REGMPE

Revista de Empreendedorismo e Gestão de Micro e Pequenas Empresas

USO DE LA INVESTIGACIÓN OPERATIVA EN LA GESTIÓN DE UNA MICROEMPRESA

Yudi Matsuguma Yoshida[1]
Stella Jacyszyn Bachega[2]
Dalton Matsuo Tavares[3]

resumen
La búsqueda del crecimiento de una empresa con los costos más bajos es hoy uno de los mayores desafíos para los gerentes. Una de varias opciones es el uso de técnicas de optimización presentes en la Investigación de operaciones. Existen varios optimizadores basados en lenguajes de programación destinados a resolver problemas de optimización. Sin embargo, cada idioma tiene sus particularidades y, en consecuencia, sus limitaciones. El lenguaje Julia fue creado con el objetivo de combinar las ventajas de varios idiomas con una sintaxis relativamente simple, lo que lo hace más fácil de usar. El propósito de este artículo es aplicar técnicas de investigación operativa para ayudar a la gestión de una microempresa. Para esto, se utilizó el enfoque cuantitativo y el procedimiento de investigación experimental. Con las traducciones adecuadas de los problemas al idioma Julia,*mezcla*producción de bocadillos para una empresa de alimentos y la ruta más corta para su entrega. Al abordar las técnicas de programación lineal y el problema del vendedor ambulante, este artículo puede explorarse como una herramienta instructiva en futuras investigaciones científicas y / o para ayudar en el proceso de toma de decisiones de una organización.
Palabras clave: Investigación Operativa. Lenguaje julia. Programación lineal. Problema de vendedor ambulante.

1. Introducción

La Investigación Operativa (PO) es una investigación de operaciones, que se utiliza en problemas que requieren una conducta, coordinación y toma de decisiones en relación con las operaciones de una organización (HILLIER; LIEBERMAN, 2013). Por lo tanto, la PO se puede abordar en varios segmentos, como producción, logística, gestión financiera y planificación y control de la producción, lo que denota su interdisciplinariedad y multidisciplinariedad (ARENALES et al., 2007).

Varios de estos enfoques, que requerían un gran volumen de cuentas y que a menudo no eran factibles de resolver manualmente, podrían resolverse con el avance de la tecnología

[1]Universidad Federal de Goiás - Regional Catalão - Unidad Especial de Ingeniería Académica (FENG). Correo electrónico: yudimatsuguma@gmail.com. ORCID: https://orcid.org/0000-0002-5701-7837
[2]Universidad Federal de Goiás - Regional Catalão - Unidad Especial de Ingeniería Académica (FENG). Correo electrónico: stella@ufg.br. ORCID: https://orcid.org/0000-0002-7533-5361
[3]Universidad Federal de Goiás - Regional Catalana - Unidad Académica Especial de Biotecnología (IBIOTEC). Correo electrónico: dalton_tavares@ufg.br. ORCID: https://orcid.org/0000-0001-8531-5578

Yoshida, YM, Bachega, SJ, Tavares, DM; Uso de la investigación operativa en la gestión de una microempresa.Revista de Emprendimiento y Gestión de Micro y Pequeñas Empresas V.5, Nº1, p.58-78, Jan / Abr. 2019. Artículo recibido el 15/02/2020. Última versión recibida el 10/03/2020. Aprobado el 05/04/2019.

informática y la creación de varios programas de optimización especializados (MOREIRA, 2010). Dichos programas tienen ventajas y desventajas en relación con la productividad y el rendimiento. Como ejemplo, los lenguajes C y C ++, que pueden usarse en estudios de PO, permiten la implementación de algoritmos para una ejecución rápida, pero tienen una sintaxis compleja que requiere habilidades informáticas específicas. El lenguaje Phyton, por otro lado, contiene varias funciones y herramientas de alto nivel, que ofrecen la ejecución de algoritmos con pocas líneas de código. Por otro lado, su rendimiento es inferior a C / C ++ (JULIALANG.ORG, 2018).

Para reemplazar varios idiomas, para combinar sus ventajas con una sintaxis relativamente simple, el lenguaje Julia apareció en 2009 en el Massachusetts Institute of Technology (MIT). Esto se basó en varios lenguajes en un intento de mantenerlo genérico, similar a C, Fortran y Phyton, con el objetivo de ser utilizado por programadores, matemáticos, estadísticos e investigadores (BEZANSON et al., 2012).

Aunque el lenguaje es relativamente 'joven', ya hay varios trabajos científicos publicados en diferentes áreas y disciplinas, como genética, álgebra lineal, estadística, econometría, neurociencia computacional, análisis de riesgos y también PO. En base a este contexto, la presente investigación se justifica por la importancia del tema, como se puede ver en publicaciones recientes que usaron el lenguaje Julia, como en Anderson et al. (2017), Baldassi (2017), Bezanson et al. (2017), Castelluccia (2017), Mogensen y Riseth (2018), Rackauckas et al. (2018) Aún así, autores como Almeida et al. (2017), Santana et al. (2018), Sousa et al. (2019), Souza et al. (2019) y Teixeira et al. (2017) alentaron la aplicación de técnicas de PO en micro y pequeñas empresas para mejorar la gestión empresarial. Así,

Por lo tanto, el objetivo general del artículo es aplicar técnicas de investigación operativa para ayudar a la gestión de una microempresa que opera en la industria alimentaria. Para eso, se utilizó el lenguaje Julia. A través del análisis de la realidad empresarial, se seleccionó un problema de Programación Lineal (PL) para maximizar los ingresos de la compañía al determinar la mejor combinación de refrigerios, siempre que se respeten sus restricciones. Además, la entrega de los bocadillos la realiza la propia empresa, y también existe la oportunidad de aplicar el problema del vendedor ambulante.

La estructura del artículo es la siguiente: la sección dos presenta el marco teórico y la sección tres discute el enfoque de la metodología utilizada para llevar a cabo el trabajo. La sección cuatro trata los resultados y las discusiones. Finalmente, en la última sección están las consideraciones finales.

2. Marco teórico

2.1 Julia

Creado en 2009 en el MIT, el lenguaje de programación Julia se considera de alto nivel, dinámico y tiene un alto rendimiento para gráficos y números de computadora. Tiene un compilador sofisticado, ejecución paralela distribuida, precisión numérica y una extensa biblioteca de funciones matemáticas. Además, tiene una sintaxis genérica basada en varios lenguajes (C, Fortran, Matlab, Phyton, entre otros), por lo que el software puede ser utilizado no solo por programadores, sino también por matemáticos, estadísticos e investigadores (BEZANSON et al., 2012).

