciones). En el *eje X* se ubican las desviaciones promedio, y en el *eje Y* el promedio de los tiempos de los valores de GRASP usados para calcular las desviaciones.

Este gráfico ilustra que si el algoritmo es totalmente voraz (i.e. Alfa=0) las soluciones encontradas tienen un tiempo de cómputo bajo, sin embargo sus resultados se agrupan por encima de los obtenidos con el de Alfa=0.1. Para valores de Alfa=0.5, los datos se encuentran más dispersos, no obstante es posible ver que cuando se combina con MaxIteraciones=50 se obtienen datos con una calidad similar a la obtenida con el uso de MaxIteraciones=10 pero con mayores tiempos de ejecución. Finalmente, para

valores de Alfa=0.1 se obtienen soluciones de buena calidad, teniendo valores de desviación siempre menores al 10% sin importar el valor que tome MaxIteraciones, pero es de notar que si MaxIteraciones=50 el tiempo de ejecución será mayor en el 65% de las veces.

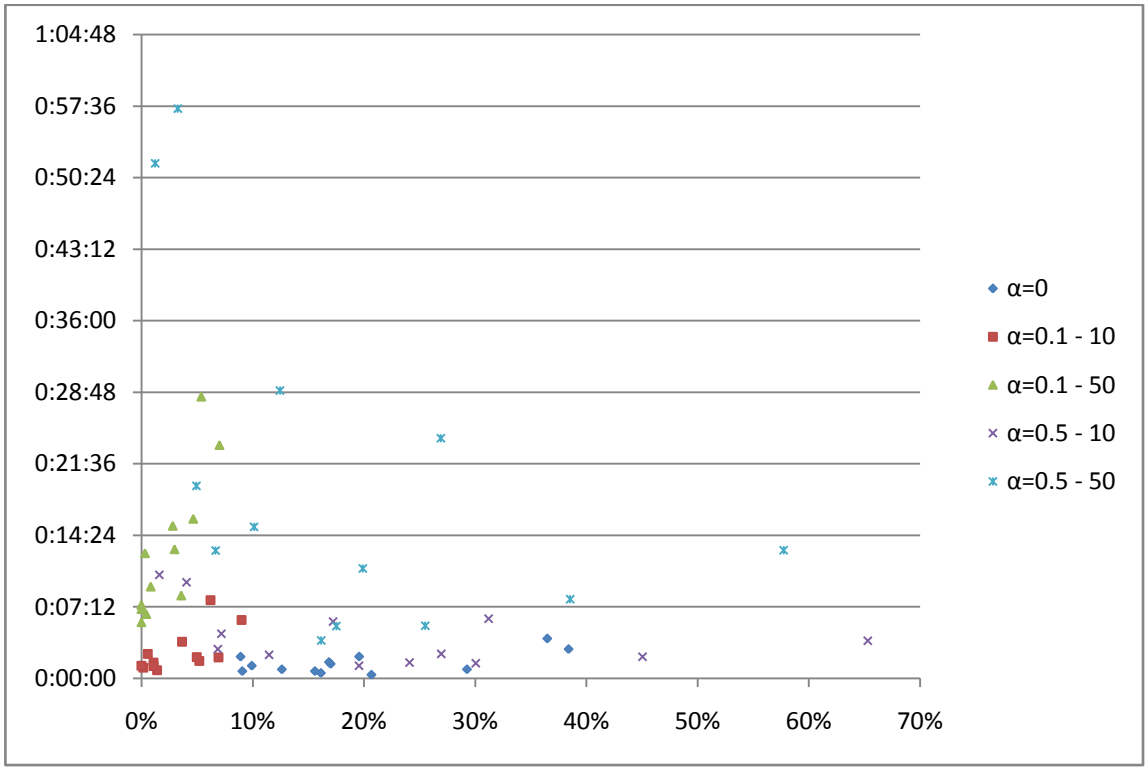

Figura 18: Dispersión de las desviaciones promedio contra tiempos

Los análisis hasta el momento realizados se han hecho basados en los promedios de los datos para describir el comportamiento global de GRASP, sin embargo es importante ver en detalle este rendimiento para diferentes grupos de instancias con todos los datos disponibles. Con este fin se presentan las Figura 19 a Figura 21, donde se muestran gráficos de dispersión de datos con las mismas características que el de la gráfica Dispersión de las desviaciones promedio contra tiempos, con la diferencia que los datos acá representados no serán promediados. Es de notar, que cada uno de estos gráficos exhibe los datos de un grupo de instancias con igual número de instalaciones (i.e. se agruparon por el valor que tenga el factor j).

Adicionalmente se muestran líneas que representan la frontera de eficiencia para cada uno de los grupos de instancias, las cuales representan los mejores valores para cada uno de los criterios (i.e. un punto que este sobre la frontera de eficiencia no será mejorado en tiempo de ejecución ni en calidad al a la vez)

La Figura 19 presenta los datos para todas las instancias con el mayor número de instalaciones propuesto (j=50). Para este caso se usaron las desviaciones respecto a las relajaciones encontradas debido a los pocos valores óptimos encontrados para este grupo de instancias. En esta gráfica se puede ver que la combinación de parámetros dada por Alfa=0.1 y MaxIteraciones=10 son los valores que presentan los resultados con las desviaciones más bajas en el menor tiempo, notando que los diez primeros puntos de la frontera de eficiencia es conformada por esta combinación de parámetros.

En las siguientes dos gráficas es posible ver un comportamiento de los datos similar al descrito en la Figura 19, diferenciándose por tener una frontera de eficiencia que parte de tiempos más pequeños y que la conforman sólo unos pocos puntos. No obstante, teniendo en cuenta el número de puntos de cada combinación que están sobre dicha frontera se puede ver que la combinación de Alfa=0.1 y MaxIteraciones=50 es la que genera los mejores resultados al comparar las replicas individuales obtenidas.

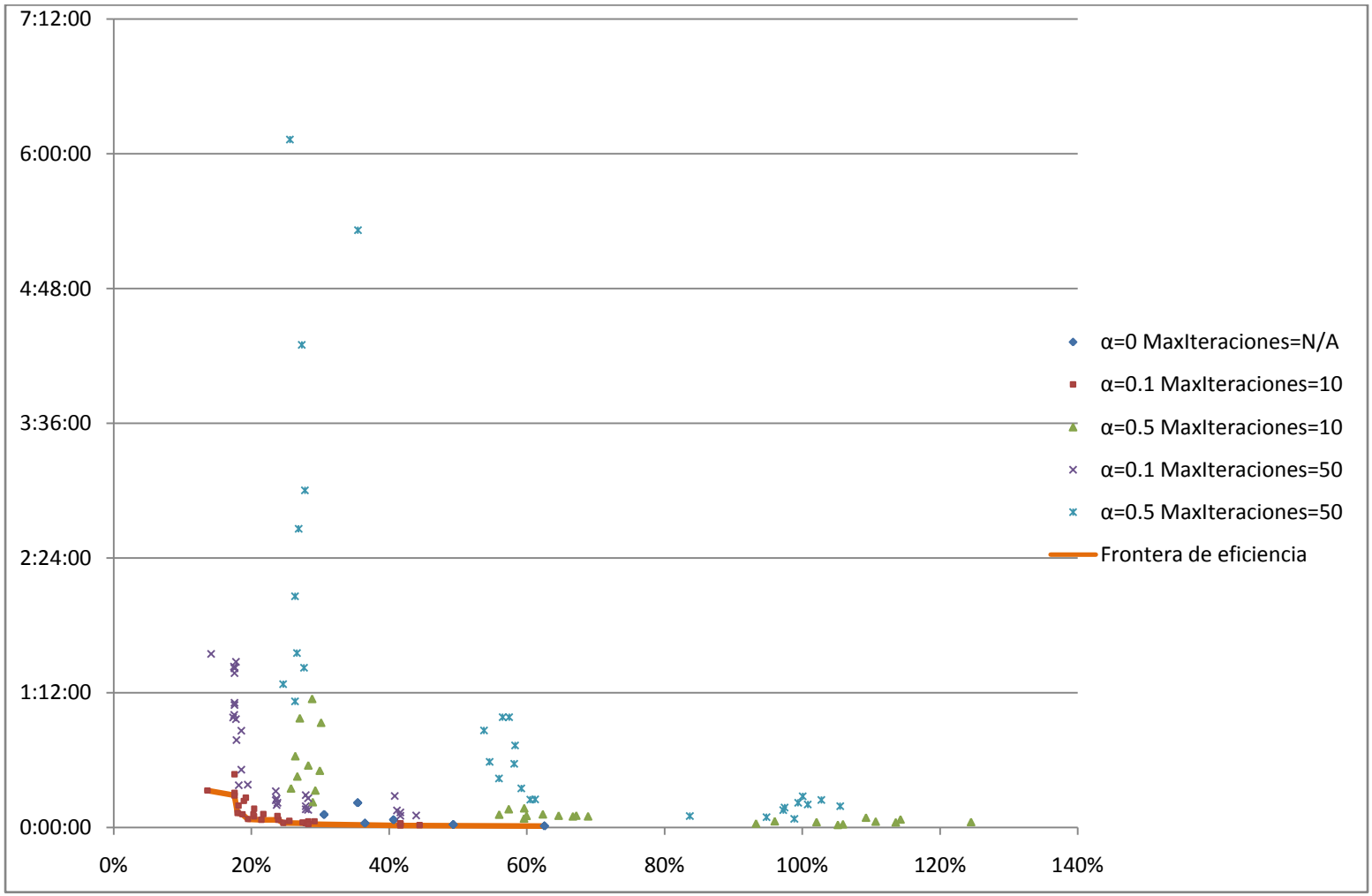

Figura 19: Frontera de eficiencia para instalaciones con j=50 respecto a los valores de la relajación

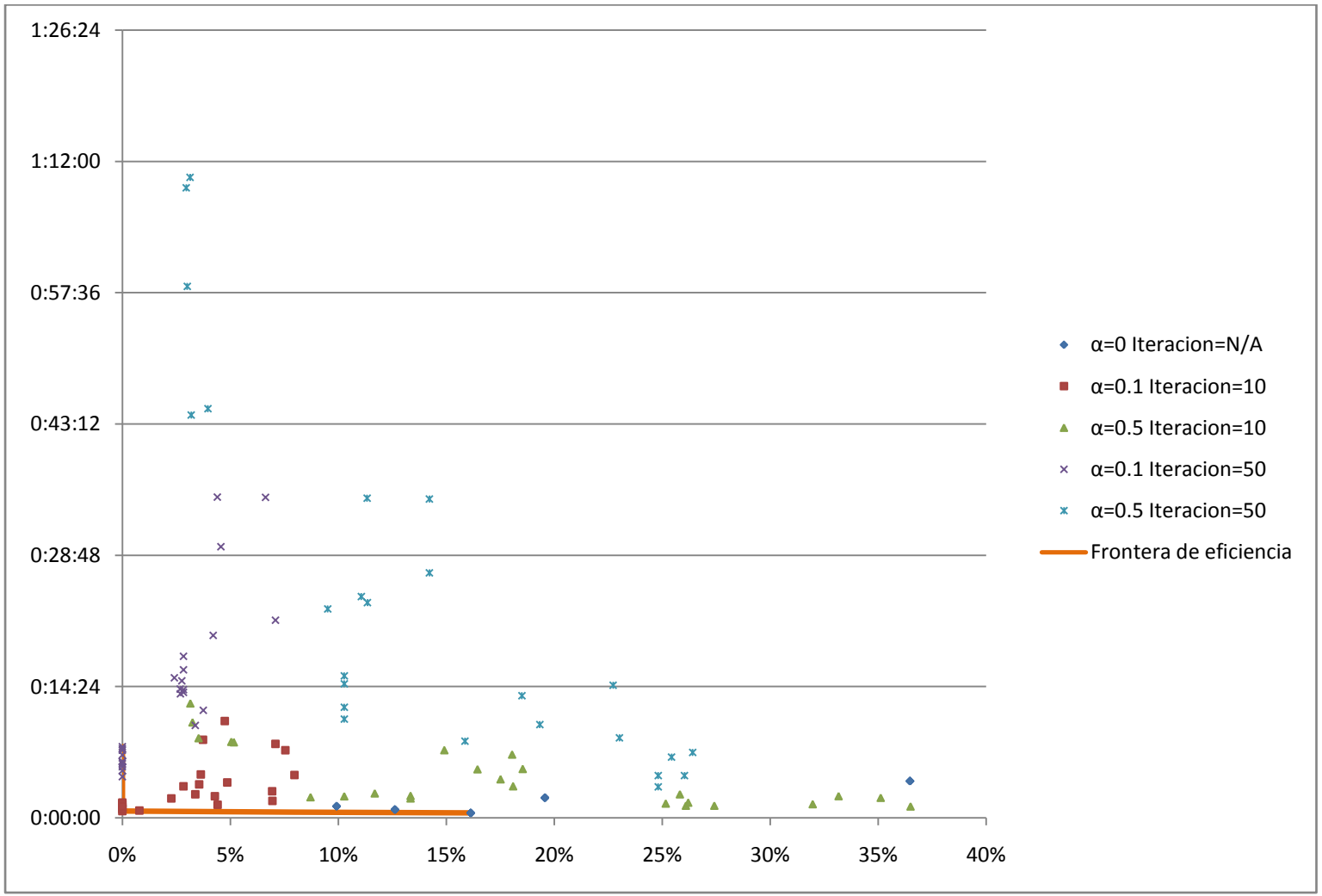

Figura 20: Frontera de eficiencia para instalaciones con j=25 respecto a los valores óptimos

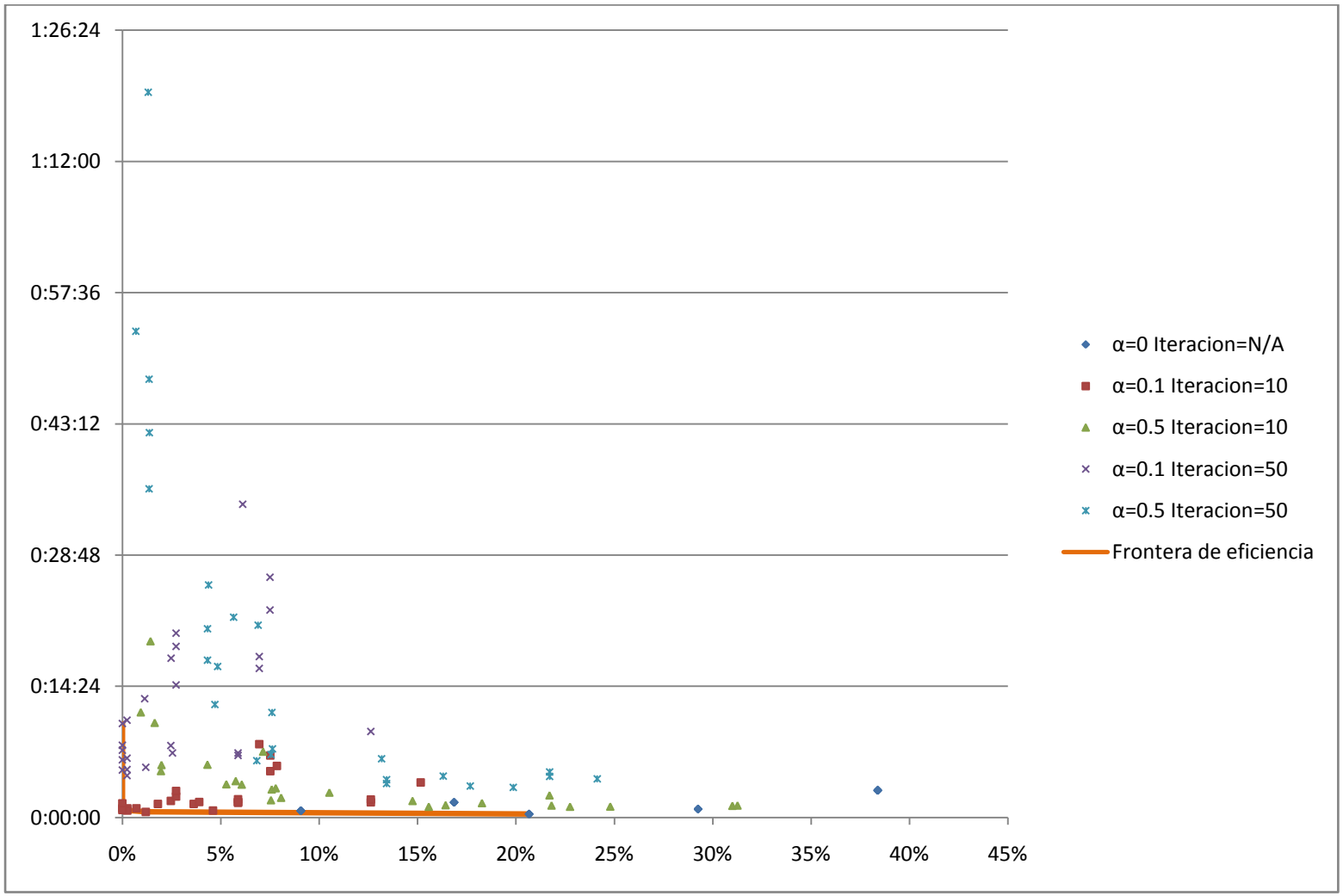

Figura 21: Frontera de eficiencia para instalaciones con j=16 respecto a los valores óptimos

CAPITULO IV
CONCLUSIONES Y RECOMENDACIONES

1. CONCLUSIONES

En este trabajo se presentó una propuesta de solución al TUFLP, el cual es un problema de localización de instalaciones de múltiples productos en dos niveles, utilizado para determinar la mejor configuración entre las plantas, centros de distribución y clientes de una cadena logística. La revisión del estado del arte recopila la información acerca de lo que se ha trabajado del problema comenzando por los problemas de localización, su importancia y los avances que se han presentado en esta línea de investigación; hasta la presentación de técnicas de solución para problemas similares al estudiado. Entre las más utilizadas se encuentran las técnicas de descomposición y aproximación (descomposición de Benders, ramificación y acotamiento, y relajación Lagrangiana) y algunas meta-heurísticas como son recocido simulado y algoritmos genéticos.

La metodología de solución propuesta se basó en GRASP. Ésta es una meta-heurística que surgió a finales de la década de 1990 (aunque se han visto desarrollos de procedimientos basados en el mismo principio desde finales de la década de 1980) y ha sido aplicada en gran variedad de problemas computacionalmente complejos. Es considerada también como un método de multi-arranque basado en la premisa de que soluciones iníciales diversas y de buena calidad juegan un papel importante en el éxito de métodos locales de búsqueda. Su implementación fue sencilla ya que solo requiere ajustar dos parámetros: uno para determinar el criterio de parada del algoritmo y el otro para la escogencia de los valores utilizados para la construcción de la lista de candidatos

En la primera fase (construcción) se creó una solución completa (a diferencia del método de descomposición de Benders) y posteriormente se mejoró en una búsqueda local. Debido a que tiene en cuenta diversas soluciones, el tiempo de cómputo y la calidad de las soluciones presentaron variaciones según el valor asignado para el parámetro Alfa (es el que restringe el número de elementos a seleccionar). Por ejemplo para la instancia 1 al cambiar el valor de alfa de 0, a 0.1 las desviaciones del valor de la función objetivo con respecto al valor óptimo presentan diferencias hasta del 20%, igualmente sucede al cambiar el valor de alfa a 0.5. Otro aspecto importante para destacar es que cuando las soluciones son totalmente voraces (si alfa es igual a 0, sólo se escoge el mejor candidato de la lista) el algoritmo

siempre arrojará las mismas soluciones en cada iteración por lo que no vale realizar más iteraciones.

En general, puede concluirse que la selección de las instalaciones generada en la fase de construcción fue acertada. Esto se refleja en el porcentaje de desviación del óptimo ya que si en esta fase no se hubieran escogido las instalaciones correctamente, no se hubiera encontrado ninguno.

En cuanto a la fase de búsqueda local, la decisión de utilizar el método simplex facilitó el diseño del algoritmo, además de garantizar la obtención del óptimo para la configuración de los datos resultantes de la fase de construcción.

