Selección de Componentes Discretos para un Filtro Activo mediante *Programación por Restricciones* y *Optimización por Colonia de Hormigas*

Leandro Demarco Vedelago

Director:
Dr. Eduardo Abel Romero

Trabajo especial presentado para acceder al grado de
Licenciado en Ciencias de la Computación

Facultad de Matemática, Astronomía, Física y Computación
Universidad Nacional de Córdoba

MMXIX

Abstract

In the current active filter design, one of the possible implementations is the so called RC, in which the filter is built with operational amplifiers, resistors and capacitors. In order to satisfy the filter specifications it's of great importance the selection of the discrete components that make up the filter.

Given the wide range of values that these components can take, it results inefficient to enumerate all possible combinations and select amongst them the best one. In this work we use a metaheuristic called ACOR which allows to solve this kind of constrained combinatorial optimization problems in reasonable time while guaranteeing that the obtained solutions satisfy all the restrictions, though they might not be of the best quality (where quality is defined with respect to some characteristic that depends on the chosen values).

All the code developed for this work was implemented in Python.

Resumen

En el diseño actual de filtros activos una de las opciones de implementación es la denominada RC (resistencia/capacitor), en la cual el filtro se construye a partir de amplificadores operacionales, resistencias y capacitores. Para satisfacer el cumplimiento de las especificaciones del filtro resulta de gran importancia la selección de los componentes discretos del filtro.

Dado el amplio espectro de valores que los componentes pueden tomar, resulta ineficiente enumerar todas las combinaciones posibles y seleccionar entre ellas la mejor. En este trabajo usamos una metaheurística denominada ACOR la cual permite resolver este tipo de problemas combinatorios con restricciones en tiempos de ejecución razonables al tiempo que garantiza que las soluciones obtenidas satisfacen todas las restricciones aunque pueden no ser de la mejor calidad (donde la calidad se define respecto a alguna característica dependiente de los valores elegidos).

Todo el código desarrollado para este trabajo fue implementado en Python.

Keywords: Ant Colony Optimization; Metaheuristics; Filter Design; Constraint Programming; Combinatorial Optimization

CCS 2012:

- **Theory of computation~Constraint and logic programming**
- **Computing methodologies~Heuristic function construction**
- *Theory of computation~Design and analysis of algorithms*
- *Mathematics of computing~Mathematical optimization*

Índice general

1. **Fundamentos electrónicos** 3
   1.1. Sensibilidad de un filtro IGMFB 3
2. **Programación por Restricciones** 5
   2.1. Noción de una restricción 6
   2.2. Definición formal de un CSP 6
   2.3. Complejidad de un CSP 7
   2.4. Problemas de optimización relacionados a CSP 7
      2.4.1. Maximización de satisfacción de restricciones 7
      2.4.2. Optimización restringida 8
   2.5. Definición del problema del filtro como CSP 8
3. **Algoritmo Exacto** 9
4. **Solución Analítica** 12
5. **Optimización por Colonia de Hormigas** 17
   5.1. La metaheurística ACO 18
   5.2. ACO en dominios continuos: $ACO_{\mathbb{R}}$ 20
      5.2.1. *La función de densidad de probabilidad* 20
      5.2.2. *Implementación algorítmica de* $ACO_{\mathbb{R}}$ 23
6. **$ACO_{\mathbb{R}}$ Aplicado al Problema del Filtro** 24
   6.1. Elección de la función costo 25
   6.2. Versión continua con todas las variables libres 27
   6.3. Versión continua con R_1 fijado 33
   6.4. Versión continua con búsqueda de vecinos discretos cercanos 37
   6.5. Versión continua con todas las variables libres y búsqueda de vecinos cercanos 39
   6.6. Versión Discreta *Pura* 40
7. **Conclusiones y Trabajos a Futuro** 44
A. **Tablas de Resultados** 45
   A.1. Algoritmo Exacto 45
   A.2. Óptimos de Pareto 50
   A.3. Versión Continua con R_1 fijado 51

1. Fundamentos electrónicos

Tomaremos como caso de estudio el mismo filtro que se utiliza en [2]. Dicho filtro es de tipo IGMFB (Infinite-Gain Multiple Feedback) pasabajo de segundo orden. Son filtros bicuadráticos que emplean múltiples lazos de retroalimentación y un amplificador operacional. Una descripción más detallada de este tipo de filtros puede encontrarse en [3] y [4]. En la figura (1) se muestra el esquemático del filtro a analizar.

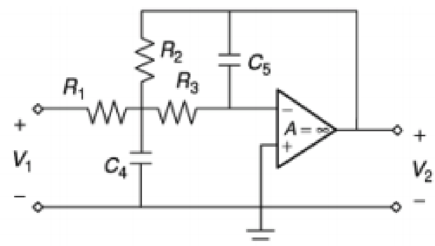


Figura 1: Filtro IGMFB pasabajo de segundo orden.

Las especificaciones del filtro están dadas por la ganancia en la banda de paso (G), la frecuencia de polo (ω) y el factor de calidad (Q). En función de estos valores la función de transferencia del filtro queda expresada de la siguiente manera:

$$F(s) = \frac{G\,\omega^2}{s^2 + (\frac{\omega}{Q})s + \omega^2} \tag{1}$$

Para el filtro de la figura (1) los valores de G, ω y Q están dados por las siguientes expresiones,

$$G = R_2/R_1 \tag{2}$$

$$\omega = 1/\sqrt{R_2 R_3 C_4 C_5} \tag{3}$$

$$Q^{-1} = \sqrt{C_5/C_4}\left(\sqrt{R_2 R_3}/R_1 + \sqrt{R_3/R_2} + \sqrt{R_2/R_3}\right) \tag{4}$$

1.1. Sensibilidad de un filtro IGMFB

El término sensibilidad es utilizado para expresar una medida de la variación del rendimiento como resultado de cambios en los valores de los componentes. Dichas variaciones pueden ocurrir debido al envejecimiento de los mismos, tolerancias de fabricación, condiciones ambientales (temperatura), entre otros factores [3,4]. Mientras menos sensible es un filtro a los cambios en sus componentes, más estables permanecen sus características y, por lo tanto, existen más posibilidades de que pueda permanecer dentro de sus especificaciones, independientemente de la presencia de dichos cambios. Una ventaja de los filtros IGMFB es que presentan sensibilidades más bajas respecto a otros filtros bicuadráticos.

De manera general, si F es una función de varias variables, $F(x_1, x_2, ..., x_n)$, entonces la sensibilidad de F con respecto a x_i está definida por:

$$S_{x_i}^{F} = \frac{\%\ cambio\ en\ F}{\%\ cambio\ en\ x_i} = \frac{\partial F/F}{\partial x_i/x_i} \tag{5}$$

Se considera que un filtro tiene baja sensibilidad cuando la sensibilidad respecto a cada uno de sus componentes es inferior a la unidad [4].

Para el filtro de la figura (1) las sensibilidades de Q y ω con respecto a cada uno de los componentes pasivos vienen dadas, luego de aplicar la definición (5), por las siguientes expresiones:

$$S_{R_1}^{\omega} = 0 \tag{6}$$

$$S_{R_2}^{\omega} = S_{R_3}^{\omega} = S_{C_4}^{\omega} = S_{C_5}^{\omega} = -\left(\frac{1}{2}\right) \tag{7}$$

$$S_{C_4}^{Q} = -S_{C_5}^{Q} = \left(\frac{1}{2}\right) \tag{8}$$

$$S_{R_1}^{Q} = Q\left(\frac{1}{R_1}\sqrt{\frac{R_2R_3C_5}{C4}}\right) \tag{9}$$

$$S_{R_2}^{Q} = -\frac{Q}{2}\left(\frac{1}{R1}\sqrt{\frac{R_2R_3C_5}{C_4}} - \sqrt{\frac{R_3C_5}{R_2C_4}} + \sqrt{\frac{R_2C_5}{R_3C_4}}\right) \tag{10}$$

$$S_{R_3}^{Q} = -\frac{Q}{2}\left(\frac{1}{R1}\sqrt{\frac{R_2R_3C_5}{C_4}} + \sqrt{\frac{R_3C_5}{R_2C_4}} - \sqrt{\frac{R_2C_5}{R_3C_4}}\right) \tag{11}$$

Las sensibilidades de las ecuaciones (6), (7) y (8) adoptan valores fijos, mientras que las otras dependen de los valores que adopten los componentes. En consecuencia, debemos considerar las sensibilidades (9), (10) y (11) a la hora de seleccionar los valores para cada componente.

Idealmente, querríamos hallar una solución que minimice las tres sensibilidades $S_{R_1}^{Q}$, $S_{R_2}^{Q}$ y $S_{R_3}^{Q}$ simultáneamente. En general, tal solución no existe y entonces es necesario adoptar una solución de compromiso. En nuestro caso elegimos minimizar la *sensibilidad total* S del filtro, que definimos de la siguiente manera:

$$S \equiv S_1 + S_2 + S_3 \tag{12}$$

donde, para simplificar la notación, hemos definido $S_i \equiv |S_{R_i}^{Q}|$.

En el caso de estudio seleccionado, la especificación elegida para el filtro consiste en establecer valores objetivo para G, ω y Q que deben cumplirse dentro de una tolerancia especificada ϵ. Los valores objetivo que utilizaremos son $G_0 = 3$, $\omega_0 = 2\pi \cdot 10^3$ rad/s y $Q_0 = 1/\sqrt{2}$.

Consideraremos dos casos, uno, que denominaremos *escenario 1*, donde las resistencias y capacitores pueden tomar valores de las series E96 y E24, respectivamente, y otro, que denominaremos *escenario 2*, donde las series utilizadas son la E24 para las resistencias y la E12 para los capacitores. En ambos casos los extremos de los valores de las series son similares, cambiando la cantidad de valores disponibles entre esos extremos. De este modo, el rango de valores posibles para las resistencias es en ambos escenarios aproximadamente 10^3-$10^6 \Omega$, mientras que para los capacitores es 10^{-9}-10^{-6}F. Se supone que valores fuera de estos rangos conducirían a efectos negativos debido a capacidades parásitas o señales de corriente muy grandes. De esta manera, el espacio de búsqueda total consta de $N_R^3 \cdot N_C^2$ configuraciones posibles, donde N_R es el número de resistencias en la serie industrial utilizada, y N_C es el correspondiente valor para la serie utilizada para los capacitores. De este modo, el escenario 1 consta aproximadamente de $1,24 \cdot 10^{11}$ configuraciones mientras que el escenario 2 consta de $4{,}84 \cdot 10^8$ configuraciones. Consideraremos un error máximo admisible $\epsilon = 0,5\,\%$ para el escenario 1 y $\epsilon = 2,5\,\%$ para el escenario 2.

Si bien efectuamos los cálculos en los dos escenarios, en este documento, por razones de espacio, los resultados presentados corresponden, cuando no se especifique, al escenario más pequeño (el 2). Los resultados que se presenten correspondientes al otro escenario serán indicados explícitamente.

2. Programación por Restricciones

Intuitivamente, los problemas de satisfacción de restricciones (*CSP*, por sus siglas en inglés) pueden definirse como *encontrar una solución que satisface determinadas restricciones o propiedades*. Muchos problemas de la vida real pueden ser reducidos a este tipo de problemas, por ejemplo: el planeamiento y control del tráfico aéreo en un aeropuerto, la confección de una agenda de clases o el diseño de una dieta saludable. Todos estos problemas suelen ser *NP*-Complejos.

Existen dos técnicas para intentar solucionar este tipo de problemas [5]: una posibilidad es utilizar *métodos exactos*, que exploran el espacio de combinaciones de forma exacta estructurándolo como un árbol de búsqueda. Para acotar el problema de la explosión combinatoria, la búsqueda se combina con técnicas de filtrado, para *podar* subconjuntos de combinaciones, y con heurísticas de ordenación, para orientar la búsqueda hacia las ramas más promisorias primero.

Cuando las técnicas de filtrado y las heurísticas de ordenamiento no son suficientes para prevenir la explosión combinatoria, se debe dejar de lado la exactitud y utilizar *métodos heurísticos* que exploran el espacio de búsqueda de forma incompleta, utilizando (meta)heurísticas para guiar la búsqueda hacia las áreas más promisorias ignorando deliberadamente otras áreas. Dentro de este enfoque hay dos variantes: las *heurísticas perturbativas* -ej: Algoritmos Genéticos- que van modificando iterativamente combinaciones existentes a fin de crear nuevas combinaciones, y las *heurísticas constructivas* que construyen nuevas combinaciones iterativamente desde cero.

2.1. Noción de una restricción

Una restricción puede pensarse como una relación lógica entre un conjunto de valores, en principio, desconocidos, también llamados *variables*. De esta forma, una restricción restringe el conjunto de valores que pueden asignarse simultáneamente a sus variables.

Una posibilidad para definirla consiste en enumerar todo el conjunto de tuplas que pertenece a la relación o, de manera dual, especificar aquellos que **no** pertenecen a ella.

Existen diferentes tipos de restricciones, de acuerdo a los valores que pueden asignarse a las variables. Así, tenemos restricciones numéricas, booleanas, sobre conjuntos, etc. Una restricción numérica puede ser entonces una igualdad ($=$) o bien una desigualdad ($\neq, \geq, \leq, <, >$) entre dos expresiones aritméticas.

De manera similar, las restricciones sobre conjuntos expresan relaciones entre variables cuyo rango son conjuntos y pueden incluir relaciones de igualdad ($=$), desigualdad ($\neq$), inclusión ($\subset, \subseteq$) o pertenencia ($\in$).

Las restricciones sobre booleanos, pueden expresarse en términos de los operadores lógicos como por ejemplo $\wedge$, $\vee$, $\neg$, $\rightarrow$, etc.

2.2. Definición formal de un CSP

Formalmente, un CSP es una 3-upla (X, D, C) tal que:

- X es un conjunto finito de variables que corresponde a las incógnitas del problema.
- D es una función que asocia un dominio $D(x_i)$ con cada variable $x_i \in X$, es decir $D(x_i)$ es el conjunto de valores que la variable x_i puede tomar.
- C es un conjunto finito de restricciones y cada restricción $c_j \in C$ es una relación entre algunas variables de X; este conjunto de variables se denota como $var(c_j)$.

Resolver un CSP (X, D, C) consiste en asignar valores a todas las variables de forma tal que todas las restricciones sean satisfechas. Más formalmente, se definen:

- Una *asignación* es un conjunto de pares (variables, valor) denotado

  $$A = \{(x_1, v_1), ..., (x_n, v_n)\}$$

  donde la misma variable no puede asignarse a dos valores distintos, es decir:

  $$\forall((x_i, v_i), (x_j, v_j)) \in A \times A,\ x_i = x_j \implies v_i = v_j$$

  y el valor asignado a una variable pertenece a su dominio:

  $$\forall(x_i, v_i) \in A,\ v_i \in D(x_i)$$

  Se denota al conjunto de variables que tienen un valor asignado en la asignación A como $var(A)$:

$$var(A) = \{x_i \mid (x_i, v_i) \in A\}$$

- Una asignación A se dice *completa* si asigna un valor a todas las variables del problema, es decir, si $var(A) = X$; en caso contrario se dice que es *parcial.*
- Una asignación A *satisface* una restricción c_k tal que $var(c_k) \subseteq var(A)$ si la relación definida por c_k es verdadera para los valores de las variables de c_k definidos en A. Caso contrario, la asignación *viola* la restricción.
- Una asignación A (completa o parcial) es *consistente* si satisface todas las restricciones y es *inconsistente* si viola al menos una.
- Una *solución* es una asignación completa y consistente.

2.3. Complejidad de un CSP

Dado que los dominios pueden ser intervalos numéricos continuos, no todos los CSP son problemas combinatorios. La complejidad teórica de un CSP depende del dominio de las variables y del tipo de restricciones utilizadas.

En algunos casos, el problema puede ser polinomial. Este es el caso, por ejemplo, cuando todas las restricciones son (in)ecuaciones lineales y todos los dominios son intervalos numéricos continuos.

En otros casos, el problema puede ser indecidible. Por ejemplo, cuando las restricciones pueden ser cualquier fórmula matemática arbitraria y los dominios de las variables son intervalos numéricos continuos.

Sin embargo, en muchos casos, el problema es $\mathcal{NP}$-completo. En particular, los CSP con dominios finitos suelen pertenecer a este grupo en general.

2.4. Problemas de optimización relacionados a CSP

Los problemas de satisfacción de restricciones implican encontrar una solución que las satisfaga a todas o bien probar la inconsistencia si no existe ninguna solución. Sin embargo, en muchos casos también suelen haber involucrados problemas de optimización. En particular, el resolver problemas *excesivamente restringidos* suele convertirse en uno de maximización del número de restricciones satisfechas. Otras veces el resolver un CSP también involucra optimizar alguna función objetivo al tiempo que se satisfacen todas las restricciones.

2.4.1. Maximización de satisfacción de restricciones Cuando las restricciones de un CSP son tales que no pueden ser todas satisfechas simultáneamente, se dice que el CSP está excesivamente restringido. En este caso, usualmente se intenta hallar una asignación completa que satisfaga tantas restricciones como sea posible o, a la inversa, viole la menor cantidad posible. Este problema se llama *CSP parcial* o *MaxCSP* [12].