Según el sitio web oficial, la programación tiene las siguientes características principales: (i) métodos múltiples, donde tiene la capacidad de definir el comportamiento de la función a través de varias combinaciones de tipos de argumentos; (ii) paralelismo y capacidad informática distribuida, lo que permite dividir el problema en partes y distribuirlo entre varias computadoras; (iii) tiene un administrador de paquetes práctico y fácil de usar; (iv) buen desempeño, acercándolo al lenguaje C en varios aspectos; y (v) poderosa capacidad para gestionar otros procesos (JULIALANG.ORG, 2018).

2.2 Investigación operativa: definición y algunas técnicas

Según Hillier y Lieberman (2013), PO se trata de investigar operaciones, de usarse en problemas que requieren gestión, coordinación y toma de decisiones con respecto a las operaciones en una organización. Por lo tanto, varias áreas diferentes (como logística, gestión financiera, planificación y control de producción, salud, entre otras) pueden resolver sus problemas utilizando técnicas de PO.

Una de las técnicas más generales utilizadas en PO es la simulación de sistemas, que consiste en tratar de representar con precisión un sistema real en un sistema informático, permitir la prueba de hipótesis de nuevos escenarios y, en consecuencia, ayudar en el proceso de toma de decisiones de la organización. (PEREIRA et al., 2015). La introducción de una nueva bomba de combustible en una estación de servicio es un ejemplo de un estudio de simulación.

La programación lineal (PL) es responsable de resolver los problemas de asignación de recursos limitados a actividades que compiten entre sí por dichos recursos. Sin embargo, PL tiene otras aplicaciones, siempre que la formulación del problema se ajuste a su formato matemático genérico (HILLIER; LIEBERMAN, 2013).

Los problemas que tienen una relación entre sus componentes, para que puedan formularse como una red, se resuelven mediante la optimización de la red. Su uso es muy amplio ya que las redes existen en varias áreas, como las redes de transporte, electricidad, comunicaciones, producción y gestión de proyectos (ARENALES et al., 2007).

Dentro de la optimización de la red, existen varios algoritmos que ayudan a resolver problemas y tomar decisiones, tales como: algoritmo de árbol de expansión mínima, cuyo objetivo es conectar todos los nodos en la red, directa o indirectamente, de modo que la longitud total de los arcos son lo más pequeños posible; algoritmo de flujo máximo, cuyo objetivo es determinar la ruta de paso desde el nodo de origen al destino (sumidero) con la mayor cantidad total de flujo posible; algoritmo de ruta crítica, ampliamente utilizado en la gestión de proyectos para determinar actividades críticas del proyecto; problema de ruta más corta, que trata de determinar la ruta que tiene la longitud más corta entre el nodo de origen y el nodo de destino; y el problema del vendedor ambulante (TAHA, 2008).

2.3 programación lineal

La programación lineal (PL) surgió a fines de los años cuarenta y, con la creación de la computadora en la década siguiente, su desarrollo se aceleró y se generalizó (PRADO, 2007). Para Moreira (2010), PL es uno de los modelos matemáticos más populares, configurado para resolver problemas que tienen variables medibles y correlacionadas, expresadas por ecuaciones matemáticas y / o desigualdades. La popularización del modelo se debe al hecho de que existen numerosos problemas en las áreas científicas y sociales que pueden formularse de esta manera.

Según Ehrlich (1985), PL es una herramienta de planificación que ayuda a decidir qué actividades (variables de decisión) realizar, ya que compiten entre sí por el uso de recursos escasos (restricciones), ya que dichos recursos son insuficientes para que todas las actividades se llevan a cabo en el nivel máximo deseado.

Hoy, se ha convertido en una herramienta estándar que ha ahorrado millones de dólares para muchas empresas. Vale la pena mencionar que PL no está restringido solo al sector industrial, sino también a varias áreas de la sociedad (HILLIER; LIEBERMAN, 2013).

Según Passos (2008), PL es una técnica de optimización aplicada a un sistema de ecuaciones, o desigualdades, matemática lineal de un problema. Su objetivo es maximizar o minimizar una función lineal mediante la determinación de los valores de las variables de decisión, teniendo en cuenta que las restricciones deben cumplirse.

Según Passos (2008), el patrón modelo de un problema de LP se da de acuerdo con las ecuaciones 1 a 7.

$$\text{Máx. (O Mín.) } Z = c1x1 + c2x2 + c3x3 + ... + cnxn \quad (1)$$

$$\text{Sujeto a: } a11x1 + a12x2 + a13x3 + ... + a1nxn = b1 \quad (2)$$

$$a21x1 + a22x2 + a23x3 + ... + a2nxn = b2 \quad (3)$$

$$a31x1 + a32x2 + a33x3 + ... + a3nxn = b3 \quad (4)$$

$$... \ ... \ ... \qquad ... \ ... \quad (5)$$

$$amnx1 + am2x2 + am3x3 + ... + amnxn = bm \quad (6)$$

$$x1 \geq 0, x2 \geq 0, x3 \geq 0, ..., xn \geq 0 \quad (7)$$

Donde X: {x1, x2, x3, ..., xn}: son las variables de decisión; A: {a1, a2, a3, ..., a_{metro}}: son los coeficientes de las variables; y B: {b1, b2, b3, ..., bm}: son los términos independientes que representan los recursos disponibles.Las variables de decisión son esenciales para resolver el problema. La función objetivo, dada por la ecuación (1), quiere optimizar los recursos involucrados maximizándolos o minimizándolos. Las restricciones, dadas de acuerdo con las ecuaciones 2 a 7, son las limitaciones del problema, como, por ejemplo, la capacidad productiva del sistema, la cantidad de mano de obra, materia prima o stock (LINS; CALÔBA, 2010, p. 7).

Para que se satisfaga la linealidad, la estructura del problema debe cumplir tres propiedades básicas: proporcionalidad, aditividad y certeza. La primera propiedad propone que la contribución de cada variable de decisión es directamente proporcional al valor de la variable, tanto en la función objetivo como en las restricciones. La segunda propiedad establece que la contribución total de las variables de la función objetivo y de las restricciones es la suma de las contribuciones individuales de cada variable. Finalmente, la tercera propiedad se refiere a todos los coeficientes de la función objetivo y las restricciones del modelo PL, que son constantes conocidas (TAHA, 2008).

2.4 Problema del vendedor ambulante

El problema del vendedor ambulante (PCV) es un problema de optimización combinatoria, por lo tanto, se usa ampliamente en experimentos de varios métodos de optimización, ya que es fácil de describir, comprender y tiene una amplia aplicabilidad (KARP, 1975). Según Hillier y Lieberman (2013), el PCV puede describirse como un problema para definir la ruta más corta (en términos de distancia o costos) de un vendedor que recorre varias ciudades (nodos) a través de una red y, al final, regresa a la ciudad de origen. Vale la pena mencionar que cada ciudad solo se puede visitar una vez.