En la experimentación, se comprobó que la metodología utilizada para solucionar este problema, en las 18 instancias generadas con una distribución uniforme tiene resultados favorables. Se encontró el óptimo en el 10% y diferencias inferiores al 10% en el 54% de las comparaciones realizadas. Adicionalmente GRASP fue capaz de solucionar todos los tipos de instancias obteniendo desviaciones con respecto a los valores relajados cercanas al 22%

El efecto de los parámetros Alfa y MaxIteraciones en el rendimiento del algoritmo es claro, para la mayoría de instancias los valores mínimos de la función objetivo se encontraron para valores de Alfa igual 0,1 excepto para las instancias I10, I13, I14 y I16 donde valores cercanos a 0.5 podrían generar mejores soluciones. Conclusiones semejantes se obtuvieron al realizar las comparaciones de este parámetro teniendo en cuenta la desviación del valor de la función objetivo de GRASP respecto al valor óptimo encontrado, donde los menores porcentajes de desviación se presentaron para Alfa igual a 0.1 y MaxIteraciones igual a 50.

El parámetro MaxIteraciones afecta principalmente el tiempo de ejecución del algoritmo. Para 50 iteraciones el tiempo de ejecución es el aproximadamente el doble al empleado en 10 iteraciones, sin embargo en la mayoría de los casos estas soluciones son las que mejor calidad tienen.

En cuanto a la eficiencia del algoritmo, el tiempo empleado para generar cada solución es sustancialmente inferior al compararlo con las soluciones óptimas y relajadas, especialmente para las instancias más difíciles (con mayor número total de variables o mayor número de variables binarias), por ejemplo en la Instancia 9, GRASP genera soluciones en promedio de 2 minutos, mientras que al solucionarlas con el programa óptimo, el mejor valor registrado hasta las 16 horas es más alto que se obtuvo en GRASP .

Para finalizar es importante destacar que uno de los principales valores agregados de la metodología propuesta es la sencillez y practicidad que la caracteriza; las herramientas

computacionales necesarias para utilizarla son básicas y es capaz de resolver problemas superiores 500.000 variables. Sólo se necesita contar con el programa Microsoft Excel, el cual generalmente viene instalado en todos los computadores y es usado frecuentemente en oficinas y hogares. Adicionalmente la forma de ingresar de los datos (Hoja de cálculo) y la visualización de los resultados (archivo plano de texto) es muy sencilla.

2. RECOMENDACIONES

Si bien el procedimiento propuesto mostró buenos resultados en los indicadores de desempeño analizados, es posible realizar algunas mejoras. Por ejemplo, debido al gran tamaño del campo de exploración de soluciones y a los tiempos de ejecución elevados en algunos casos (particularmente en las instancias grandes), se podría pensar en disminuir el número de iteraciones al correr estas instancias. Por otro lado, puesto que en la resolución de las instancias pequeñas se observaron tiempos de cálculo cortos, un incremento en el número de iteraciones podría permitir lograr mejorar los resultados de la función objetivo.

El uso del método simplex en la búsqueda local simplificó el diseño del algoritmo. Sin embargo, se podría obtener un mejor desempeño (en cuanto al tiempo de cálculo) si se lograrán incluir los valores de la función objetivo generados en la solución de la fase de construcción como una cota inferior para la inicialización de este algoritmo.

Debido a que el algoritmo se divide en dos fases principales, es recomendable registrar el valor de la función objetivo y tiempos empleados en cada fase, lo que permite realizar análisis sobre la contribución de cada fase en el desempeño total del algoritmo, con el fin de determinar mejoras para cada una, por ejemplo agregando o eliminando algunas movidas en la búsqueda local o incluyendo otros criterios para la asignación de las probabilidades de selección de candidatos en la fase de construcción.

Finalmente, el diseño actual del procedimiento puede ser incorporado en una herramienta para la toma de decisiones empresariales. En este sentido, este trabajo se convierte en un insumo para futuros proyectos de investigación.

BIBLIOGRAFÍA

Albernathy, W.J. & Hershey, J.C., 1972. A Spatial-Allocation Model for Regional Health-Services Planning. *Operations Resarch*, 20(3), pp.629-42.

Alvarez, V., Parreño, F. & Tamarit, J., 2008. Reactive GRASP for the Strip Packing Problem. *Computers and Operations Research*, 35(4), pp.1065-83.

Aponte, A.F., Rosas, P.A., Montoya, J.R. & Caballero, J.P., 2009. Solving the two-echelon facility location problem using GRASP. *Target journal: Operations Research Letters*.

Barbarosogl, G. & Ozgur, D., 1999. Hierarchical design of an integrated production and 2-echelon distribution system. *European Journal of Operational Research 118*, p.464±484.

Bard, J. & Rojanasoonthon, S., 2005. A GRASP for Parallel Machine Scheduling with Time Windows. *Informs journal on computing*, 17(1), pp.32-51.

Birge, J.R. & Malyshko, V., 1985. Methods for a network design problem in solar power systems. *Computers & Operational Research*, 12(1), pp.125-38.

Bortfeld, A., 2005. A genetic algorithm fot the two-dimensional strip packing problem with rectangular pieces. *European Journal of Operational Research, in press*.

Burke, E.K.G.W.G., 2006. Metaheuristic enhancements of the Best-Fit heuristic for the orthogonal stock cutting problem. *INFORMS Journal on Computing (2nd revision)*.

Burke, E.K.G.W.G., 2006. Metaheuristic enhancements of the Best-Fit heuristic for the orthogonal stock cutting problem. *INFORMS Journal on Computing(2nd revision)*.

Chardaire, P., McKeown, G. & Maki, J., 2001. Application of GRASP to the Multiconstraint knapsack Problem. *Applications of Evolutionary Computing*, pp.30-39.

Chase, J. & Aquiliano, 2004. *Administración de la producción y las operaciones*. Mc Graw Hill 2004.

Chen, Z.-L., 2004. Integrated Production and Distribution Operations:Taxonomy, Models, and Review/Handbook of Quantitative Supply Chain Analysis: Modeling in the E-Business Era. Ch. 17.

Cohen, M.A. & Lee, H.L., 1988. Strategic Analysis of Integrated Production-Distribution Systems. *Operations research*, 36, pp.216-28.

Cohon, R.L. et al., 1980. Application of a Multiobjective Location Model to Power Plant Siting in a Six State Region of the U.s. *Computers and Operations Research*, 7, pp.107-24.

Cornuejols, G., Nemhauser, G. & Wolsey, L., 1990. The uncapacitated facility location problem. In *Discrete Location Theory*. New York: John Wiley and Sons Inc. pp.119-71.

Council of supply Chain management professionals, 2008. *SUPPLY CHAIN MANAGEMENT TERMS and GLOSSARY*. [Online] Available at: www.cscmp.org/Website/AboutCSCMP/Definitions/Definitions.asp [Accessed 10 SEPTIEMBRE 2008].

Current, J.D.M.S.D., 2002. Facility Location: Applications and Theory Discrete network location models models. In: Drezner, Z., Hamacher, H. (Eds.). *Springer-Verlag, Berlin*, p.81–118.

Daskin, 1995. *Network and Discrete Location*. Sabim S.A. A wiley publication.

Delmaire, H., Diaz, J., Fernandez, H. & Ortega, M., 1999. Comparring new heuristics for the pure integer capacitated plant location problem. *Investigación Operativa*, pp.217-42.

Dezner, Z., 1988. Location Strategies For Satellites Orbita. *Naval Research Logistics*, 35, pp.503-12.

Dhaenens-Flipo, C. & Finke, G., 2001. An Integrated Model for an Industrial Production-Distribution Problem. *IIE Transactions*, 33, pp.705-15.

Dowsland, K., 1993. Some experiments with simulated annealing techniques for packing problems. *European Journal of Operational Research 68*, pp.389-99.

Flynn, J. & Ratick, S., 1988. A Multiobjective Hierarchical Covering Model For The Essential Air Services Program. *Transportation science*, 22, pp.139-47.

Gao, L.-L. & Robinson Jr., E., 1992. A dual-based optimization procedure for the two-echelon uncapacitated facility locationproblem. *Naval Research Logistics*, 39, pp.191-212.

Geoffrion, A.M. & Graves, G.W., 1974. Multicommodity Distribution System Design By Benders Decomposition. *Informs*, 20(5), pp.822-44.

Gleason, J., 1975. A Set Covering Approach To Bus Stop Location. *Omega*, 3, pp.605-08.

Higgins, A., Kozan, E. & Ferreira, L., 1977. Modeling The Number And Location Of Sidings On A Single Line Railway. *Computers and operations research*, 24, pp.209-20.

Hillier, F. & Lieberman, G.j., 2001. *Investigación de operaciones*. México : Mc Graw Hill.

Hindi, S. & Basta, T., 1994. Efficient Solution of a Multi-commodity, Two-stage Distribution Problem with Constraints on Assignment of Customers to Distribution Centres. *International Transactions in Operations Research*, 5(6), p.519–528.

Hodder, J.E. & Dincer, M.C., 1986. Multifactor model for international plant location and financing under uncertainty. *Operations Research*, 13(5), pp.601-09.

Hodgson, M.J., Rosing, K.E. & Zhang, J., 1996. Locating Vehicle Inspection Stations to Protect a Transportation Network. *Geographical Analysis*, 28, pp.299-314.

Hogan, K., 1990. Reducing Errors in Rainfall Estimates through Rain Gauge Location. *Geographical Analysis*, 22, pp.33-49.

Jang, Y., Chang, B. & Park, J., 2002. A combined model of network design and production distribution planning for a supply network. *Computers and Industrial Engineering*, 43, pp.263-81.

Jayaraman, V. & Pirkul, H., 2001. Planning and Coordination of Production and Distribution facilities for Multiple Commodities. *European Journal of Operational Research*, 133, pp.394-408.

José A. Caballero, I.E.G., 2007. UNA REVISION DEL ESTADO DEL ARTE EN OPTIMIZACIÖN. *Revista Iberoamericana de Automática e Informática Industriall*, pp.5-23.

Joseph B. Mazzola, A.W.N., 1998. Lagrangian-relaxation-based solution procedures for a multiproduct capacitated facility location problem with choice of facility type. *European Journal of Operational Research*, pp.285-99.

Klose, A. & Drexl, A., 2003. Facility location models for distribution system design. *European Journal of Operational Research*.

Lacefiel, S., 2005. Bullseye! Tools to help you target the right site. *Logistics Management*, 44(1), p.63.

Machuca, J.A., 1995. *Dirección de Operaciones*. McGrawHill cap. 8, p245.

Marín, A., 2007. Lower bounds for the two-stage uncapacitated facility location problem. *European Journal of Operational Research*, pp.1126-42.

Mauricio G. C. Resende, J.L.G.V., 2003. GRASP: Procedimientos de búsqueda miopes aleatorizados y adaptativos. *Revista Iberoamericana de Inteligencia Artificial,* p.61'76.

Melo, M., Nickel, S. & Saldanha-da-Gama, F., 2008. Facility location and supply chain management – A review. *European Journal of Operational Research (2008)*, pp., doi:10.1016/j.ejor.2008.05.007.

Miguel Angel Gutiérrez Andrade, S.D.l.C.S., 2000. Un problema de localización de plantas de gran escala. *Revista de Matem´atica: Teor´ıa y Aplicaciones 2000*, p.117–124.

Mohamed, Z.M., 1999. An Integrated Production-Distribution Model for a Multi-National Company. *International Journal of Production Economics*, 58, pp.81-92.

Moore, G.C. & ReVelle, C., 1982. The hierarchical service location problem. *Management Science*, 28, pp.775-80.

Moreno, J. & Melián, B., 2005. *METAHEURÍSTICAS PARA LA PLANIFICACIÓN LOGÍSTICA*. Ministerio de Ciencia y Tecnología proyecto TIC 2002-04242-C03-01. Tenerife: Universidad de la Laguna.

Oliveira, J. & Moura, A., 2005. A GRASP approach to the container loading problem. *IEEE Computer intelligent systems, Transportation and logistics*, pp.50-57.

Ozgur, G.B.D., 1999. Hierarchical design of an integrated production and 2-echelon distribution system. *European Journal of Operational Research 118*, p.464±484.

Pirkul, H. & Jayaraman, V., 1998. A multi-commodity, multi-plant, capacitated facility location problem: formulation and efficient heuristic solution. *Computers Operational Research*, 25(10), pp.869-78.

Resende, M. & Werneck, R., 2002. *A GRASP with path-relinking for the p-median problem*. Florham Park, NJ 07932: AT&T Labs Research.

ReVelle, C.S. & Eiselt, A.H., (2005). Location analysis: A synthesis and survey. *European Journal of Operational Research*, p.165 1–19.

Richard S. Barr, B.L.G.J.P.K.G.R.W.R.S., 1995. Designing and Reporting on Computational Experiments with Heuristic Methods.

Ross, V.J.&.A., 2003. A simulated annealing methodology to distribution network design and management, 144. *European Journal of Operational Research*, p. 629–645.

Sambola, M.A., Díaz, J.A., Fernández, E. & Martínez Ojea, C., 2004. *Dpto. Estadıstica Universidad Carlos III, Madrid*. [Online] Available at: http://www-eio.upc.es/personal/homepages/elena/Tutoriales/UPC-EIO.pdf [Accessed Septiembre 2008].

Sharma, P., 2004. Local Search for Combinatorial Optimisation Problems. *Directions*, pp.25-29.

Shen, Z.-J.M. & Daskin, M.S., 2005. Trade-off between Customer service and cost in integrated supply chain design. *Manufacturing & Service Operations Management*, pp.7, 3; ABI/INFORM Global pg. 188.

Troncoso, N. & Barriga, N., 2008. *Inteligencia Artificial Avanzada problema de las n-reinas*. Valparaiso: Universidad Técnica Federico Santa María, Depto. de informatica.

Vidal, C.J. & Goetschalckx, M., 1997. Strategic production-distribution models:A critical review with emphasis on global supply chain models. *European Journal of Operational Research*, 98, pp.1-18.

Yilmaz, P. & Catay, B., 2006. Strategic level three-stage production distribution planning with capacity expansion. *Computers and Industrial Engineering*, 51, pp.609-20.

Zuo, M., Kuo, W. & McRoberts, K.L., 1991. Application of Mathematical Programming to a Large Scale Programming to a Large Scale Agricultural Production and Distribution System. *journal of Operational Research Society*, pp.639-48.

ANEXOS

ANEXO 1

FECHA	AUTORES	CARÁCTERÍSITICAS	TÉCNICA	REFERENCIA
1974	Geoffrion y Graves	Multicomoditis, en un solo periodo, problema de producción/ distribución	Método de descomposición de Benders	Strategic level three-stage production distribution planning with capacity expansión, Computers & Industrial Engineering 51 (2006) 609–620
1988	Cohen, M.A. and H.L. Lee	Multicomoditis, problema abastecimiento /producción /mayoristas/minoristas, donde las demandas de los productos son estocásticas	MIP utilizando aproximación heurística de jerarquización	Integrated Production and Distribution Operations: Taxonomy, Models, and Review, Chapter 17 of the book "*Handbook of Quantitative Supply Chain Analysis: Modeling in the E-Business Era*", edited by D. Simchi-Levi, S.D. Wu, and Z.-J. Shen, Kluwer Academic Publishers, 2004.
1991	Zuo, M., W. Kuo	Multiproducto , Caso de producción/distribución de semillas de maíz	MIP, utilizando aproximación heurística (utilizando dos fases, para manejar restricciones)	Integrated Production and Distribution Operations: Taxonomy, Models, and Review, Chapter 17 of the book "*Handbook of Quantitative Supply Chain Analysis:*

				Modeling in the E-Business Era", edited by D. Simchi-Levi, S.D. Wu, and Z.-J. Shen, Kluwer Academic Publishers, 2004.
1997	Hasan Pirkul, y Vaidyanathan Jayaraman	Multicomoditis, problema producción/distribución con capacidad limitada de planta	Modelo de relajación Lagrange unido con pasos que permiten encontrar un límite inferior en cada iteración	A multi-commodity, multiplant, capacited facility location problem: formulation and efficient heuristic solution, Computers Ops Res. Vol 25, No 10 pp 869-878
1994	Hindi y Basta	Multicomoditis, problema producción/distribución	MIP, branch and bound	Efficient solution of a multi-commodity, two-stage distribution problem with constraints on assignment of customers to distribution centers. International Transactions in Operations Research 5 (6), 519–528.
1995	Arntzen, B.C., G.G. Brown, T.P. Harrison, and L.L. Trafton,	Multiproducto, problema producción/distribución	MIP , técnica de restricciones elásticas	"Global Supply Chain Management" , *Interfaces*, 25 (1995), 69 – 93.
1998	Mazzola y Neebe	Multiproducto problema producción/distribución con capacidad limitada	Relajacion Lagrange	Lagrangian-relaxation-based solution procedures for a multiproduct capacitated facility location problem

				with choice of facility type, European Journal of Operational Research 115 (1999) 285±299
1999	Barbarosoglu y Ozgur	Multiproducto, problema producción/distribución con una sola planta (3 capas)	MIP, relajación lagrangean	Hierarchical Design of an Integrated Production and 2-Echelon Distribution System", *European Journal of Operational Research*, 118 (1999), 464 – 484.
1999	Dogan y Goeteschlackx	Multiproducto, problema producción/distribución (4 capas)	MIP, Método de descomposición de Bender	A Primal Decomposition Method for the Integrated Design of Multi-Period Production-Distribution Systems", *IIE Transactions*, 31 (1999), 1027 – 1036.
1999	Mohamed	Multiproducto, problema producción/distribución en distintos paises	MIP	An Integrated Production-Distribution Model for a Multi-National Company Operating under varying exchange rates", *International Journal of Production Economics*, 58 (1999), 81 – 92.
1999	Ozdamar y Yazgac	Multiproducto, problema producción/distribución	Relajación de restricciones	Integrated Production and Distribution Operations: Taxonomy, Models, and Review, Chapter 17 of the book "*Handbook of*

				Quantitative Supply Chain Analysis: Modeling in the E-Business Era", edited by D. Simchi-Levi, S.D. Wu, and Z.-J. Shen, Kluwer Academic Publishers, 2004.
1999	Byrne y Bakir	Multiproducto, multiperiodo	Hibrido entre método de simulación y análitico	Fuzzy multi-objective production/distribution planning decisions with multi-product and multi-time period in a supply chain Computers & Industrial Engineering 55 (2008) 676–694
2001	Dhanenens-Flipo y FInke	Multiproducto, problema producción/distribución	Formulado en MIP, resuelto en Cplex	Integrated Production and Distribution Operations: Taxonomy, Models, and Review, Chapter 17 of the book "*Handbook of Quantitative Supply Chain Analysis: Modeling in the E-Business Era*", edited by D. Simchi-Levi, S.D. Wu, and Z.-J. Shen, Kluwer Academic Publishers, 2004.
2001	Vaidyanathan Jayaraman , Anthony Ross	Multiproducto, Problema de producción/distribucion	SA (simulated-annealing)	A simulated annealing methodology to distribution network design and

				managemen, European Journal of Operational Research 144 (2003) 629–645
2002	Jang-Chang y Park	Problema producción/distribucion	GA (Genetic-Algoritm),y CPLEX(para probar efectividad de GA)	Strategic level three-stage production distribution planning with capacity expansión, Computers & Industrial Engineering 51 (2006) 609–620
2005	Gen Syarif	Multiproductos, problema producción/distribución de inventarios para múltiples periodos	Algoritmo híbrido de GA	Fuzzy multi-objective production/distribution planning decisions with multi-product and multi-time period in a supply chain Computers & Industrial Engineering 55 (2008) 676–694
2008	Tien-Fu-Liang	Multiproducto, multiperido, problemas de producción/distribución con información imprecisa	FMOLP (Fuzzy multi-objective linear programming)	Fuzzy multi-objective production/distribution planning decisions with multi-product and multi-time period in a supply chain Computers & Industrial Engineering 55 (2008) 676–694