También suele ocurrir en muchos problemas que no todas las restricciones sean igualmente importantes. Algunas pueden ser *duras*, con lo cual no deben ser violadas, mientras que otras pueden ser *blandas*, permitiendo que se las viole a algún costo dado. En este caso se le asocia un peso a cada restricción blanda que define el costo de violarla, y el objetivo es encontrar la asignación completa que satisface todas las restricciones duras y minimiza la suma ponderada de las restricciones blandas violadas. Este problema se llama CSP ponderado, (WCSP, por sus siglas en inglés) [13].

Para estos CSP excesivamente restringidos, el espacio de búsqueda está definido por el conjunto de todas las asignaciones completas posibles y el objetivo es encontrar aquella que maximiza el nivel de satisfacción: el número de restricciones satisfechas en el caso de MaxCSP y la suma ponderada de las restricciones satisfechas en el caso los WCSP. En la mayoría de los casos, estos problemas son $\mathcal{NP}$-complejos ya que los problemas de decisión asociados son $\mathcal{NP}$-completos. Debe notarse además que son generalizaciones de CSP, con lo cual un algoritmo diseñado para resolver cualquiera de estos problemas puede utilizarse para resolver un CSP.

2.4.2. Optimización restringida Cuando las restricciones de un CSP son tales que existen múltiples soluciones que las satisfacen a todas, el CSP se dice estar *sub-restringido*. En este caso, algunas soluciones pueden ser preferibles a otras. Estas preferencias pueden ser expresadas añadiendo una función objetivo a ser maximizada (o minimizada), definiendo así un un problema de optimización restringida. Formalmente, un problema de optimización restringida está definido por un CSP (X, D, C) y una función objetivo $f : X \to \mathbb{R}$. El objetivo es hallar una solución del CSP que maximiza (o minimiza) f. En estos casos la dificultad no radica en hallar una solución, si no en hallar una óptima con respecto a f.

2.5. Definición del problema del filtro como CSP

Como se estableció en la sección 1, el problema consiste en seleccionar los valores para las resistencias R_1, R_2 y R_3 y los capacitores C_4 y C_5 de una lista de posibles valores (las series industriales E24 y E12 para resistencias y capacitores respectivamente). Si definimos

$$\begin{aligned} G_\pm &= (1 \pm \epsilon)G_0 \\ \omega_\pm &= (1 \pm \epsilon)\omega_0 \\ Q_\pm &= (1 \pm \epsilon)Q_0 \end{aligned} \tag{13}$$

Entonces el problema puede modelarse como un CSP (X, D, C) donde:

- $X = \{R_1, R_2, R_3, C_4, C_5\}$
- $D(R_i) = \text{E24}$, $i \in \{1, 2, 3\}$ y $D(C_i) = \text{E12}$, $i \in \{4, 5\}$
- $C = \{G_+ > G > G_-,\ \omega_+ > \omega > \omega_-,\ Q_+ > Q > Q_-\}$

Además, el problema está *sub-restringido* dado que hay múltiples soluciones que satisfacen las restricciones. En nuestro caso, la preferencia entre las soluciones está determinada por aquella que presente menor sensibilidad total. Es decir, $f : X \to \mathbb{R} = S(R_1, R_2, R_3, C_4, C_5)$ y queremos minimizarla.

Como mencionamos en la sección 1.1, idealmente querríamos hallar una solución que minimice las tres sensibilidades simultáneamente, pero desconocemos si existe o no tal solución, por lo cual hemos definido la sensibilidad total S como la suma de los valores absolutos de cada una de las tres sensibilidades $S^{Q}_{R_i}$. Otra posibilidad consistiría en ver, dado un conjunto de soluciones, cuáles de ellas son *óptimos de Pareto* de acuerdo a las siguientes definiciones:

- Dados dos vectores $\boldsymbol{u} = (u_1, \ldots, u_k)$ y $\boldsymbol{v} = (v_1, \ldots, v_k)$, se dice que $\boldsymbol{u}$ *domina* a $\boldsymbol{v}$ si y sólo si $\langle \forall i : i \in 1, \ldots, k : u_i \leq v_i \rangle$.
- Dado un conjunto de vectores V, se dice que $\boldsymbol{v}^\star \in V$ es un *óptimo de Pareto* si y sólo si $\neg\langle \exists \boldsymbol{v} : \boldsymbol{v} \in V \wedge \boldsymbol{v} \neq \boldsymbol{v}^\star : \boldsymbol{v}$ domina a $\boldsymbol{v}^\star \rangle$.

En nuestro caso, dadas dos soluciones:

$$\boldsymbol{u} = (R^u_1, R^u_2, R^u_3, C^u_4, C^u_5)$$
$$\boldsymbol{v} = (R^v_1, R^v_2, R^v_3, C^v_4, C^v_5)$$

$\boldsymbol{u}$ domina a $\boldsymbol{v}$ si y sólo si

$$S_1(\boldsymbol{u}) \leq S_1(\boldsymbol{v}) \ \wedge \ S_2(\boldsymbol{u}) \leq S_2(\boldsymbol{v}) \ \wedge \ S_3(\boldsymbol{u}) \leq S_3(\boldsymbol{v}) \tag{14}$$

3. Algoritmo Exacto

En esta sección mostraremos un algoritmo exacto que encuentra todas las configuraciones que satisfacen las restricciones. La ventaja de contar con una solución de este tipo es que nos permitirá evaluar luego el rendimiento de los métodos heurísticos. Este algoritmo que desarrollamos pertenece al tipo de algoritmos exactos conocidos como de propagación de restricciones [5]. El algoritmo recorre el espacio de búsqueda propagando las restricciones de manera de filtrar parte del espacio de búsqueda. Primero filtramos las configuraciones que satisfacen la restricción sobre G. A las configuraciones que sobreviven las volvemos a filtrar aplicando la restricción sobre ω y finalmente aplicamos la restricción sobre Q. De este modo obtenemos todas las configuraciones que cumplen las restricciones. Finalmente ordenamos este conjunto de soluciones de acuerdo a la sensibilidad para obtener la configuración óptima.

Es claro que a partir de (2) se tiene que para cada valor de R_1 hay un conjunto de valores de R_2 que denotaremos $\{R_2\}_{R_1}$ tales que $G_+ > G > G_-$

Definiendo las siguientes frecuencias,

$$\begin{aligned} \omega_1 &= 1/(R_2C_4) \\ \omega_2 &= 1/(R_3C_5) \end{aligned} \tag{15}$$

podemos reescribir (3) de la siguiente forma: $\omega = \sqrt{\omega_1\omega_2}$

A partir de los valores que pueden tomar las resistencias y capacitores, es posible calcular la lista de valores que pueden tomar las frecuencias definidas en (15).

Para cada uno de los valores posibles para ω_1, habrá un conjunto de pares de valores (R_2, C_4) tales que $\omega_1 = 1/(R_2 C_4)$. Designamos a tal conjunto como $\{(R_2, C_4)\}_{\omega_1}$.

De manera análoga se puede proceder con ω_2, R_3 y C_5 (obsérvese que el conjunto de valores posibles para ω_2 es el mismo que para ω_1 y que $\{(R_2, C_4)\}_{\omega_1} = \{(R_3, C_5)\}_{\omega_2}$ cuando $\omega_1 = \omega_2$).

Luego, dada la restricción $\omega \in [\omega_-, \omega_+]$ y elegido un valor de ω_1, los valores admisibles de ω_2 son aquellos para los que $\omega = \sqrt{\omega_1 \omega_2}$ satisface dicha restricción, y por lo tanto cumplen las siguiente relación:

$$\frac{(\omega_+)^2}{\omega_1} > \omega_2 > \frac{(\omega_-)^2}{\omega_1}$$

De esta manera, para cada valor realizable de ω_1 existe un conjunto finito $\{\omega_2\}_{\omega_1}$ de valores de ω_2 tales que $\omega_+ > \omega > \omega_-$

Por otro lado, a partir de (4) y (15) podemos reescribir Q^{-1} de la siguiente forma,

$$Q^{-1} = \sqrt{\frac{\omega_1}{\omega_2}} \left(1 + \frac{R_2}{R_1} + \frac{R_2}{R_3}\right) \tag{16}$$

Finalmente, para cada par (R_1, R_2), (ω_1, ω_2) que satisface las restricciones sobre G y ω, utilizando la expresión (16) buscamos los valores R_3 admisibles que satisfagan la condición

$$(Q_-)^{-1} > Q^{-1} > (Q_+)^{-1}$$

y de esta manera obtenemos todas las soluciones de la forma (R_1, R_2, ω_1, ω_2, R_3) que luego podemos *mapear* a las soluciones de la forma (R_1, R_2, R_3, C_4, C_5).

A continuación mostramos el *pseudocódigo* del algoritmo de búsqueda exacta que acabamos de describir.

Algoritmo 1 Búsqueda exacta

R_{vals}: conjunto de valores que pueden tomar las resistencias
C_{vals}: conjunto de valores que pueden tomar los capacitores
G_-: valor mínimo aceptable para la ganancia
G_+: valor máximo aceptable para la ganancia
ω_-: valor mínimo aceptable para la frecuencia de polo
ω_+: valor máximo aceptable para la frecuencia de polo
Q_-: valor mínimo aceptable para el factor de calidad
Q_+: valor máximo aceptable para el factor de calidad

procedure ExhaustiveSearch(R_{vals}, C_{vals})
    soluciones $\leftarrow$ []
    $G_{constraint} \leftarrow$ []   ▷ Obtener pares $(r1, r2)$ que satisfagan la restricción sobre G
    for cada $r1$ en R_{vals}, cada $r2$ en R_{vals} **do**

$G \leftarrow r2/r1$
if $G_+ > G > G_-$ **then**
Agregar el par $(r1, r2)$ a $G_{constraint}$

$\omega_{constraint} \leftarrow \{\}$ ▷ Diccionario que para cada ω_1 contiene el conjunto de los ω_2 posibles para no violar la restricción
$wToRCMap \leftarrow \{\}$ ▷ Diccionario que para cada valor de w contiene una lista con los pares (r, c) que lo generan

$w_{vals} = []$
for cada r en R_{vals}, cada c en C_{vals} **do**
$w \leftarrow 1/(r * c)$
Agregar el par (r, c) a $wToRCMap[w]$
Agregar w a la lista w_{vals}
for cada w_1 en w_{vals} **do**
$Posibles\_\omega_2 \leftarrow []$
for cada w_2 en w_{vals} **do**
$minVal \leftarrow (\omega_-)^2/w_1$
$maxVal \leftarrow (\omega_+)^2/w_1$
if $maxVal > w2 > minVal$ **then**
Agregar $w2$ a la lista $Posibles\_\omega_2$
if $Posibles\_\omega_2$ no es vacía **then**
Agregar $Posibles\_\omega_2$ a $\omega_{constraint}[\omega_1]$

for cada par $(r1, r2)$ en $G_{constraint}$ **do**
for cada ω_1 en las keys de $\omega_{constraint}$ **do**
$Posibles\_\omega_2 \leftarrow \omega_{constraint}[\omega_1]$
$genera \leftarrow \exists (r, c) \in wToRCMap[\omega_1] \mid r = r_2$ ▷ Chequear si r_2 *genera* ω_1
if *genera* **then**
for cada ω_2 en $Posibles\_\omega_2$, cada r_3 en R_{vals} **do**
$genera \leftarrow \exists (r, c) \in wToRCMap[\omega_2] \mid r = r_3$ ▷ Chequear si r_3 *genera* ω_2
if *genera* **then**
$(Q^{-1})_{r_3} \leftarrow \sqrt{\omega_1/\omega_2} * (1 + r_2/r_1 + r_2/r_3)$
if $(Q_-)^{-1} > (Q^{-1})_{r_3} > (Q_+)^{-1}$ **then**
$c_4 \leftarrow 1/(r_2 * \omega_1)$
$c_5 \leftarrow 1/(r_3 * \omega_2)$
Agregar la tupla $(r_1, r_2, r3, c4, c5)$ a la lista *soluciones*
return *soluciones*

La ejecución del código *Python* que implementa este algoritmo nos permite encontrar todas las soluciones que satisfacen las restricciones sobre G, ω y Q así como también el subconjunto de éstas que son óptimos de Pareto.

En el apéndice A.1 mostramos en una tabla estas soluciones numeradas por un índice n en orden creciente de la sensibilidad total S. Por razones de espacio y legibilidad las sensibilidades individuales se representan en ese mismo apéndice en un gráfico en lugar de incorporarlas a la tabla.

En el apéndice A.2 mostramos el subconjunto de soluciones del apéndice A.1 que son óptimos de Pareto.

4. Solución Analítica

En esta sección presentaremos una solución analítica del problema que permite encontrar todas las soluciones del problema. Al igual que la solución exacta, esta solución es una herramienta útil a la hora de evaluar el rendimiento de los métodos heurísticos. El problema consiste en minimizar una función -la sensibilidad total S- de cinco variables -las tres resistencias y los dos capacitores-, que están sujetas a tres restricciones,

$$\begin{aligned} \underset{R_i, C_j}{\text{minimize}} \quad & S(R_i, C_j) \\ \text{subject to} \quad & G(R_i, C_j) \in [G_-, G_+], \\ & \omega(R_i, C_j) \in [\omega_-, \omega_+], \\ & Q(R_i, C_j) \in [Q_-, Q_+], \\ & R_i \in D(R_i), \\ & C_j \in D(C_j) \end{aligned} \tag{17}$$

con $i = 1, 2, 3$, $j = 4, 5$, $D(R_i)$ y $D(C_j)$ son los dominios discretos definidos por las series industriales utilizadas y siendo

$$G(R_i, C_j) = R_2/R_1 \tag{18}$$

$$\omega(R_i, C_j) = \frac{1}{\sqrt{R_2 C_4 R_3 C_5}} \tag{19}$$

$$Q(R_i, C_j) = \sqrt{\frac{C_4}{C_5}} \left\{ \frac{\sqrt{R_2 R_3}}{R_1} + \sqrt{\frac{R_3}{R_2}} + \sqrt{\frac{R_2}{R_3}} \right\}^{-1} \tag{20}$$

$$\begin{aligned} S(R_i, C_j) = \frac{Q}{2} \sqrt{\frac{C_5}{C_4}} \Bigg\{ 2 \left| \frac{\sqrt{R_2 R_3}}{R_1} \right| + \left| \frac{\sqrt{R_2 R_3}}{R_1} - \sqrt{\frac{R_3}{R_2}} + \sqrt{\frac{R_2}{R_3}} \right| + \\ \left| \frac{\sqrt{R_2 R_3}}{R_1} + \sqrt{\frac{R_3}{R_2}} - \sqrt{\frac{R_2}{R_3}} \right| \Bigg\} \end{aligned} \tag{21}$$

En el problema (17) las restricciones están todas especificadas en base a un mismo error admisible respecto al valor objetivo de cada variable de acuerdo a como fue especificado en (13).

Para simplificar la discusión por ahora consideremos el problema (17) con variables continuas en lugar de discretas lo que permite considerar el problema de optimización como uno con restricciones de igualdad (si no considerásemos variables continuas, el problema con restricciones de igualdad en general no tiene solución),

$$\begin{aligned}
\underset{R_i, C_j}{\text{minimize}} \quad & S(R_i, C_j) \\
\text{subject to} \quad & G(R_i, C_j) = G_0, \\
& \omega(R_i, C_j) = \omega_0, \\
& Q(R_i, C_j) = Q_0, \\
& R_i \in I_R, \\
& C_j \in I_C
\end{aligned} \tag{22}$$

donde I_R e I_C son los intervalos reales determinados por los extremos de los dominios $D(R_i)$ y $D(C_j)$ respectivamente.

A partir de (20) y (21) es fácil ver que la sensibilidad depende sólo de los valores de las resistencias,

$$\begin{aligned}
S(R_1, R_2, R_3) = \frac{1}{2}&\left\{\frac{\sqrt{R_2 R_3}}{R_1} + \sqrt{\frac{R_3}{R_2}} + \sqrt{\frac{R_2}{R_3}}\right\}^{-1} \times \\
&\left\{2\left|\frac{\sqrt{R_2 R_3}}{R_1}\right| + \left|\frac{\sqrt{R_2 R_3}}{R_1} - \sqrt{\frac{R_3}{R_2}} + \sqrt{\frac{R_2}{R_3}}\right| + \right. \\
&\left.\left|\frac{\sqrt{R_2 R_3}}{R_1} + \sqrt{\frac{R_3}{R_2}} - \sqrt{\frac{R_2}{R_3}}\right|\right\}
\end{aligned} \tag{23}$$

Pero de (23) se puede ver que la sensibilidad depende de los cocientes entre pares de resistencias, por lo que siendo tres las resistencias, dependerá sólo de dos cocientes independientes. Tomemos tales cocientes los siguientes,

$$\begin{aligned}
a &= R_1/R_2 \\
b &= R_1/R_3
\end{aligned} \tag{24}$$

En función de estas dos nuevas variables, la sensibilidad toma la forma,

$$S(a, b) = \frac{1}{2}\frac{\{2 + |1 - a + b| + |1 + a - b|\}}{(1 + a + b)} \tag{25}$$

Teniendo en cuenta que en función de las nuevas variables a y b la ganancia es simplemente $G(a) = 1/a$, el problema (22) queda reducido a,

$$\begin{aligned}
\underset{a, b}{\text{minimize}} \quad & S(a, b) \\
\text{subject to} \quad & a = a_0, \\
& \omega(R_i, C_j) = \omega_0, \\
& Q(R_i, C_j) = Q_0
\end{aligned} \tag{26}$$

donde definimos $a_0 = 1/G_0$. Teniendo en cuenta entonces que la restricción sobre la ganancia no hace más que fijar el valor de a en a_0, resulta entonces que la sensibilidad es función sólo de b. La minimización de S es entonces un problema de una sola variable. Para simplificar la notación definamos $d = 1 + a$ con lo que la sensibilidad queda,

$$S(b) = \frac{1}{2} \frac{\{2 + |2 - d + b| + |d - b|\}}{(d + b)} \tag{27}$$

La presencia de la función valor absoluto introduce dos puntos en la función $S(b)$ en los que su derivada es discontinua: $b = d - 2$ y $b = d$. Entonces el o los mínimos de S estarán en los puntos donde la derivada sea nula, en los puntos de discontinuidad de la derivada o en los extremos del dominio. Si suponemos que la ganancia es mayor que 1, entonces $a < 1$ y por lo tanto $d < 2$ con lo que el punto de discontinuidad de la derivada $b = d - 2$ no es de interés ya que es negativo y b sólo puede tomar valores positivos. Un cálculo explícito permite mostrar que no existe ningún punto en el que la derivada se anule (para valores positivos de b). Respecto a los extremos del dominio, si bien la función está definida para $b \in (-\infty, +\infty)$, los valores negativos de b no son de interés. Consideramos entonces la función $S(b)$ definida en $(0, +\infty)$ (aunque estrictamente hablando, 0 y $+\infty$ tampoco son valores realizables). A partir de (27) obtenemos $S(0) = 2/d > 1$, $\lim_{b \to +\infty} S(b) = 1$ y el valor de S en el punto $b_0 = d = 1 + a_0$ de discontinuidad de la derivada es $S(b_0) = 1/d < 1$. Por lo tanto

$$\min S(b) = S(d) = 1/d \tag{28}$$

siendo $S(0)$ el máximo de la función (en el intervalo $(0, +\infty)$). En la figura (2) mostramos el gráfico de la función $S(a, b)$ para $a = 1/3$.