Tal problema puede resolverse mediante programación lineal, más específicamente, mediante el método Simplex, en el que se analizarán todas las rutas posibles para el problema.

Sin embargo, a partir de un cierto número de nodos en el problema, es imposible resolverlo con este método. Teniendo en cuenta que "n" es el número de ciudades y que el PCV es simétrico, es decir, cuando la ruta de la ciudad A a la ciudad B es igual a la ruta de la ciudad B a la ciudad A, el número de rutas posibles viene dado por (n-1)! / 2, donde hay (n-1) posibilidades para la primera ciudad, (n-2) para la segunda y así sucesivamente. La división por 2 indica que cada ruta tiene una ruta con el mismo módulo, pero en la dirección opuesta. Si el PCV es asimétrico, el número de rutas posibles viene dado por (n-1)! Por lo tanto, un problema con solo 10 ciudades tiene 181.

Como G = (N, E) es un gráfico, donde N = {1, 2, ..., n} es el conjunto de nodos, E = {1, 2, ..., m} es el conjunto de aristas de G , el objetivo es determinar el ciclo hamiltoniano más pequeño del gráfico G (ARENALES et al., 2007).

Es posible notar que, dependiendo del valor de "n", la solución del problema por los métodos exactos se vuelve impracticable. Ante tales dificultades, se han aplicado varios enfoques heurísticos y se han proporcionado soluciones de buena calidad, sin embargo, no garantizan la optimización de un resultado, ya que está determinado por métodos intuitivos (SILVEIRA, 2000).

Un método heurístico bien conocido es el "vecino más cercano", que selecciona la ruta más corta a los nodos vecinos en los que se encuentra el empleado. Por lo tanto, el nodo más cercano al nodo fuente se define como el segundo nodo y, a partir de ahí, selecciona el nodo más cercano entre los nodos no seleccionados, que será el tercer nodo. Repita el proceso hasta que todos los nodos estén conectados (HILLIER; LIEBERMAN, 2013).

3. Metodología

La explicación científica utilizada en esta investigación fue hipotética-deductiva, según las consideraciones de Carvalho (2000). Por lo tanto, la proposición de que es posible aplicar técnicas de PO en una microempresa para mejorar la gestión empresarial se corroboró al final de la investigación.

Aún así, se utilizó el enfoque cuantitativo según Creswell (2010), debido a la forma en que se manipularon los datos en la investigación. Como resultado, se generaron modelos matemáticos con ecuaciones estructuradas y desigualdades, incorporando pautas causales e identificando la correlación de múltiples variables.

Se adoptó el procedimiento de investigación experimental. Esto es más adecuado para enfoques cuantitativos y puede relacionarse con modelos matemáticos y simulaciones

computacionales (BRYMAN, 1989). Gil (2009) agrega que este procedimiento cubre tanto la determinación del objeto de estudio como la selección de variables que lo influyen.

Los pasos para llevar a cabo el estudio PO fueron apoyados por Hillier y Lieberman (2013). Los pasos realizados en esta investigación fueron:

Paso 1) Definición del problema de interés y recopilación de datos: en este paso, se identificaron problemas clásicos de investigación operativa para seleccionar el problema a modelar y se recopilaron los datos para el modelado matemático. Se seleccionaron la programación lineal y los problemas del vendedor ambulante. Los datos para respaldar los ejemplos presentes en este trabajo se obtuvieron de un microempresario en la industria alimentaria.

Paso 2) Formulación del modelo matemático que representa un problema dado: aquí el modelo matemático se formalizó en funciones, ecuaciones y desigualdades, de acuerdo con la técnica de PO apropiada para la representación y resolución posterior.

Paso 3) Desarrollo del procedimiento computacional para encontrar una solución al problema modelado: el modelo matemático se tradujo al lenguaje Julia y los optimizadores empleados determinaron una posible solución óptima.

Paso 4) Prueba y mejora del modelo: existía una preocupación por validar los modelos, haciéndolos representar fielmente la realidad. Por lo tanto, la estructura del modelo se verificó nuevamente y fue posible encontrar la respuesta correcta a sus respectivos problemas.

Paso 5) Documentación: la documentación se realizó para favorecer la comprensión de la investigación realizada y para garantizar su continuidad futura.

4. Resultados obtenidos

En esta sección, Existe la presentación de los resultados obtenidos primero en el problema de programación lineal y, más tarde, en el problema del vendedor ambulante.

4.1 Programación lineal

La programación lineal (PL) se define como un problema de optimización donde la función objetivo y las restricciones están representadas por ecuaciones lineales o desigualdades. Además, PL es responsable de resolver problemas de asignación de recursos limitados para actividades que compiten entre sí por dichos recursos. Como el problema definido para llevar a cabo el estudio tiene el objetivo de determinar la mejor combinación de refrigerios que maximice los ingresos de la empresa, en función de las cantidades de materias primas disponibles en el stock, la capacidad de producción y el tiempo promedio empleado por producto y que la función objetivo y las restricciones pueden representarse de forma lineal, es

evidente que puede tratarse como un problema de LP. Además, las variables serán tratadas como enteros; por lo tanto, es posible aplicar la Programación lineal entera (PLI).

La compañía salada, ubicada en una ciudad del sureste de Goiás, quiere maximizar sus ingresos vendiendo seis tipos de salados. La información sobre cada sal se puede encontrar en la Tabla 1.

Tabla 1 - Datos sobre los ingresos por la venta de aperitivos

Salado	Variable (unidad)	Ingresos (R $) por
Coxinha	x1	2,50
Rollo de salchicha	x2	2,50
Empanada de carne	x3	2,50
Masa de jamón y queso	x4	2,50
Kebab de carne	x5	2,50
Kebab de queso	x6	2,50

Cada salado está compuesto de diferentes cantidades de materias primas (Tabla 2). Como se puede ver, una coxinha (la cantidad de coxinhas está representada por la variable x1) consume aproximadamente 25 g de papa y 65 g de pollo. Cabe señalar que los elementos de materia prima representados en estas restricciones fueron solo aquellos puestos a disposición por el empresario, ya que representan un mayor costo por elemento.