ANEXO 2

	CARACTERISTICAS / MODELO	01	02	03	04	05	6	07	08	09	10	11	12	13	14	15	TOTAL	1,3,4,5
1	Kuehn and Hamburger (1963)	0	1	0	1	0	1	0	1	0	1	1	1	1	1	1	10	1
2	Efroymson and Ray (1966)	0	1	0	1	0	1	0	1	0	1	1	1	1	1	1	10	1
3	Spielberg (1969)	0	1	0	1	0	1	0	1	0	1	1	1	1	1	1	10	1
4	Khumawala (1972)	0	1	0	1	0	1	0	1	0	1	1	1	1	1	1	10	1
5	Elson (1972)	0	1	0	1	1	1	0	1	1	1	1	1	1	1	1	12	2
6	Warskawski (1973)	0	1	0	0	1	1	0	1	0	1	1	1	1	1	1	10	1
7	Geoffrion and Graves (1974)	1	1	0	1	1	0	0	1	1	1	1	1	1	1	1	12	3
8	Balachandran and Jain (1976)	0	1	0	0	0	1	0	0	1	1	0	1	1	1	1	8	0
9	Khumawala and Whybark (1976)	0	1	0	1	0	1	0	1	0	1	1	1	1	1	1	10	1
10	leblanc (1977)	0	1	0	0	0	1	0	0	1	1	0	1	1	1	1	8	0
11	Akinc and Khumawala (1977)	0	1	0	1	0	0	0	1	0	1	1	1	1	1	1	9	1
12	Kaufman et al. (1977)	1	1	0	1	0	1	0	1	1	1	1	1	1	1	1	12	2
13	Nauss (1978)	0	1	0	1	0	0	0	1	0	1	1	1	1	1	1	9	1

14	Khumawala and Neebe (1978)	0	1	0	1	1	1	0	1	0	1	1	1	1	1	1	11	2
15	Erlenkotter (1978)	0	1	0	1	0	1	0	1	0	1	1	1	1	1	1	10	1
16	Geoffrion and McBride (1978)	0	1	0	1	0	0	0	1	0	1	1	1	1	1	1	9	1
17	Geoffrion et al. (1978)	1	1	0	1	1	0	0	1	1	1	1	1	1	1	1	12	3
18	Dearing and Newruck (1979)	0	1	0	1	0	0	0	1	0	1	1	1	1	1	1	9	1
19	Neebe and Khumawala (1981)	0	1	0	1	1	1	0	1	0	1	1	1	1	1	1	11	2
20	Karkazis and Boffey (1981)	0	1	0	1	1	1	0	1	0	1	1	1	1	1	1	11	2
21	Gross et al. (1981)	0	1	0	0	1	1	0	0	0	1	0	1	1	1	1	8	1
22	Roy and Erlenkotter (1982)	0	1	0	0	0	1	0	1	0	1	1	1	1	1	1	9	0
23	Cabot and Erenguc (1984)	0	1	0	1	0	1	0	1	0	1	1	1	1	1	1	10	1
24	Tcha and Lee (1984)	0	1	0	1	0	1	0	1	0	1	1	1	1	1	1	10	1
25	Cohen and Lee (1989)	1	0	0	1	1	1	0	1	1	1	1	1	0	1	1	11	3
26	Cohen and Moon (1991)	1	1	0	1	1	1	0	1	1	1	1	1	0	1	1	12	3
27	Fleishmann (1993)	0	1	0	1	1	1	0	1	0	1	1	1	1	1	1	11	2

28	Arntzen et al. (1995)	1	1	0	0	1	0	0	1	0	1	0	0	0	1	0	6	2
29	Cole (1995)	1	1	1	0	1	0	0	0	1	0	0	0	1	1	0	7	3
30	Goetschalckx et al. (1995)	1	1	0	0	1	0	0	0	0	0	0	0	1	1	0	5	2
31	Dogan and Goetschalckx (1999)	1	1	0	0	1	0	0	1	1	1	0	0	1	1	0	8	2
32	Vidal and Goetschalckx (2000)	1	1	0	1	1	1	0	1	1	1	0	1	0	0	1	10	3
33	Novick and Turnquist (2001)	0	1	0	0	1	1	0	0	1	1	0	1	1	1	1	9	1
34	Our proposed model	0	1	0	0	1	0	1	0	0	0	0	0	1	0	0	4	1

EXPLICACIÓN DE LAS CARACTERÍSTICAS

01	PRODUCCION/ DISTRIBUCION	04	DEMANDA DETERMINISTICA	07	VARIOS CENTROS DE DISTRIBUCIÓN POR CLIENTE	10	MANEJA COSTO DE INSTALACION	13	NO MANEJA TAXES INTERNACINALES
02	OBJETIVO DE MINIMIZAR COSTOS	05	MULTIPLES PRODUCTOS	08	NO MANEJA NIVEL DE SERVICIO	11	NO MANEJA NIVEL DE INVENTARIOS	14	NO MANEJA LEAD TIME
03	2 ESLABONES	6	SIN RESTRICCION DE CAPACIDAD	09	INCLUYE COSTOS LINEALES DE TRANPOSRTE	12	NO MANEJA NIVEL DE MANIPULACIÓN DEL PRODUCTO	15	NO MANEJA PIPELINE COST

Anexo 3

Código fuente del algoritmo GRASP en lenguaje VBA de Excel

```
Option Explicit
Dim VAFO As Double              Contiene el Valor Actual Funcion Objetivo
Dim cantidad As Double          'Es la cantidad de sitios utilizados o escogidos
Dim Alfa, Max_iteraciones As Double 'Parametros del algoritmo
Public Instancia As String      'la instancia que se esta corriendo
Public k, m, j, i As Double     'Variables que determinan el tamaño de la instancia
Dim ki, mi, ji, ii As Double    'variables de control de ciclos
Dim ruta As String              'la ruta donde se encientran lo archivos para el funcionamiento
Dim MejorSol As Double          'la mejor solucion encintrada hasta el momento
Dim Replica As Double           'numero de la replica que se esta corriendo
Dim NumIter As Double           'Numero de iteraciones totales del algoritmo

Sub main()
'Esta funcion es la que se encarga de automatizar el proceso de ejecución del algoritmo leyendo de una tabla en MS-Excel
tanto los parametros como la instancia que se necesitaba correr

Dim HojaActual As String
Dim fs As Object
Dim tiempo As Variant

ruta = "C:\pruebaglpk\TESIS"
HojaActual = ActiveSheet.Name

While ActiveCell <> ""

   Replica = ActiveCell.Offset(0, -1)
   Instancia = ActiveCell
   Alfa = ActiveCell.Offset(0, 4)
   Max_iteraciones = ActiveCell.Offset(0, 5)

   On Error Resume Next
   Set fs = CreateObject("Scripting.FileSystemObject")
   fs.createfolder (ruta & "\RESULTADOS\GRASP " & Str(Alfa) & " - " & Max_iteraciones)
   On Error GoTo 0

   tiempo = Time 'lleva el control de tiempo de ejecucion del algoritmo

   MejorSol = 0
   Call GRASP(Alfa, Max_iteraciones)
   Sheets(HojaActual).Select
   ActiveCell.Offset(0, 1) = MejorSol
   ActiveCell.Offset(0, 2) = Time - tiempo
   ActiveCell.Offset(0, 3) = NumIter
   ActiveCell.Offset(1, 0).Select

Wend

End Sub

Private Sub Cargar_Valores()
'Segun la instancia que se esté corriendo esta funcion lee el tamaño estos valores se usan a lo largo de todo el algoritmo.

Dim HojaActual As String

'Valores de prueba
'Instancia = "I 1"
'HojaActual = ActiveSheet.Name
With ActiveWorkbook.Sheets(Instancia)

   k = .Cells(1, 1).End(xlDown)
   m = .Cells(1, 2).End(xlDown)
   j = .Cells(1, 3).End(xlDown)
   i = .Cells(1, 4).End(xlDown)
```

```
End With
'ActiveWorkbook.Sheets(HojaActual).Select

End Sub

Sub GRASP(Alfa, Max_iteraciones)
'Este procedimiento es el procedimiento principal de 'la metaheuristica GRASP, y es el que llama las funcione adicionales.
Cada procedimiento adicional será explicado mas adelante

Dim iteracion As Double
Dim semilla As Double
Dim solucion As Double
Dim tiempo As Date
'Valores de pruba
'Dim Max_iteraciones, Alfa As Double
'Max_iteraciones = 2
'Instancia = "Instancia"
'Alfa = 0.3
'On Error GoTo DisplayMsg

iteracion = 1
NumIter = 0
'Worksheets("solucion").Cells.Clear

Call Cargar_Valores 'Lee los valores de las variables globales del tamano del problema

    While iteracion <= Max_iteraciones
      Call Fase_Construccion     'Esta rutina genrera una solucion factible al problema en la hoja de solucion parcial

      Call BusquedaLocal         'Llama un progama de solucion lineal para solucionar el problema unicamente
                                 'con las instalaciones escogidas en la construcción

      Call actualizar(iteracion)  'Compara la respuesta obtenida despues de la busqueda local
                                 'con la mejor solucion encontrada hasta ese momento para escoger la mejor
      iteracion = iteracion + 1
      NumIter = NumIter + 1
    Wend

End Sub

Private Sub actualizar(ByRef iteracion)
'En caso que la solucion encontrada despues de la busqueda local sea mejor que la que se etenia hasta ese entonces
'se reemplaza por la que estuviera en ese momento. generando un archivo de texto dentro de la "ruta" con el
'nombre de la instancia y de la corrida

Dim solucion As Double
Dim fs As Object
Dim f As Object

solucion = LeerSolucion

  If MejorSol <= solucion And MejorSol <> 0 Then

    On Error Resume Next
    Set fs = CreateObject("Scripting.FileSystemObject")
    Set f = fs.GetFile(ruta & "\SolucParcial.txt")
    f.Delete
    On Error GoTo 0

  Else

    'On Error Resume Next
    Set fs = CreateObject("Scripting.FileSystemObject")
    Set f = fs.GetFile(ruta & "\SolucParcial.txt")
    f.Copy ruta & "\RESULTADOS\GRASP " & Str(Alfa) & " - " & Max_iteraciones & "\" & Instancia & "-" &
Str(Replica) & ".txt"
    'On Error GoTo 0
    MejorSol = solucion
    iteracion = 0
```

```
    End If
End Sub

Private Sub Fase_Construccion()
'Esta rutina genrera una solucion factible al problema en la hoja de solucion parcial

Dim Tinicio, Ttotal As Double
Dim HojaActual As String
'Alfa = 0.5
'Dim semilla As Double
'Instancia = "I 16"
'Call Cargar_Valores
Tinicio = Time
'semilla = 0
Dim DemTotal As Double

Dim cliente, producto, instalacion, demanda, planta As Double

Dim Evaluaciones() As Double
Dim RCL() As Double ' la matriz rcl va a tener tres renglones y multiples columnas  entonces las prob estaran en el ultimo renglon

ReDim Evaluaciones(m * j - 1)
'Crea una copia de los datos de la hoja "parametros" en la hoja CDatos para modificarla sin danar los datos originales
Call CopiarDatos
'Borra la hoja SolucParcial para generar una nueva solución  VAFO pasa por referencia para ser inicializado en 0
Call ResetSolucion(VAFO)
DemTotal = 1
Sheets("CDatos").Select
Do
    'Implica saber cual es el cliente y el producto mas demandado. Se basa en la hoja CDatos. los datos 'pasan por referencia para ser modificados. En pocas palabras escoge la demanda mas alta 'e identifica cliente y producto de dicha demanda
    Call EscogerDemanda(cliente, producto, demanda, DemTotal)
    'Evaluar me retorna un vector de tamaño  plantas*instalaciones (contiene los costos totales) 'hay que tener en cuenta que si no se tiene capacidad de produccion disponible no se multiplica toda la dda. 'sino lo q sea posible transportar. Recibe por ref el vector de evaluaciones para ser llenado. si la capacidad de produccion es cero entonces la evaluacion sera cero
    Call EvaluarCostosIncrementales(cliente, producto, demanda, Evaluaciones)
    'Hay que ajustar el valor de "Alfa" para obtener el mejor desempeño del algotirmo. Pasa RCL por ref para modificarlo (no se puede mandar por referencia xq no se puede modificar el tamano dentro de la rutina) 'guarda el numero de planta y numero de instalación y probabilidad de ser escogida. si se genera un error de desbordamiento por culpa de una division en cero, es posible que la capacidad de 'produccion de algun producto no sea suficiente para poder suplir la demanda, luego no tiene solucion.
    RCL = CrearRCL(Evaluaciones)
    'se escoge una instalacion y una planta para suplir la 'demanda escogida. utiliza la semilla
    Call EscogerCandidato(RCL, planta, instalacion)
    'se adiciona el candidato que se escogio a la solución 'que existia hasta el momento, y se actualizan los valores de demanda y capacidad de produccion reiniciar la RCL
    Call Actualice_SolucParcial(cliente, producto, planta, instalacion)
Loop While Application.WorksheetFunction.Sum( _
        Range(Cells(2, 13), Cells((k * i + 2), 13))) > 0 'la condicion de salida del ciclo se cumple cuando en la hoja cdatos no hay mas demanda de ningun producto

HojaActual = ActiveSheet.Name
'ActiveWorkbook.Sheets("SolucParcial").Select
With Sheets("SolucParcial")
    .Cells(2, 3) = .Application.WorksheetFunction.Sum(Range(.Cells(7, 8), .Cells(7, 7).End(xlDown)))
End With
'Ttotal = Time - Tinicio
'Cells(2, 4) = Ttotal
'ActiveWorkbook.Sheets(HojaActual).Select

End Sub

Private Sub CopiarDatos() 'toma la copia de la hoja de parámetros sheets(instancia).select
    Cells.Copy
    Sheets("CDatos").Select
    Cells.Select
    ActiveSheet.Paste
End Sub
```

```
Private Sub ResetSolucion(ByRef VAFO)
'Inicializa la hoja de solucion para comenzar de cero una nueva fase de construcción

   With Sheets("SolucParcial")

      .Cells.ClearContents

      .Cells(2, 1) = "VALOR DE LA FUNCION = "
      .Cells(4, 1) = "ASIGNACIONES"
      .Cells(6, 1) = "k"
      .Cells(6, 2) = "m"
      .Cells(6, 3) = "j"
      .Cells(6, 4) = "i"
      .Cells(6, 5) = "Costo por unidad"
      .Cells(6, 6) = "Y kmji"
      .Cells(6, 7) = "Costo Variable Total"
      .Cells(6, 8) = "Costos Fijos"

   End With

   VAFO = 0
End Sub

Private Sub EscogerDemanda(ByRef cliente, ByRef producto, ByRef demanda, ByRef DemTotal)
'Escoge la demanda por la que inicia el algoritmo a construir la solución

Dim tamano, fila As Double

   tamano = i * k
   'Sheets("CDatos").Select
   demanda = Application.WorksheetFunction.Max(Range(Cells(3, 13), Cells(2 + tamano, 13)))
   fila = Application.WorksheetFunction.Match(demanda, Range( _
            Cells(3, 13), Cells(2 + tamano, 13)), 0)
   cliente = Cells(fila + 2, 12)
   producto = Cells(fila + 2, 11)
   DemTotal = Application.WorksheetFunction.Sum(Range(Cells(3, 13), Cells(2 + tamano, 13)), 0)

End Sub

Private Sub EvaluarCostosIncrementales(cliente, producto, demanda, ByRef Evaluaciones) '
'Evaluar me retorna un vector de tamano plantas*instalaciones (contiene los costos totales) 'hay que tener en cuenta que si no se tiene capacidad de produccion disponible no se multiplica toda la dda. sino lo q sea posible transportar segun por la capacidad disponible.

Dim c, CAPACIDAD, CV, CF As Double

'Valores de prueba
'Dim VAFO, cliente, producto, demanda As Double
'Dim Evaluaciones() As Double
'Call Cargar_Valores
'ReDim Evaluaciones(m * j - 1)

c = 0
While c < (m * j)

   CV = Cells(((producto - 1) * m * j * i) - 1 + 3 + c * i + cliente, 5)
   CAPACIDAD = Cells(((producto - 1) * m) + ((c \ j) + 1) + 2, 9)

   If (c + 1) Mod j = 0 Then
      CF = Cells(j + 2, 16)
    Else
      CF = Cells((c + 1) Mod j + 2, 16)
   End If

   If CAPACIDAD = 0 Then
      Evaluaciones(c) = 0
   Else
      Evaluaciones(c) = (CV * Application.WorksheetFunction.Min( _
                CAPACIDAD, demanda)) + CF
   End If
```

```
    c = c + 1
Wend
'renglon para probar la funcion de crearRCL
'Dim RCL() As Double

'RCL = CrearRCL(0.5, Evaluaciones)

End Sub
Private Function CrearRCL(Evaluaciones)
'En esta funcion se genera la lista de candidatos del algoritmo. Con base a esta 'lista son seleccionados los cadidatos que se adicionan a la solución

Dim Umbral, c, Suma As Double
Dim RCL() As Double
Dim Cmin, Cmax As Double
Dim MINI, MA As Double

ReDim RCL(2, 0)
If Alfa < 0 Or Alfa > 1 Then
 MsgBox "Error en el parametro Alfa: " & Alfa
End If

Cmin = MinSinCeros(Evaluaciones) 'Application.WorksheetFunction.Min(Evaluaciones)
Cmax = Application.WorksheetFunction.Max(Evaluaciones)
Umbral = Cmin + (Cmax - Cmin) * Alfa
Suma = 0

c = 0
MINI = Cmin
MA = 0
While c < (m * j)

   If Evaluaciones(c) <= Umbral And Evaluaciones(c) > 0 Then
      ReDim Preserve RCL(2, (UBound(RCL, 2) + 1))
      RCL(0, UBound(RCL, 2) - 1) = (c Mod j) + 1 'instalacion
      RCL(1, UBound(RCL, 2) - 1) = (c \ j) + 1   'planta
      RCL(2, UBound(RCL, 2) - 1) = Evaluaciones(c)

      If MA < Evaluaciones(c) Then
         MA = Evaluaciones(c)
      Else
         If MINI > Evaluaciones(c) Then
            MINI = Evaluaciones(c)
         End If
      End If

   End If
   c = c + 1

Wend

Call ArreglarValores(RCL, MINI, MA)
Suma = SumarValores(RCL)

c = 1
RCL(2, 0) = RCL(2, 0) / Suma
While c < UBound(RCL, 2)

   RCL(2, c) = RCL(2, c) / Suma
   RCL(2, c) = RCL(2, c - 1) + RCL(2, c)
   c = c + 1

Wend

CrearRCL = RCL
End Function
Private Function MinSinCeros(Evaluaciones)
'devuelve el minimo del vector de evaluaciones sin tener en cuenta los datos = 0
Dim c, MinSinC As Double
```

```
c = 0
MinSinC = 4.94E+304
While c < (m * j)

   If Evaluaciones(c) < MinSinC And Evaluaciones(c) > 0 Then
      MinSinC = Evaluaciones(c)
   End If
   c = c + 1

Wend

If MinSinC = 4.94E+304 Then
   MsgBox "ERROR BUSCANDO EL MINIMO DE LA LISTA DE EVALUACIONES"
   Exit Function
End If

MinSinCeros = MinSinC

End Function
Private Sub ArreglarValores(ByRef RCL, MINI, MA)
'Es llamada desde la creacion de la RCL y el objetivo es ponerle al mayor valor un numero que sea el menor de todos los de
la lista, hay que recordar que RCL es una matriz con tres renglones y solo necesitamos el tercer renglon
Dim c As Double

c = 0
While c < UBound(RCL, 2)

   RCL(2, c) = (MA - RCL(2, c)) + MINI
   c = c + 1

Wend

End Sub
Private Function SumarValores(ByRef RCL)
'Devuelve el valor de la suma de los valores de los candidatos en la RCL con el fin asignatles una probabilidad

Dim c, Suma As Double

c = 1
Suma = RCL(2, 0)
While c < UBound(RCL, 2)

   Suma = RCL(2, c) + Suma
   c = c + 1

Wend

SumarValores = Suma

End Function

Private Sub EscogerCandidato(RCL, ByRef planta, ByRef instalacion)
'Genera un número aleatorio bajo una distribucion uniforme y 'escoge un candidato de la RCL

Dim a As Boolean
Dim c As Double
Dim r  As Double
Randomize 'semilla

c = 0
a = 0
r = Rnd()

  While a = 0

    If r <= RCL(2, c) Then
      instalacion = RCL(0, c)
      planta = RCL(1, c)
      a = 1
    End If
```

```
    c = c + 1
  Wend
End Sub

Private Sub Actualice_SolucParcial(cliente, producto, planta, instalacion)
'Actualiza la solución parcial con el candidato que se haya escogido de la RCL
Dim CV, CF, DDA, CAP, Ymjik, fila As Double
Dim HojaActual As String
With Worksheets("CDatos")
  CV = .Cells(((producto - 1) * m * j * i) + (planta - 1) * j * i + (instalacion - 1) * i + instalacion - 1 + 3 + cliente, 5)
  CF = .Cells(instalacion + 2, 16)
  'Call organizarDDA
  DDA = .Cells((producto - 1) * i + cliente + 2, 13)
  CAP = .Cells((producto - 1) * m + planta + 2, 9)
  If DDA < CAP Then
    CV = DDA * CV
    Ymjik = DDA
    .Cells((producto - 1) * i + cliente + 2, 13) = 0
    .Cells((producto - 1) * m + planta + 2, 9) = CAP - DDA
  Else
    CV = CAP * CV
    .Cells((producto - 1) * i + cliente + 2, 13) = DDA - CAP
    .Cells((producto - 1) * m + planta + 2, 9) = 0
    Ymjik = CAP
  End If
  .Cells(instalacion + 2, 16) = 0
End With
  'HojaActual = ActiveSheet.Name
  'ActiveWorkbook.Sheets("SolucParcial").Select
  With Worksheets("SolucParcial")
    If .Cells(6, 1).End(xlDown).row >= 1048576 Then
     fila = 7
    Else
     fila = .Cells(6, 1).End(xlDown).row + 1
    End If
    .Cells(fila, 1) = producto
    .Cells(fila, 2) = planta
    .Cells(fila, 3) = instalacion
    .Cells(fila, 4) = cliente
    .Cells(fila, 5) = CV / Ymjik
    .Cells(fila, 6) = Ymjik
    .Cells(fila, 7) = CV
    If CF > 0 Then .Cells(fila, 8) = CF
    End With
 ' ActiveWorkbook.Sheets(HojaActual).Select

End Sub
Private Sub BusquedaLocal()

  Call CREAR_DATOS

  Call LLamaGLP

End Sub

Private Sub CREAR_DATOS()
'variables de prueba del procedimiento
'Alfa = 0.5
'Instancia = "I 1"
'k = 3
'm = 20
'j = 25
'i = 30
Dim fs As Object
Dim a As Object
Dim c As Double

Sheets(Instancia).Select
'ruta = "C:\pruebaglpk\TESIS"
Set fs = CreateObject("Scripting.FileSystemObject")
```

```
Set a = fs.CreateTextFile(ruta & "\data.txt", True)

a.Writeline ("data;") 'escribe los conjuntos
a.Writeline ("set K:=")
For ki = 1 To k
   a.Writeline (Str(ki))
Next
a.Writeline (";")

a.Writeline ("set M:=")
For mi = 1 To m
   a.Writeline (Str(mi))
Next
a.Writeline (";")

a.Writeline ("set J:=")
ji = 3
While Cells(ji, 15) <> ""
   If Sheets("CDatos").Cells(ji, 16) = 0 Then
      a.Writeline (Str(Cells(ji, 15)))
   End If
   ji = ji + 1
Wend
a.Writeline (";")

a.Writeline ("set I:=")
For ii = 1 To i
   a.Writeline (Str(ii))
Next
a.Writeline (";") 'escribe las capacidades de produccion
c = 3
a.Writeline ("param s:= [*,*]")
While Cells(c, 7) <> ""
   a.Writeline (Str(Cells(c, 7)) & " " & Str(Cells(c, 8)) & " " & Str(Cells(c, 9)))
   c = c + 1
Wend
a.Writeline (";") 'escribe las demandas de los clientes
c = 3
a.Writeline ("param h:= [*,*]")
While Cells(c, 11) <> ""
   a.Writeline (Str(Cells(c, 11)) & " " & Str(Cells(c, 12)) & " " & Str(Cells(c, 13)))
   c = c + 1
Wend
a.Writeline (";") 'escribe los costos de transporte o Variables
c = 3
a.Writeline ("param c:= [*,*,*,*]")

While Cells(c, 1) <> ""
   If Sheets("CDatos").Cells(Cells(c, 3) + 2, 16) = 0 Then
      a.Writeline (Str(Cells(c, 1)) & " " & _
      Str(Cells(c, 2)) & " " & Str(Cells(c, 3)) & " " & _
      Str(Cells(c, 4)) & " " & Str(Cells(c, 5)))
   End If
   c = c + 1
Wend
a.Writeline (";") 'Escribe los costos de las instalaciones escogidas
c = 3
a.Writeline ("param f:= [*]")
While Cells(c, 15) <> ""
   If Sheets("CDatos").Cells(c, 16) = 0 Then
      a.Writeline (Str(Cells(c, 15)) & " " & _
      Str(Cells(c, 16)))
   End If
   c = c + 1
Wend
a.Writeline (";")
a.Writeline ("end;")
a.Close

End Sub
```

```
Private Sub LLamaGLP()
Dim fs As Object
Dim f As Object
Dim operacion As Double

'ruta = "C:\pruebaglpk\TESIS"
On Error Resume Next
Set fs = CreateObject("Scripting.FileSystemObject")
Set f = fs.GetFile(ruta & "\SolucParcial.txt")
f.Delete
On Error GoTo 0

operacion = Shell(ruta & "\glpsol.exe --model " & ruta & "\Modelo.txt --data " & ruta & "\Data.txt --output " & ruta & "\SolucParcial.txt", vbNormalFocus)

End Sub

Private Function LeerSolucion()
Dim NRenglon, cont, reng As Double
Dim fs As Object
Dim f As Object
Dim ts As Object
Dim cadena As String
Dim newHour, newMinute, newSecond, waitTime As Variant

NRenglon = 6
'ruta = "C:\pruebaglpk\TESIS"
Set fs = CreateObject("Scripting.FileSystemObject")

On Error Resume Next
cont = 0
Do
   Err.Clear
   Set f = fs.GetFile(ruta & "\SolucParcial.txt")
   cont = cont + 1

Loop While Err.Number

On Error GoTo 0
newHour = Hour(Now())
newMinute = Minute(Now())
newSecond = Second(Now()) + 2
waitTime = TimeSerial(newHour, newMinute, newSecond)
Application.Wait waitTime

Set ts = f.openastextstream(1, -2)
For reng = 1 To NRenglon - 1

 cadena = ts.readline

Next reng

Dim CodeAux, caracter As String

cadena = ts.read(19)
Do
   caracter = ts.read(1)
   CodeAux = CodeAux & caracter

Loop While caracter <> " "
cadena = Trim(CodeAux)
LeerSolucion = CDbl(cadena)

End Function
```