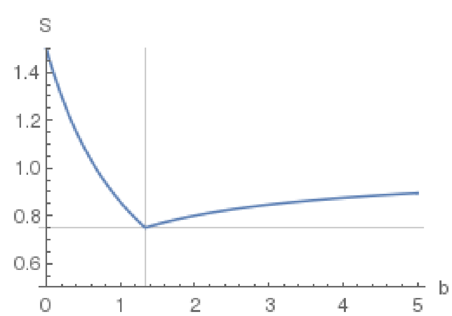


Figura 2: Sensibilidad total S para $a = 1/3$ en función de b para $b > 0$. En este dominio el mínimo se encuentra en $b = 1 + a = 4/3$ siendo su valor $S_{min} = 3/4$.

Resumiendo lo hecho hasta ahora, el problema se redujo al siguiente sistema de ecuaciones,

$$R_1/R_2 = a_0 \tag{29}$$

$$R_1/R_3 = 1 + R_1/R_2 \tag{30}$$

$$\omega(R_2, R_3, C_4, C_5) = \omega_0 \tag{31}$$

$$Q(R_1, R_2, R_3, C_4, C_5) = Q_0 \tag{32}$$

Hemos llegado entonces a un sistema de 4 ecuaciones con 5 incógnitas, por lo tanto una variable es independiente. Elegimos R_1 como variable independiente. Luego, las ecuaciones (42) y (42) permiten obtener R_2 y R_3 en función del valor elegido de R_1. Las últimas dos ecuaciones, estando ya determinados los valores de las tres resistencias, forman un sistema de dos ecuaciones con dos incógnitas (C_4 y C_5) que permiten determinar los valores de los capacitores.

En síntesis, fijamos el valor de R_1. Luego de (42) obtenemos R_2 y con (42) R_3. Luego, de (42) y (42) obtenemos C_4 y C_5. El resultado final es,

$$\begin{aligned} R_2 &= G_0 R_1 \\ R_3 &= \frac{R_1 R_2}{R_1 + R_2} \\ C_4 &= \frac{Q_0 D}{\omega_0 E} \\ C_5 &= \frac{DE}{\omega_0 Q_0} \end{aligned} \tag{33}$$

donde hemos definido,

$$\begin{aligned} D &= 1/\sqrt{R_2 R_3} \\ E &= \left\{ \frac{\sqrt{R_2 R_3}}{R_1} + \sqrt{\frac{R_3}{R_2}} + \sqrt{\frac{R_2}{R_3}} \right\}^{-1} \end{aligned} \tag{34}$$

A modo de ejemplo mostramos en el cuadro 1 el resultado de utilizar (33) con los valores de R_1 del escenario 2. Luego de calcular el conjunto de valores $(R_1, R_2, R_3, C_4, C_5)$ para cada valor de R_1, descartamos todas la tuplas en las que alguno de sus cinco elementos se salen del rango correspondiente al escenario que establecimos al final de la sección 1. Para todas las soluciones la sensibilidad total es la mínima (0.75). Obsérvese que de la ecuación (27) se pueden identificar las contribuciones de cada una de las sensibilidades S_i a la sensibilidad total. Teniendo en cuenta que en el mínimo de S es $b = d = 1 + a_0 = 4/3$ resultan para todas las soluciones de ese cuadro los siguientes valores para las sensibilidades: $S_1 = S_2 = 0{,}375$ y $S_3 = 0$.

Podemos intentar utilizar las soluciones del cuadro 1 para hallar soluciones para el caso en que las variables del circuito tienen su dominio en las series discretas industriales en lugar de los números reales. Una opción es hallar para cada solución del cuadro 1 la configuración discreta más próxima (cuando alguno

R_1	R_2	R_3	C_4	C_5
1500	4500	1125	2,00 10^{-7}	2,50 10^{-8}
1600	4800	1200	1,87 10^{-7}	2,34 10^{-8}
1800	5400	1350	1,66 10^{-7}	2,08 10^{-8}
2000	6000	1500	1,50 10^{-7}	1,87 10^{-8}
2200	6600	1650	1,36 10^{-7}	1,70 10^{-8}
2400	7200	1800	1,25 10^{-7}	1,56 10^{-8}
2700	8100	2025	1,11 10^{-7}	1,38 10^{-8}
3000	9000	2250	1,00 10^{-7}	1,25 10^{-8}
3300	9900	2475	9,09 10^{-8}	1,13 10^{-8}
3600	10800	2700	8,33 10^{-8}	1,04 10^{-8}
3900	11700	2925	7,69 10^{-8}	9,61 10^{-9}
4300	12900	3225	6,97 10^{-8}	8,72 10^{-9}
4700	14100	3525	6,38 10^{-8}	7,98 10^{-9}
5100	15300	3825	5,88 10^{-8}	7,35 10^{-9}
5600	16800	4200	5,35 10^{-8}	6,69 10^{-9}
6200	18600	4650	4,84 10^{-8}	6,05 10^{-9}
6800	20400	5100	4,41 10^{-8}	5,51 10^{-9}

R_1	R_2	R_3	C_4	C_5
7500	22500	5625	4,00 10^{-8}	5,00 10^{-9}
8200	24600	6150	3,29 10^{-8}	4,12 10^{-9}
9100	27000	6800	3,30 10^{-8}	3,90 10^{-9}
10000	30000	7500	3,00 10^{-8}	3,75 10^{-9}
11000	33000	8250	2,72 10^{-8}	3,41 10^{-9}
12000	36000	9000	2,50 10^{-8}	3,12 10^{-9}
13000	39000	9750	2,30 10^{-8}	2,88 10^{-9}
15000	45000	11250	2,00 10^{-8}	2,50 10^{-9}
16000	48000	12000	1,87 10^{-8}	2,34 10^{-9}
18000	54000	13500	1,66 10^{-8}	2,08 10^{-9}
20000	60000	15000	1,50 10^{-8}	1,87 10^{-9}
22000	66000	16500	1,36 10^{-8}	1,70 10^{-9}
24000	72000	18000	1,25 10^{-8}	1,56 10^{-9}
27000	81000	20250	1,11 10^{-8}	1,38 10^{-9}
30000	90000	22500	1,00 10^{-8}	1,25 10^{-9}
33000	99000	24750	9,09 10^{-9}	1,13 10^{-9}
36000	108000	27000	8,33 10^{-9}	1,04 10^{-9}

Cuadro 1: Resultados obtenidos al aplicar las ecuaciones (33) para los valores de R_1 del escenario 2.

de los valores de algún parámetro del cuadro 1 es equidistante a dos valores permitidos de la serie discreta correspondiente, tomamos como más cercano el de la izquierda) Luego, esta solución se preserva o se desecha si cumple o no con las restricciones sobre G, ω y Q. Procediendo de este modo sólo sobreviven las tres soluciones que mostramos en el cuadro 2. Comparando con la solución exacta que se muestra en el apéndice A.1, vemos que las soluciones del cuadro 2 son precisamente las tres mejores soluciones del caso discreto que se obtuvieron por el método exacto.

Observemos que esta última metodología propuesta mueve el problema discreto original con restricciones de desigualdad a uno de variable real con restricciones de igualdad. Una vez resuelto este problema se aproxima la solución obtenida a valores discretos de la serie E retornando a las restricciones de desigualdad. Esta estrategia muestra buenos resultados al compararlos con los brindados por el método exacto.

Podemos todavía utilizar la solución analítica de otra manera para obtener más soluciones discretas si procedemos de la siguiente manera. Para cada valor de R_1 de la serie industrial buscamos los valores de esa serie para R_2 que satisfacen la restricción sobre G (en nuestro caso, si existe este valor es único debido la tolerancia pequeña admitida). Luego, mediante la segunda de las ecuaciones (33) calculamos un valor de R_3 que en general es un número real que no pertenece a la serie industrial pero que es el que minimiza la sensibilidad para los

R_1	R_2	R_3	C_4	C_5
3600	11000	2700	$8{,}2\ 10^{-8}$	$1{,}0\ 10^{-8}$
11000	33000	8200	$2{,}7\ 10^{-8}$	$3{,}3\ 10^{-9}$
36000	110000	27000	$8{,}2\ 10^{-9}$	$1{,}0\ 10^{-9}$

Cuadro 2: Resultados discretos más cercanos a los del cuadro 1 (y que satisfacen las tres restricciones) para el escenario 2 teniendo en cuenta un solo valor de R_3 por cada par (R_1,R_2)

valores elegidos de R_1 y R_2. Elegimos entonces para R_3 el valor de la serie más próximo a este calculado. Ahora, con la terna de valores (R_1, R_2, R_3) de la serie discreta utilizamos las últimas dos restricciones para obtener todos los valores de capacitores que las satisfacen. Una manera simple de hacer esto es calcular, dados R_1, R_2 y R_3, los valores máximos y mínimos de C_4 y C_5 que satisfacen las restricciones para ω y Q. De las últimas dos ecuaciones (33) obtenemos:

$$\begin{aligned} C_4^{min} &= \frac{Q_- D}{\omega_+ E} \\ C_4^{max} &= \frac{Q_+ D}{\omega_- E} \\ C_5^{min} &= \frac{DE}{\omega_+ Q_+} \\ C_5^{max} &= \frac{DE}{\omega_- Q_-} \end{aligned} \tag{35}$$

Luego basta con elegir todos los C_4 de la serie discreta que caen en el rango (C_4^{min}, C_4^{max}). Análogamente con C_5. Por este procedimiento obtenemos las mismas soluciones del cuadro 2. Podemos obtener más soluciones si en lugar de tomar sólo el valor más próximo de la serie discreta al valor calculado de R_3, incluimos además los vecinos a izquierda y derecha. De esta manera se obtienen 12 soluciones adicionales, que mostramos en el cuadro 3. Agregando más vecinos se obtienen cada vez más soluciones, eventualmente hasta obtener todas las soluciones que se obtuvieron por el método exacto. Es evidente que cuando se consideran todos los valores posibles de R_3 para cada par (R_1,R_2) este método se convierte en exacto.

5. Optimización por Colonia de Hormigas

La Optimización por Colonia de Hormigas -en inglés *Ant Colony Optimization*- es una metaheurística introducida en los años '90 inspirada en el comportamiento de las hormigas para solucionar problemas de optimización combinatoria [7,8,10].

Al buscar comida, las hormigas inicialmente exploran el área circundante a su nido de manera aleatoria. En cuanto una hormiga encuentra una fuente de

R_1	R_2	R_3	C_4	C_5
1600	4700	1300	$1{,}8\ 10^{-7}$	$2{,}2\ 10^{-8}$
2700	8200	1800	$1{,}2\ 10^{-7}$	$1{,}5\ 10^{-8}$
3000	9100	2400	$1{,}0\ 10^{-7}$	$1{,}2\ 10^{-8}$
3300	10000	2200	$1{,}0\ 10^{-7}$	$1{,}2\ 10^{-8}$
3600	11000	2700	$8{,}2\ 10^{-8}$	$1{,}0\ 10^{-8}$
4300	13000	3600	$6{,}8\ 10^{-8}$	$8{,}2\ 10^{-9}$
5100	15000	4300	$5{,}6\ 10^{-8}$	$6{,}8\ 10^{-9}$
11000	33000	8200	$2{,}7\ 10^{-8}$	$3{,}3\ 10^{-9}$
12000	36000	8200	$2{,}7\ 10^{-8}$	$3{,}3\ 10^{-9}$
13000	39000	11000	$2{,}2\ 10^{-8}$	$2{,}7\ 10^{-9}$
16000	47000	13000	$1{,}8\ 10^{-8}$	$2{,}2\ 10^{-9}$
27000	82000	18000	$1{,}2\ 10^{-8}$	$1{,}5\ 10^{-9}$
30000	91000	24000	$1{,}0\ 10^{-8}$	$1{,}2\ 10^{-9}$
33000	100000	22000	$1{,}0\ 10^{-8}$	$1{,}2\ 10^{-9}$
36000	110000	27000	$8{,}2\ 10^{-9}$	$1{,}0\ 10^{-9}$

Cuadro 3: Resultados discretos más cercanos a los del cuadro 1 (y que satisfacen las tres restricciones) para el escenario 2 teniendo en cuenta tres valores de R_3 por cada par (R_1,R_2)

comida, la evalúa y lleva un poco hacia el nido. Durante el viaje de vuelta, la hormiga va depositando feromona en el suelo. La cantidad de feromona depositada depende de la cantidad y calidad de comida, y sirve para guiar otras hormigas hacia la fuente de comida. Esta forma de comunicación indirecta a través de la feromona les permite hallar los caminos más cortos entre el nido y la fuente de comida [14]. Este comportamiento ha inspirado la definición de colonias artificiales de hormigas que pueden encontrar soluciones aproximadas -en el sentido que son buenas soluciones pero no necesariamente óptimas- a problemas de optimización combinatorios complejos.

5.1. La metaheurística ACO

Recordemos de la sección 2.2 que en un CSP (X, D, C) cada variable x_i del problema en X, tiene un dominio $D(x_i)$ indicando los valores que puede tomar dicha variable. Definimos una *componente de solución* c_{ij} a la asignación $x_i = v_j$ donde $v_j \in D(x_i)$. Y definimos también C_s como el conjunto de todas las componentes de solución posibles.

En el algoritmo 2 se presenta la metaheurística ACO, que consiste de tres actividades, que se detallan a continuación.

Algoritmo 2 Metaheurística ACO

while condiciones de terminación no alcanzadas **do**
    ConstruirSolución()
    ActualizarFeromonas()
    AccionesDaemon() ▷ Opcional

ConstruirSolución(): Las hormigas artificiales construyen soluciones creando una secuencia de componentes elegidas de un conjunto finito de valores posibles para cada componente.

La construcción de esta secuencia comienza con una solución parcial vacía $s^p = \emptyset$. Luego, en cada paso de la construcción, s^p se extiende agregando una componente perteneciente al conjunto $N(s^p) \in C_s \setminus s^p$, que está definido por el mecanismo de construcción de la solución.

El proceso de construir una solución puede ser considerado como un camino dentro de un grafo de construcción $G_C = (V, E)$ donde el conjunto de componentes de solución C_s está asociado o bien con el conjunto de vértices V del grafo, o bien con el de sus aristas E. Los caminos permitidos en G_C están definidos implícitamente por el mecanismo de construcción de la solución que define el conjunto $N(s^p)$ respecto a la solución parcial s^p.

La elección de una componente de solución del conjunto $N(s^p)$ se realiza de forma probabilística en cada paso de la construcción. Las reglas exactas para esta elección varían entre cada una de las diferentes variantes de ACO. Una de las más conocidas es la que utiliza *Ant System* [8]:

$$p(c_{ij} \mid s^p) = \frac{\tau_{ij}^{\alpha} \cdot \eta(c_{ij})^{\beta}}{\sum_{c_{il} \in N(s^p)} \tau_{il}^{\alpha} \cdot \eta(c_{il})^{\beta}}, \forall c_{ij} \in N(s^p) \tag{36}$$

donde τ_{ij} es la cantidad de feromona asociada a la componente c_{ij} y $\eta(\cdot)$ es una función de ponderación que en cada paso de la construcción le asigna un valor heurístico a cada componente $c_{ij} \in N(s^p)$. Los valores que devuelve la función de ponderación suelen llamarse *información heurística*. α y β son parámetros con valores positivos que determinan la relación entre la información obtenida de las feromonas y la heurística.