Tabla 2 - Cantidad de materia prima utilizada para cada sal y disponible en stock

Recurso	x1	x2	x3	x4	x5	x6	Stock (g)
Papa (g)	25	0 0	0 0	0 0	0 0	0 0	2,500
Carne (g)	0 0	0 0	40	0 0	75	sesenta	28,000
Pollo (g)	sesenta	0 0	0 0	0 0	0 0	0 0	6,000
Maíz (g)	0 0	0 0	20	0 0	0 0	0 0	4,000
Mozzarella (g)	0 0	35	0 0	50	0 0	25	10,000
Jamón (g)	0 0	0 0	0 0	50	0 0	0 0	3.000
Salchicha (g)	0 0	100	0 0	0 0	0 0	0 0	4,000
Trigo (g)	0 0	0 0	0 0	0 0	sesenta	50	16,000

Además, cada producto salado tiene su demanda diaria mínima y máxima en función de la capacidad de producción, como se puede ver en la Tabla 3. Cada producto salado debe tener una producción diaria mínima de cuatro artículos para satisfacer la demanda mínima. La producción máxima, por ejemplo de coxinha, debe ser de 90 unidades.

Tabla 3 - Demanda mínima y máxima para cada tipo de salado

salado	Demanda mínima	Demanda máxima
Coxinha	$x1 \geq 4$	$x1 \leq 90$
Rollo de salchicha	$x2 \geq 4$	$x2 \leq 40$
Empanada de carne	$x3 \geq 4$	$x3 \leq 180$
Masa de jamón y queso	$x4 \geq 4$	$x4 \leq 60$
Kebab de carne	$x5 \geq 4$	$x5 \leq 160$
Kebab de queso	$x6 \geq 4$	$x6 \leq 120$

Finalmente, los bocadillos salados tardan unos 25 segundos en ensamblarse, independientemente del tipo. El tiempo total reservado para el montaje de los bocadillos es de 4 horas y 30 minutos, o 16.200 segundos. Por lo tanto, la ecuación 8 se refiere a la última restricción del problema.

$$25x1 + 25x2 + 25x3 + 25x4 + 25x5 + 25x6 \leq 16{,}200 \quad (8)$$

Por lo tanto, es posible representar el problema de la compañía mediante ecuaciones / desigualdades matemáticas lineales. La ecuación 9 se refiere a la función objetivo, mientras que la ecuación 10 a la ecuación 32 son las restricciones.

$$\text{Máx. } Z = 2.5x1 + 2.5x2 + 2.5x3 + 2.5x4 + 2.5x5 + 2.5x6 \quad (9)$$

Sujeto a:

$$25x1 \leq 2500 \quad (10)$$

$$40x3 + 75x5 + 65x6 \leq 28{,}000 \quad (11)$$

$$65x1 \leq 6{,}000 \quad (12)$$

$$20x3 \leq 4{,}000 \quad (13)$$

$$35x2 + 50x4 + 25x6 \leq 10{,}000 \quad (14)$$

$$50x4 \leq 3{,}000 \quad (15)$$

$$100x2 \leq 4{,}000 \quad (16)$$

$$65x5 + 50x6 \leq 16{,}000 \quad (17)$$

$$x1 \leq 90 \quad (18)$$

$$x2 \leq 40 \quad (19)$$

$$x3 \leq 180 \quad (20)$$

$$x4 \leq 60 \quad (21)$$

$$x5 \leq 160 \quad (22)$$

$$x6 \leq 120 \quad (23)$$

$$x1 \geq 4 \quad (24)$$

$$x2 \geq 4 \quad (25)$$

$$x3 \geq 4 \quad (26)$$

$$x4 \geq 4 \quad (27)$$

$$x5 \geq 4 \quad (28)$$

$$x6 \geq 4 \quad (29)$$

$$25x1 + 25x2 + 25x3 + 25x4 + 25x5 + 25x6 \leq 16{,}200 \quad (30)$$

$$x1, x2, x3, x4, x5, x6 \geq 0 \quad (31)$$

$$\text{enteros } x1, x2, x3, x4, x5, x6 \quad (32)$$

Una vez definido el modelo matemático, el siguiente paso fue traducirlo al lenguaje Julia, donde el optimizador encontrará una posible solución. Primero, fue necesario instalar el paquete Julia for Mathematical Programming (JuMP) para que Julia pudiera leer los problemas de optimización. Es un lenguaje matemático utilizado para resolver problemas de programación lineales, no lineales, enteros mixtos, cónicos, de segundo orden y semi-definidos. Sin embargo, el paquete no puede resolver problemas de optimización por sí solo, por lo que los optimizadores también deben instalarse juntos para interactuar con JuMP. En este caso, en particular, se instaló el optimizador "Cbc", que es gratuito y admite problemas de programación de PL y enteros mixtos.

La instalación de paquetes requiere solo dos líneas de comando, una línea para cada paquete. Se recomienda que, al final de la instalación, se actualicen a la última versión utilizando otra línea de comando. Luego, es necesario informar qué paquetes se utilizarán para la resolución, además de crear el objeto que almacenará el modelo del problema (Figuras 1 y 2). La última línea de comando en la Figura 2 sirvió para crear el objeto llamado ModeloMat. El nombre del modelo no necesariamente tiene que ser el mismo, y se le puede dar cualquier nombre. Vale la pena mencionar que en el futuro el objeto debe escribirse de la misma manera, ya que existe una diferencia entre letras mayúsculas y minúsculas. El argumento entre paréntesis indica que el optimizador utilizado para resolver el problema fue Cbc.

```
julia> import Pkg; Pkg.add("JuMP")
julia> import Pkg; Pkg.add("Cbc")
julia> Pkg.update()
```

Figura 1 - Comandos para instalar y actualizar paquetes JuMP y Cbc en Julia

```
julia> using JuMP

julia> using Cbc

julia> ModeloMat = Model(with_optimizer(Cbc.Optimizer))
```

Figura 2 - Comandos para cargar los paquetes instalados y crear el objeto "ModeloMat"

Luego, de acuerdo con la Figura 3, las variables de decisión del modelo se definieron con el comando @variable (nombre del objeto, nombre y límites de la variable, tipo de variable). Los límites pueden ser inferiores, superiores o ambos. Si no se especifica el tipo de la variable, se considera real. "Bin" se usa para variables binarias e "Int" para enteros.

```
julia> @variable(ModeloMat, x1 >= 0, Int)

julia> @variable(ModeloMat, x2 >= 0, Int)

julia> @variable(ModeloMat, x3 >= 0, Int)

julia> @variable(ModeloMat, x4 >= 0, Int)

julia> @variable(ModeloMat, x5 >= 0, Int)

julia> @variable(ModeloMat, x6 >= 0, Int)
```