ANEXO 4

Ejemplo archivo de texto solución arrojado por el algoritmo

Problem: Modelo
Rows: 61
Columns: 1500
Non-zeros: 4500
Status: OPTIMAL
Objective: cost = 1 87 194.8075 (MINi mum)

No.	Row name	St	Activity	Lower bound	Upperbound
1	cost	B	160577		
2	demand[1,1]	NL	98.6099	98.6099	
3	demand[1,2]	NL	164.97	164.97	
4	demand[1,3]	NL	73.6904	73.6904	
5	demand[1,4]	NL	11.6277	11.6277	
6	demand[1,5]	NL	43.0185	43.0185	
7	demand[1,6]	NL	100.738	100.738	
8	demand[1,7]	NL	9.40346	9.40346	
9	demand[1,8]	NL	25.4358	25.4358	
10	demand[1,9]	NL	188.752	188.752	
11	demand[1,10]	NL	152.062	152.062	
12	demand[1,11]	NL	39.856	39.856	
13	demand[1,12]	NL	30.8065	30.8065	
14	demand[1,13]	NL	132.455	132.455	
15	demand[1,14]	NL	91.5118	91.5118	
16	demand[1,15]	NL	159.836	159.836	
17	demand[1,16]	NL	105.877	105.877	
18	demand[1,17]	NL	52.2134	52.2134	
19	demand[1,18]	NL	147.323	147.323	
20	demand[1,19]	NL	197.138	197.138	
21	demand[1,20]	NL	29.7614	29.7614	
22	demand[1,21]	NL	23.1876	23.1876	
23	demand[1,22]	NL	56.5197	56.5197	
24	demand[1,23]	NL	53.191	53.191	
25	demand[1,24]	NL	160.675	160.675	
26	demand[1,25]	NL	183.231	183.231	
27	demand[1,26]	NL	168.039	168.039	
28	demand[1,27]	NL	193.681	193.681	
29	demand[1,28]	NL	164.059	164.059	
30	demand[1,29]	NL	148.518	148.518	
31	demand[1,30]	NL	195.859	195.859	
32	demand[1,31]	NL	189.176	189.176	
33	demand[1,32]	NL	190.707	190.707	
34	demand[1,33]	NL	79.169	79.169	
35	demand[1,34]	NL	22.1204	22.1204	
36	demand[1,35]	NL	105.754	105.754	
37	demand[1,36]	NL	104.89	104.89	

38 demand[1,37]	NL	194.873	194.873		
39 demand[1,38]	NL	150.253	150.253		
40 demand[1,39]	NL	145.624	145.624		
41 demand[1,40]	NL	162.478	162.478		
42 demand[1,41]	NL	140.515	140.515		
43 demand[1,42]	NL	8.10151	8.10151		
44 demand[1,43]	NL	137.752	137.752		
45 demand[1,44]	NL	139.071	139.071		
46 demand[1,45]	NL	1.8016	1.8016		
47 demand[1,46]	NL	101.049	101.049		
48 demand[1,47]	NL	78.49	78.49		
49 demand[1,48]	NL	108.118	108.118		
50 demand[1,49]	NL	170.883	170.883		
51 demand[1,50]	NL	15.1428	15.1428		
52 capaci[1,1]	NU	729.016			729.016
53 capaci[1,2]	NU	69.968			69.968
54 capaci[1,3]	NU	1076.2			1076.2
55 capaci[1,4]	NU	335.882			335.882
56 capaci[1,5]	B	997.24			1008.24
57 capaci[1,6]	NU	183.045			183.045
58 capaci[1,7]	NU	422.251			422.251
59 capaci[1,8]	NU	140.426			140.426
60 capaci[1,9]	NU	453.877			453.877
61 capaci[1,10]	NU	1040.11			1040.11

No.	Column name	St	Activity	Lower bound	Upper bound	Marginal
1	y[1,1,5,1]	NL	0	0		49.998
2	y[1,1,5,2]	NL	0	0		101.693
3	y[1,1,5,3]	NL	0	0		75.6867
4	y[1,1,5,4]	NL	0	0		45.0257
5	y[1,1,5,5]	NL	0	0		74.7569
6	y[1,1,5,6]	NL	0	0		44.6306
7	y[1,1,5,7]	NL	0	0		37.7801
8	y[1,1,5,8]	NL	0	0		64.5372
9	y[1,1,5,9]	NL	0	0		11.8658
10	y[1,1,5,10]	NL	0	0		76.2389
11	y[1,1,5,11]	NL	0	0		30.311
12	y[1,1,5,12]	NL	0	0		59.0803
13	y[1,1,5,13]	NL	0	0		123.101
14	y[1,1,5,14]	B	91.5118	0		
15	y[1,1,5,15]	NL	0	0		89.0427
16	y[1,1,5,16]	NL	0	0		25.9767
17	y[1,1,5,17]	NL	0	0		28.418
18	y[1,1,5,18]	NL	0	0		82.0038
19	y[1,1,5,19]	NL	0	0		49.47
20	y[1,1,5,20]	NL	0	0		38.1441
21	y[1,1,5,21]	NL	0	0		94.8342
22	y[1,1,5,22]	NL	0	0		66.1793
23	y[1,1,5,23]	NL	0	0		28.7546
24	y[1,1,5,24]	NL	0	0		53.4326
25	y[1,1,5,25]	NL	0	0		11.8466
26	y[1,1,5,26]	NL	0	0		36.1263
27	y[1,1,5,27]	NL	0	0		69.1767

28 y[1,1,5,28]	NL	0	0	106.018
29 y[1,1,5,29]	NL	0	0	18.9029
30 y[1,1,5,30]	NL	0	0	49.401
31 y[1,1,5,31]	NL	0	0	110.583
32 y[1,1,5,32]	B	190.707	0	
33 y[1,1,5,33]	NL	0	0	9.7008
34 y[1,1,5,34]	NL	0	0	47.0491
35 y[1,1,5,35]	NL	0	0	92.634
36 y[1,1,5,36]	NL	0	0	108.341
37 y[1,1,5,37]	NL	0	0	67.1144
38 y[1,1,5,38]	NL	0	0	108.373
39 y[1,1,5,39]	NL	0	0	88.4586
40 y[1,1,5,40]	NL	0	0	80.9511
41 y[1,1,5,41]	NL	0	0	78.9367
42 y[1,1,5,42]	NL	0	0	85.1489
43 y[1,1,5,43]	NL	0	0	92.5465
44 y[1,1,5,44]	NL	0	0	58.3623
45 y[1,1,5,45]	NL	0	0	23.2571
46 y[1,1,5,46]	B	101.049	0	
47 y[1,1,5,47]	NL	0	0	93.9068
48 y[1,1,5,48]	NL	0	0	46.938
49 y[1,1,5,49]	NL	0	0	51.0459
50 y[1,1,5,50]	NL	0	0	99.3651
51 y[1,1,8,1]	NL	0	0	90.8075
52 y[1,1,8,2]	NL	0	0	31.6782
53 y[1,1,8,3]	NL	0	0	50.3641
54 y[1,1,8,4]	NL	0	0	61.469
55 y[1,1,8,5]	NL	0	0	0.875428
56 y[1,1,8,6]	NL	0	0	74.3042
57 y[1,1,8,7]	NL	0	0	71.5469
58 y[1,1,8,8]	NL	0	0	79.2494
59 y[1,1,8,9]	NL	0	0	41.5665
60 y[1,1,8,10]	B	152.062	0	
61 y[1,1,8,11]	NL	0	0	17.3132
62 y[1,1,8,12]	NL	0	0	57.5922
63 y[1,1,8,13]	NL	0	0	45.885
64 y[1,1,8,14]	NL	0	0	86.0638
65 y[1,1,8,15]	NL	0	0	64.4785
66 y[1,1,8,16]	NL	0	0	8.16244
67 y[1,1,8,17]	NL	0	0	28.2568
68 y[1,1,8,18]	NL	0	0	102.675
69 y[1,1,8,19]	NL	0	0	98.7695
70 y[1,1,8,20]	NL	0	0	40.6819
71 y[1,1,8,21]	NL	0	0	57.7865
72 y[1,1,8,22]	B	56.5197	0	
73 y[1,1,8,23]	NL	0	0	56.1347
74 y[1,1,8,24]	NL	0	0	94.141
75 y[1,1,8,25]	NL	0	0	68.0508
76 y[1,1,8,26]	NL	0	0	66.9002
77 y[1,1,8,27]	NL	0	0	50.5704
78 y[1,1,8,28]	NL	0	0	50.0669
79 y[1,1,8,29]	NL	0	0	68.9443
80 y[1,1,8,30]	NL	0	0	112.091
81 y[1,1,8,31]	NL	0	0	80.9445

82 y[1,1,8,32]	NL	0	0	97.9951
83 y[1,1,8,33]	NL	0	0	110.353
84 y[1,1,8,34]	NL	0	0	45.5521
85 y[1,1,8,35]	NL	0	0	117.618
86 y[1,1,8,36]	NL	0	0	81.2876
87 y[1,1,8,37]	NL	0	0	51.4586
88 y[1,1,8,38]	NL	0	0	55.1668
89 y[1,1,8,39]	NL	0	0	95.4748
90 y[1,1,8,40]	NL	0	0	2.08683
91 y[1,1,8,41]	NL	0	0	35.851
92 y[1,1,8,42]	NL	0	0	37.8159
93 y[1,1,8,43]	NL	0	0	61.1905
94 y[1,1,8,44]	NL	0	0	31.8707
95 y[1,1,8,45]	NL	0	0	11.2858
96 y[1,1,8,46]	NL	0	0	44.6267
97 y[1,1,8,47]	NL	0	0	105.763
98 y[1,1,8,48]	NL	0	0	58.5331
99 y[1,1,8,49]	NL	0	0	46.6408
100 y[1,1,8,50]	NL	0	0	5.81463
101 y[1,1,16,1]	NL	0	0	11.0294
102 y[1,1,16,2]	NL	0	0	2.09148
103 y[1,1,16,3]	NL	0	0	70.8868
104 y[1,1,16,4]	NL	0	0	79.6128
105 y[1,1,16,5]	NL	0	0	82.3084
106 y[1,1,16,6]	NL	0	0	47.4126
107 y[1,1,16,7]	B	9.40346	0	
108 y[1,1,16,8]	NL	0	0	125.555
109 y[1,1,16,9]	NL	0	0	32.654
110 y[1,1,16,10]	NL	0	0	11.7227
111 y[1,1,16,11]	NL	0	0	27.1891
112 y[1,1,16,12]	NL	0	0	17.8326
113 y[1,1,16,13]	NL	0	0	70.5591
114 y[1,1,16,14]	NL	0	0	45.8551
115 y[1,1,16,15]	NL	0	0	83.6641
116 y[1,1,16,16]	NL	0	0	4.92674
117 y[1,1,16,17]	B	52.2134	0	
118 y[1,1,16,18]	NL	0	0	51.3368
119 y[1,1,16,19]	NL	0	0	36.109
120 y[1,1,16,20]	NL	0	0	3.68804
121 y[1,1,16,21]	NL	0	0	93.6949
122 y[1,1,16,22]	NL	0	0	117.495
123 y[1,1,16,23]	NL	0	0	27.9914
124 y[1,1,16,24]	NL	0	0	80.5484
125 y[1,1,16,25]	NL	0	0	27.896
126 y[1,1,16,26]	NL	0	0	55.1625
127 y[1,1,16,27]	NL	0	0	107.39
128 y[1,1,16,28]	NL	0	0	64.7889
129 y[1,1,16,29]	NL	0	0	94.2363
130 y[1,1,16,30]	NL	0	0	77.8411
131 y[1,1,16,31]	NL	0	0	58.817
132 y[1,1,16,32]	NL	0	0	39.0115
133 y[1,1,16,33]	NL	0	0	38.4031
134 y[1,1,16,34]	NL	0	0	8.10714
135 y[1,1,16,35]	NL	0	0	114.84

136 y[1,1,16,36]	NL	0	0	59.9569
137 y[1,1,16,37]	NL	0	0	79.44
138 y[1,1,16,38]	NL	0	0	79.2548
139 y[1,1,16,39]	NL	0	0	74.114
140 y[1,1,16,40]	NL	0	0	38.4591
141 y[1,1,16,41]	NL	0	0	9.86397
142 y[1,1,16,42]	NL	0	0	116.387
143 y[1,1,16,43]	NL	0	0	73.6451
144 y[1,1,16,44]	B	75.5501	0	
145 y[1,1,16,45]	NL	0	0	90.4185
146 y[1,1,16,46]	NL	0	0	106.667
147 y[1,1,16,47]	NL	0	0	21.3266
148 y[1,1,16,48]	NL	0	0	6.78974
149 y[1,1,16,49]	NL	0	0	117.032
150 y[1,1,16,50]	NL	0	0	49.1573
151 y[1,2,5,1]	NL	0	0	54.6706
152 y[1,2,5,2]	NL	0	0	107.284
153 y[1,2,5,3]	NL	0	0	128.327
154 y[1,2,5,4]	NL	0	0	57.1036
155 y[1,2,5,5]	NL	0	0	31.7649
156 y[1,2,5,6]	NL	0	0	72.1218
157 y[1,2,5,7]	NL	0	0	138.001
158 y[1,2,5,8]	NL	0	0	94.0293
159 y[1,2,5,9]	NL	0	0	61.1628
160 y[1,2,5,10]	NL	0	0	18.2695
161 y[1,2,5,11]	NL	0	0	62.6956
162 y[1,2,5,12]	NL	0	0	14.7388
163 y[1,2,5,13]	NL	0	0	100.93
164 y[1,2,5,14]	NL	0	0	35.3419
165 y[1,2,5,15]	NL	0	0	79.7746
166 y[1,2,5,16]	NL	0	0	16.7496
167 y[1,2,5,17]	NL	0	0	45.4351
168 y[1,2,5,18]	NL	0	0	66.1868
169 y[1,2,5,19]	NL	0	0	119.46
170 y[1,2,5,20]	NL	0	0	13.7993
171 y[1,2,5,21]	NL	0	0	114.235
172 y[1,2,5,22]	NL	0	0	47.116
173 y[1,2,5,23]	NL	0	0	96.7646
174 y[1,2,5,24]	NL	0	0	70.1835
175 y[1,2,5,25]	NL	0	0	52.4918
176 y[1,2,5,26]	NL	0	0	19.7371
177 y[1,2,5,27]	NL	0	0	54.9457
178 y[1,2,5,28]	NL	0	0	92.1488
179 y[1,2,5,29]	NL	0	0	108.441
180 y[1,2,5,30]	NL	0	0	15.2391
181 y[1,2,5,31]	NL	0	0	137.951
182 y[1,2,5,32]	NL	0	0	21.4888
183 y[1,2,5,33]	NL	0	0	137.695
184 y[1,2,5,34]	NL	0	0	88.9256
185 y[1,2,5,35]	NL	0	0	67.6684
186 y[1,2,5,36]	NL	0	0	134.488
187 y[1,2,5,37]	NL	0	0	138.167
188 y[1,2,5,38]	NL	0	0	74.9819
189 y[1,2,5,39]	NL	0	0	27.5776