ActualizarFeromonas(): El objetivo de la actualización de feromonas es incrementar el valor asociado con soluciones buenas o promisorias, y decrementar el valor para soluciones malas. Usualmente esto se consigue aumentando los niveles de feromona asociados a una buena solución escogida s_{ch} en un determinado valor $\Delta\tau$ y reduciendo todos los valores de feromona a través del mecanismo de *evaporación de feromonas*:

$$\tau_{ij} \leftarrow \begin{cases} (1-\rho)\tau_{ij} + \rho\Delta\tau, & \text{si } \tau_{ij} \in s_{ch} \\ (1-\rho)\tau_{ij}, & \text{caso contrario} \end{cases}$$

donde $\rho \in (0, 1]$ es la *tasa de evaporación*. Este mecanismo es necesario para evitar una convergencia demasiado rápida del algoritmo. Implementa una manera útil de *olvidar*, para favorecer la exploración de nuevas áreas del espacio de búsqueda.

En general, las soluciones buenas halladas tempranamente por las hormigas se utilizan para actualizar las feromonas de forma tal que se aumente la probabilidad de búsqueda por hormigas que las siguen en las áreas promisorias del espacio de búsqueda. Cada variante de ACO suele implementar distintas formas de actualización de feromonas. En principio, pueden escoger entre dos estrategias: basándose en la mejor solución encontrada en la última iteración, o bien en la mejor solución encontrada hasta el momento, desde que el algoritmo comenzó a ejecutarse. La primera favorece una mayor exploración del espacio de búsqueda, mientras que la segunda lleva a una convergencia más rápida [9].

AccionesDaemon(): Son acciones que pueden ser usadas para implementar acciones centralizadas que las hormigas no puedan realizar individualmente. Por ejemplo, aplicar búsqueda local a las soluciones encontradas, o recolectar información global que puede ser utilizada para decidir si es útil o no depositar feromona adicional para influir el proceso de búsqueda desde una perspectiva no local.

5.2. ACO en dominios continuos: $\mathbf{ACO}_{\mathbb{R}}$

Dado un CSP (X, D, C) diremos que el mismo tiene *dominio continuo* si $D_i \subseteq \mathbb{R}, \forall x_i \in X$.

La idea central a la forma en la que ACO trabaja es la construcción incremental de soluciones basadas en la selección probabilista -influenciada por la feromona- de componentes de la solución. Cuando se aplica ACO a problemas de optimización combinatorios, el conjunto de las componentes de solución está definido por la formulación del problema. En cada paso de la construcción, la hormiga escoge de manera probabilística una componente $c_{ij} \in N(s^p)$ de acuerdo a la ecuación (36). Las probabilidades asociadas con los elementos del conjunto $N(s^p)$ de componentes disponibles forman una distribución de probabilidad *discreta* que cada hormiga muestrea para escoger la componente a agregar a la solución parcial s^p.

En $\mathrm{ACO}_{\mathbb{R}}$, la idea fundamental es pasar de utilizar esta distribución discreta a utilizar una *continua*, es decir, una función de densidad de probabilidad.

5.2.1. ***La función de densidad de probabilidad*** Una *función de densidad de probabilidad* (FDP) puede ser cualquier función $P(x) \geq 0, \forall x$ tal que

$$\int_{-\infty}^{\infty} P(x)\, dx = 1$$

Para una FDP $P(x)$ dada, se puede definir una *función de distribución acumulada* (FDA) $F(x)$ que es útil para muestrear la correspondiente FDP. La FDA $F(x)$ asociada a la FDP $P(x)$ se define como sigue:

$$F(x) = \int_{-\infty}^{x} P(t)\, dt$$

Para muestrear la FDP $P(x)$ suele utilizarse la inversa de su FDA, $F^{-1}(x)$. Basta generar números reales *pseudo aleatorios* uniformemente distribuidos. Sin embargo, es importante notar que para una FDP arbitraria $P(x)$ no siempre es trivial hallar la inversa de su FDA, $F^{-1}(x)$.

Una de las funciones más populares para utilizar como FDP es la función gaussiana. Tiene algunas ventajas como por ejemplo que es relativamente fácil de muestrear, sin embargo tiene también desventajas: una función gaussiana sola no sirve para describir una situación donde dos áreas disjuntas del espacio de búsqueda son promisorias, ya que tiene un único máximo. Debido a esto, $\mathrm{ACO}_{\mathbb{R}}$ utiliza una FDP basada en funciones gaussianas pero ligeramente mejoradas: un *núcleo* gaussiano.

Un núcleo gaussiano $G^i(x)$ se define como la suma ponderada de varias funciones gaussianas unidimensionales $g_l^i(x)$

$$G^i(x) = \sum_{l=1}^{k} w_l \cdot g_l^i(x) = \sum_{l=1}^{k} w_l \cdot \frac{1}{\sigma_l^i \cdot \sqrt{2\pi}} \mathrm{e}^{-\frac{(x-\mu_l^i)^2}{2{\sigma_l^i}^2}} \tag{37}$$

$\mathrm{ACO}_{\mathbb{R}}$ utiliza tantos núcleos como variables tiene el CSP, es decir si $n = |X|$, se usan n núcleos $G^i, i = 1, \ldots, n$. Como se observa en la ecuación (37), el núcleo $G^i(x)$ está parametrizado con tres vectores de parámetros: w es el vector de pesos asociados con cada función gaussiana individual, μ^i es el vector de las medias y σ^i el de las desviaciones estándar. La cardinalidad de cada uno de estos vectores es igual al número de funciones gaussianas que componen el núcleo. Por conveniencia, lo denotaremos con k, es decir $|w| = |\mu^i| = |\sigma^i| = k$

Al utilizar un núcleo como FDP seguimos manteniendo la facilidad de muestreo pero obtenemos una flexibilidad en cuanto a la forma que puede tomar si lo comparamos con una función gaussiana individual. La figura (3) muestra un ejemplo de la forma que puede tomar la FDP de un núcleo gaussiano.

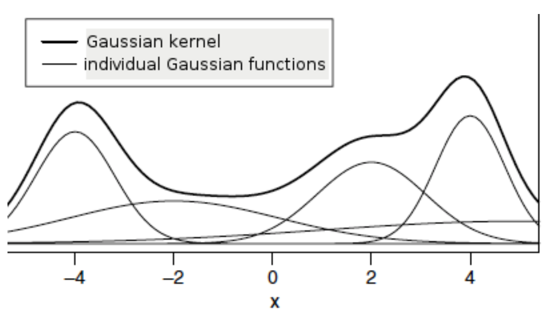


Figura 3: Ejemplo de cinco funciones gaussianas y su superposición que es el núcleo resultante. Extraída de [6]

Al usar ACO en problemas de optimización combinatoria, en cada iteración, al elegir una componente para ser agregada a la solución (de acuerdo a la ecuación (36)), una hormiga usa la información de las feromonas -almacenada como tabla- como una distribución de probabilidad discreta. En el caso continuo, la elección de la hormiga no está restringida a un conjunto finito. Por lo tanto, la información de las feromonas no puede almacenarse en una tabla y debe adoptarse un enfoque diferente.

En $\text{ACO}_{\mathbb{R}}$, se mantiene un registro de un número de soluciones dentro de un *archivo* T. Para cada solución s_l de un problema de n dimensiones, $\text{ACO}_{\mathbb{R}}$ almacena en T los valores de las n variables y el valor de la función objetivo $f(s_l)$. De esta forma, la i-ésima variable de la l-ésima solución está denotada por s_l^i.

La información almacenada en T se utiliza luego para ir generando funciones de densidad de probabilidad dinámicamente. Como se indicó en la ecuación (37), el núcleo gaussiano G^i está parametrizado por tres vectores w, μ y σ, cada uno con la misma cardinalidad. Las soluciones en el archivo T se usan para calcular los valores de estos tres parámetros y por consiguiente darle forma a la FDP del núcleo gaussiano usado para guiar a las hormigas en el proceso de búsqueda. El número de soluciones almacenadas en el archivo T será k, la cardinalidad de cada uno de los vectores, y determina la complejidad de la FDP: habrá k gaussianas individuales conformando el núcleo.

Para cada variable del problema $i = 1, \ldots, n$, hay una FDP definida por el núcleo G^i. Para cada uno de estos núcleos G^i, el vector μ^i se compone de los valores de la i-ésima variable de cada una de las soluciones en el archivo:

$$\mu^i = \{s_1^i, \ldots, s_k^i\} \tag{38}$$

El vector de pesos w, se crea de la siguiente forma: cada solución que se agrega al archivo T se evalúa y puntúa (en caso de empate se define aleatoriamente). Las soluciones luego son ordenadas de acuerdo a su puntuación, es decir, la solución s_l está l-ésima en el ranking. En el caso de un problema de minimización de una función f como el que estamos tratando, la puntuación de la solución s_i está dada por $f(s_i)$ y, por lo tanto, será mejor mientras más bajo sea ese valor. El peso de la solución s_l se calcula con la siguiente fórmula:

$$\omega_l = \frac{1}{qk\sqrt{2\pi}} e^{-\frac{(l-1)^2}{2q^2k^2}} \tag{39}$$

lo cual define que el peso es el valor de una gaussiana con argumento l, media 1 y desviación estándar qk, donde q es un parámetro del algoritmo. Cuando q es pequeño, las soluciones mejor puntuadas son fuertemente preferidas, mientras que si es grande el algoritmo se vuelve más uniforme. La influencia de este parámetro en $\text{ACO}_{\mathbb{R}}$ es similar a ajustar el balance entre usar las estrategias de usar la mejor solución de la iteración o la mejor hallada hasta el momento para actualizar las feromonas en ACO. La figura (4) muestra la estructura del archivo T, el vector w y los núcleos gaussianos.

Para terminar de definir el núcleo G^i, resta determinar el vector de desviaciones estándar σ^i, lo cual haremos en la siguiente sección.

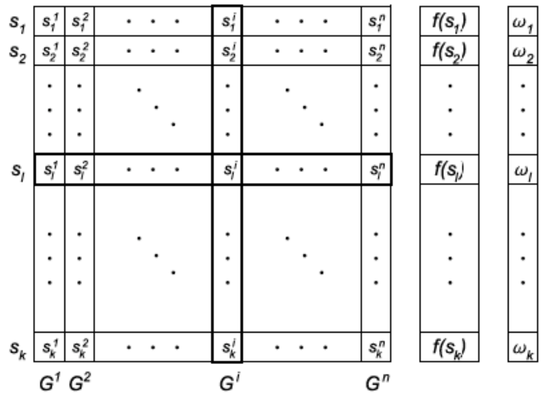


Figura 4: El archivo de soluciones utilizado por $\text{ACO}_{\mathbb{R}}$. Las k soluciones son ordenadas de acuerdo a su calidad. Es decir, para un problema de minimización, $f(s_1) \leq f(s_2) \leq \cdots \leq f(s_k)$. Cada solución tiene asociado un peso proporcional a la calidad de la solución, es decir $w_1 \geq w_2 \geq \cdots \geq w_k$. La FDP G^i se construye utilizando solamente los i-ésimos valores de cada una de las k soluciones.

5.2.2. *Implementación algorítmica de $ACO_{\mathbb{R}}$* En esta sección describiremos las actividades referidas en el algoritmo 2 pero aplicadas a $\text{ACO}_{\mathbb{R}}$.

ConstruirSolución(): Dadas las variables $x_i, i = 1, \ldots, n$, una hormiga construye una solución realizando n pasos de construcción. En el paso i, la hormiga escoge un valor para la variable x_i. Como se mencionó en la sección anterior, la FDP dada por el núcleo gaussiano está compuesta de un número dado de funciones gaussianas individuales. Dicho número es igual al tamaño k del archivo T. En el paso i, sólo la información de la variable x_i es utilizada. De este modo, en cada paso el núcleo G^i utilizado es diferente. De acuerdo a la ecuación (37), para poder definir G^i los vectores w, μ^i y σ^i deben estar definidos.En la sección anterior se explicó cómo crear w y μ^i; el cálculo de σ^i es bastante más complejo. Antes de presentar cómo realizarlo, es conveniente explicar la implementación de la ecuación (37).

En la práctica, el proceso de muestreo se realiza como sigue. Primero, los elementos del vector de pesos w se computan de acuerdo a la ecuación (39). Luego el proceso sigue en dos fases: en la primera se escoge alguna de las funciones gaussianas que componen el núcleo. La probabilidad p_l de escoger la l-ésima gaussiana está dada por:

$$p_l = \frac{w_l}{\sum_{r=1}^{k} w_r}$$

La segunda fase consiste en muestrear la gaussiana escogida (*i.e*, en el paso i la función g_l^i). Esto puede hacerse utilizando un generador de números aleatorios capaz de generar números aleatorios de acuerdo a una distribución normal parametrizada, o utilizando un generador aleatorio uniforme junto con, por ejemplo, el método de Box-Müller. Este muestreo de dos fases equivale a muestrear el núcleo G^i definido en la ecuación (37).

En el paso i, la desviación estándar necesita ser conocida únicamente para la gaussiana individual $g_l^i(x)$ escogida en la fase uno. Por lo tanto, no es necesario computar el vector completo σ^i, sino solamente la σ_l^i requerida.

Para establecer el valor de la desviación estándar σ_l^i en el paso i, calculamos la distancia promedio de la solución escogida s_l a las otras soluciones dentro del archivo T, y las multiplicamos por un parámetro ξ:

$$\sigma_l^i = \xi \sum_{e=1}^{k} \frac{|s_e^i - s_l^i|}{k-1}$$

El parámetro $\xi > 0$, que es el mismo para todas las variables, tiene un efecto similar al de la tasa de evaporación en ACO. Mientras más alto sea su valor, más baja será la velocidad de convergencia del algoritmo.

Mientras que en ACO la tasa de evaporación afecta la memoria de largo plazo -*i.e*, las peores soluciones son olvidadas más rápido-, ξ en $\text{ACO}_{\mathbb{R}}$ afecta el modo en que la memoria de largo plazo es usada -*i.e*, la búsqueda está menos influenciada hacia los puntos en el espacio de búsqueda que ya han sido explorados (y son mantenidos en el archivo T)-.

ActualizarFeromonas(): Como se mencionó previamente, la información de feromonas en $\text{ACO}_{\mathbb{R}}$ se mantiene en el archivo de soluciones. Esto implica que el proceso de actualización de feromonas debe realizar alguna forma de actualización de dicho archivo.

El tamaño k del archivo T es un parámetro del algoritmo. Sin embargo, k no puede ser menor que el número de variables del problema que se intenta resolver [6].

Al comenzar el algoritmo, el archivo de soluciones T se inicializa generando k soluciones mediante un muestreo aleatorio uniforme. La actualización de feromonas se consigue agregando el conjunto de nuevas soluciones generadas al archivo T y luego eliminando el mismo número de peores soluciones de forma que el tamaño de T no cambie. Este proceso garantiza que sólo las mejores soluciones son mantenidas en el archivo, guiando efectivamente a las hormigas en el proceso de búsqueda.

AccionesDaemon(): Se va actualizando la mejor solución encontrada hasta el momento de forma que pueda ser devuelta una vez que se alcanzan las condiciones de terminación. Se podrían aplicar heurísticas de búsqueda local para mejorar la *rendimiento* del algoritmo.

6. $\text{ACO}_{\mathbb{R}}$ Aplicado al Problema del Filtro

En esta sección detallamos la implementación de la metaheurística $\text{ACO}_{\mathbb{R}}$ para resolver el problema del filtro especificado en la sección (1).

En la sección 6.1 mostramos un argumento que sirve para dirigir la búsqueda de una función costo adecuada.

En la sección 6.2 presentamos primero el caso en que todas las variables son libres y reales para ver la convergencia del algoritmo. En la sección 6.3 modificamos ligeramente el algoritmo para que el valor de R_1 quede fijo a alguno de los valores admisibles definidos por la serie E y analizamos las soluciones continuas obtenidas.

A continuación, en la sección 6.4, extendemos el algoritmo de la sección anterior para que, una vez obtenida la solución para variables continuas, realice una búsqueda local entre los vecinos discretos más cercanos a esa solución de variables continuas.

Luego, en la sección 6.6, introducimos una modificación al algoritmo $\mathrm{ACO}_{\mathbb{R}}$ para que trabaje con los valores discretos de las componentes del circuito.

Todas las versiones del algoritmo utilizaban los mismos valores para los parámetros del mismo, a excepción del número máximo de iteraciones que se lo ejecuta, ya que algunas versiones convergen mucho más rápido que otras y por lo tanto se las ejecutó un número menor de veces.

Los parámetros comunes utilizados fueron los siguientes,

- Tamaño de archivo (k): 10
- Tamaño de la muestra: 40
- Factor de intensificación (q): 0.5
- Relación de distancia de desviación (ξ): 1.0

6.1. Elección de la función costo

Para cada una de las variantes, definimos una única función *costo* que es el objetivo a minimizar. En el problema que estamos tratando, la elección de la forma funcional de la función costo no es única. La sensibilidad es un sumando en la función costo ya que es la cantidad a minimizar. Pero las restricciones duras (sobre G, ω y Q) se agregan a la función costo mediante funciones que penalicen aquellas configuraciones que no las satisfacen. Cualquier función que tenga un único mínimo en los valores de G, ω y Q elegidos como objetivos es adecuada, pero el rendimiento del algoritmo dependerá de la forma funcional explícita que elijamos. Por ejemplo, para penalizar las configuraciones que no satisfacen la restricción sobre G podemos agregar a la función costo un sumando de la forma $w_G((G - G_o)/G_o)^2$ donde w_G es un número positivo. Pero también podemos usar por ejemplo $w_G|(G - G_o)/G_o|$ o $w_G(\log(G/G_o))^2$, etc. Lo mismo sucede con las restricciones sobre ω y Q.