Figura 3 - Definición de las variables problemáticas en Julia

Como las variables presentes en el modelo elaborado son los tipos de bocadillos, la cantidad producida debe ser entera, lo que justifica ser clasificada como entera. El siguiente paso consistió en definir la función objetivo usando el comando @objective (nombre del modelo, Min o Max, ecuación), como se puede ver en la Figura 4. Finalmente, se agregaron restricciones usando el comando @constraint (nombre del modelo, ecuación / desigualdad), como se muestra en la Figura 5.

```
julia> @objective(ModeloMat, Max, 2.5x1 + 2.5x2 + 2.5x3 + 2.5x4 + 2.5x5 + 2.5x6)
```

Figura 4 - Definición de la función objetivo del problema en Julia

```
julia> @constraint(ModeloMat, 25x1 <= 2500)
julia> @constraint(ModeloMat, 40x3 + 75x5 + 65x6 <= 28000)
julia> @constraint(ModeloMat, 65x1 <= 6000)
julia> @constraint(ModeloMat, 20x3 <= 4000)
julia> @constraint(ModeloMat, 35x2 + 50x4 + 25x6 <= 10000)
julia> @constraint(ModeloMat, 50x4 <= 3000)
julia> @constraint(ModeloMat, 100x2 <= 4000)
julia> @constraint(ModeloMat, 65x5 + 50x6 <= 16000)
julia> @constraint(ModeloMat, x1 <= 90)
julia> @constraint(ModeloMat, x2 <= 40)
julia> @constraint(ModeloMat, x3 <= 180)
julia> @constraint(ModeloMat, x4 <= 60)
julia> @constraint(ModeloMat, x5 <= 160)
julia> @constraint(ModeloMat, x6 <= 120)
julia> @constraint(ModeloMat, x1 >= 4)
julia> @constraint(ModeloMat, x2 >= 4)
julia> @constraint(ModeloMat, x3 >= 4)
julia> @constraint(ModeloMat, x4 >= 4)
julia> @constraint(ModeloMat, x5 >= 4)
julia> @constraint(ModeloMat, x6 >= 4)
julia> @constraint(ModeloMat, 25x1 + 25x2 + 25x3 + 25x4 + 25x5 + 25x6 <= 16200)
```

Figura 5 - Definición de las restricciones del problema en Julia

Después de definir las restricciones del modelo, el comando de impresión (nombre del modelo) se puede usar para mostrar todo el modelo construido (Figura 6).

```
julia> print(ModeloMat)
Max 2.5 x1 + 2.5 x2 + 2.5 x3 + 2.5 x4 + 2.5 x5 + 2.5 x6
Subject to
 x1 integer
 x2 integer
 x3 integer
 x4 integer
 x5 integer
 x6 integer
 x1 >= 0.0
 x2 >= 0.0
 x3 >= 0.0
 x4 >= 0.0
 x5 >= 0.0
 x6 >= 0.0
 x1 >= 4.0
 x2 >= 4.0
 x3 >= 4.0
 x4 >= 4.0
 x5 >= 4.0
 x6 >= 4.0
 25 x1 <= 2500.0
 40 x3 + 75 x5 + 65 x6 <= 28000.0
 65 x1 <= 6000.0
 20 x3 <= 4000.0
 35 x2 + 50 x4 + 25 x6 <= 10000.0
 50 x4 <= 3000.0
 100 x2 <= 4000.0
 65 x5 + 50 x6 <= 16000.0
 x1 <= 90.0
 x2 <= 40.0
 x3 <= 180.0
 x4 <= 60.0
 x5 <= 160.0
 x6 <= 120.0
```

Figura 6 - Modelo definido demostrado en Julia

Finalmente, el comando optimizar! (Nombre del modelo) sirve para resolver el problema y determinar la solución óptima (Figura 7). En este paso, son posibles algunas respuestas: (i) Óptimo, significa que el problema tiene una solución óptima y se ha determinado; (ii) Inviable, es decir, el problema se considera inviable porque no tiene una solución óptima; y (iii) Sin límites, donde el valor de la función objetivo crece (o disminuye) indefinidamente a medida que los valores de las variables también aumentan (o disminuyen).

Como se muestra en la Figura 7, el problema tiene una solución óptima. Además, el optimizador de Cbc describe algunas características para resolver el problema. En la parte superior, está la descripción del optimizador. Justo debajo, hay un comentario sobre la solución del problema por otros métodos. Si el problema permitiera valores continuos en las variables,

el valor de la función objetivo sería R $ 1,609.62. El programa también menciona otros métodos de resolución, pero no se utilizaron porque no cumplían los requisitos previos, como lo demuestra la frase "se intentó 0 veces [...]".