190 y[1,2,5,40]	B	69.968	0	
191 y[1,2,5,41]	NL	0	0	115.813
192 y[1,2,5,42]	NL	0	0	95.6914
193 y[1,2,5,43]	NL	0	0	75.6452
194 y[1,2,5,44]	NL	0	0	68.1285
195 y[1,2,5,45]	NL	0	0	120.013
196 y[1,2,5,46]	NL	0	0	84.8109
197 y[1,2,5,47]	NL	0	0	92.4817
198 y[1,2,5,48]	NL	0	0	45.4318
199 y[1,2,5,49]	NL	0	0	76.1501
200 y[1,2,5,50]	NL	0	0	90.5366
201 y[1,2,8,1]	NL	0	0	45.6583
202 y[1,2,8,2]	NL	0	0	12.9067
203 y[1,2,8,3]	NL	0	0	72.5767
204 y[1,2,8,4]	NL	0	0	100.378
205 y[1,2,8,5]	NL	0	0	70.7508
206 y[1,2,8,6]	NL	0	0	58.1152
207 y[1,2,8,7]	NL	0	0	43.0432
208 y[1,2,8,8]	NL	0	0	24.2135
209 y[1,2,8,9]	NL	0	0	131.557
210 y[1,2,8,10]	NL	0	0	83.6226
211 y[1,2,8,11]	NL	0	0	83.906
212 y[1,2,8,12]	NL	0	0	33.6727
213 y[1,2,8,13]	NL	0	0	93.3071
214 y[1,2,8,14]	NL	0	0	20.0473
215 y[1,2,8,15]	NL	0	0	37.02
216 y[1,2,8,16]	NL	0	0	20.3033
217 y[1,2,8,17]	NL	0	0	80.7791
218 y[1,2,8,18]	NL	0	0	56.5139
219 y[1,2,8,19]	NL	0	0	31.2758
220 y[1,2,8,20]	NL	0	0	114.335
221 y[1,2,8,21]	NL	0	0	101.554
222 y[1,2,8,22]	NL	0	0	77.7594
223 y[1,2,8,23]	NL	0	0	73.9108
224 y[1,2,8,24]	NL	0	0	40.8903
225 y[1,2,8,25]	NL	0	0	105.81
226 y[1,2,8,26]	NL	0	0	72.8403
227 y[1,2,8,27]	NL	0	0	53.3357
228 y[1,2,8,28]	NL	0	0	108.255
229 y[1,2,8,29]	NL	0	0	73.1687
230 y[1,2,8,30]	NL	0	0	66.2926
231 y[1,2,8,31]	NL	0	0	70.9573
232 y[1,2,8,32]	NL	0	0	73.3471
233 y[1,2,8,33]	NL	0	0	101.368
234 y[1,2,8,34]	NL	0	0	74.5406
235 y[1,2,8,35]	NL	0	0	27.8003
236 y[1,2,8,36]	NL	0	0	112.536
237 y[1,2,8,37]	NL	0	0	50.5637
238 y[1,2,8,38]	NL	0	0	32.5386
239 y[1,2,8,39]	NL	0	0	117.539
240 y[1,2,8,40]	NL	0	0	76.598
241 y[1,2,8,41]	NL	0	0	18.4499
242 y[1,2,8,42]	NL	0	0	99.8626
243 y[1,2,8,43]	NL	0	0	98.7611

244 y[1,2,8,44]	NL	0	0	126.268
245 y[1,2,8,45]	NL	0	0	110.008
246 y[1,2,8,46]	NL	0	0	105.96
247 y[1,2,8,47]	NL	0	0	102.505
248 y[1,2,8,48]	NL	0	0	29.3227
249 y[1,2,8,49]	NL	0	0	129.467
250 y[1,2,8,50]	NL	0	0	24.9917
251 y[1,2,16,1]	NL	0	0	28.7617
252 y[1,2,16,2]	NL	0	0	60.3006
253 y[1,2,16,3]	NL	0	0	65.2827
254 y[1,2,16,4]	NL	0	0	92.5968
255 y[1,2,16,5]	NL	0	0	58.0828
256 y[1,2,16,6]	NL	0	0	79.1515
257 y[1,2,16,7]	NL	0	0	30.8377
258 y[1,2,16,8]	NL	0	0	33.12
259 y[1,2,16,9]	NL	0	0	62.9672
260 y[1,2,16,10]	NL	0	0	40.0017
261 y[1,2,16,11]	NL	0	0	87.9378
262 y[1,2,16,12]	NL	0	0	27.0706
263 y[1,2,16,13]	NL	0	0	96.6341
264 y[1,2,16,14]	NL	0	0	59.5048
265 y[1,2,16,15]	NL	0	0	20.3322
266 y[1,2,16,16]	NL	0	0	15.3129
267 y[1,2,16,17]	NL	0	0	10.9116
268 y[1,2,16,18]	NL	0	0	145.633
269 y[1,2,16,19]	NL	0	0	0.368744
270 y[1,2,16,20]	NL	0	0	122.705
271 y[1,2,16,21]	NL	0	0	79.4943
272 y[1,2,16,22]	NL	0	0	134.784
273 y[1,2,16,23]	NL	0	0	55.3043
274 y[1,2,16,24]	NL	0	0	14.3125
275 y[1,2,16,25]	NL	0	0	57.7361
276 y[1,2,16,26]	NL	0	0	100.486
277 y[1,2,16,27]	NL	0	0	70.1315
278 y[1,2,16,28]	NL	0	0	133.673
279 y[1,2,16,29]	NL	0	0	19.4965
280 y[1,2,16,30]	NL	0	0	34.5602
281 y[1,2,16,31]	NL	0	0	119.402
282 y[1,2,16,32]	NL	0	0	68.2727
283 y[1,2,16,33]	NL	0	0	70.8144
284 y[1,2,16,34]	NL	0	0	53.5294
285 y[1,2,16,35]	NL	0	0	53.8467
286 y[1,2,16,36]	NL	0	0	20.359
287 y[1,2,16,37]	NL	0	0	119.21
288 y[1,2,16,38]	NL	0	0	75.2576
289 y[1,2,16,39]	NL	0	0	118.41
290 y[1,2,16,40]	NL	0	0	61.8992
291 y[1,2,16,41]	NL	0	0	98.8014
292 y[1,2,16,42]	NL	0	0	125.043
293 y[1,2,16,43]	NL	0	0	99.5122
294 y[1,2,16,44]	NL	0	0	20.1327
295 y[1,2,16,45]	NL	0	0	115.059
296 y[1,2,16,46]	NL	0	0	30.8697
297 y[1,2,16,47]	NL	0	0	132.586

298 y[1,2,16,48]	NL	0	0	24.6527
299 y[1,2,16,49]	NL	0	0	11.7633
300 y[1,2,16,50]	NL	0	0	23.2447
301 y[1,3,5,1]	NL	0	0	92.4382
302 y[1,3,5,2]	NL	0	0	32.5666
303 y[1,3,5,3]	NL	0	0	65.8694
304 y[1,3,5,4]	NL	0	0	58.4532
305 y[1,3,5,5]	NL	0	0	39.9342
306 y[1,3,5,6]	NL	0	0	90.1054
307 y[1,3,5,7]	NL	0	0	35.7219
308 y[1,3,5,8]	B	25.4358	0	
309 y[1,3,5,9]	B	188.752	0	
310 y[1,3,5,10]	NL	0	0	60.6557
311 y[1,3,5,11]	NL	0	0	62.9894
312 y[1,3,5,12]	NL	0	0	85.6456
313 y[1,3,5,13]	NL	0	0	1.98992
314 y[1,3,5,14]	NL	0	0	69.9652
315 y[1,3,5,15]	NL	0	0	91.6384
316 y[1,3,5,16]	NL	0	0	88.1027
317 y[1,3,5,17]	NL	0	0	89.6876
318 y[1,3,5,18]	NL	0	0	112.639
319 y[1,3,5,19]	NL	0	0	21.6168
320 y[1,3,5,20]	NL	0	0	11.9293
321 y[1,3,5,21]	NL	0	0	85.1886
322 y[1,3,5,22]	NL	0	0	67.3731
323 y[1,3,5,23]	NL	0	0	65.7894
324 y[1,3,5,24]	NL	0	0	27.8663
325 y[1,3,5,25]	NL	0	0	39.3179
326 y[1,3,5,26]	NL	0	0	58.7816
327 y[1,3,5,27]	NL	0	0	18.3896
328 y[1,3,5,28]	NL	0	0	39.2305
329 y[1,3,5,29]	NL	0	0	84.1016
330 y[1,3,5,30]	NL	0	0	91.6867
331 y[1,3,5,31]	NL	0	0	102.467
332 y[1,3,5,32]	NL	0	0	0.510976
333 y[1,3,5,33]	NL	0	0	37.0653
334 y[1,3,5,34]	NL	0	0	85.6499
335 y[1,3,5,35]	NL	0	0	72.1355
336 y[1,3,5,36]	NL	0	0	91.3123
337 y[1,3,5,37]	NL	0	0	61.1614
338 y[1,3,5,38]	NL	0	0	4.73918
339 y[1,3,5,39]	NL	0	0	96.227
340 y[1,3,5,40]	NL	0	0	49.4338
341 y[1,3,5,41]	NL	0	0	30.5108
342 y[1,3,5,42]	NL	0	0	116.746
343 y[1,3,5,43]	NL	0	0	46.1844
344 y[1,3,5,44]	NL	0	0	9.5492
345 y[1,3,5,45]	NL	0	0	65.7813
346 y[1,3,5,46]	NL	0	0	16.5594
347 y[1,3,5,47]	NL	0	0	69.2198
348 y[1,3,5,48]	NL	0	0	3.41213
349 y[1,3,5,49]	NL	0	0	116.771
350 y[1,3,5,50]	NL	0	0	54.134
351 y[1,3,8,1]	NL	0	0	104.649

352 y[1,3,8,2]	NL	0	0	75.6919
353 y[1,3,8,3]	NL	0	0	119.095
354 y[1,3,8,4]	NL	0	0	85.345
355 y[1,3,8,5]	NL	0	0	71.4262
356 y[1,3,8,6]	NL	0	0	18.5026
357 y[1,3,8,7]	NL	0	0	103.787
358 y[1,3,8,8]	NL	0	0	95.4121
359 y[1,3,8,9]	NL	0	0	111.82
360 y[1,3,8,10]	NL	0	0	15.4036
361 y[1,3,8,11]	B	39.856	0	
362 y[1,3,8,12]	NL	0	0	105.226
363 y[1,3,8,13]	NL	0	0	123.287
364 y[1,3,8,14]	NL	0	0	25.3337
365 y[1,3,8,15]	NL	0	0	64.629
366 y[1,3,8,16]	NL	0	0	11.2178
367 y[1,3,8,17]	NL	0	0	90.9569
368 y[1,3,8,18]	NL	0	0	31.3628
369 y[1,3,8,19]	NL	0	0	44.7284
370 y[1,3,8,20]	NL	0	0	89.1245
371 y[1,3,8,21]	NL	0	0	85.8873
372 y[1,3,8,22]	NL	0	0	102.798
373 y[1,3,8,23]	NL	0	0	18.8248
374 y[1,3,8,24]	NL	0	0	100.202
375 y[1,3,8,25]	NL	0	0	74.858
376 y[1,3,8,26]	NL	0	0	53.8919
377 y[1,3,8,27]	NL	0	0	24.7428
378 y[1,3,8,28]	NL	0	0	29.494
379 y[1,3,8,29]	NL	0	0	7.21657
380 y[1,3,8,30]	B	195.859	0	
381 y[1,3,8,31]	NL	0	0	16.4282
382 y[1,3,8,32]	NL	0	0	76.3008
383 y[1,3,8,33]	NL	0	0	28.5528
384 y[1,3,8,34]	NL	0	0	63.9921
385 y[1,3,8,35]	NL	0	0	75.57
386 y[1,3,8,36]	B	104.89	0	
387 y[1,3,8,37]	B	194.873	0	
388 y[1,3,8,38]	NL	0	0	52.893
389 y[1,3,8,39]	NL	0	0	29.631
390 y[1,3,8,40]	B	92.5104	0	
391 y[1,3,8,41]	NL	0	0	98.3527
392 y[1,3,8,42]	NL	0	0	13.9738
393 y[1,3,8,43]	NL	0	0	99.1141
394 y[1,3,8,44]	NL	0	0	101.294
395 y[1,3,8,45]	NL	0	0	85.8191
396 y[1,3,8,46]	NL	0	0	55.0025
397 y[1,3,8,47]	NL	0	0	80.096
398 y[1,3,8,48]	NL	0	0	76.5421
399 y[1,3,8,49]	NL	0	0	1.98007
400 y[1,3,8,50]	NL	0	0	69.085
401 y[1,3,16,1]	NL	0	0	113.524
402 y[1,3,16,2]	NL	0	0	61.3946
403 y[1,3,16,3]	B	65.2076	0	
404 y[1,3,16,4]	NL	0	0	53.5926
405 y[1,3,16,5]	NL	0	0	83.7978

406 y[1,3,16,6]	NL	0	0	105.045
407 y[1,3,16,7]	NL	0	0	98.1882
408 y[1,3,16,8]	NL	0	0	100.122
409 y[1,3,16,9]	NL	0	0	8.9436
410 y[1,3,16,10]	NL	0	0	90.2671
411 y[1,3,16,11]	NL	0	0	101.703
412 y[1,3,16,12]	NL	0	0	71.2344
413 y[1,3,16,13]	NL	0	0	36.9076
414 y[1,3,16,14]	NL	0	0	99.5354
415 y[1,3,16,15]	NL	0	0	21.8334
416 y[1,3,16,16]	NL	0	0	100.176
417 y[1,3,16,17]	NL	0	0	67.3693
418 y[1,3,16,18]	NL	0	0	122.267
419 y[1,3,16,19]	NL	0	0	86.5906
420 y[1,3,16,20]	NL	0	0	19.8126
421 y[1,3,16,21]	B	23.1876	0	
422 y[1,3,16,22]	NL	0	0	116.93
423 y[1,3,16,23]	NL	0	0	82.0206
424 y[1,3,16,24]	NL	0	0	93.8424
425 y[1,3,16,25]	NL	0	0	44.2551
426 y[1,3,16,26]	NL	0	0	44.7482
427 y[1,3,16,27]	NL	0	0	105.03
428 y[1,3,16,28]	NL	0	0	5.06198
429 y[1,3,16,29]	NL	0	0	56.2208
430 y[1,3,16,30]	NL	0	0	50.8634
431 y[1,3,16,31]	NL	0	0	20.2101
432 y[1,3,16,32]	NL	0	0	95.8388
433 y[1,3,16,33]	NL	0	0	2.28727
434 y[1,3,16,34]	NL	0	0	45.7428
435 y[1,3,16,35]	NL	0	0	46.4559
436 y[1,3,16,36]	NL	0	0	88.9293
437 y[1,3,16,37]	NL	0	0	103.451
438 y[1,3,16,38]	NL	0	0	6.82116
439 y[1,3,16,39]	B	145.624	0	
440 y[1,3,16,40]	NL	0	0	92.4331
441 y[1,3,16,41]	NL	0	0	60.4333
442 y[1,3,16,42]	NL	0	0	34.5924
443 y[1,3,16,43]	NL	0	0	66.6774
444 y[1,3,16,44]	NL	0	0	110.348
445 y[1,3,16,45]	NL	0	0	73.4296
446 y[1,3,16,46]	NL	0	0	47.8596
447 y[1,3,16,47]	NL	0	0	109.459
448 y[1,3,16,48]	NL	0	0	89.6486
449 y[1,3,16,49]	NL	0	0	34.0718
450 y[1,3,16,50]	NL	0	0	8.57639
451 y[1,4,5,1]	NL	0	0	52.6416
452 y[1,4,5,2]	NL	0	0	63.7567
453 y[1,4,5,3]	NL	0	0	58.904
454 y[1,4,5,4]	NL	0	0	16.5404
455 y[1,4,5,5]	NL	0	0	51.1058
456 y[1,4,5,6]	NL	0	0	97.2973
457 y[1,4,5,7]	NL	0	0	89.0325
458 y[1,4,5,8]	NL	0	0	62.4154
459 y[1,4,5,9]	NL	0	0	92.7184

460 y[1,4,5,10]	NL	0	0	14.6134
461 y[1,4,5,11]	NL	0	0	76.7836
462 y[1,4,5,12]	NL	0	0	114.267
463 y[1,4,5,13]	NL	0	0	28.1227
464 y[1,4,5,14]	NL	0	0	102.586
465 y[1,4,5,15]	NL	0	0	107.725
466 y[1,4,5,16]	NL	0	0	20.002
467 y[1,4,5,17]	NL	0	0	50.5166
468 y[1,4,5,18]	NL	0	0	32.5778
469 y[1,4,5,19]	NL	0	0	51.5319
470 y[1,4,5,20]	B	29.7614	0	
471 y[1,4,5,21]	NL	0	0	84.5346
472 y[1,4,5,22]	NL	0	0	0.666678
473 y[1,4,5,23]	NL	0	0	43.92
474 y[1,4,5,24]	NL	0	0	81.4468
475 y[1,4,5,25]	NL	0	0	111.666
476 y[1,4,5,26]	NL	0	0	89.4759
477 y[1,4,5,27]	NL	0	0	5.09942
478 y[1,4,5,28]	NL	0	0	39.7295
479 y[1,4,5,29]	NL	0	0	22.7253
480 y[1,4,5,30]	NL	0	0	27.4599
481 y[1,4,5,31]	NL	0	0	112.222
482 y[1,4,5,32]	NL	0	0	92.0325
483 y[1,4,5,33]	NL	0	0	97.1528
484 y[1,4,5,34]	NL	0	0	98.4379
485 y[1,4,5,35]	NL	0	0	26.6264
486 y[1,4,5,36]	NL	0	0	71.2806
487 y[1,4,5,37]	NL	0	0	37.9396
488 y[1,4,5,38]	NL	0	0	21.361
489 y[1,4,5,39]	NL	0	0	27.4978
490 y[1,4,5,40]	NL	0	0	9.21859
491 y[1,4,5,41]	NL	0	0	87.371
492 y[1,4,5,42]	NL	0	0	84.5281
493 y[1,4,5,43]	NL	0	0	49.7552
494 y[1,4,5,44]	NL	0	0	100.09
495 y[1,4,5,45]	NL	0	0	62.4036
496 y[1,4,5,46]	NL	0	0	43.9616
497 y[1,4,5,47]	NL	0	0	7.21212
498 y[1,4,5,48]	NL	0	0	75.8449
499 y[1,4,5,49]	NL	0	0	62.2497
500 y[1,4,5,50]	NL	0	0	51.3734
501 y[1,4,8,1]	NL	0	0	0.870347
502 y[1,4,8,2]	B	101.767	0	
503 y[1,4,8,3]	NL	0	0	79.2605
504 y[1,4,8,4]	NL	0	0	77.5871
505 y[1,4,8,5]	NL	0	0	48.4924
506 y[1,4,8,6]	NL	0	0	47.9323
507 y[1,4,8,7]	NL	0	0	80.6183
508 y[1,4,8,8]	NL	0	0	41.5612
509 y[1,4,8,9]	NL	0	0	90.4462
510 y[1,4,8,10]	NL	0	0	75.3088
511 y[1,4,8,11]	NL	0	0	29.9361
512 y[1,4,8,12]	NL	0	0	83.4682
513 y[1,4,8,13]	NL	0	0	56.4156