A partir de la ecuación (3) y de los valores que pueden tomar las resistencias y capacitores en las series industriales con las que trabajamos, es fácil ver que los valores extremos que puede tomar ω son $\omega_{min} \simeq 1{,}34$ rad/seg y $\omega_{max} = 10^6$ rad/seg. De manera análoga, para la ganancia G, a partir de la ecuación (2), se obtiene $G_{min} \simeq 0{,}001$ y $G_{max} = 910$. Para determinar los valores extremos que puede obtener Q observemos que a partir de (4) podemos escribir,

$$Q^{-1} = \sqrt{C_5/C_4} \cdot f(x, y) \tag{40}$$

donde $f(x,y) \equiv \sqrt{xy} + \sqrt{x/y} + \sqrt{y/x}$ siendo $x = R_2/R_1$ e $y = R_3/R_1$. Hallando los valores máximos y mínimos de $f(x,y)$ dentro del rango permitido para las variables x e y cuando las resistencias toman los valores de las series discretas utilizadas y teniendo también en cuenta los valores permitidos para los capacitores encontramos los siguientes valores máximo y mínimo para Q en el escenario 1, $Q_{min} \simeq 3{,}4 \cdot 10^{-5}$ y $Q_{max} \simeq 15{,}07$, mientras que para el escenario 2 obtenemos $Q_{min} \simeq 3{,}8 \cdot 10^{-5}$ y $Q_{max} \simeq 14{,}31$. Vemos entonces que tanto los rangos de ω como de Q se extienden a lo largo de varios órdenes de magnitud. Por otro lado, vimos en la sección 4 que la sensibilidad varía, cuando $G = 3$, entre 0.75 y 1.5. Esta gran diferencia entre el rango de la sensibilidad, de G, Q y ω hace que los distintos sumandos que integran la función costo dificulten el proceso de convergencia si las funciones elegidas tienen también rangos muy diferentes. Encontramos en nuestro caso que los mejores resultados los obtenemos eligiendo para G, ω y Q funciones de penalización que tienen un rango que se extiende en varias unidades en lugar de varios órdenes de magnitud para que no haya un desbalance entre los términos que integran la función costo. Siguiendo este criterio y luego de haber probado con distintas formas funcionales, encontramos que la función costo que mejor resultados permitió obtener es la siguiente,

$$costo = w_S S^2 + w_G |\log(G/G_o)| + w_\omega \left(\log(\omega/\omega_o)\right)^2 + w_Q \left(\log(Q/Q_o)\right)^2 \quad (41)$$

En todas las variantes del algoritmo utilizamos $w_S = w_G = w_\omega = w_Q = 1{,}0$.

A continuación el pseudocódigo muestra la función costo utilizada en todos los casos.

Algoritmo 3 Función costo a minimizar

R_{min}, R_{max}: Valores mínimo y máximo que pueden tomar las resistencias
C_{min}, C_{max}: Valores mínimo y máximo que pueden tomar los capacitores
$W_G \leftarrow 1{,}0$: Peso de la restricción sobre la ganancia
$W_\omega \leftarrow 1{,}0$: Peso de la restricción sobre ω
$W_Q \leftarrow 1{,}0$: Peso de la restricción sobre Q
$W_S \leftarrow 1{,}0$: Peso de la restricción sobre la sensibilidad
$W_{Max} \leftarrow 1e11$: Penalización para variables fuera de rango
$G_{obj} \leftarrow 3$: Ganancia objetivo
$\omega_{obj} \leftarrow 2000 * \pi$: Frecuencia de polo objetivo
$Q_{obj} \leftarrow 1/\sqrt{2}$: Factor de calidad objetivo

function EnRango(Comp)
    if EsResistencia($Comp$) **then**
        return $R_{min} < Comp < R_{max}$
    else
        return $C_{min} < Comp < C_{max}$

function Costo(R_1, R_2, R_3, C_4, C_5)
    $RangosOK \leftarrow \langle \forall\, c : c \in \{R_1, R_2, R_3, C_4, C_5\} : \text{EnRango}(c) \rangle$
    if $RangosOK$ **then**

$Sens \leftarrow$ SENSIBILIDAD$(R_1, R_2, R_3, C_4, C_5)$
$Q \leftarrow$ FACTORCALIDAD$(R_1, R_2, R_3, C_4, C_5)$
$G \leftarrow$ GANANCIA$(R_1, R_2, R_3, C_4, C_5)$
$\omega \leftarrow$ FRECUENCIAPOLO$(R_1, R_2, R_3, C_4, C_5)$
$Costo \leftarrow W_S * Sens^2 + W_G * |\log(G/G_{obj})| + W_\omega * (\log(\omega/\omega_{obj}))^2 + W_Q * (\log(Q/Q_{obj}))^2$
return $Costo$
else
return W_{Max}

6.2. Versión continua con todas las variables libres

En esta primera versión del algoritmo el dominio de las variables es todo el intervalo continuo entre los valores admisibles máximos y mínimos especificados por las series E. Es conveniente que definamos algunas funciones elementales aquí para hacer el pseudocódigo más legible [1].

- $zeros(n)$: devuelve un arreglo A de n elementos, todos inicializados en 0.
- $zeros(m, n)$: devuelve una matriz M de tamaño $m \times n$ con todos sus elementos inicializados en 0.
- $repmat(M, f, c)$: dada una matriz M devuelve una matriz que replica f copias de M en las filas y c copias en las columnas.
- $randUniform(m, n)$: dados $m < n$ devuelve un número aleatorio con distribución uniforme en el intervalo $[m, n]$. Si m y n se omiten, toman por defecto el valor 0 y 1 respectivamente.
- $randn()$: Devuelve un valor aleatorio correspondiente a una distribución de probabilidad normal estándar.
- $cumsum(p)$: Devuelve la suma acumulada de elementos a lo largo de un dado eje.
- $sort(M, c)$: ordena las filas de la matriz M de menor a mayor de acuerdo al valor de la columna c.
- $concat(M, N)$: dadas las matrices M de tamaño $m \times n$ y N de tamaño $l \times n$ devuelve una matriz M' de tamaño $(m + l) \times n$ donde

$$M'[i][j] = \begin{cases} M[i][j], & \text{si } i \leq m \\ N[l + (i - m - 1)][j], & \text{caso contrario} \end{cases}$$

- $sum(A)$: Dado un arreglo numérico A, devuelve la suma de sus elementos.

El pseudocódigo correspondiente a esta versión puede verse a continuación:

Algoritmo 4 ACO$_{\mathbb{R}}$ Continuo Libre

R_{min}, R_{max}: Valores mínimo y máximo que pueden tomar las resistencias

[1] Estas funciones pueden encontrarse ya implementadas en varias librerías o lenguajes como por ejemplo *Python* -en la librería *Numpy*- y *Matlab*

C_{min}, C_{max}: Valores mínimo y máximo que pueden tomar los capacitores
$TamArchivo$: cantidad de soluciones que se mantienen en el archivo T
$TamMuestra$: cantidad de hormigas que se lanzan por iteración
$NumIt$: número de iteraciones
$FactorIntensidad$: ξ
$NumVars = 5$ ▷ R_1, R_2, R_3, C_4, C_5
$zeta$: Deviation Distance Ratio

function ElegirNucleo(p)
    $r \leftarrow randUniform()$
    $C \leftarrow cumsum(p)$
    return $\left\langle Min\ j : 0 \leq j < size(C) : C[j] > r \right\rangle$

procedure InicializarArchivo(T) ▷ Inicializa aleatoriamente el archivo
    for cada fila f en T **do**
        $f[0 \ldots 2] \leftarrow randUniform(R_{min}, R_{max})$
        $f[3 \ldots 4] \leftarrow randUniform(C_{min}, C_{max})$
        $f[5] \leftarrow$ Costo($f[0 \ldots 4]$)

function HallarSoluciones()
    $fila \leftarrow zeros(NumVars + 1)$
    $T \leftarrow repmat(fila, TamArchivo, 1)$
    InicializarArchivo(T)
    $sort(T, NumVars)$
    $w \leftarrow zeros(TamArchivo)$ ▷ Arreglo con los pesos
    for l en $[0 \ldots TamArchivo)$ **do**
        $b \leftarrow \frac{1}{\sqrt{2\pi} \cdot FactorIntensidad \cdot TamArchivo}$
        $c \leftarrow e^{-\frac{l}{2 \cdot (FactorIntensidad \cdot TamArchivo)^2}}$
        $w[l] \leftarrow b \cdot c$
    $p \leftarrow zeros(TamArchivo)$ ▷ Arreglo con las probabilidades
    for l en $[0 \ldots TamArchivo)$ **do**
        $p[l] \leftarrow \frac{w[l]}{sum(w)}$

    for it en $[0 \ldots NumIt)$ **do** ▷ Loop principal
        $\mu \leftarrow zeros(TamArchivo, NumVars)$ ▷ Arreglo con las medias
        for j en $[0 \ldots TamArchivo)$ **do**
            $\mu[j] \leftarrow T[j][0 \ldots NumVars - 1]$
        $\sigma \leftarrow zeros(TamArchivo, NumVars)$ ▷ Arreglo con las desv. est.
        for j en $[0 \ldots TamArchivo)$ **do**
            $D \leftarrow \sum_{r=0}^{TamArchivo-1} |\mu[j] - \mu[r]|$
            $\sigma[j] \leftarrow \frac{zeta * D}{TamArchivo - 1}$

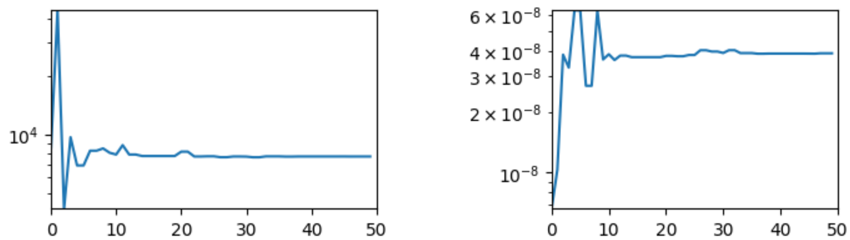
        $NuevaPobl \leftarrow repmat(fila, TamMuestra, 1)$

```
        for t en [0 ... TamMuestra) do
            NuevaPobl[t][0 ... NumVars) ← zeros(NumVars)
            for i en [0 ... NumVars) do
                g ← ElegirNucleo(p)                ▷ Escoger núcleo Gaussiano
                NuevaPobl[t][i] ← μ[g][t] + σ[g][i] · rand()
            NuevaPobl[t][NumVars] ← Cost(NuevaPobl[0 ... NumVars))
        PobTotal ← concat(T, NuevaPobl)
        sort(PobTotal, NumVars)
        T ← PobTotal[0 ... TamArchivo)        ▷ Actualizar archivo con las mejores soluciones
    Sol ← Archivo[0]
    return Sol
```

Es de hacer notar que hemos utilizado dos variantes del algoritmo: en la primera el muestreo y selección de las variables se hace directamente sobre su valor, mientras que en la segunda se hace utilizando el logaritmo de los valores (denominaremos a estas dos variantes lineal y logarítmica respectivamente). Esta diferenciación se hace efectiva en el algoritmo simplemente modificando el procedimiento de inicialización del archivo donde se cambian los límites de la llamada a *randUniform* utilizando $\log(R_{min})$ y $\log(R_{max})$ -lo mismo se hace con los capacitores-. El rendimiento de la versión logarítmica es sensiblemente superior al de la versión lineal: en el escenario 2 y para los mismos valores de los parámetros del algoritmo y para una tolerancia en ambos casos del 2,5 % sobre las variables objetivo, en 100 repeticiones la versión logarítmica convergió a una solución aceptable en el 100 % de los casos mientras que la versión lineal lo hizo en el 68 %.

Toda vez que el algoritmo converge se obtiene el mínimo valor de la función sensibilidad total S que ya fuera obtenido mediante el cálculo analítico de la sección 4. Así mismo, las contribuciones de las sensibilidades individuales S_i adoptan los mismos valores indicados en esa sección cuando se obtiene el mínimo de la sensibilidad total.

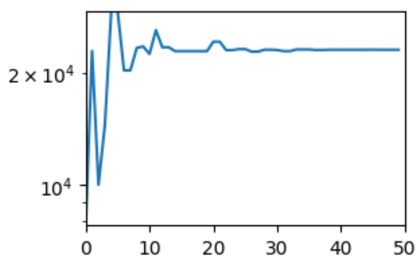
A modo de ejemplo, en la figura (5) mostramos la convergencia de la función costo así como también de las cinco variables del circuito para la versión logarítmica del algoritmo para una ejecución del mismo de 500 iteraciones, aunque se representan sólo los resultados de las primeras 50 iteraciones. El resultado obtenido luego de las 500 iteraciones para las valores de los componentes del filtro es $R_1 = 7710\Omega$, $R_2 = 23129\Omega$, $R_3 = 5782\Omega$, $C_4 = 3,89 \cdot 10^{-8} F$ y $C_5 = 4,87 \cdot 10^{-9} F$. La configuración tiene la sensibilidad mínima ($S = 0,75$), y los errores porcentuales en las variables objetivos son $\epsilon_G = 6 \cdot 10^{-14}\,\%$, $\epsilon_\omega = 4 \cdot 10^{-8}\,\%$ y $\epsilon_Q = 5 \cdot 10^{-7}\,\%$. El valor final de la función costo es $0,5625$ (=$0,75^2$).



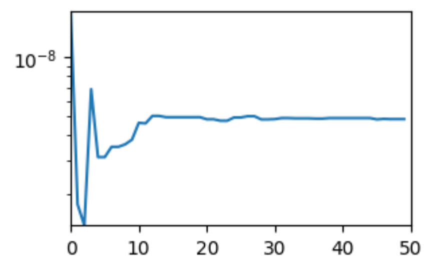
(a) (b)

En la figura (5f) pareciera que la función costo converge a su valor límite antes que las otras variables. Esto en realidad no es así ya que si al costo calculado le restamos su valor asintótico se puede apreciar cómo converge esta función. En la figura (6) mostramos en escala logarítmica la función costo a la que se le restó su valor asintótico de 0,5625.

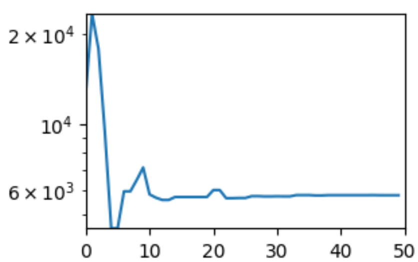
A fin de ilustrar el comportamiento del algoritmo a lo largo de las iteraciones se muestra en la figura (7) la evolución del núcleo gaussiano generado por el algoritmo para cada una de las variables (componentes del filtro) a lo largo de las iteraciones. Se hace notar que estas figuras corresponden a otra corrida del algoritmo distinta a la de las figuras (5) y (6). Esta opción se ha adoptado a fines meramente ilustrativos. En todos los casos puede observarse que para las primeras iteraciones se obtiene una distribución de probabilidad con una dispersión grande la cual se va reduciendo a medida que transcurren las mismas al tiempo que el valor medio tiende al de la solución definitiva. Si bien la construcción del núcleo se basa en la suma ponderada de gaussianas como se describió oportunamente en la sección 5.2, se observa de las gráficas de la figura (7) que los núcleos gaussianos lucen como una única gaussiana. Esto es consecuencia del valor de ξ elegido ($\xi = 1$), el cual ha dado resultados satisfactorios en nuestras simulaciones.



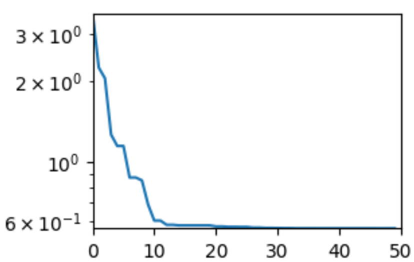
(c)



(d)



(e)



(f)

Figura 5: Convergencia de las distintas variables y la función costo para la versión logarítmica con todas las variables libres. El eje vertical corresponde al valor de la variable y el horizontal al número de iteración. (a) R1, (b) C4, (c) R2, (d) C5, (e) R3 y (f) costo.