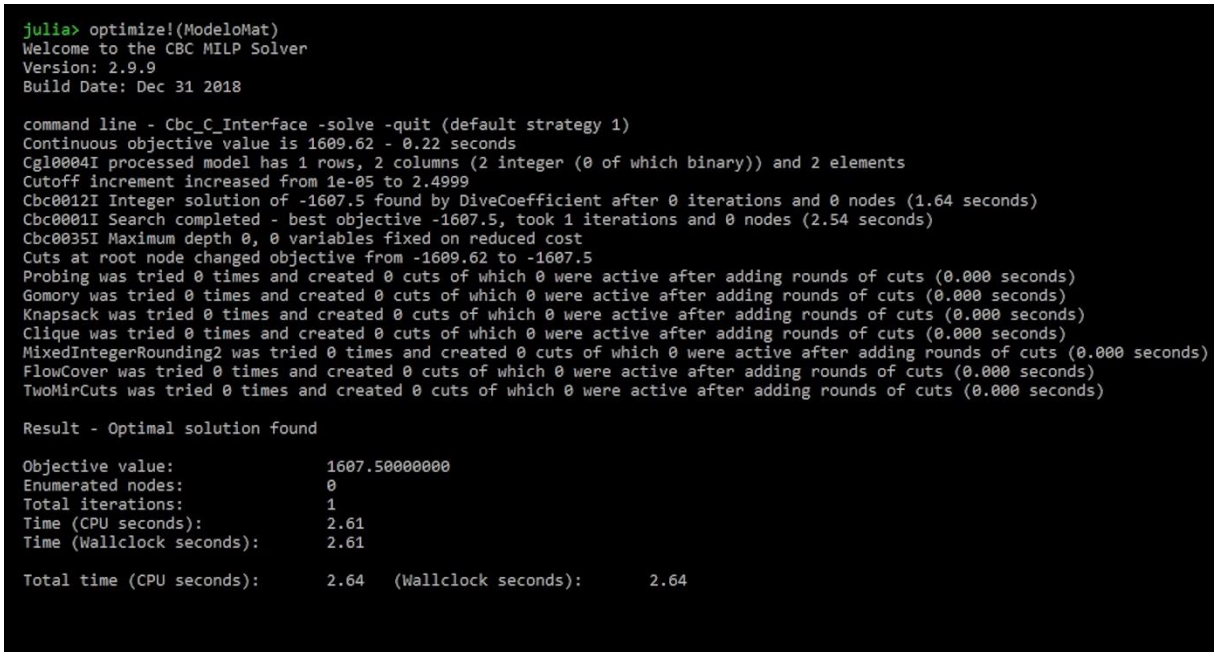

Figura 7 - Solución óptima determinada por Julia

Finalmente, el informe incluye información sobre el número de iteraciones para lograr la solución óptima y los tiempos de CPU y Wallclock, donde solo el primero considera los tiempos de parada durante la ejecución de una tarea. El tiempo de Wallclock cuenta todo el tiempo dedicado a realizar una tarea, incluidas las interrupciones. Por lo tanto, como se puede ver, no hubo interrupción en la ejecución de la actividad, ya que el tiempo de Wallclock era igual al tiempo de la CPU.

Los resultados se imprimen utilizando la combinación del comando print () y el comando que desea saber. JuMP.objective_value (nombre del modelo) se usa para el valor de la función objetivo y JuMP.value (variable) para las variables de decisión. Por lo tanto, de acuerdo con la Figura 8, la empresa tendrá un ingreso máximo de R $ 1,607.50 si decide producir 90 muslos, 40 rollos de salchicha, 180 pasteles de carne, 60 pasteles de jamón y queso, 153 kibbehs de carne y 120 kibbehs de queso. .

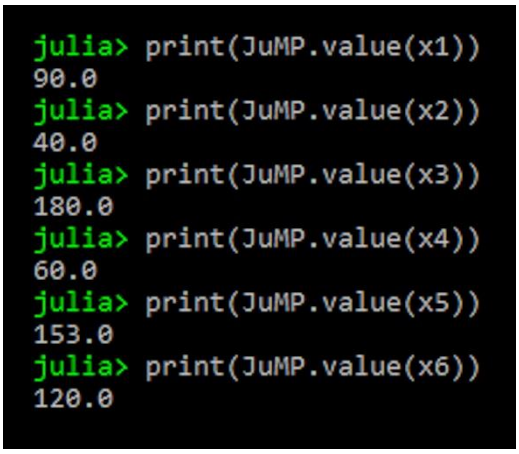

Figura 8 - Valores de las variables determinadas por Julia

4.2 Problema de vendedor ambulante

Además de definir la mejor combinación de bocadillos que se producirán, la compañía también quiere optimizar su ruta de entrega. Para esto, la técnica utilizada fue el problema del vendedor ambulante, que determina la ruta más corta para recorrer una serie de puntos solo una vez y, al final, regresa al punto de origen.

En total, la compañía entrega los refrigerios a cuatro clientes. Cada cliente se define como un punto (o nodo) en la red. Esto, a su vez, se define como un conjunto de nodos conectados por arcos. En este caso particular, los nodos representan a los clientes y los arcos son las distancias entre los clientes. El nodo de origen es C1, es decir, es el nodo donde comenzará el viaje (la empresa estudió). Por lo tanto, desde la empresa de enfoque del estudio (C1) hasta el cliente C2, la distancia recorrida es de 1,5 km. La Tabla 4 muestra las distancias dadas en kilómetros (km).

Tabla 4 - Distancia entre clientes en kilómetros

Nosotros	C1	C2	C3	C4	C5
C1	0 0	1,5	1.4	2.6	3.5
C2	1,5	0 0	0.8	2,0	3.2
C3	1.4	0.8	0 0	1,5	2.2 2.2
C4	2.6	2,0	1,5	0 0	0.8
C5	3.5	3.2	2.2 2.2	0.8	0 0

Es posible percibir que el problema es simétrico, ya que la distancia para viajar del cliente i al cliente j es la misma en sentido contrario. La figura 9 representa el problema visualmente.

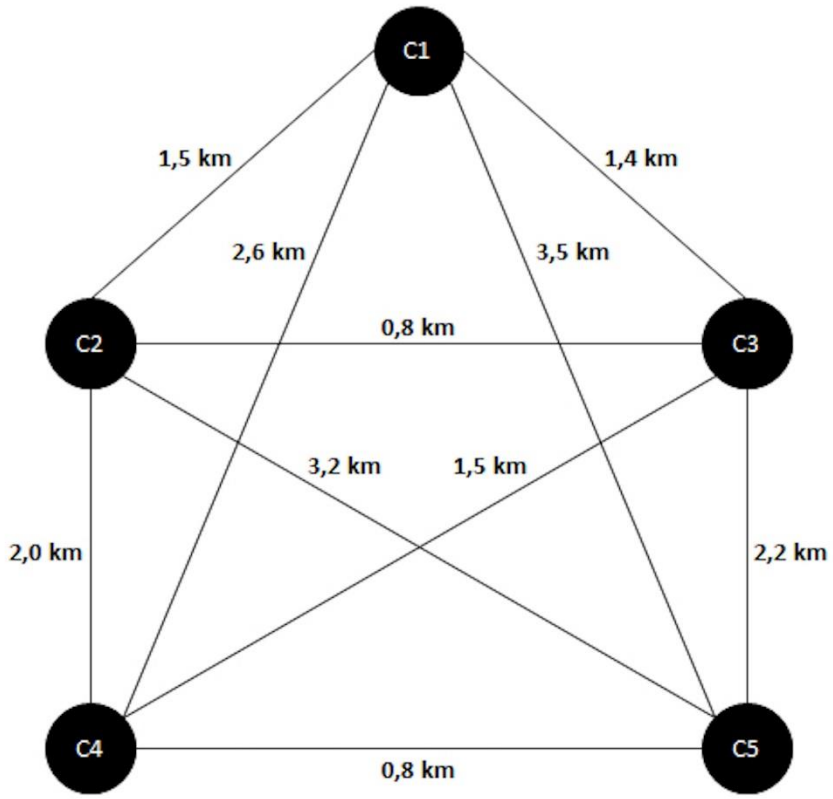

Figura 9 - Representación visual del problema del vendedor ambulante

Antes de traducir el problema propuesto al lenguaje Julia, es necesario instalar los paquetes necesarios para la resolución. Por lo tanto, el paquete utilizado fue "TravellingHalesmanHeuristics" (Figura 10).

```
julia> import Pkg; Pkg.add("TravelingSalesmanHeuristics")

julia> Pkg.update()
```

Figura 10 - Representación visual del problema del vendedor ambulante

Dentro del paquete hay varias funciones, tales como: (i) cheap_insertion, donde la ruta más barata se calcula en términos monetarios; (ii) farthest_insertion, que calcula la ruta en función de los vértices más alejados del vértice de origen; y (iii) el vecino más cercano, que es la heurística del vecino más cercano.