514 y[1,4,8,14]	NL	0	0	69.3889
515 y[1,4,8,15]	NL	0	0	99.1465
516 y[1,4,8,16]	NL	0	0	104.622
517 y[1,4,8,17]	NL	0	0	41.881
518 y[1,4,8,18]	NL	0	0	57.1876
519 y[1,4,8,19]	NL	0	0	28.4685
520 y[1,4,8,20]	NL	0	0	26.2672
521 y[1,4,8,21]	NL	0	0	56.3771
522 y[1,4,8,22]	NL	0	0	42.5826
523 y[1,4,8,23]	NL	0	0	92.7176
524 y[1,4,8,24]	NL	0	0	20.793
525 y[1,4,8,25]	NL	0	0	83.2849
526 y[1,4,8,26]	NL	0	0	40.0209
527 y[1,4,8,27]	NL	0	0	104.133
528 y[1,4,8,28]	B	59.0277	0	
529 y[1,4,8,29]	NL	0	0	41.6788
530 y[1,4,8,30]	NL	0	0	5.67937
531 y[1,4,8,31]	NL	0	0	119.198
532 y[1,4,8,32]	NL	0	0	105.572
533 y[1,4,8,33]	B	79.169	0	
534 y[1,4,8,34]	NL	0	0	43.8727
535 y[1,4,8,35]	NL	0	0	25.2677
536 y[1,4,8,36]	NL	0	0	54.8947
537 y[1,4,8,37]	NL	0	0	70.3585
538 y[1,4,8,38]	NL	0	0	83.696
539 y[1,4,8,39]	NL	0	0	33.5928
540 y[1,4,8,40]	NL	0	0	21.0087
541 y[1,4,8,41]	NL	0	0	8.65035
542 y[1,4,8,42]	NL	0	0	60.1157
543 y[1,4,8,43]	NL	0	0	76.5905
544 y[1,4,8,44]	NL	0	0	18.1655
545 y[1,4,8,45]	NL	0	0	109.31
546 y[1,4,8,46]	NL	0	0	109.22
547 y[1,4,8,47]	NL	0	0	115.376
548 y[1,4,8,48]	NL	0	0	73.9074
549 y[1,4,8,49]	NL	0	0	22.2707
550 y[1,4,8,50]	NL	0	0	43.0604
551 y[1,4,16,1]	NL	0	0	123.409
552 y[1,4,16,2]	NL	0	0	35.3395
553 y[1,4,16,3]	NL	0	0	14.3797
554 y[1,4,16,4]	NL	0	0	23.8162
555 y[1,4,16,5]	NL	0	0	80.0444
556 y[1,4,16,6]	NL	0	0	92.5586
557 y[1,4,16,7]	NL	0	0	28.8926
558 y[1,4,16,8]	NL	0	0	75.2785
559 y[1,4,16,9]	NL	0	0	103.674
560 y[1,4,16,10]	NL	0	0	67.485
561 y[1,4,16,11]	NL	0	0	82.8265
562 y[1,4,16,12]	NL	0	0	86.5401
563 y[1,4,16,13]	NL	0	0	61.9704
564 y[1,4,16,14]	NL	0	0	8.41299
565 y[1,4,16,15]	NL	0	0	40.0088
566 y[1,4,16,16]	NL	0	0	8.23127
567 y[1,4,16,17]	NL	0	0	27.464

568 y[1,4,16,18]	NL	0	0	11.2053
569 y[1,4,16,19]	NL	0	0	59.1155
570 y[1,4,16,20]	NL	0	0	6.22581
571 y[1,4,16,21]	NL	0	0	25.8029
572 y[1,4,16,22]	NL	0	0	31.3999
573 y[1,4,16,23]	NL	0	0	59.9815
574 y[1,4,16,24]	NL	0	0	67.854
575 y[1,4,16,25]	NL	0	0	95.544
576 y[1,4,16,26]	NL	0	0	42.9164
577 y[1,4,16,27]	NL	0	0	101.428
578 y[1,4,16,28]	NL	0	0	115.171
579 y[1,4,16,29]	B	58.0558	0	
580 y[1,4,16,30]	NL	0	0	94.2167
581 y[1,4,16,31]	NL	0	0	88.0834
582 y[1,4,16,32]	NL	0	0	120.426
583 y[1,4,16,33]	NL	0	0	65.9128
584 y[1,4,16,34]	NL	0	0	14.7133
585 y[1,4,16,35]	NL	0	0	107.009
586 y[1,4,16,36]	NL	0	0	76.8837
587 y[1,4,16,37]	NL	0	0	77.7558
588 y[1,4,16,38]	NL	0	0	91.4114
589 y[1,4,16,39]	NL	0	0	45.7251
590 y[1,4,16,40]	NL	0	0	3.77691
591 y[1,4,16,41]	NL	0	0	2.8507
592 y[1,4,16,42]	B	8.10151	0	
593 y[1,4,16,43]	NL	0	0	114.481
594 y[1,4,16,44]	NL	0	0	81.8606
595 y[1,4,16,45]	NL	0	0	11.1205
596 y[1,4,16,46]	NL	0	0	125.103
597 y[1,4,16,47]	NL	0	0	28.7882
598 y[1,4,16,48]	NL	0	0	75.4934
599 y[1,4,16,49]	NL	0	0	42.0486
600 y[1,4,16,50]	NL	0	0	35.1183
601 y[1,5,5,1]	NL	0	0	107.208
602 y[1,5,5,2]	NL	0	0	44.6569
603 y[1,5,5,3]	NL	0	0	60.6086
604 y[1,5,5,4]	NL	0	0	56.4177
605 y[1,5,5,5]	NL	0	0	49.7071
606 y[1,5,5,6]	B	100.738	0	
607 y[1,5,5,7]	NL	0	0	63.6108
608 y[1,5,5,8]	NL	0	0	93.8278
609 y[1,5,5,9]	NL	0	0	11.2454
610 y[1,5,5,10]	NL	0	0	98.9453
611 y[1,5,5,11]	NL	0	0	57.2556
612 y[1,5,5,12]	NL	0	0	100.654
613 y[1,5,5,13]	NL	0	0	38.7556
614 y[1,5,5,14]	NL	0	0	39.5064
615 y[1,5,5,15]	NL	0	0	18.7125
616 y[1,5,5,16]	B	105.877	0	
617 y[1,5,5,17]	NL	0	0	99.8496
618 y[1,5,5,18]	NL	0	0	44.8042
619 y[1,5,5,19]	NL	0	0	37.3826
620 y[1,5,5,20]	NL	0	0	103.064
621 y[1,5,5,21]	NL	0	0	96.701

622 y[1,5,5,22]	NL	0	0	3.29918
623 y[1,5,5,23]	NL	0	0	46.8338
624 y[1,5,5,24]	NL	0	0	43.4776
625 y[1,5,5,25]	NL	0	0	66.4629
626 y[1,5,5,26]	NL	0	0	80.6223
627 y[1,5,5,27]	NL	0	0	93.2527
628 y[1,5,5,28]	NL	0	0	93.6983
629 y[1,5,5,29]	NL	0	0	33.7396
630 y[1,5,5,30]	NL	0	0	103.861
631 y[1,5,5,31]	NL	0	0	57.8921
632 y[1,5,5,32]	NL	0	0	108.606
633 y[1,5,5,33]	NL	0	0	114.885
634 y[1,5,5,34]	NL	0	0	96.0922
635 y[1,5,5,35]	NL	0	0	9.31855
636 y[1,5,5,36]	NL	0	0	74.4453
637 y[1,5,5,37]	NL	0	0	52.9289
638 y[1,5,5,38]	NL	0	0	31.1499
639 y[1,5,5,39]	NL	0	0	87.0676
640 y[1,5,5,40]	NL	0	0	66.9067
641 y[1,5,5,41]	NL	0	0	83.3211
642 y[1,5,5,42]	NL	0	0	92.8407
643 y[1,5,5,43]	NL	0	0	39.535
644 y[1,5,5,44]	NL	0	0	89.8044
645 y[1,5,5,45]	NL	0	0	94.3072
646 y[1,5,5,46]	NL	0	0	73.3198
647 y[1,5,5,47]	NL	0	0	47.1361
648 y[1,5,5,48]	NL	0	0	75.2826
649 y[1,5,5,49]	NL	0	0	84.5136
650 y[1,5,5,50]	NL	0	0	51.0571
651 y[1,5,8,1]	B	98.6099	0	
652 y[1,5,8,2]	NL	0	0	98.3833
653 y[1,5,8,3]	B	8.48284	0	
654 y[1,5,8,4]	NL	0	0	45.9064
655 y[1,5,8,5]	NL	0	0	80.1271
656 y[1,5,8,6]	NL	0	0	21.4567
657 y[1,5,8,7]	NL	0	0	82.9657
658 y[1,5,8,8]	NL	0	0	18.9633
659 y[1,5,8,9]	NL	0	0	83.1137
660 y[1,5,8,10]	NL	0	0	75.8909
661 y[1,5,8,11]	NL	0	0	95.6417
662 y[1,5,8,12]	NL	0	0	62.2017
663 y[1,5,8,13]	NL	0	0	95.8704
664 y[1,5,8,14]	NL	0	0	27.2655
665 y[1,5,8,15]	B	159.836	0	
666 y[1,5,8,16]	NL	0	0	81.818
667 y[1,5,8,17]	NL	0	0	74.229
668 y[1,5,8,18]	NL	0	0	71.5407
669 y[1,5,8,19]	NL	0	0	29.4235
670 y[1,5,8,20]	NL	0	0	19.5651
671 y[1,5,8,21]	NL	0	0	91.2008
672 y[1,5,8,22]	NL	0	0	22.1647
673 y[1,5,8,23]	NL	0	0	30.0168
674 y[1,5,8,24]	NL	0	0	83.9647
675 y[1,5,8,25]	NL	0	0	21.7685

676 y[1,5,8,26]	NL	0	0	93.7799
677 y[1,5,8,27]	NL	0	0	88.4326
678 y[1,5,8,28]	NL	0	0	113.575
679 y[1,5,8,29]	NL	0	0	4.73285
680 y[1,5,8,30]	NL	0	0	83.3832
681 y[1,5,8,31]	NL	0	0	113.695
682 y[1,5,8,32]	NL	0	0	67.4637
683 y[1,5,8,33]	NL	0	0	31.3867
684 y[1,5,8,34]	NL	0	0	76.7348
685 y[1,5,8,35]	NL	0	0	48.821
686 y[1,5,8,36]	NL	0	0	121.023
687 y[1,5,8,37]	NL	0	0	89.8167
688 y[1,5,8,38]	B	150.253	0	
689 y[1,5,8,39]	NL	0	0	113.852
690 y[1,5,8,40]	NL	0	0	45.9267
691 y[1,5,8,41]	NL	0	0	15.0203
692 y[1,5,8,42]	NL	0	0	50.8406
693 y[1,5,8,43]	NL	0	0	78.1177
694 y[1,5,8,44]	NL	0	0	95.6843
695 y[1,5,8,45]	NL	0	0	8.65886
696 y[1,5,8,46]	NL	0	0	18.6652
697 y[1,5,8,47]	NL	0	0	67.7742
698 y[1,5,8,48]	NL	0	0	52.7209
699 y[1,5,8,49]	NL	0	0	81.0164
700 y[1,5,8,50]	NL	0	0	9.47046
701 y[1,5,16,1]	NL	0	0	74.0916
702 y[1,5,16,2]	NL	0	0	44.6877
703 y[1,5,16,3]	NL	0	0	30.3494
704 y[1,5,16,4]	NL	0	0	97.0702
705 y[1,5,16,5]	B	15.1758	0	
706 y[1,5,16,6]	NL	0	0	41.7453
707 y[1,5,16,7]	NL	0	0	57.3782
708 y[1,5,16,8]	NL	0	0	114.891
709 y[1,5,16,9]	NL	0	0	112.837
710 y[1,5,16,10]	NL	0	0	34.2079
711 y[1,5,16,11]	NL	0	0	78.2349
712 y[1,5,16,12]	NL	0	0	41.7903
713 y[1,5,16,13]	B	132.455	0	
714 y[1,5,16,14]	NL	0	0	36.1901
715 y[1,5,16,15]	NL	0	0	59.2858
716 y[1,5,16,16]	NL	0	0	20.782
717 y[1,5,16,17]	NL	0	0	31.8732
718 y[1,5,16,18]	B	147.323	0	
719 y[1,5,16,19]	NL	0	0	89.8709
720 y[1,5,16,20]	NL	0	0	50.7475
721 y[1,5,16,21]	NL	0	0	110.08
722 y[1,5,16,22]	NL	0	0	3.24588
723 y[1,5,16,23]	NL	0	0	98.6142
724 y[1,5,16,24]	NL	0	0	92.6499
725 y[1,5,16,25]	NL	0	0	102.28
726 y[1,5,16,26]	NL	0	0	32.5427
727 y[1,5,16,27]	NL	0	0	106.251
728 y[1,5,16,28]	NL	0	0	57.8017
729 y[1,5,16,29]	NL	0	0	54.0117

730 y[1,5,16,30]	NL	0	0	39.6726
731 y[1,5,16,31]	NL	0	0	57.4488
732 y[1,5,16,32]	NL	0	0	48.3359
733 y[1,5,16,33]	NL	0	0	27.3685
734 y[1,5,16,34]	NL	0	0	22.9934
735 y[1,5,16,35]	NL	0	0	32.4327
736 y[1,5,16,36]	NL	0	0	78.0246
737 y[1,5,16,37]	NL	0	0	120.301
738 y[1,5,16,38]	NL	0	0	79.0807
739 y[1,5,16,39]	NL	0	0	115.013
740 y[1,5,16,40]	NL	0	0	77.2331
741 y[1,5,16,41]	NL	0	0	66.731
742 y[1,5,16,42]	NL	0	0	83.8188
743 y[1,5,16,43]	NL	0	0	39.8521
744 y[1,5,16,44]	NL	0	0	28.4743
745 y[1,5,16,45]	NL	0	0	6.3096
746 y[1,5,16,46]	NL	0	0	12.6515
747 y[1,5,16,47]	B	78.49	0	
748 y[1,5,16,48]	NL	0	0	13.4897
749 y[1,5,16,49]	NL	0	0	51.3711
750 y[1,5,16,50]	NL	0	0	40.423
751 y[1,6,5,1]	NL	0	0	120.72
752 y[1,6,5,2]	NL	0	0	11.7232
753 y[1,6,5,3]	NL	0	0	91.8142
754 y[1,6,5,4]	NL	0	0	120.791
755 y[1,6,5,5]	NL	0	0	87.8193
756 y[1,6,5,6]	NL	0	0	22.1696
757 y[1,6,5,7]	NL	0	0	42.7878
758 y[1,6,5,8]	NL	0	0	99.4436
759 y[1,6,5,9]	NL	0	0	120.162
760 y[1,6,5,10]	NL	0	0	100.107
761 y[1,6,5,11]	NL	0	0	25.2364
762 y[1,6,5,12]	NL	0	0	112.515
763 y[1,6,5,13]	NL	0	0	85.9696
764 y[1,6,5,14]	NL	0	0	104.683
765 y[1,6,5,15]	NL	0	0	32.9313
766 y[1,6,5,16]	NL	0	0	33.3028
767 y[1,6,5,17]	NL	0	0	102.268
768 y[1,6,5,18]	NL	0	0	60.7749
769 y[1,6,5,19]	B	129.854	0	
770 y[1,6,5,20]	NL	0	0	13.8272
771 y[1,6,5,21]	NL	0	0	48.7688
772 y[1,6,5,22]	NL	0	0	49.2267
773 y[1,6,5,23]	NL	0	0	32.862
774 y[1,6,5,24]	NL	0	0	44.1647
775 y[1,6,5,25]	NL	0	0	18.2905
776 y[1,6,5,26]	NL	0	0	68.6771
777 y[1,6,5,27]	NL	0	0	53.6804
778 y[1,6,5,28]	NL	0	0	18.843
779 y[1,6,5,29]	NL	0	0	44.2258
780 y[1,6,5,30]	NL	0	0	44.8462
781 y[1,6,5,31]	NL	0	0	22.8098
782 y[1,6,5,32]	NL	0	0	55.437
783 y[1,6,5,33]	NL	0	0	79.8425

784 y[1,6,5,34]	NL	0	0	115.069
785 y[1,6,5,35]	NL	0	0	41.043
786 y[1,6,5,36]	NL	0	0	43.5124
787 y[1,6,5,37]	NL	0	0	33.2102
788 y[1,6,5,38]	NL	0	0	8.0619
789 y[1,6,5,39]	NL	0	0	108.267
790 y[1,6,5,40]	NL	0	0	102.704
791 y[1,6,5,41]	NL	0	0	7.9422
792 y[1,6,5,42]	NL	0	0	53.1397
793 y[1,6,5,43]	NL	0	0	36.355
794 y[1,6,5,44]	NL	0	0	46.3974
795 y[1,6,5,45]	NL	0	0	88.5733
796 y[1,6,5,46]	NL	0	0	78.5903
797 y[1,6,5,47]	NL	0	0	22.323
798 y[1,6,5,48]	NL	0	0	6.9313
799 y[1,6,5,49]	NL	0	0	48.1438
800 y[1,6,5,50]	NL	0	0	89.6413
801 y[1,6,8,1]	NL	0	0	60.369
802 y[1,6,8,2]	NL	0	0	1.04797
803 y[1,6,8,3]	NL	0	0	120.895
804 y[1,6,8,4]	NL	0	0	26.759
805 y[1,6,8,5]	NL	0	0	62.1613
806 y[1,6,8,6]	NL	0	0	131.782
807 y[1,6,8,7]	NL	0	0	37.9098
808 y[1,6,8,8]	NL	0	0	126.575
809 y[1,6,8,9]	NL	0	0	48.1534
810 y[1,6,8,10]	NL	0	0	22.356
811 y[1,6,8,11]	NL	0	0	61.6981
812 y[1,6,8,12]	NL	0	0	77.8825
813 y[1,6,8,13]	NL	0	0	12.4825
814 y[1,6,8,14]	NL	0	0	91.6695
815 y[1,6,8,15]	NL	0	0	69.2706
816 y[1,6,8,16]	NL	0	0	41.7622
817 y[1,6,8,17]	NL	0	0	21.3321
818 y[1,6,8,18]	NL	0	0	79.6283
819 y[1,6,8,19]	NL	0	0	37.1745
820 y[1,6,8,20]	NL	0	0	105.474
821 y[1,6,8,21]	NL	0	0	86.1897
822 y[1,6,8,22]	NL	0	0	109.251
823 y[1,6,8,23]	NL	0	0	7.80332
824 y[1,6,8,24]	NL	0	0	13.6737
825 y[1,6,8,25]	NL	0	0	98.6398
826 y[1,6,8,26]	NL	0	0	95.3749
827 y[1,6,8,27]	NL	0	0	92.2239
828 y[1,6,8,28]	NL	0	0	31.6762
829 y[1,6,8,29]	NL	0	0	42.2099
830 y[1,6,8,30]	NL	0	0	50.8176
831 y[1,6,8,31]	NL	0	0	51.9994
832 y[1,6,8,32]	NL	0	0	60.9315
833 y[1,6,8,33]	NL	0	0	81.0441
834 y[1,6,8,34]	NL	0	0	42.7845
835 y[1,6,8,35]	NL	0	0	10.8109
836 y[1,6,8,36]	NL	0	0	109.841
837 y[1,6,8,37]	NL	0	0	79.2056