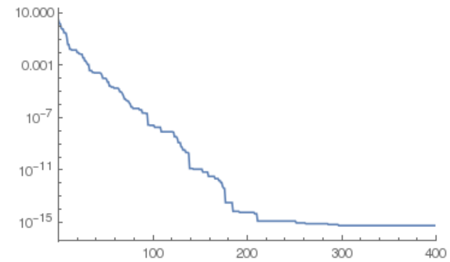
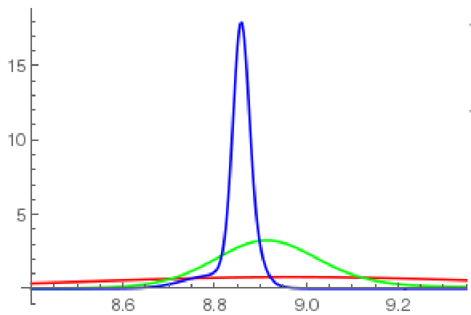
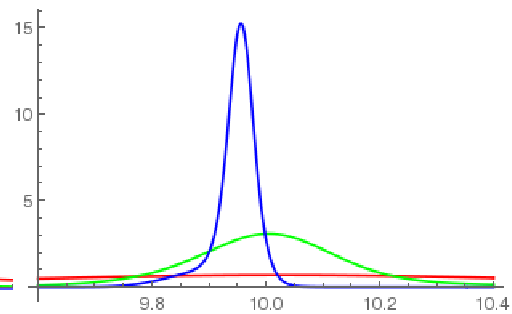


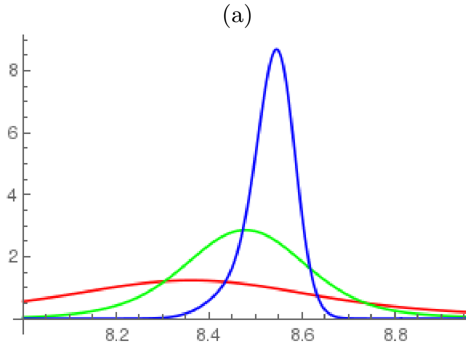
Figura 6: Convergencia de la función costo de la figura (5f) a la que se le restó el valor asintótico.



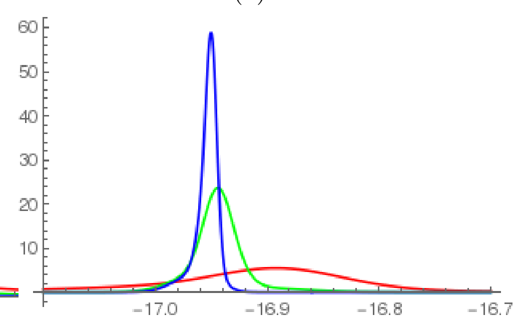
(a)



(b)



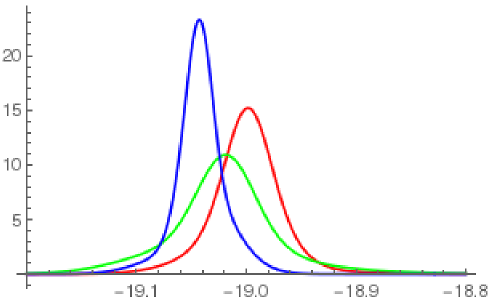
(c)



(d)

Para ilustrar la presencia de las distintas gaussianas involucradas en la construcción del núcleo durante las iteraciones del algoritmo se muestran en la figura (8) las gaussianas individuales que componen el núcleo para el componente R_1 en la primera iteración. Puede verse de esta última figura que las distintas gaussianas se encuentran solapadas en diferentes grados no observándose ninguna aislada.

Con propósitos ilustrativos adoptamos un valor menor de ξ ($\xi = 0{,}3$) a los fines de visualizar la presencia de varios picos en el núcleo gaussiano. Se hace notar en este punto que este valor de ξ no es el que da los mejores resultados



(e)

Figura 7: Núcleos gaussianos para cada una de las componentes del filtro. (a) R_1, (b) R_2, (c) R_3, (d) C_4 y (e) C_5. Para las resistencias la curva roja corresponde a la iteración 20, la verde a la 25 y la azul a la número 30. Para los capacitores, la curva roja corresponde a la iteración 20, la verde a la 30 y la azul a la 40. En todos los casos las abscisas corresponden al logaritmo natural de la variable.

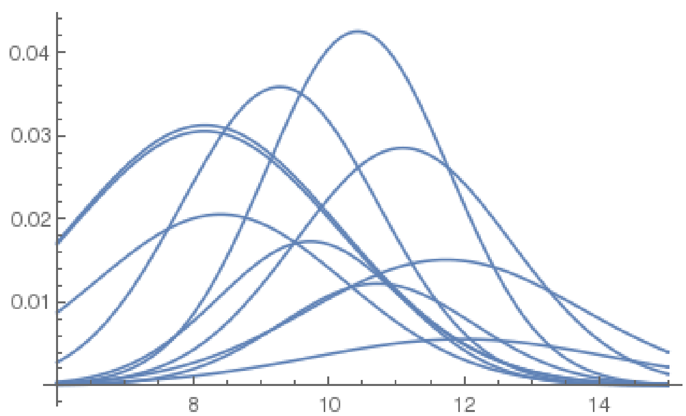


Figura 8: Gaussianas individuales componentes del núcleo gaussiano para un archivo de tamaño $k = 10$

(la tasa de éxitos es baja). En la figura (9) mostramos el núcleo para la primera iteración para la variable R_1, pudiéndose observar dos máximos. En la figura (10) mostramos el núcleo (para R_1) para tres iteraciones cercanas. De este gráfico puede observarse que el núcleo sigue manteniendo en general una distribución con más de un pico, pero con una dispersión que tiende a disminuir.

6.3. Versión continua con R_1 fijado

En la sección 6.2 vimos cómo $\mathrm{ACO}_{\mathbb{R}}$ converge hacia valores dentro del rango continuo que cada componente puede tomar, a la vez que minimiza la función costo. Sin embargo, en nuestro problema, el conjunto de valores que puede tomar

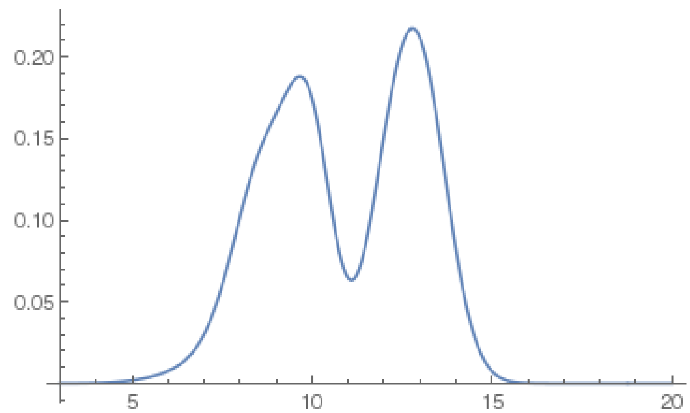


Figura 9: Núcleo gaussiano para la primera iteración correspondiente a la componente R_1 y para $\xi = 0{,}3$

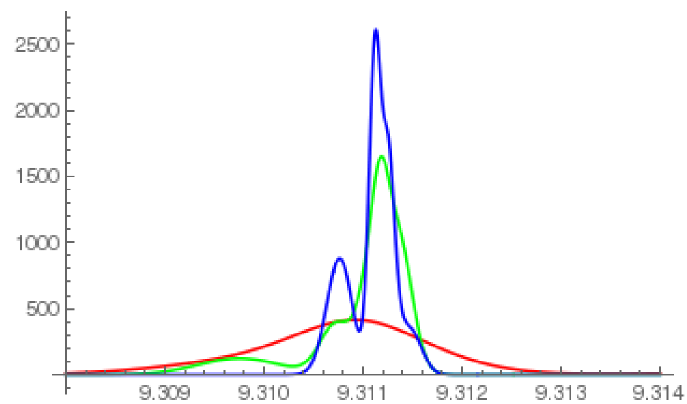


Figura 10: Núcleo gaussiano para R_1 y para $\xi = 0{,}3$. La curva roja corresponde a la iteración número 12, la verde a la 14 y la azul a la 16.

cada componente del filtro proviene de un conjunto finito. Si pretendemos utilizar $\mathrm{ACO}_{\mathbb{R}}$ para obtener soluciones en el dominio discreto, debemos comenzar por reconocer que aun cuando el algoritmo encuentra soluciones en el dominio continuo cuando no especificamos al inicio ninguna de las cinco variables, esto no es de gran utilidad ya que la solución a la que converge puede no ser próxima a ninguna de las configuraciones discretas disponibles. Una manera de comenzar a remediar este inconveniente es restringir el algoritmo a determinar cuatro de las variables fijando de antemano una de ellas a uno de los valores disponibles en el dominio discreto de esta variable. Esto es posible ya que en la sección 4 mostramos que en el dominio de variables continuas, el problema se reduce a uno de cuatro variables con cinco incógnitas y por lo tanto podemos especificar el valor de una de las cinco y así quedan determinados los valores de las otras cuatro. De esta manera la solución que obtenemos tiene al menos una de las variables con una asignación de valor permitido. Más adelante veremos en la subsección 6.4 cómo utilizar esta solución para obtener una en el dominio discreto que satisfaga todas las restricciones impuestas.

Con esta motivación es que en esta sección presentamos una variante del algoritmo 4 en el que el valor de R_1 está especificado (obviamente fijado en uno de los valores discretos permitidos) y obtenemos una solución continua utilizando la función **HallarSoluciones**.

En este punto recordamos que el problema que se resuelve es el de la solución analítica descripta por las ecuaciones (32) que por conveniencia reproducimos a continuación

$$\begin{aligned} R_1/R_2 &= a_0 \\ R_1/R_3 &= 1 + R_1/R_2 \\ \omega(R_2, R_3, C_4, C_5) &= \omega_0 \\ Q(R_1, R_2, R_3, C_4, C_5) &= Q_0 \end{aligned}$$

Como se puede ver, en este caso, el número de variables $NumVars$ pasa a ser 4 en vez de 5, ya que R1 queda fijado. De esta manera para cada valor de R_1 se resuelve un sistema de cuatro ecuaciones con cuatro incógnitas, que tiene solución única (ver sección 6.4). Nuestros resultados muestran que el método empleado en la presente sección encuentra siempre esa solución. Luego, iteramos sobre los valores discretos permitidos de R_1 y obtenemos así una solución para cada valor de R_1. En la subsección 6.4 utilizaremos esta solución para construir una (o más) soluciones en el dominio discreto.

Como en la sección 6.2 tendremos una versión que trabaja directamente con los valores de los componentes y otra que lo hace con los logaritmos. Sin embargo, en este caso, los resultados obtenidos son prácticamente idénticos, aunque la versión logarítmica presenta una convergencia más rápida. El pseudocódigo - donde omitimos redefinir aquellos parámetros que son los mismos que en el algoritmo 4- se muestra a continuación.

Algoritmo 5 ACO$_{\mathbb{R}}$ continuo con R_1 fijo

```
NumVars = 4                                          ▷ R2, R3, C4, C5
Soluciones ← []
for R1 en ValoresResistencias do
    Sol ← [R1] + HallarSoluciones()   ▷ Función definida en algoritmo 4
    Agregar Sol a la lista Soluciones
```

A continuación comparamos las cinco mejores soluciones obtenidas con el algoritmo exacto de la sección 3, y que reproducimos en el cuadro 4, con las correspondientes soluciones obtenidas con las versiones lineales y logarítmicas del algoritmo 5 que mostramos en los cuadros 5 y 6 respectivamente. Estos dos últimos cuadros corresponden ambos a simulaciones en las que el algoritmo fue limitado a efectuar 50 iteraciones. Vemos que para el número de cifras que mostramos no se puede apreciar diferencia en los valores de los elementos del circuito, sin embargo en las tres últimas columnas, donde mostramos el error alcanzado al finalizar por las variables objetivo G, ω y Q, se puede apreciar que nuevamente la versión logarítmica tiene un mejor rendimiento que la versión lineal, que en este caso se manifiesta en un menor error de estas variables objetivos para un mismo número de iteraciones.

R_1	R_2	R_3	C_4	C_5	ϵ_G	ϵ_ω	ϵ_Q	$Sens$
11000	33000	8200	2,7 10^{-8}	3,3 10^{-9}	0,0	2,5	1,2	0,7508
3600	11000	2700	8,2 10^{-8}	1,0 10^{-8}	1,9	2,0	0,6	0,7540
36000	110000	27000	8,2 10^{-9}	1,0 10^{-9}	1,9	2,0	0,6	0,7540
12000	36000	8200	2,7 10^{-8}	3,3 10^{-9}	0,0	1,9	1,0	0,7616
3300	10000	2200	1,0 10^{-7}	1,2 10^{-8}	1,0	2,0	1,5	0,7668

Cuadro 4: Resultados obtenidos con la versión exacta. Los errores son porcentuales

R_1	R_2	R_3	C_4	C_5	ϵ_G	ϵ_ω	ϵ_Q
11000	33000	8250	2,73 10^{-8}	3,41 10^{-9}	7 10^{-9}	2 10^{-4}	8 10^{-4}
3600	10800	2700	8,34 10^{-8}	1,04 10^{-8}	1 10^{-8}	1 10^{-3}	3 10^{-4}
36000	10800	27000	8,34 10^{-9}	1,04 10^{-9}	1 10^{-7}	4 10^{-4}	3 10^{-4}
12000	36000	9000	2,50 10^{-8}	3,13 10^{-9}	3 10^{-7}	4 10^{-4}	6 10^{-3}
3300	9900	2475	9,09 10^{-8}	1,14 10^{-8}	2 10^{-8}	1 10^{-4}	6 10^{-4}

Cuadro 5: Resultados obtenidos con la versión lineal fijando R_1 y dejando el resto de las variables libres. La sensibilidad obtenida fue la mínima en todos los casos (0,75). Los errores son porcentuales.

La tabla del apéndice A.3 reporta los resultados completos para la versión logarítmica para todos aquellos valores de R_1 para los cuales es posible hallar una solución en el dominio continuo con la mínima sensibilidad teórica (0.75). Otros valores de R_1 dentro del rango especificado y que no figuran en esa tabla brindan soluciones que cumplen con las especificaciones pero no logran el mínimo de sensibilidad. Esto es debido a la penalización implementada en la función costo para valores de los componentes que se salen del rango permitido. Los resultados correspondientes al *escenario 1* presentan un comportamiento similar razón por la cual no son reportados en este trabajo.

R_1	R_2	R_3	C_4	C_5	ϵ_G	ϵ_ω	ϵ_Q
11000	33000	8250	$2{,}73\ 10^{-8}$	$3{,}41\ 10^{-9}$	$1\ 10^{-12}$	$1\ 10^{-5}$	$4\ 10^{-5}$
3600	10800	2700	$8{,}34\ 10^{-8}$	$1{,}04\ 10^{-8}$	$5\ 10^{-11}$	$2\ 10^{-5}$	$2\ 10^{-5}$
36000	10800	27000	$8{,}34\ 10^{-9}$	$1{,}04\ 10^{-9}$	$2\ 10^{-10}$	$7\ 10^{-5}$	$1\ 10^{-4}$
12000	36000	9000	$2{,}50\ 10^{-8}$	$3{,}13\ 10^{-9}$	$3\ 10^{-10}$	$2\ 10^{-4}$	$9\ 10^{-5}$
3300	9900	2475	$9{,}09\ 10^{-8}$	$1{,}14\ 10^{-8}$	$8\ 10^{-12}$	$2\ 10^{-6}$	$6\ 10^{-5}$

Cuadro 6: Resultados obtenidos con la versión logarítmica fijando R_1 y dejando el resto de las variables libres. La sensibilidad obtenida fue la mínima en todos los casos (0,75). Los errores son porcentuales.

6.4. Versión continua con búsqueda de vecinos discretos cercanos

Como ya mencionamos en la sección 6.3, vamos ahora a utilizar las soluciones continuas obtenidas por $\mathrm{ACO}_{\mathbb{R}}$ cuando se fija el valor de R_1 para hallar soluciones del problema original en el dominio discreto. Vimos que las soluciones obtenidas en la sección 6.3 tienen valores para R_2, R_3, C_4 y C_5 que, en general, no corresponden a los valores permitidos en las series industriales. Pero si complementamos el algoritmo $\mathrm{ACO}_{\mathbb{R}}$ con uno de búsqueda local, a partir de éstas podemos construir soluciones en el dominio discreto. Inicialmente, la búsqueda se limitó a los vecinos inmediatamente más cercanos, sin embargo las soluciones obtenidas no resultaron demasiadas y decidimos extender la búsqueda a dos vecinos, tanto por arriba como por debajo. Así mismo también incluimos a R_1 en la búsqueda de vecinos para extender el espacio de búsqueda local. Los resultados se muestran en el cuadro 7. A continuación mostramos el pseudocódigo para este algoritmo.