Después de instalar y actualizar el paquete, es necesario declararlo e ingresar las distancias entre los clientes en formato de matriz cuadrada (Figura 11). Por lo tanto, el número de filas debe ser igual al número de columnas. Si no se completa este paso, se devolverá un

error en los siguientes pasos. La matriz debe almacenarse en un objeto, vale la pena recordar que el nombre del objeto es libre, sin embargo, debe escribirse de la misma manera en el futuro, ya que las letras mayúsculas y minúsculas se diferencian en el idioma.

```
julia> using TravelingSalesmanHeuristics

julia> distclientes = [
       0 1.5 1.4 2.6 3.5;
       1.5 0 0.8 2.0 3.2;
       1.4 0.8 0 1.5 2.2;
       2.6 2.0 1.5 0 0.8;
       3.5 3.2 2.2 0.8 0;
       ]
5×5 Array{Float64,2}:
 0.0  1.5  1.4  2.6  3.5
 1.5  0.0  0.8  2.0  3.2
 1.4  0.8  0.0  1.5  2.2
 2.6  2.0  1.5  0.0  0.8
 3.5  3.2  2.2  0.8  0.0
```

Figura 11: declarar el paquete y definir el objeto con las distancias entre clientes

El objeto creado para almacenar las distancias entre clientes era "distclientes". Los valores están separados por espacios y al final de cada línea, es necesario insertar un punto y coma para indicar el final de la misma. La matriz se inicia y termina con corchetes. Justo debajo es posible verificar si la matriz se creó correctamente. La matriz creada es de orden cinco, tiene datos de tipo real de 64 bits y es de dimensión 2.

Entonces, se llama la función heurística correspondiente elegida para resolver el problema. Los parámetros necesarios para la función del vecino más cercano son el vecino más cercano (distmat, firstcity). El parámetro distmat corresponde a la matriz con las distancias y el segundo parámetro define qué nodo será el nodo fuente. Como la matriz "distclientes" tiene 5 líneas, el parámetro puede recibir un valor del uno al cinco, ya que Julia comienza a contar desde el número uno (que indica el primer nodo en la red), como se ve en la Figura 12.

```
julia> nearest_neighbor(distclientes,firstcity=1)
([1, 2, 3, 5, 4, 1], 7.9)
```

Figura 12 - Solución encontrada en Julia

Por lo tanto, analizando la Figura 12, la mejor ruta para la entrega de refrigerios viene dada por los valores entre corchetes. Entonces, primero tiene que abandonar la empresa (nodo C1) e ir al cliente 2. Luego, cliente 3, bajar al cliente 5 y finalmente, ir al cliente 4 antes de regresar al nodo de origen (C1), que es la empresa. El valor después de la coma corresponde a la distancia total de la ruta, que vale 7,9 km. La Figura 12 ilustra el camino más corto resaltado.

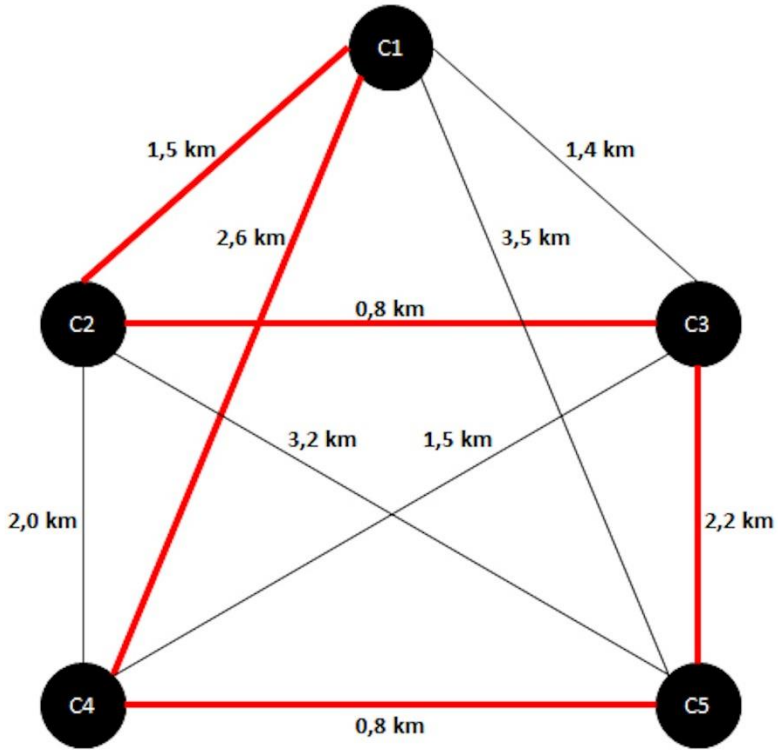

Figura 11 - Solución encontrada en Julia

5. Consideraciones finales

Se identificó la solución óptima al problema de programación lineal que tenía como objetivo identificar la mejor combinación de bocadillos que se produciría para maximizar los ingresos de la microempresa estudiada. Además, la mejor ruta de entrega, minimizando las distancias de entrega entre todos los clientes, se encontró a través del problema del vendedor ambulante. Por lo tanto, se logró el objetivo buscado en el artículo.

Con respecto al primer problema, para que la empresa obtenga el máximo ingreso posible, es necesario que mantenga su producción cerca de la demanda máxima, ya que hubo una diferencia de solo siete kibbeh de carne. Con respecto al problema del vendedor ambulante, debe tenerse en cuenta que la ruta para la entrega de bocadillos tendrá la misma distancia si se realiza de manera inversa.

Para ambos problemas, el método de resolución es relativamente simple porque el lenguaje de programación Julia contiene funciones y paquetes específicos para cada uno. En consecuencia, Julia aprovecha los diferentes lenguajes de programación para lograr eficiencia y productividad. También destaca el hecho de que es un lenguaje gratuito y de código abierto. Aún así, debe tenerse en cuenta que el artículo presenta dos de las diversas técnicas de optimización que están disponibles para su resolución en el lenguaje Julia. El hecho de que reúna varias técnicas ayuda a la comodidad de los analistas, ya que solo pueden especializarse en el lenguaje y resolver diversos problemas comerciales.