838 y[1,6,8,38]	NL	0	0	119.511
839 y[1,6,8,39]	NL	0	0	72.4883
840 y[1,6,8,40]	NL	0	0	48.71
841 y[1,6,8,41]	NL	0	0	75.7937
842 y[1,6,8,42]	NL	0	0	116.355
843 y[1,6,8,43]	NL	0	0	93.2768
844 y[1,6,8,44]	NL	0	0	120.307
845 y[1,6,8,45]	NL	0	0	54.6961
846 y[1,6,8,46]	NL	0	0	101.044
847 y[1,6,8,47]	NL	0	0	114.37
848 y[1,6,8,48]	NL	0	0	111.939
849 y[1,6,8,49]	NL	0	0	11.5484
850 y[1,6,8,50]	NL	0	0	98.5212
851 y[1,6,16,1]	NL	0	0	48.9045
852 y[1,6,16,2]	NL	0	0	94.6455
853 y[1,6,16,3]	NL	0	0	37.49
854 y[1,6,16,4]	NL	0	0	59.8106
855 y[1,6,16,5]	NL	0	0	114.496
856 y[1,6,16,6]	NL	0	0	20.311
857 y[1,6,16,7]	NL	0	0	110.726
858 y[1,6,16,8]	NL	0	0	67.9151
859 y[1,6,16,9]	NL	0	0	119.762
860 y[1,6,16,10]	NL	0	0	120.642
861 y[1,6,16,11]	NL	0	0	77.5096
862 y[1,6,16,12]	NL	0	0	98.4409
863 y[1,6,16,13]	NL	0	0	121.828
864 y[1,6,16,14]	NL	0	0	125.573
865 y[1,6,16,15]	NL	0	0	6.74546
866 y[1,6,16,16]	NL	0	0	111.784
867 y[1,6,16,17]	NL	0	0	38.928
868 y[1,6,16,18]	NL	0	0	93.8573
869 y[1,6,16,19]	NL	0	0	43.4379
870 y[1,6,16,20]	NL	0	0	64.8337
871 y[1,6,16,21]	NL	0	0	23.6636
872 y[1,6,16,22]	NL	0	0	100.174
873 y[1,6,16,23]	B	53.191	0	
874 y[1,6,16,24]	NL	0	0	17.1862
875 y[1,6,16,25]	NL	0	0	22.7947
876 y[1,6,16,26]	NL	0	0	18.8332
877 y[1,6,16,27]	NL	0	0	109.081
878 y[1,6,16,28]	NL	0	0	119.411
879 y[1,6,16,29]	NL	0	0	114.087
880 y[1,6,16,30]	NL	0	0	79.9372
881 y[1,6,16,31]	NL	0	0	105.388
882 y[1,6,16,32]	NL	0	0	103.526
883 y[1,6,16,33]	NL	0	0	94.9855
884 y[1,6,16,34]	NL	0	0	75.7893
885 y[1,6,16,35]	NL	0	0	62.3089
886 y[1,6,16,36]	NL	0	0	128.808
887 y[1,6,16,37]	NL	0	0	126.917
888 y[1,6,16,38]	NL	0	0	37.5353
889 y[1,6,16,39]	NL	0	0	9.94408
890 y[1,6,16,40]	NL	0	0	36.7577
891 y[1,6,16,41]	NL	0	0	85.4054

892 y[1,6,16,42]	NL	0	0	41.2549
893 y[1,6,16,43]	NL	0	0	82.3702
894 y[1,6,16,44]	NL	0	0	111.645
895 y[1,6,16,45]	NL	0	0	79.8279
896 y[1,6,16,46]	NL	0	0	28.2121
897 y[1,6,16,47]	NL	0	0	75.1758
898 y[1,6,16,48]	NL	0	0	2.5935
899 y[1,6,16,49]	NL	0	0	20.3705
900 y[1,6,16,50]	NL	0	0	29.6966
901 y[1,7,5,1]	NL	0	0	17.3374
902 y[1,7,5,2]	NL	0	0	60.9929
903 y[1,7,5,3]	NL	0	0	107.933
904 y[1,7,5,4]	NL	0	0	86.9482
905 y[1,7,5,5]	NL	0	0	27.1046
906 y[1,7,5,6]	NL	0	0	72.3433
907 y[1,7,5,7]	NL	0	0	44.4758
908 y[1,7,5,8]	NL	0	0	19.0498
909 y[1,7,5,9]	NL	0	0	93.6309
910 y[1,7,5,10]	NL	0	0	114.137
911 y[1,7,5,11]	NL	0	0	61.1385
912 y[1,7,5,12]	NL	0	0	27.135
913 y[1,7,5,13]	NL	0	0	31.4271
914 y[1,7,5,14]	NL	0	0	81.6526
915 y[1,7,5,15]	NL	0	0	43.2939
916 y[1,7,5,16]	NL	0	0	59.6976
917 y[1,7,5,17]	NL	0	0	106.933
918 y[1,7,5,18]	NL	0	0	51.5272
919 y[1,7,5,19]	NL	0	0	19.7964
920 y[1,7,5,20]	NL	0	0	109.577
921 y[1,7,5,21]	NL	0	0	52.4004
922 y[1,7,5,22]	NL	0	0	46.9865
923 y[1,7,5,23]	NL	0	0	19.9168
924 y[1,7,5,24]	NL	0	0	23.2954
925 y[1,7,5,25]	NL	0	0	16.3106
926 y[1,7,5,26]	NL	0	0	24.6774
927 y[1,7,5,27]	NL	0	0	91.795
928 y[1,7,5,28]	NL	0	0	67.4508
929 y[1,7,5,29]	NL	0	0	40.8459
930 y[1,7,5,30]	NL	0	0	8.95275
931 y[1,7,5,31]	NL	0	0	24.8869
932 y[1,7,5,32]	NL	0	0	122.313
933 y[1,7,5,33]	NL	0	0	41.1882
934 y[1,7,5,34]	NL	0	0	1.40525
935 y[1,7,5,35]	NL	0	0	77.2121
936 y[1,7,5,36]	NL	0	0	105.769
937 y[1,7,5,37]	NL	0	0	90.446
938 y[1,7,5,38]	NL	0	0	110.634
939 y[1,7,5,39]	NL	0	0	109.009
940 y[1,7,5,40]	NL	0	0	56.3986
941 y[1,7,5,41]	NL	0	0	35.3966
942 y[1,7,5,42]	NL	0	0	61.4624
943 y[1,7,5,43]	NL	0	0	120.627
944 y[1,7,5,44]	NL	0	0	97.1567
945 y[1,7,5,45]	NL	0	0	31.864

946 y[1,7,5,46]	NL	0	0	93.3103
947 y[1,7,5,47]	NL	0	0	75.6849
948 y[1,7,5,48]	NL	0	0	60.5783
949 y[1,7,5,49]	NL	0	0	127.303
950 y[1,7,5,50]	NL	0	0	13.1633
951 y[1,7,8,1]	NL	0	0	74.8897
952 y[1,7,8,2]	NL	0	0	22.7813
953 y[1,7,8,3]	NL	0	0	116.107
954 y[1,7,8,4]	NL	0	0	116.182
955 y[1,7,8,5]	NL	0	0	75.0073
956 y[1,7,8,6]	NL	0	0	131.196
957 y[1,7,8,7]	NL	0	0	57.113
958 y[1,7,8,8]	NL	0	0	22.932
959 y[1,7,8,9]	NL	0	0	3.47762
960 y[1,7,8,10]	NL	0	0	104.491
961 y[1,7,8,11]	NL	0	0	102.267
962 y[1,7,8,12]	NL	0	0	101.548
963 y[1,7,8,13]	NL	0	0	11.6643
964 y[1,7,8,14]	NL	0	0	14.8882
965 y[1,7,8,15]	NL	0	0	43.6206
966 y[1,7,8,16]	NL	0	0	17.9919
967 y[1,7,8,17]	NL	0	0	26.1023
968 y[1,7,8,18]	NL	0	0	21.2377
969 y[1,7,8,19]	NL	0	0	100.884
970 y[1,7,8,20]	NL	0	0	5.03252
971 y[1,7,8,21]	NL	0	0	121.756
972 y[1,7,8,22]	NL	0	0	56.1284
973 y[1,7,8,23]	NL	0	0	12.7395
974 y[1,7,8,24]	NL	0	0	93.4571
975 y[1,7,8,25]	NL	0	0	81.8113
976 y[1,7,8,26]	NL	0	0	109.593
977 y[1,7,8,27]	NL	0	0	39.6688
978 y[1,7,8,28]	NL	0	0	100.34
979 y[1,7,8,29]	NL	0	0	109.522
980 y[1,7,8,30]	NL	0	0	35.2699
981 y[1,7,8,31]	NL	0	0	45.7738
982 y[1,7,8,32]	NL	0	0	119.513
983 y[1,7,8,33]	NL	0	0	41.8843
984 y[1,7,8,34]	NL	0	0	6.809
985 y[1,7,8,35]	NL	0	0	85.3996
986 y[1,7,8,36]	NL	0	0	117.385
987 y[1,7,8,37]	NL	0	0	118.938
988 y[1,7,8,38]	NL	0	0	69.5167
989 y[1,7,8,39]	NL	0	0	21.1654
990 y[1,7,8,40]	NL	0	0	62.8959
991 y[1,7,8,41]	NL	0	0	83.8827
992 y[1,7,8,42]	NL	0	0	71.4452
993 y[1,7,8,43]	NL	0	0	46.2301
994 y[1,7,8,44]	NL	0	0	63.0701
995 y[1,7,8,45]	NL	0	0	77.8341
996 y[1,7,8,46]	NL	0	0	108.644
997 y[1,7,8,47]	NL	0	0	116.827
998 y[1,7,8,48]	NL	0	0	35.0984
999 y[1,7,8,49]	NL	0	0	81.7789

1000 y[1,7,8,50]	B	15.1428	0	
1001 y[1,7,16,1]	NL	0	0	65.7598
1002 y[1,7,16,2]	B	63.2028	0	
1003 y[1,7,16,3]	NL	0	0	84.9639
1004 y[1,7,16,4]	NL	0	0	8.07476
1005 y[1,7,16,5]	NL	0	0	3.94392
1006 y[1,7,16,6]	NL	0	0	30.543
1007 y[1,7,16,7]	NL	0	0	50.599
1008 y[1,7,16,8]	NL	0	0	92.8889
1009 y[1,7,16,9]	NL	0	0	17.3618
1010 y[1,7,16,10]	NL	0	0	16.574
1011 y[1,7,16,11]	NL	0	0	129.813
1012 y[1,7,16,12]	NL	0	0	102.529
1013 y[1,7,16,13]	NL	0	0	132.865
1014 y[1,7,16,14]	NL	0	0	28.8482
1015 y[1,7,16,15]	NL	0	0	119.05
1016 y[1,7,16,16]	NL	0	0	64.7745
1017 y[1,7,16,17]	NL	0	0	66.5406
1018 y[1,7,16,18]	NL	0	0	107.564
1019 y[1,7,16,19]	NL	0	0	93.9788
1020 y[1,7,16,20]	NL	0	0	19.5218
1021 y[1,7,16,21]	NL	0	0	96.965
1022 y[1,7,16,22]	NL	0	0	74.4715
1023 y[1,7,16,23]	NL	0	0	0.801086
1024 y[1,7,16,24]	B	160.675	0	
1025 y[1,7,16,25]	B	183.231	0	
1026 y[1,7,16,26]	NL	0	0	66.5751
1027 y[1,7,16,27]	NL	0	0	34.08
1028 y[1,7,16,28]	NL	0	0	21.0371
1029 y[1,7,16,29]	NL	0	0	10.6381
1030 y[1,7,16,30]	NL	0	0	92.2978
1031 y[1,7,16,31]	NL	0	0	93.5621
1032 y[1,7,16,32]	NL	0	0	69.5318
1033 y[1,7,16,33]	NL	0	0	36.6759
1034 y[1,7,16,34]	NL	0	0	112.888
1035 y[1,7,16,35]	NL	0	0	120.8
1036 y[1,7,16,36]	NL	0	0	75.2718
1037 y[1,7,16,37]	NL	0	0	53.0164
1038 y[1,7,16,38]	NL	0	0	94.0624
1039 y[1,7,16,39]	NL	0	0	92.1813
1040 y[1,7,16,40]	NL	0	0	40.8881
1041 y[1,7,16,41]	NL	0	0	76.7861
1042 y[1,7,16,42]	NL	0	0	62.0955
1043 y[1,7,16,43]	NL	0	0	41.1982
1044 y[1,7,16,44]	NL	0	0	52.4091
1045 y[1,7,16,45]	NL	0	0	62.0876
1046 y[1,7,16,46]	NL	0	0	49.0718
1047 y[1,7,16,47]	NL	0	0	115.978
1048 y[1,7,16,48]	NL	0	0	76.3421
1049 y[1,7,16,49]	NL	0	0	91.959
1050 y[1,7,16,50]	NL	0	0	67.7263
1051 y[1,8,5,1]	NL	0	0	22.1469
1052 y[1,8,5,2]	NL	0	0	89.4258
1053 y[1,8,5,3]	NL	0	0	115.3

1054 y[1,8,5,4]	NL	0	0	8.0985
1055 y[1,8,5,5]	NL	0	0	30.1479
1056 y[1,8,5,6]	NL	0	0	109.981
1057 y[1,8,5,7]	NL	0	0	12.5097
1058 y[1,8,5,8]	NL	0	0	93.3564
1059 y[1,8,5,9]	NL	0	0	31.7366
1060 y[1,8,5,10]	NL	0	0	37.9937
1061 y[1,8,5,11]	NL	0	0	46.2966
1062 y[1,8,5,12]	NL	0	0	22.7253
1063 y[1,8,5,13]	NL	0	0	37.7129
1064 y[1,8,5,14]	NL	0	0	54.8754
1065 y[1,8,5,15]	NL	0	0	118.635
1066 y[1,8,5,16]	NL	0	0	69.8944
1067 y[1,8,5,17]	NL	0	0	88.9306
1068 y[1,8,5,18]	NL	0	0	39.0208
1069 y[1,8,5,19]	NL	0	0	103.107
1070 y[1,8,5,20]	NL	0	0	68.5234
1071 y[1,8,5,21]	NL	0	0	20.1807
1072 y[1,8,5,22]	NL	0	0	81.0385
1073 y[1,8,5,23]	NL	0	0	76.8332
1074 y[1,8,5,24]	NL	0	0	96.5797
1075 y[1,8,5,25]	NL	0	0	35.6084
1076 y[1,8,5,26]	NL	0	0	95.5832
1077 y[1,8,5,27]	NL	0	0	88.9312
1078 y[1,8,5,28]	NL	0	0	42.7317
1079 y[1,8,5,29]	NL	0	0	61.1851
1080 y[1,8,5,30]	NL	0	0	80.6406
1081 y[1,8,5,31]	NL	0	0	132.958
1082 y[1,8,5,32]	NL	0	0	49.9449
1083 y[1,8,5,33]	NL	0	0	99.0067
1084 y[1,8,5,34]	NL	0	0	27.0611
1085 y[1,8,5,35]	NL	0	0	124.161
1086 y[1,8,5,36]	NL	0	0	64.4256
1087 y[1,8,5,37]	NL	0	0	137.221
1088 y[1,8,5,38]	NL	0	0	48.3271
1089 y[1,8,5,39]	NL	0	0	33.1283
1090 y[1,8,5,40]	NL	0	0	43.6996
1091 y[1,8,5,41]	NL	0	0	15.7691
1092 y[1,8,5,42]	NL	0	0	14.7687
1093 y[1,8,5,43]	NL	0	0	48.6871
1094 y[1,8,5,44]	NL	0	0	45.2922
1095 y[1,8,5,45]	NL	0	0	86.7643
1096 y[1,8,5,46]	NL	0	0	76.9396
1097 y[1,8,5,47]	NL	0	0	26.0567
1098 y[1,8,5,48]	NL	0	0	101.934
1099 y[1,8,5,49]	NL	0	0	47.0749
1100 y[1,8,5,50]	NL	0	0	93.583
1101 y[1,8,8,1]	NL	0	0	112.397
1102 y[1,8,8,2]	NL	0	0	29.2931
1103 y[1,8,8,3]	NL	0	0	85.7226
1104 y[1,8,8,4]	NL	0	0	86.1359
1105 y[1,8,8,5]	B	27.8427	0	
1106 y[1,8,8,6]	NL	0	0	72.9071
1107 y[1,8,8,7]	NL	0	0	53.1601

1108 y[1,8,8,8]	NL	0	0	42.4958
1109 y[1,8,8,9]	NL	0	0	17.9211
1110 y[1,8,8,10]	NL	0	0	63.0064
1111 y[1,8,8,11]	NL	0	0	67.4345
1112 y[1,8,8,12]	NL	0	0	30.1578
1113 y[1,8,8,13]	NL	0	0	99.7505
1114 y[1,8,8,14]	NL	0	0	100.132
1115 y[1,8,8,15]	NL	0	0	85.6348
1116 y[1,8,8,16]	NL	0	0	94.9669
1117 y[1,8,8,17]	NL	0	0	32.3748
1118 y[1,8,8,18]	NL	0	0	137.079
1119 y[1,8,8,19]	NL	0	0	70.6362
1120 y[1,8,8,20]	NL	0	0	115.199
1121 y[1,8,8,21]	NL	0	0	79.2347
1122 y[1,8,8,22]	NL	0	0	41.0071
1123 y[1,8,8,23]	NL	0	0	82.4103
1124 y[1,8,8,24]	NL	0	0	32.7749
1125 y[1,8,8,25]	NL	0	0	40.1177
1126 y[1,8,8,26]	NL	0	0	127.145
1127 y[1,8,8,27]	NL	0	0	30.8522
1128 y[1,8,8,28]	NL	0	0	91.526
1129 y[1,8,8,29]	NL	0	0	88.0045
1130 y[1,8,8,30]	NL	0	0	89.9498
1131 y[1,8,8,31]	NL	0	0	132.603
1132 y[1,8,8,32]	NL	0	0	77.6676
1133 y[1,8,8,33]	NL	0	0	107.742
1134 y[1,8,8,34]	NL	0	0	84.5183
1135 y[1,8,8,35]	NL	0	0	122.672
1136 y[1,8,8,36]	NL	0	0	40.6163
1137 y[1,8,8,37]	NL	0	0	90.3475
1138 y[1,8,8,38]	NL	0	0	28.2265
1139 y[1,8,8,39]	NL	0	0	122.468
1140 y[1,8,8,40]	NL	0	0	47.9443
1141 y[1,8,8,41]	NL	0	0	108.122
1142 y[1,8,8,42]	NL	0	0	31.8221
1143 y[1,8,8,43]	NL	0	0	37.0626
1144 y[1,8,8,44]	NL	0	0	70.9342
1145 y[1,8,8,45]	NL	0	0	84.4079
1146 y[1,8,8,46]	NL	0	0	94.6746
1147 y[1,8,8,47]	NL	0	0	112.729
1148 y[1,8,8,48]	NL	0	0	31.6596
1149 y[1,8,8,49]	NL	0	0	110.543
1150 y[1,8,8,50]	NL	0	0	79.6167
1151 y[1,8,16,1]	NL	0	0	68.4922
1152 y[1,8,16,2]	NL	0	0	126.461
1153 y[1,8,16,3]	NL	0	0	22.856
1154 y[1,8,16,4]	NL	0	0	88.8225
1155 y[1,8,16,5]	NL	0	0	49.6794
1156 y[1,8,16,6]	NL	0	0	125.651
1157 y[1,8,16,7]	NL	0	0	39.5813
1158 y[1,8,16,8]	NL	0	0	24.2722
1159 y[1,8,16,9]	NL	0	0	124.472
1160 y[1,8,16,10]	NL	0	0	87.7144
1161 y[1,8,16,11]	NL	0	0	85.2298