Algoritmo 6 $\mathrm{ACO}_{\mathbb{R}}$ Discreto con Búsqueda de Vecinos Cercanos

G_+, G_-: Valores máximo y mínimo aceptables para la ganancia
ω_+, ω_-: Valores máximo y mínimo aceptables para la frecuencia de polo
Q_+, Q_-: Valores máximo y mínimo aceptables para el factor de calidad
$Soluciones \leftarrow []$
for R_1 en $ValoresResistencias$ **do**
    $Sol \leftarrow [R_1] + \textsc{HallarSoluciones}()$ ▷ Función definida en el algoritmo 4
    $Vecinos \leftarrow []$
    for $Componente$ en Sol **do** ▷ Buscar vecinos de cada componente
        $(V_l, V_r) \leftarrow VecinosCercanos(Componente, PosiblesValores)$
        Agregar (V_l, V_r) a la lista $Vecinos$
    for r_1 en $Vecinos[0]$ **do**
        for r_2 en $Vecinos[1]$ **do**
            for r_3 en $Vecinos[2]$ **do**
                for c_4 en $Vecinos[3]$ **do**
                    for c_5 en $Vecinos[4]$ **do**

$SolTentativa \leftarrow [r_1, r_2, r_3, c_4, c_5]$
$Sens \leftarrow$ Sensibilidad($SolTentativa$)
$G \leftarrow$ Ganancia($SolTentativa$)
$Q \leftarrow$ FactorCalidad($SolTentativa$)
$\omega \leftarrow$ FrecuenciaPolo($SolTentativa$)
$G^{ok} \leftarrow G_- < G < G_+$
$Q^{ok} \leftarrow Q_- < Q < Q_+$
$\omega^{ok} \leftarrow \omega_- < \omega < \omega_+$
if $Sens < 1 \wedge G^{ok} \wedge \omega^{ok} \wedge Q^{ok}$ **then**
    Agregar $SolTentativa$ a la lista $Soluciones$

R_1	R_2	R_3	C_4	C_5	ϵ_G	ϵ_Q	ϵ_ω	$Sens$	Búsq. ext.
1300	3900	1100	2,2 10^{-7}	2,7 10^{-8}	0	0,74	0,3	0,795	**X**
1600	4700	1300	1,8 10^{-7}	2,2 10^{-8}	2,08	1,84	2,32	0,778	**X**
2700	8200	1800	1,2 10^{-7}	1,5 10^{-8}	1,23	0,64	2,36	0,767	**X**
2700	8200	2700	1,0 10^{-7}	1,2 10^{-8}	1,23	0,57	2,36	0,859	**X**
3000	9100	2400	1,0 10^{-7}	1,2 10^{-8}	1,11	1,59	1,69	0,775	
3300	10000	2200	1,0 10^{-7}	1,2 10^{-8}	1,01	1,49	2,05	0,767	**X**
3300	10000	3000	8,2 10^{-8}	1,0 10^{-8}	1,01	0,41	1,47	0,823	**X**
3600	11000	2700	8,2 10^{-8}	1,0 10^{-8}	1,85	0,55	1,98	0,754	
4300	13000	2400	8,2 10^{-8}	1,0 10^{-8}	0,78	0,16	0,5	0,788	**X**
4300	13000	3600	6,8 10^{-8}	8,2 10^{-9}	0,78	1,37	1,48	0,792	
5100	15000	3000	6,8 10^{-8}	8,2 10^{-9}	1,96	1,85	0,47	0,776	**X**
5100	15000	4300	5,6 10^{-8}	6,8 10^{-9}	1,96	2,02	1,55	0,792	**X**
6800	20000	6800	3,9 10^{-8}	4,7 10^{-9}	1,96	1,51	0,8	0,855	**X**
7500	22000	4300	4,7 10^{-8}	5,6 10^{-9}	2,22	2,4	0,86	0,779	**X**
9100	27000	5100	3,9 10^{-8}	4,7 10^{-9}	1,1	1,21	0,18	0,784	**X**
10000	30000	9100	2,7 10^{-8}	3,3 10^{-9}	0	0,66	2,05	0,822	
11000	33000	6200	3,3 10^{-8}	3,9 10^{-9}	0	1,8	1,92	0,785	**X**
11000	33000	8200	2,7 10^{-8}	3,3 10^{-9}	0	1,13	2,49	0,751	
12000	36000	8200	2,7 10^{-8}	3,3 10^{-9}	0	1,02	1,87	0,762	
13000	39000	7500	2,7 10^{-8}	3,3 10^{-9}	0	0,27	1,41	0,783	**X**
13000	39000	11000	2,2 10^{-8}	2,7 10^{-9}	0	0,74	0,3	0,795	**X**
16000	47000	13000	1,8 10^{-8}	2,2 10^{-9}	2,08	1,84	2,32	0,778	**X**
27000	82000	18000	1,2 10^{-8}	1,5 10^{-9}	1,23	0,64	2,36	0,767	**X**
27000	82000	27000	1,0 10^{-8}	1,2 10^{-9}	1,23	0,57	2,36	0,859	**X**
30000	91000	24000	1,0 10^{-8}	1,2 10^{-9}	1,11	1,59	1,69	0,775	
33000	100000	22000	1,0 10^{-8}	1,2 10^{-9}	1,01	1,49	2,05	0,767	**X**
33000	100000	30000	8,2 10^{-9}	1,0 10^{-9}	1,01	0,41	1,47	0,823	**X**
36000	110000	27000	8,2 10^{-9}	1,0 10^{-9}	1,85	0,55	1,98	0,754	
43000	130000	24000	8,2 10^{-9}	1,0 10^{-9}	0,78	0,16	0,5	0,788	**X**

Cuadro 7: Soluciones obtenidas con búsqueda de vecinos discretos. En la última columna se marcan con una **X** aquellas soluciones que fueron halladas extendiendo el rango de búsqueda a dos vecinos. Los errores ϵ_G, ϵ_Q y ϵ_ω son porcentuales.

6.5. Versión continua con todas las variables libres y búsqueda de vecinos cercanos

Como una generalización del algoritmo anterior se implementó la versión continua con todas las variables libres y búsqueda de vecinos cercanos. Un algoritmo como éste se utilizaría en aquellos casos en que ninguna de las variables puede ser fijada de antemano. Implementamos solamente el caso en que la búsqueda local se extiende a dos vecinos. Para evaluar el desempeño del algoritmo se realizaron 100 corridas del mismo registrándose la cantidad de soluciones obtenidas y la tasa de éxitos. En este punto definimos la tasa de éxito como la proporción de corridas en que el algoritmo encuentra al menos una solución discreta. Para la versión logarítmica en el escenario 2 se obtuvo una tasa de éxito del 85 % y un total de 203 soluciones, mientras que para la versión lineal estas cantidades fueron 70 % y 195 respectivamente. Para la versión logarítmica en el escenario 1 se obtuvo una tasa de éxito del 10 % y un total de 15 soluciones, mientras que para la versión lineal estas cantidades fueron 8 % y 16 respectivamente. En un primer análisis de estos resultados puede observarse un rendimiento mucho más bajo en el caso del escenario 1. Atribuimos este comportamiento a que en el escenario 1 las especificaciones requeridas al filtro son mucho más exigentes. Es de hacer notar que las soluciones encontradas se encuentran siempre en el lote de las mejores soluciones del conjunto total de configuraciones aceptables obtenidas oportunamente por el método de la sección 3. Por ejemplo, en el caso del escenario 1, todas las soluciones obtenidas se encuentran en el lote de las 15 mejores soluciones de un total de 333 configuraciones aceptables. Esto es así porque cuando uno limita la búsqueda local a una cierta cantidad de vecinos (dos en nuestro caso), se está determinando el tamaño del entorno de búsqueda alrededor de la solución continua calculada. Como la función costo es una función continua, limitar el entorno es equivalente a poner una cota superior a la sensibilidad de las soluciones discretas que se pueden obtener (teniendo en cuenta que la solución continua obtenida corresponde a un mínimo).

6.6. Versión Discreta *Pura*

Para finalizar, presentamos ahora una variante donde, en lugar de *discretizar* las soluciones continuas encontradas por el algoritmo una vez que el mismo finaliza, en el archivo T almacenamos soluciones en las que las variables tienen su dominio D_i en las series discretas industriales en lugar de ser $D_i \subseteq \mathbb{R}$. Esto lo logramos asignando a la componente seleccionada en cada paso de la iteración el valor discreto posible más cercano al continuo generado. Este procedimiento se ilustra en la figura (11), en la cual puede observarse el núcleo gaussiano para un dado componente y una dada iteración. En este gráfico se ha indicado el valor discreto x_i correspondiente a una serie industrial y sus correspondientes vecinos x_{i-1} y x_{i+1}. Al componente x_i se le asigna un valor de probabilidad discreta que resulta de integrar la densidad de probabilidad entre a y b, siendo estas cantidades los puntos medios por abajo y por encima del valor x_i. Esto es equivalente a reemplazar el núcleo gaussiano (37) por el siguiente,

$$G^i(x_j) = \sum_{l=1}^{k} w_l \cdot g_l^i(x_j) = \sum_{l=1}^{k} w_l \cdot \frac{1}{\sigma_l^i \cdot \sqrt{2\pi}} \int_{a_{j-1}}^{a_j} e^{-\frac{(y-\mu_l^i)^2}{2{\sigma_l^i}^2}} \, dy \qquad (42)$$

donde $a_j = (x_j + x_{j+1})/2$. Ahora $g_l^i(x_j)$ es una función discreta de probabilidad para la variable discreta x_j. Para implementar este núcleo gaussiano basta simplemente reemplazar en el algoritmo 4 las líneas 43 y 44 por las siguientes:

Algoritmo 7 Selección Discreta de Componente

1: $g \leftarrow$ ElegirNucleo(p)
2: $randVar \leftarrow \mu[g][t] + \sigma[g][i] \cdot rand()$
3: $esResistencia \leftarrow i < 3$
4: $valores \leftarrow esResistencia\ ?\ ValoresResistencias : ValoresCapacitores$
5: $NuevaPobl[t][i] \leftarrow ValorMasCercano(valores, randVar)$

donde ***ValorMasCercano*** es una función que dada una lista de valores posibles l y un número n, devuelve el valor de la lista e más cercano a n.

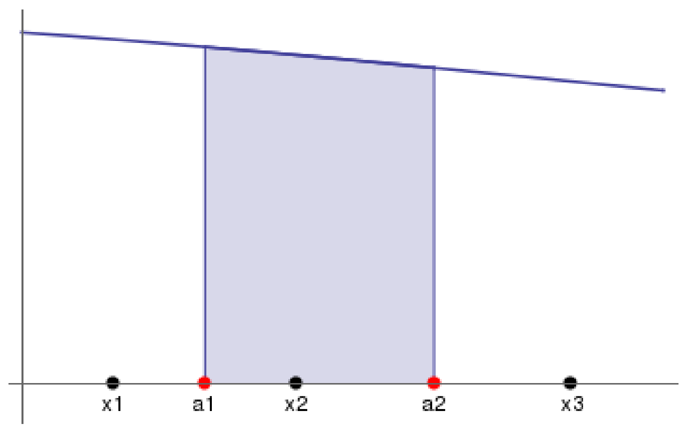


Figura 11: Obtención de una probabilidad discreta a partir de un nucleo gaussiano. Las abscisas corresponden a una variable resistiva siendo $x1 = 22000\Omega$, $x2 = 22000\Omega$ y $x3 = 27000\Omega$ valores de la serie industrial. $a1 = (x1 + x2)/2$ y $a2 = (x2 + x3)/2$ son los puntos medios entre los valores de las resistencias. La probabilidad discreta que se asigna a $x2$ es la integral entre $a1$ y $a2$.

Como en las variantes anteriores, trabajaremos con una versión lineal y una logarítmica. Comenzamos implementando el algoritmo manteniendo R_1 fijo y efectuamos un bucle que recorre todos los valores posibles de R_1 dentro de la serie industrial utilizada. Ejecutamos el algoritmo catorce veces por cada valor de R_1 de manera que al recorrer todos los valores de R_1 efectuamos un total de 1008 ejecuciones. Esto es así porque más adelante vamos a mostrar los resultados de una versión también discreta en la que R_1 no está fija sino que es otra variable a optimizar. En ese caso ejecutaremos 1000 veces el algoritmo. De este modo, al ser prácticamente el mismo el número de ejecuciones, podremos comparar el rendimiento de las dos versiones.

En el cuadro 8 mostramos los resultados para esta versión del algoritmo en sus versiones lineal y logarítmica.

n	R_1	R_2	R_3	lineal	log
45	1200	3600	1200	-	1
27	1300	3900	1100	9	10
2	3600	11000	2700	10	12
1	11000	33000	8200	4	10
4	12000	36000	8200	1	-
3	36000	110000	27000	-	9
		TOTAL		24	42

Cuadro 8: Resultados obtenidos con la versión discreta fijando R_1 y efectuando un bucle sobre todos los valores posibles de R_1

En este cuadro, la primer columna identifica la solución obtenida con el índice de la misma que le fue asignado en el apéndice A.1. En la tabla se muestran también los valores de las resistencias correspondientes a dicha solución. La quinta columna, identificada con *lineal*, indicamos la cantidad de veces que se obtuvo dicha solución con la versión lineal del algoritmo al efectuar las 14 repeticiones para el correspondiente valor de R_1. La última columna muestra el resultado para la versión logarítmica del algoritmo. Finalmente, en la última fila mostramos el total de veces en que se encontró solución. Otra vez se puede apreciar en el cuadro que la versión logarítmica presenta un rendimiento superior respecto a la versión lineal. El tiempo de ejecución de la versión lineal fue de 227 segundos mientras que la de la versión logarítmica fue de 43 segundos.[2] La diferencia se debe a que como la versión logarítmica converge en menos iteraciones que la lineal, el número máximo de iteraciones por ciclo se fijó en 15 para la versión logarítmica mientras que para la lineal, de convergencia más lenta, se fijó en 50. El cuadro 9 muestra los resultados para el escenario 1.

n	R_1	R_2	R_3	lineal	log
1	1870	5620	1400	4	3
4	2490	7500	1870	2	4
2	18700	56200	14000	4	4
3	24900	75000	18700	3	2
		TOTAL		13	13

Cuadro 9: Resultados obtenidos con la versión discreta fijando R_1 y efectuando un bucle sobre todos los valores posibles de R_1 para el escenario 1

[2] En una máquina con un i7 de 3GHz y 16 GB de RAM

Finalmente implementamos la versión discreta del algoritmo pero sin fijar el valor de R_1, sino que permitimos que sea una variable más a determinar. Efectuamos un bucle de mil iteraciones de este algoritmo, tanto en su versión lineal como logarítmica. En el cuadro 10 presentamos los resultados. El tiempo de ejecución de la versión lineal fue de 327 segundos mientras que la de la versión logarítmica fue de 124 segundos. Podemos apreciar que nuevamente la versión logarítmica presenta un mejor desempeño, tanto en tiempo de ejecución (al presentar una convergencia más rápida) como en el número de veces que encuentra solución.

n	R_1	R_2	R_3	lineal	log
7	2700	8200	1800	-	1
2	3600	11000	2700	2	11
1	11000	33000	8200	29	43
4	12000	36000	8200	-	2
60	16000	47000	3000	-	1
8	27000	82000	18000	5	6
10	30000	91000	24000	1	-
3	36000	110000	27000	-	35
		TOTAL		37	99

Cuadro 10: Resultados obtenidos con la versión discreta con todas las variables libres.

En el cuadro 11 mostramos los resultados para el escenario 1. En esta oportunidad fue necesario utilizar el mismo número de iteraciones por ciclo (50) en las dos versiones (lineal y logarítmica) para obtener rendimientos comparables.

n	R_1	R_2	R_3	lineal	log
1	1870	5620	1400	-	1
4	2490	7500	1870	-	2
2	18700	56200	14000	18	19
3	24900	75000	18700	12	12
		TOTAL		30	34

Cuadro 11: Resultados obtenidos con la versión discreta con todas las variables libres para el escenario 1.

7. Conclusiones y Trabajos a Futuro

En este trabajo presentamos la idea de resolver un problema concreto de la industria electrónica aplicando la técnica metaheurística de *Optimización por Colonia de Hormigas* en su extensión sobre dominios continuos -$ACO_{\mathbb{R}}$-.

En el proceso encontramos, dadas las características de diseño del filtro, que era posible resolver el problema de manera analítica e implementar un algoritmo exacto que encontrase todas las soluciones. Esto nos sirvió para luego poder evaluar la calidad de las soluciones encontradas al aplicar $ACO_{\mathbb{R}}$.

Finalmente, realizamos la implementación de $ACO_{\mathbb{R}}$ para este problema en particular enfocándonos primero en resolver el problema asumiendo un dominio continuo para las componentes del filtro. Esto nos permitió corroborar que el algoritmo convergía en tiempos razonables. Una vez que tuvimos esto fuimos trabajando en pos de discretizar los resultados: primero fijamos el valor de R_1 a sus valores discretos posibles dejando las otras variables libres; sobre esa versión trabajamos en otra la cual tomaba las soluciones obtenidas y analizaba las soluciones discretas cercanas mediante una búsqueda local sobre los valores discretos alcanzables más cercanos al real obtenido para cada una de las componentes. Finalmente, probamos realizando la búsqueda local directamente en cada iteración del algoritmo tomando allí el valor discreto posible más cercano.

Los resultados obtenidos fueron satisfactorios. En primer lugar porque al resolver esta clase de problemas combinatorios con restricciones, el objetivo es encontrar *alguna* solución; dicho objetivo fue superado ya que logramos hallar más de una solución y pudimos determinar, más allá de las restricciones, que la *calidad* respecto a la sensibilidad de las mismas era buena, gracias a haber obtenido una lista de todas las soluciones mediante el algoritmo exacto. Además, los tiempos de ejecución fueron relativamente bajos, en el orden de decenas de segundos.

Entre las posibilidades de trabajo a futuro que surgieron durante el desarrollo del trabajo podemos mencionar:

- Extender la solución implementada a otros filtros con una configuración de componentes diferentes.
- Intentar solucionar el problema con el enfoque ACO clásico sobre dominios discretos.

A. Tablas de Resultados

A.1. Algoritmo Exacto

Para el escenario 2 el número de soluciones 171, obtenidas en un tiempo de ejecución 0.80 segundos.