En cuanto a la contribución en el ámbito empresarial, este artículo se puede usar como tutorial para usar estas técnicas de investigación operativa en micro y pequeñas empresas, ya que las resoluciones de problemas se hicieron paso a paso para que el lector pueda usarlas. ellos. El uso del lenguaje Julia se consideró útil y posible ser aplicable en pequeñas empresas. En cuanto al campo académico, este artículo puede usarse como tutorial para futuros proyectos e investigaciones que incluyen programación lineal y / o el problema del vendedor ambulante. Además, revela la aplicación práctica de los problemas de OP en las microempresas, lo que aumenta la revelación de que un área compleja se puede utilizar para el apoyo gerencial también en micro y pequeñas empresas.

Se sugiere, para futuras investigaciones, la aplicación del lenguaje Julia en otras micro y pequeñas empresas con el fin de obtener comentarios de los empresarios sobre la experiencia de uso y los beneficios de gestión percibidos después de la implementación de los resultados alcanzados.

Referencias bibliográficas

ALMEIDA, WS; SILVA, YO; SANTOS, NVM; RIBEIRO, LM; BACHEGA, SJ Aplicación de toda la programación lineal para maximizar el beneficio de una empresa en el sector de la belleza y la estética. En: REUNIÓN NACIONAL DE INGENIERÍA DE PRODUCCIÓN, 37., 2017, Joinville. Anais .... Río de Janeiro: ABEPRO, 2017. p. 01-16.

ANDERSON, TA, LIU, H., KUPER, L., TOTONI, E., VITEK, J.; SHPEISMAN, T. Paralelizando a Julia con un DSL no invasivo. DARDOS. v. 3, n. 2, p. 01-29, 2017.

ARENALES, M .; MORABITO, R .; ARMENTANO, V. YANASSE, H. Investigación operativa. 1. ed. Río de Janeiro: Elsevier, 2007.

BALDASSI, C. Un método para reducir la tasa de rechazo en las cadenas de Monte Carlo Markov. Revista de Mecánica Estadística: Teoría y Experimento. Vol. 2017, pp. 01-19, 2017.

BEZANSON, J.; EDELMAN, A; KARPINSKI, S; SHAH, VB Julia: un nuevo enfoque de la computación numérica. Revista SIAM, Vol. 59, n. 1, pp. 65-98, 2017.

BEZANSON, J.; KARPINSKI, S; SHAH, V; EDELMAN, A. ¿Por qué creamos a Julia? 2012. Disponible en:http://julialang.org/blog/2012/02/why-we-created-julia. Consultado el 10/09/2019.

BRYMAN, A. Métodos de investigación y estudios de organización. Londres: Uniwin Hyman, 1989.

CARVALHO, MCM de. La construcción del conocimiento científico: algunas proposiciones. En: CARVALHO, MCM de (org.). Construyendo conocimiento. 2.ed. Campinas: Papirus, 2000.

CASTELLUCCIA, PB JULIA Y JuMP: NUEVAS HERRAMIENTAS PARA PROGRAMACIÓN. Investigación operativa para el desarrollo, v. 9, n. 2, p. 48-61, 2017.

CRESWELL, JW Proyecto de investigación. Métodos cualitativos, cuantitativos y mixtos. 3. ed. Porto Alegre: Artmed, 2010.

EHRLICH, PJ Investigación operativa. 5. ed. São Paulo: Atlas, 1985.

GIL, AC Cómo preparar proyectos de investigación. 4. ed. São Paulo: Atlas, 2009.

HILLIER, FS; LIEBERMAN, GJ Introducción a la Investigación de Operaciones. 9. ed. Porto Alegre: AMGH, 2013.

JULIALANG.ORG. Julia Micro-Benchmarks. 2018. Disponible en:https://julialang.org/benchmarks/. Consultado el 28/03/2019.

KARP, RM Sobre la complejidad computacional de los problemas combinatorios. Newtorks 5, 1975.

LINS, MPE; CALÔBA, Guilherme M. Programación lineal. 4. ed. Río de Janeiro: Interciência, 2010.

MOGENSEN, PK; RISETH, AN Optim: un paquete de optimización matemática para Julia. Revista de software de código abierto, v. 3, n. 24, p. 01-03, 2018.

MOREIRA, DA Investigación Operacional - Curso introductorio. 2. ed. São Paulo: Cengage Learning, 2010.

PASSOS, EJPF Programación lineal como instrumento de investigación operativa. 1. ed. São Paulo: Atlas, 2008.

PEREIRA, CD; CUNHA, GF; SILVA, MG Simulación en investigación operativa: una revisión literaria. EEPA, Campo Mourão. Disponible:http://www.fecilcam.br/anais/ix_eepa/data/uploads/3-pesquisa-operacional/3-03.pdf. Consultado el 10/10/2019.

PRADO, Darci. Programación lineal. 5. ed. Belo Horizonte: INDG, 2007.

RACKAUCKAS, C; SCHILLING, T .; NIE, Q. Control de ruido independiente del promedio de destinos celulares a través de estados intermedios. IScience, v. 3, pp. 11-20, 2018.

SANTANA, TMN; GUIMARAES, RRS; FONSECA, GS; GONCALVES, LF; BACHEGA, SJ Aplicación del problema del árbol generador mínimo en una empresa proveedora de internet. En: SIMPOSIO DE INGENIERÍA DE PRODUCCIÓN, 2018, catalán. Anais ... catalán: UFG-RC, 2018. p. 01-09.

SILVEIRA, JFP Problema de vendedor ambulante. UFRGS, Rio Grande do Sul. Disponible en:http://www.mat.ufrgs.br/~portosil/caixeiro.html. Consultado el 17/10/2018.

SOUSA, LT; VEIGA, AG; CHAVES, MFC; VAZ, MV; BACHEGA, SJ Aplicación de optimización de redes en una empresa del sector avícola. En: MACHADO, MWK (Org.). Ingeniería de Producción: ¿Cuál es tu plan? Ponta Grossa: Atena Editora, 2019, p. 356-367.

SOUZA, LC; VIANA, GG; YOSHIDA, YM; TELLAROLI, LF; BACHEGA, SJ El problema del camino más corto aplicado a una compañía de periódicos. En: POISSON (Org.). Gestión de la producción en foco - Volumen 26. Belo Horizonte: Poisson, 2019, v. 26, p. 127-135.

TAHA, HA Investigación Operativa. 8. ed. São Paulo: Pearson Pretince Hall, 2008.

TEIXEIRA, NB; SILVA, CF; VIRGOLINO, GA; SOUZA, A; BACHEGA, SJ Aplicación de toda la programación lineal en una pequeña fábrica de parrillas prefabricadas. En: SIMPOSIO DE INGENIERÍA DE PRODUCCIÓN, 24., 2017, Bauru. Anais ... Bauru: UNESP, 2017. p. 01-12.