1162 y[1,8,16,12]	NL	0	0	76.0154
1163 y[1,8,16,13]	NL	0	0	39.4484
1164 y[1,8,16,14]	NL	0	0	49.2264
1165 y[1,8,16,15]	NL	0	0	58.7847
1166 y[1,8,16,16]	NL	0	0	89.2135
1167 y[1,8,16,17]	NL	0	0	58.5461
1168 y[1,8,16,18]	NL	0	0	31.8314
1169 y[1,8,16,19]	NL	0	0	91.5785
1170 y[1,8,16,20]	NL	0	0	61.7716
1171 y[1,8,16,21]	NL	0	0	86.3192
1172 y[1,8,16,22]	NL	0	0	104.348
1173 y[1,8,16,23]	NL	0	0	13.6024
1174 y[1,8,16,24]	NL	0	0	125.551
1175 y[1,8,16,25]	NL	0	0	102.731
1176 y[1,8,16,26]	NL	0	0	41.4783
1177 y[1,8,16,27]	NL	0	0	106.333
1178 y[1,8,16,28]	NL	0	0	34.6385
1179 y[1,8,16,29]	B	90.4626	0	
1180 y[1,8,16,30]	NL	0	0	4.96423
1181 y[1,8,16,31]	NL	0	0	59.5571
1182 y[1,8,16,32]	NL	0	0	30.8147
1183 y[1,8,16,33]	NL	0	0	46.2748
1184 y[1,8,16,34]	B	22.1204	0	
1185 y[1,8,16,35]	NL	0	0	57.9898
1186 y[1,8,16,36]	NL	0	0	64.3753
1187 y[1,8,16,37]	NL	0	0	29.9348
1188 y[1,8,16,38]	NL	0	0	51.872
1189 y[1,8,16,39]	NL	0	0	24.3094
1190 y[1,8,16,40]	NL	0	0	49.0841
1191 y[1,8,16,41]	NL	0	0	109.708
1192 y[1,8,16,42]	NL	0	0	12.0506
1193 y[1,8,16,43]	NL	0	0	16.4209
1194 y[1,8,16,44]	NL	0	0	122.727
1195 y[1,8,16,45]	NL	0	0	84.4238
1196 y[1,8,16,46]	NL	0	0	128.441
1197 y[1,8,16,47]	NL	0	0	34.9909
1198 y[1,8,16,48]	NL	0	0	69.1954
1199 y[1,8,16,49]	NL	0	0	84.9715
1200 y[1,8,16,50]	NL	0	0	20.2219
1201 y[1,9,5,1]	NL	0	0	69.3739
1202 y[1,9,5,2]	NL	0	0	78.1229
1203 y[1,9,5,3]	NL	0	0	79.3919
1204 y[1,9,5,4]	NL	0	0	21.5931
1205 y[1,9,5,5]	NL	0	0	93.675
1206 y[1,9,5,6]	NL	0	0	53.684
1207 y[1,9,5,7]	NL	0	0	99.8654
1208 y[1,9,5,8]	NL	0	0	22.2142
1209 y[1,9,5,9]	NL	0	0	118.705
1210 y[1,9,5,10]	NL	0	0	102.779
1211 y[1,9,5,11]	NL	0	0	71.1866
1212 y[1,9,5,12]	NL	0	0	111.636
1213 y[1,9,5,13]	NL	0	0	101.553
1214 y[1,9,5,14]	NL	0	0	67.9523
1215 y[1,9,5,15]	NL	0	0	36.9312

1216 y[1,9,5,16]	NL	0	0	13.7441
1217 y[1,9,5,17]	NL	0	0	107.486
1218 y[1,9,5,18]	NL	0	0	129.357
1219 y[1,9,5,19]	NL	0	0	90.4066
1220 y[1,9,5,20]	NL	0	0	28.0171
1221 y[1,9,5,21]	NL	0	0	73.8354
1222 y[1,9,5,22]	NL	0	0	69.9836
1223 y[1,9,5,23]	NL	0	0	106.587
1224 y[1,9,5,24]	NL	0	0	88.8683
1225 y[1,9,5,25]	NL	0	0	10.4095
1226 y[1,9,5,26]	NL	0	0	12.4954
1227 y[1,9,5,27]	NL	0	0	10.565
1228 y[1,9,5,28]	NL	0	0	82.0365
1229 y[1,9,5,29]	NL	0	0	101.969
1230 y[1,9,5,30]	NL	0	0	53.5105
1231 y[1,9,5,31]	B	189.176	0	
1232 y[1,9,5,32]	NL	0	0	38.1824
1233 y[1,9,5,33]	NL	0	0	62.524
1234 y[1,9,5,34]	NL	0	0	48.1374
1235 y[1,9,5,35]	NL	0	0	22.3645
1236 y[1,9,5,36]	NL	0	0	56.8325
1237 y[1,9,5,37]	NL	0	0	45.2611
1238 y[1,9,5,38]	NL	0	0	114.741
1239 y[1,9,5,39]	NL	0	0	33.6005
1240 y[1,9,5,40]	NL	0	0	14.4581
1241 y[1,9,5,41]	NL	0	0	8.28572
1242 y[1,9,5,42]	NL	0	0	19.1595
1243 y[1,9,5,43]	NL	0	0	36.01
1244 y[1,9,5,44]	NL	0	0	28.1547
1245 y[1,9,5,45]	B	1.8016	0	
1246 y[1,9,5,46]	NL	0	0	73.0792
1247 y[1,9,5,47]	NL	0	0	26.4144
1248 y[1,9,5,48]	NL	0	0	80.8479
1249 y[1,9,5,49]	NL	0	0	116.686
1250 y[1,9,5,50]	NL	0	0	62.0012
1251 y[1,9,8,1]	NL	0	0	75.8664
1252 y[1,9,8,2]	NL	0	0	95.4343
1253 y[1,9,8,3]	NL	0	0	88.9661
1254 y[1,9,8,4]	NL	0	0	42.7209
1255 y[1,9,8,5]	NL	0	0	52.6152
1256 y[1,9,8,6]	NL	0	0	94.2675
1257 y[1,9,8,7]	NL	0	0	22.7771
1258 y[1,9,8,8]	NL	0	0	103.867
1259 y[1,9,8,9]	NL	0	0	119.46
1260 y[1,9,8,10]	NL	0	0	97.7527
1261 y[1,9,8,11]	NL	0	0	16.4254
1262 y[1,9,8,12]	NL	0	0	94.8132
1263 y[1,9,8,13]	NL	0	0	117.217
1264 y[1,9,8,14]	NL	0	0	109.75
1265 y[1,9,8,15]	NL	0	0	67.0392
1266 y[1,9,8,16]	NL	0	0	66.2881
1267 y[1,9,8,17]	NL	0	0	68.1252
1268 y[1,9,8,18]	NL	0	0	127.002
1269 y[1,9,8,19]	NL	0	0	5.65767

1270 y[1,9,8,20]	NL	0	0	42.0753
1271 y[1,9,8,21]	NL	0	0	49.1015
1272 y[1,9,8,22]	NL	0	0	76.2378
1273 y[1,9,8,23]	NL	0	0	88.5416
1274 y[1,9,8,24]	NL	0	0	0.228003
1275 y[1,9,8,25]	NL	0	0	1.53501
1276 y[1,9,8,26]	NL	0	0	97.8804
1277 y[1,9,8,27]	B	30.8193	0	
1278 y[1,9,8,28]	NL	0	0	111.336
1279 y[1,9,8,29]	NL	0	0	68.1328
1280 y[1,9,8,30]	NL	0	0	102.208
1281 y[1,9,8,31]	NL	0	0	59.2132
1282 y[1,9,8,32]	NL	0	0	103.997
1283 y[1,9,8,33]	NL	0	0	56.594
1284 y[1,9,8,34]	NL	0	0	100.763
1285 y[1,9,8,35]	NL	0	0	56.8544
1286 y[1,9,8,36]	NL	0	0	110.635
1287 y[1,9,8,37]	NL	0	0	83.9113
1288 y[1,9,8,38]	NL	0	0	7.99151
1289 y[1,9,8,39]	NL	0	0	123.121
1290 y[1,9,8,40]	NL	0	0	47.4561
1291 y[1,9,8,41]	NL	0	0	51.4885
1292 y[1,9,8,42]	NL	0	0	72.3362
1293 y[1,9,8,43]	B	137.752	0	
1294 y[1,9,8,44]	B	63.5209	0	
1295 y[1,9,8,45]	NL	0	0	39.8933
1296 y[1,9,8,46]	NL	0	0	71.4871
1297 y[1,9,8,47]	NL	0	0	98.8026
1298 y[1,9,8,48]	NL	0	0	20.2232
1299 y[1,9,8,49]	NL	0	0	0.815943
1300 y[1,9,8,50]	NL	0	0	37.2221
1301 y[1,9,16,1]	NL	0	0	85.0776
1302 y[1,9,16,2]	NL	0	0	48.8798
1303 y[1,9,16,3]	NL	0	0	35.3921
1304 y[1,9,16,4]	NL	0	0	33.1547
1305 y[1,9,16,5]	NL	0	0	92.1781
1306 y[1,9,16,6]	NL	0	0	67.9863
1307 y[1,9,16,7]	NL	0	0	74.3991
1308 y[1,9,16,8]	NL	0	0	30.6657
1309 y[1,9,16,9]	NL	0	0	94.0679
1310 y[1,9,16,10]	NL	0	0	33.9139
1311 y[1,9,16,11]	NL	0	0	2.98591
1312 y[1,9,16,12]	B	30.8065	0	
1313 y[1,9,16,13]	NL	0	0	57.0523
1314 y[1,9,16,14]	NL	0	0	74.0583
1315 y[1,9,16,15]	NL	0	0	72.6751
1316 y[1,9,16,16]	NL	0	0	103.702
1317 y[1,9,16,17]	NL	0	0	42.9202
1318 y[1,9,16,18]	NL	0	0	66.509
1319 y[1,9,16,19]	NL	0	0	95.4629
1320 y[1,9,16,20]	NL	0	0	47.684
1321 y[1,9,16,21]	NL	0	0	82.2023
1322 y[1,9,16,22]	NL	0	0	77.1554
1323 y[1,9,16,23]	NL	0	0	35.1297

1324 y[1,9,16,24]	NL	0	0	62.4411
1325 y[1,9,16,25]	NL	0	0	108.963
1326 y[1,9,16,26]	NL	0	0	18.3937
1327 y[1,9,16,27]	NL	0	0	10.0666
1328 y[1,9,16,28]	NL	0	0	16.3166
1329 y[1,9,16,29]	NL	0	0	47.6485
1330 y[1,9,16,30]	NL	0	0	80.2874
1331 y[1,9,16,31]	NL	0	0	0.0983924
1332 y[1,9,16,32]	NL	0	0	30.975
1333 y[1,9,16,33]	NL	0	0	26.7579
1334 y[1,9,16,34]	NL	0	0	61.9756
1335 y[1,9,16,35]	NL	0	0	58.1053
1336 y[1,9,16,36]	NL	0	0	77.2194
1337 y[1,9,16,37]	NL	0	0	23.1475
1338 y[1,9,16,38]	NL	0	0	48.3149
1339 y[1,9,16,39]	NL	0	0	53.0542
1340 y[1,9,16,40]	NL	0	0	104.947
1341 y[1,9,16,41]	NL	0	0	87.147
1342 y[1,9,16,42]	NL	0	0	90.9711
1343 y[1,9,16,43]	NL	0	0	67.2642
1344 y[1,9,16,44]	NL	0	0	53.7006
1345 y[1,9,16,45]	NL	0	0	112.312
1346 y[1,9,16,46]	NL	0	0	28.6691
1347 y[1,9,16,47]	NL	0	0	78.9411
1348 y[1,9,16,48]	NL	0	0	24.7542
1349 y[1,9,16,49]	NL	0	0	27.384
1350 y[1,9,16,50]	NL	0	0	87.0345
1351 y[1,10,5,1]	NL	0	0	21.072
1352 y[1,10,5,2]	NL	0	0	61.0129
1353 y[1,10,5,3]	NL	0	0	18.511
1354 y[1,10,5,4]	NL	0	0	67.6105
1355 y[1,10,5,5]	NL	0	0	17.2398
1356 y[1,10,5,6]	NL	0	0	124.88
1357 y[1,10,5,7]	NL	0	0	12.3465
1358 y[1,10,5,8]	NL	0	0	58.302
1359 y[1,10,5,9]	NL	0	0	91.59
1360 y[1,10,5,10]	NL	0	0	92.4208
1361 y[1,10,5,11]	NL	0	0	29.1122
1362 y[1,10,5,12]	NL	0	0	109.047
1363 y[1,10,5,13]	NL	0	0	22.5007
1364 y[1,10,5,14]	NL	0	0	92.3118
1365 y[1,10,5,15]	NL	0	0	3.98536
1366 y[1,10,5,16]	NL	0	0	18.9252
1367 y[1,10,5,17]	NL	0	0	24.6538
1368 y[1,10,5,18]	NL	0	0	106.464
1369 y[1,10,5,19]	B	67.2843	0	
1370 y[1,10,5,20]	NL	0	0	53.2368
1371 y[1,10,5,21]	NL	0	0	12.9183
1372 y[1,10,5,22]	NL	0	0	110.25
1373 y[1,10,5,23]	NL	0	0	64.9818
1374 y[1,10,5,24]	NL	0	0	2.84006
1375 y[1,10,5,25]	NL	0	0	52.7677
1376 y[1,10,5,26]	NL	0	0	59.3425
1377 y[1,10,5,27]	B	162.862	0	

1378 y[1,10,5,28]	NL	0	0	0.543848
1379 y[1,10,5,29]	NL	0	0	87.7515
1380 y[1,10,5,30]	NL	0	0	23.9912
1381 y[1,10,5,31]	NL	0	0	81.8155
1382 y[1,10,5,32]	NL	0	0	89.7044
1383 y[1,10,5,33]	NL	0	0	43.7936
1384 y[1,10,5,34]	NL	0	0	98.5627
1385 y[1,10,5,35]	NL	0	0	40.1275
1386 y[1,10,5,36]	NL	0	0	23.1686
1387 y[1,10,5,37]	NL	0	0	114.12
1388 y[1,10,5,38]	NL	0	0	31.3049
1389 y[1,10,5,39]	NL	0	0	66.2293
1390 y[1,10,5,40]	NL	0	0	96.3528
1391 y[1,10,5,41]	B	140.515	0	
1392 y[1,10,5,42]	NL	0	0	108.563
1393 y[1,10,5,43]	NL	0	0	100.9
1394 y[1,10,5,44]	NL	0	0	110.923
1395 y[1,10,5,45]	NL	0	0	71.1248
1396 y[1,10,5,46]	NL	0	0	53.1577
1397 y[1,10,5,47]	NL	0	0	32.5616
1398 y[1,10,5,48]	B	108.118	0	
1399 y[1,10,5,49]	NL	0	0	73.1911
1400 y[1,10,5,50]	NL	0	0	77.3466
1401 y[1,10,8,1]	NL	0	0	46.102
1402 y[1,10,8,2]	NL	0	0	98.8835
1403 y[1,10,8,3]	NL	0	0	112.892
1404 y[1,10,8,4]	NL	0	0	19.8659
1405 y[1,10,8,5]	NL	0	0	1.15669
1406 y[1,10,8,6]	NL	0	0	10.4556
1407 y[1,10,8,7]	NL	0	0	15.5175
1408 y[1,10,8,8]	NL	0	0	53.4735
1409 y[1,10,8,9]	NL	0	0	13.8975
1410 y[1,10,8,10]	NL	0	0	86.4085
1411 y[1,10,8,11]	NL	0	0	61.2937
1412 y[1,10,8,12]	NL	0	0	79.4427
1413 y[1,10,8,13]	NL	0	0	82.3678
1414 y[1,10,8,14]	NL	0	0	108.923
1415 y[1,10,8,15]	NL	0	0	37.3874
1416 y[1,10,8,16]	NL	0	0	28.3841
1417 y[1,10,8,17]	NL	0	0	89.1574
1418 y[1,10,8,18]	NL	0	0	118.687
1419 y[1,10,8,19]	NL	0	0	18.0018
1420 y[1,10,8,20]	NL	0	0	69.5896
1421 y[1,10,8,21]	NL	0	0	49.6602
1422 y[1,10,8,22]	NL	0	0	101.999
1423 y[1,10,8,23]	NL	0	0	80.6869
1424 y[1,10,8,24]	NL	0	0	92.245
1425 y[1,10,8,25]	NL	0	0	46.8668
1426 y[1,10,8,26]	NL	0	0	24.4788
1427 y[1,10,8,27]	NL	0	0	59.1658
1428 y[1,10,8,28]	NL	0	0	68.6978
1429 y[1,10,8,29]	NL	0	0	68.2108
1430 y[1,10,8,30]	NL	0	0	12.2243
1431 y[1,10,8,31]	NL	0	0	125.158

1432 y[1,10,8,32]	NL	0	0	44.9288
1433 y[1,10,8,33]	NL	0	0	94.2429
1434 y[1,10,8,34]	NL	0	0	58.2225
1435 y[1,10,8,35]	B	105.754	0	
1436 y[1,10,8,36]	NL	0	0	111.37
1437 y[1,10,8,37]	NL	0	0	117.932
1438 y[1,10,8,38]	NL	0	0	73.9903
1439 y[1,10,8,39]	NL	0	0	72.679
1440 y[1,10,8,40]	NL	0	0	32.8599
1441 y[1,10,8,41]	NL	0	0	119.785
1442 y[1,10,8,42]	NL	0	0	70.6661
1443 y[1,10,8,43]	NL	0	0	47.0961
1444 y[1,10,8,44]	NL	0	0	9.19609
1445 y[1,10,8,45]	NL	0	0	87.594
1446 y[1,10,8,46]	NL	0	0	104.26
1447 y[1,10,8,47]	NL	0	0	124.601
1448 y[1,10,8,48]	NL	0	0	97.2179
1449 y[1,10,8,49]	NL	0	0	83.402
1450 y[1,10,8,50]	NL	0	0	0.494957
1451 y[1,10,16,1]	NL	0	0	71.3197
1452 y[1,10,16,2]	NL	0	0	19.9347
1453 y[1,10,16,3]	NL	0	0	109.626
1454 y[1,10,16,4]	B	11.6277	0	
1455 y[1,10,16,5]	NL	0	0	24.7436
1456 y[1,10,16,6]	NL	0	0	47.7272
1457 y[1,10,16,7]	NL	0	0	79.606
1458 y[1,10,16,8]	NL	0	0	83.4981
1459 y[1,10,16,9]	NL	0	0	6.9534
1460 y[1,10,16,10]	NL	0	0	107.851
1461 y[1,10,16,11]	NL	0	0	120.135
1462 y[1,10,16,12]	NL	0	0	33.4118
1463 y[1,10,16,13]	NL	0	0	78.9809
1464 y[1,10,16,14]	NL	0	0	43.5224
1465 y[1,10,16,15]	NL	0	0	85.2748
1466 y[1,10,16,16]	NL	0	0	79.668
1467 y[1,10,16,17]	NL	0	0	100.467
1468 y[1,10,16,18]	NL	0	0	89.2758
1469 y[1,10,16,19]	NL	0	0	92.6856
1470 y[1,10,16,20]	NL	0	0	11.1877
1471 y[1,10,16,21]	NL	0	0	102.918
1472 y[1,10,16,22]	NL	0	0	58.0712
1473 y[1,10,16,23]	NL	0	0	114.937
1474 y[1,10,16,24]	NL	0	0	32.0978
1475 y[1,10,16,25]	NL	0	0	99.5007
1476 y[1,10,16,26]	B	168.039	0	
1477 y[1,10,16,27]	NL	0	0	107.334
1478 y[1,10,16,28]	B	105.031	0	
1479 y[1,10,16,29]	NL	0	0	46.8873
1480 y[1,10,16,30]	NL	0	0	8.44598
1481 y[1,10,16,31]	NL	0	0	89.7398
1482 y[1,10,16,32]	NL	0	0	41.4414
1483 y[1,10,16,33]	NL	0	0	58.929
1484 y[1,10,16,34]	NL	0	0	51.4935
1485 y[1,10,16,35]	NL	0	0	108.2

1486 y[1,10,16,36]	NL	0	0	22.733
1487 y[1,10,16,37]	NL	0	0	50.9584
1488 y[1,10,16,38]	NL	0	0	23.5697
1489 y[1,10,16,39]	NL	0	0	102.969
1490 y[1,10,16,40]	NL	0	0	54.9042
1491 y[1,10,16,41]	NL	0	0	89.9067
1492 y[1,10,16,42]	NL	0	0	51.5355
1493 y[1,10,16,43]	NL	0	0	55.7818
1494 y[1,10,16,44]	NL	0	0	4.25713
1495 y[1,10,16,45]	NL	0	0	39.0564
1496 y[1,10,16,46]	NL	0	0	64.9362
1497 y[1,10,16,47]	NL	0	0	47.3819
1498 y[1,10,16,48]	NL	0	0	39.4473
1499 y[1,10,16,49]	B	170.883	0	
1500 y[1,10,16,50]	NL	0	0	20.8427

Karush-Kuhn-Tucker o optimality conditions:

	=	5.51e-012 on	row 1
KKT.PE: max.abs.err.	=	4.17e-015 on	row 9
max.rel.err.			
High quality			
	=	0.00e+000 on	row 0
KKT.PB: max.abs.err.	=	0.00e+000 on	row 0
max.rel.err.			
High quality			
	=	2.84e-014 on	column 1294
KKT.DE: max.abs.err.	=	2.11e-015 on	column 268
max.rel.err.			
High quality			
	=	0.00e+000 on	row 0
KKT.DB: max.abs.err.	=	0.00e+000 on	row 0
max.rel.err.			
High quality			

End of output