Los errores ϵ_G, ϵ_ω, ϵ_Q son valores absolutos porcentuales. Las soluciones están numeradas por el índice n en orden creciente de la sensibilidad total.

n	R_1	R_2	R_3	C_4	C_5	ϵ_G	ϵ_ω	ϵ_Q	S
1	11000	33000	8200	2.7e-08	3.3e-09	0.0	2.5	1.1	0.7508
2	3600	11000	2700	8.2e-08	1.0e-08	1.9	2.0	0.6	0.7540
3	36000	110000	27000	8.2e-09	1.0e-09	1.9	2.0	0.6	0.7540
4	12000	36000	8200	2.7e-08	3.3e-09	0.0	1.9	1.0	0.7616
5	3300	10000	2200	1.0e-07	1.2e-08	1.0	2.0	1.5	0.7668
6	33000	100000	22000	1.0e-08	1.2e-09	1.0	2.0	1.5	0.7668
7	2700	8200	1800	1.2e-07	1.5e-08	1.2	2.4	0.6	0.7672
8	27000	82000	18000	1.2e-08	1.5e-09	1.2	2.4	0.6	0.7672
9	3000	9100	2400	1.0e-07	1.2e-08	1.1	1.7	1.6	0.7753
10	30000	91000	24000	1.0e-08	1.2e-09	1.1	1.7	1.6	0.7753
11	5100	15000	3000	6.8e-08	8.2e-09	2.0	0.5	1.9	0.7763
12	1600	4700	1300	1.8e-07	2.2e-08	2.1	2.3	1.9	0.7778
13	16000	47000	13000	1.8e-08	2.2e-09	2.1	2.3	1.9	0.7778
14	7500	22000	4300	4.7e-08	5.6e-09	2.2	0.9	2.4	0.7790
15	16000	47000	9100	2.2e-08	2.7e-09	2.1	0.1	0.8	0.7803
16	13000	39000	7500	2.7e-08	3.3e-09	0.0	1.4	0.3	0.7826
17	9100	27000	5100	3.9e-08	4.7e-09	1.1	0.2	1.2	0.7840
18	11000	33000	6200	3.3e-08	3.9e-09	0.0	1.9	1.8	0.7855
19	4300	13000	2400	8.2e-08	1.0e-08	0.8	0.5	0.1	0.7881
20	43000	130000	24000	8.2e-09	1.0e-09	0.8	0.5	0.1	0.7881
21	3600	11000	2000	1.0e-07	1.2e-08	1.9	2.0	0.2	0.7907
22	36000	110000	20000	1.0e-08	1.2e-09	1.9	2.0	0.2	0.7907
23	5100	15000	4300	5.6e-08	6.8e-09	2.0	1.6	2.0	0.7918
24	4300	13000	3600	6.8e-08	8.2e-09	0.8	1.5	1.4	0.7920
25	3000	9100	1600	1.2e-07	1.5e-08	1.1	1.7	1.9	0.7943
26	30000	91000	16000	1.2e-08	1.5e-09	1.1	1.7	1.9	0.7943
27	1300	3900	1100	2.2e-07	2.7e-08	0.0	0.3	0.8	0.7952
28	13000	39000	11000	2.2e-08	2.7e-09	0.0	0.3	0.8	0.7952
29	6800	20000	3300	5.6e-08	6.8e-09	2.0	0.4	0.1	0.8000
30	8200	24000	3900	4.7e-08	5.6e-09	2.4	1.4	0.8	0.8016
31	3000	9100	1500	1.2e-07	1.5e-08	1.1	1.5	2.4	0.8020
32	30000	91000	15000	1.2e-08	1.5e-09	1.1	1.5	2.4	0.8020
33	10000	30000	4700	3.9e-08	4.7e-09	0.0	1.0	0.9	0.8074
34	12000	36000	5600	3.3e-08	3.9e-09	0.0	1.2	0.0	0.8082
35	2700	8200	1200	1.5e-07	1.8e-08	1.2	2.4	1.8	0.8160

36	27000	82000	12000	1.5e-08	1.8e-09	1.2	2.4	1.8	0.8160
37	10000	30000	9100	2.7e-08	3.3e-09	0.0	2.0	0.7	0.8223
38	7500	22000	3000	5.6e-08	6.8e-09	2.2	0.4	2.4	0.8225
39	3300	10000	3000	8.2e-08	1.0e-08	1.0	1.5	0.4	0.8230
40	33000	100000	30000	8.2e-09	1.0e-09	1.0	1.5	0.4	0.8230
41	9100	27000	3600	4.7e-08	5.6e-09	1.1	0.5	2.1	0.8256
42	13000	39000	5100	3.3e-08	3.9e-09	0.0	0.5	2.3	0.8283
43	6800	20000	6800	3.9e-08	4.7e-09	2.0	0.8	1.5	0.8547
44	8200	24000	2400	6.8e-08	6.8e-09	2.4	2.5	1.6	0.8564
45	1200	3600	1200	2.2e-07	2.7e-08	0.0	0.6	0.1	0.8571
46	12000	36000	12000	2.2e-08	2.7e-09	0.0	0.6	0.1	0.8571
47	4300	13000	1300	1.2e-07	1.2e-08	0.8	2.0	0.9	0.8574
48	43000	130000	13000	1.2e-08	1.2e-09	0.8	2.0	0.9	0.8574
49	2700	8200	2700	1.0e-07	1.2e-08	1.2	2.4	0.6	0.8586
50	27000	82000	27000	1.0e-08	1.2e-09	1.2	2.4	0.6	0.8586
51	8200	24000	2200	6.8e-08	6.8e-09	2.4	1.9	0.4	0.8652
52	12000	36000	3300	4.7e-08	4.7e-09	0.0	1.8	0.9	0.8659
53	6800	20000	1800	8.2e-08	8.2e-09	2.0	2.3	0.9	0.8671
54	36000	110000	10000	1.5e-08	1.5e-09	1.9	1.2	1.5	0.8672
55	3600	11000	1000	1.5e-07	1.5e-08	1.9	1.2	1.5	0.8672
56	10000	30000	2700	5.6e-08	5.6e-09	0.0	0.1	1.3	0.8676
57	30000	91000	8200	1.8e-08	1.8e-09	1.1	2.4	1.5	0.8678
58	3000	9100	3300	8.2e-08	1.0e-08	1.1	1.4	1.0	0.8934
59	30000	91000	33000	8.2e-09	1.0e-09	1.1	1.4	1.0	0.8934
60	16000	47000	3000	4.7e-08	3.9e-09	2.1	1.0	0.9	0.8980
61	27000	82000	5100	2.7e-08	2.2e-09	1.2	1.0	1.2	0.9006
62	33000	100000	6200	2.2e-08	1.8e-09	1.0	1.6	1.5	0.9008
63	51000	150000	9100	1.5e-08	1.2e-09	2.0	1.5	0.6	0.9021
64	11000	33000	2000	6.8e-08	5.6e-09	0.0	0.4	2.3	0.9024
65	13000	39000	2000	6.8e-08	4.7e-09	0.0	0.8	1.1	0.9149
66	16000	47000	2400	5.6e-08	3.9e-09	2.1	1.4	0.8	0.9150
67	51000	150000	7500	1.8e-08	1.2e-09	2.0	2.1	2.3	0.9165
68	1100	3300	1300	2.2e-07	2.7e-08	0.0	0.3	1.6	0.9176
69	11000	33000	13000	2.2e-08	2.7e-09	0.0	0.3	1.6	0.9176
70	11000	33000	1600	8.2e-08	5.6e-09	0.0	2.2	0.2	0.9188
71	43000	130000	6200	2.2e-08	1.5e-09	0.8	2.4	0.7	0.9200
72	4300	13000	5100	5.6e-08	6.8e-09	0.8	0.2	1.4	0.9200
73	8200	24000	1100	1.2e-07	8.2e-09	2.4	1.3	1.8	0.9223
74	7500	22000	9100	3.3e-08	3.9e-09	2.2	0.8	0.7	0.9238
75	3600	11000	4300	6.8e-08	8.2e-09	1.9	2.0	1.5	0.9240
76	9100	27000	11000	2.7e-08	3.3e-09	1.1	2.2	1.3	0.9241
77	5100	15000	6200	4.7e-08	5.6e-09	2.0	1.7	0.2	0.9248
78	1300	3900	1600	1.8e-07	2.2e-08	0.0	1.2	1.9	0.9320
79	13000	39000	16000	1.8e-08	2.2e-09	0.0	1.2	1.9	0.9320
80	1600	4700	2000	1.5e-07	1.8e-08	2.1	0.1	0.4	0.9344

81	16000	47000	20000	1.5e-08	1.8e-09	2.1	0.1	0.4	0.9344
82	30000	91000	3300	3.9e-08	2.2e-09	1.1	0.8	1.1	0.9367
83	43000	130000	4700	2.7e-08	1.5e-09	0.8	1.2	0.4	0.9369
84	12000	36000	1300	1.0e-07	5.6e-09	0.0	1.7	0.8	0.9369
85	36000	110000	3900	3.3e-08	1.8e-09	1.9	0.3	0.3	0.9380
86	68000	200000	6800	1.8e-08	1.0e-09	2.0	1.7	2.4	0.9400
87	12000	36000	1100	1.2e-07	5.6e-09	0.0	2.4	2.0	0.9455
88	36000	110000	3300	3.9e-08	1.8e-09	1.9	0.3	1.7	0.9465
89	30000	91000	2700	4.7e-08	2.2e-09	1.1	0.1	0.6	0.9470
90	68000	200000	5600	2.2e-08	1.0e-09	2.0	1.4	0.0	0.9496
91	12000	36000	1000	1.2e-07	5.6e-09	0.0	2.3	1.8	0.9500
92	30000	91000	2200	5.6e-08	2.2e-09	1.1	1.3	1.1	0.9559
93	30000	91000	1800	6.8e-08	2.2e-09	1.1	1.7	2.4	0.9634
94	75000	220000	3600	3.3e-08	1.0e-09	2.2	1.6	2.3	0.9693
95	27000	82000	1200	1.0e-07	2.7e-09	1.2	2.4	1.7	0.9724
96	75000	220000	3000	3.9e-08	1.0e-09	2.2	0.8	2.1	0.9741
97	27000	82000	1000	1.2e-07	2.7e-09	1.2	2.4	0.8	0.9768
98	33000	100000	1200	1.0e-07	2.2e-09	1.0	2.0	0.4	0.9771
99	33000	100000	1000	1.2e-07	2.2e-09	1.0	2.0	0.4	0.9808
100	75000	220000	1200	1.0e-07	1.0e-09	2.2	2.0	2.3	0.9893
101	6800	20000	10000	3.3e-08	3.9e-09	2.0	0.8	2.1	0.9901
102	75000	220000	1100	1.0e-07	1.0e-09	2.2	2.3	1.9	0.9902
103	2700	8200	4700	8.2e-08	8.2e-09	1.2	1.1	2.2	1.0506
104	1000	3000	1800	2.2e-07	2.2e-08	0.0	1.6	1.9	1.0588
105	10000	30000	18000	2.2e-08	2.2e-09	0.0	1.6	1.9	1.0588
106	8200	24000	15000	2.7e-08	2.7e-09	2.4	1.8	2.4	1.0591
107	1200	3600	2200	1.8e-07	1.8e-08	0.0	0.6	1.5	1.0645
108	12000	36000	22000	1.8e-08	1.8e-09	0.0	0.6	1.5	1.0645
109	3000	9100	6200	6.8e-08	6.8e-09	1.1	1.5	1.5	1.1028
110	7500	22000	16000	2.7e-08	2.7e-09	2.2	0.6	1.2	1.1052
111	3600	11000	7500	5.6e-08	5.6e-09	1.9	1.1	1.9	1.1066
112	5100	15000	11000	3.9e-08	3.9e-09	2.0	0.5	1.5	1.1089
113	4300	13000	9100	4.7e-08	4.7e-09	0.8	1.5	1.9	1.1091
114	9100	27000	20000	2.2e-08	2.2e-09	1.1	1.6	2.3	1.1160
115	1100	3300	2400	1.8e-07	1.8e-08	0.0	0.6	2.4	1.1163
116	11000	33000	24000	1.8e-08	1.8e-09	0.0	0.6	2.4	1.1163
117	3600	11000	9100	5.6e-08	4.7e-09	1.9	1.9	2.0	1.1608
118	1600	4700	4300	1.2e-07	1.0e-08	2.1	2.2	1.8	1.1679
119	16000	47000	43000	1.2e-08	1.0e-09	2.1	2.2	1.8	1.1679
120	1300	3900	3600	1.5e-07	1.2e-08	0.0	0.1	2.4	1.1803
121	13000	39000	36000	1.5e-08	1.2e-09	0.0	0.1	2.4	1.1803
122	1600	4700	4700	1.2e-07	1.0e-08	2.1	2.2	0.8	1.1899
123	16000	47000	47000	1.2e-08	1.0e-09	2.1	2.2	0.8	1.1899
124	1000	3000	3000	1.8e-07	1.5e-08	0.0	2.1	2.0	1.2000
125	10000	30000	30000	1.8e-08	1.5e-09	0.0	2.1	2.0	1.2000

126	2700	8200	8200	6.8e-08	5.6e-09	1.2	0.5	2.1	1.2059
127	1200	3600	3900	1.5e-07	1.2e-08	0.0	0.1	2.4	1.2188
128	12000	36000	39000	1.5e-08	1.2e-09	0.0	0.1	2.4	1.2188
129	3300	10000	12000	5.6e-08	3.9e-09	1.0	1.7	0.6	1.2461
130	2700	8200	10000	6.8e-08	4.7e-09	1.2	1.7	0.3	1.2506
131	6800	20000	27000	2.7e-08	1.8e-09	2.0	1.8	0.7	1.2564
132	8200	24000	33000	2.2e-08	1.5e-09	2.4	1.6	0.7	1.2577
133	1000	3000	3900	1.8e-07	1.2e-08	0.0	0.1	0.7	1.2581
134	10000	30000	39000	1.8e-08	1.2e-09	0.0	0.1	0.7	1.2581
135	1200	3600	4700	1.5e-07	1.0e-08	0.0	0.1	0.6	1.2589
136	12000	36000	47000	1.5e-08	1.0e-09	0.0	0.1	0.6	1.2589
137	3000	9100	15000	5.6e-08	3.3e-09	1.1	0.2	2.2	1.3075
138	3600	11000	18000	4.7e-08	2.7e-09	1.9	0.4	1.1	1.3095
139	4300	13000	22000	3.9e-08	2.2e-09	0.8	1.6	0.8	1.3104
140	3000	9100	18000	5.6e-08	2.7e-09	1.1	1.1	0.9	1.3366
141	5100	15000	33000	3.3e-08	1.5e-09	2.0	1.7	1.8	1.3382
142	3600	11000	22000	4.7e-08	2.2e-09	1.9	0.6	1.5	1.3415
143	4300	13000	27000	3.9e-08	1.8e-09	0.8	1.4	1.4	1.3423
144	7500	22000	51000	2.2e-08	1.0e-09	2.2	1.3	0.2	1.3441
145	1600	4700	11000	1.0e-07	4.7e-09	2.1	2.1	2.3	1.3460
146	1100	3300	7500	1.5e-07	6.8e-09	0.0	0.2	0.8	1.3514
147	4300	13000	33000	3.9e-08	1.5e-09	0.8	0.5	2.5	1.3689
148	5100	15000	43000	3.3e-08	1.2e-09	2.0	0.4	2.1	1.3712
149	1100	3300	9100	1.5e-07	5.6e-09	0.0	0.2	1.0	1.3753
150	1300	3900	11000	1.2e-07	4.7e-09	0.0	2.3	2.3	1.3779
151	1600	4700	16000	1.0e-07	3.3e-09	2.1	1.0	0.3	1.3885
152	1100	3300	11000	1.5e-07	4.7e-09	0.0	0.5	1.8	1.3953
153	4300	13000	43000	3.9e-08	1.2e-09	0.8	1.6	2.5	1.3978
154	1600	4700	20000	1.0e-07	2.7e-09	2.1	0.1	0.0	1.4080
155	1300	3900	16000	1.2e-07	3.3e-09	0.0	1.2	0.8	1.4138
156	1600	4700	24000	1.0e-07	2.2e-09	2.1	1.0	2.1	1.4214
157	3300	10000	43000	4.7e-08	1.2e-09	1.0	2.2	0.1	1.4217
158	1300	3900	20000	1.2e-07	2.7e-09	0.0	0.1	0.7	1.4303
159	1600	4700	30000	1.0e-07	1.8e-09	2.1	0.1	1.9	1.4350
160	1300	3900	24000	1.2e-07	2.2e-09	0.0	1.2	1.2	1.4414
161	1300	3900	30000	1.2e-07	1.8e-09	0.0	0.1	0.8	1.4528
162	1600	4700	47000	1.0e-07	1.2e-09	2.1	2.2	1.1	1.4551
163	1300	3900	36000	1.2e-07	1.5e-09	0.0	0.1	1.4	1.4604
164	1600	4700	56000	1.0e-07	1.0e-09	2.1	1.9	1.9	1.4609
165	1000	3000	36000	1.5e-07	1.5e-09	0.0	2.1	0.0	1.4694
166	1300	3900	47000	1.2e-07	1.2e-09	0.0	2.0	0.2	1.4695
167	1300	3900	56000	1.2e-07	1.0e-09	0.0	1.7	0.5	1.4743
168	1000	3000	47000	1.5e-07	1.2e-09	0.0	0.1	1.7	1.4764
169	1000	3000	56000	1.5e-07	1.0e-09	0.0	0.3	1.1	1.4802
170	1000	3000	20000	1.5e-07	2.7e-09	0.0	2.1	2.1	1.5000

171	1000	3000	30000	1.5e-07	1.8e-09	0.0	2.1	2.1	1.5000

En la figura (12) mostramos las sensibilidades individuales S_i así como la sensibilidad total S que por razones de espacio no mostramos en la tabla precedente.

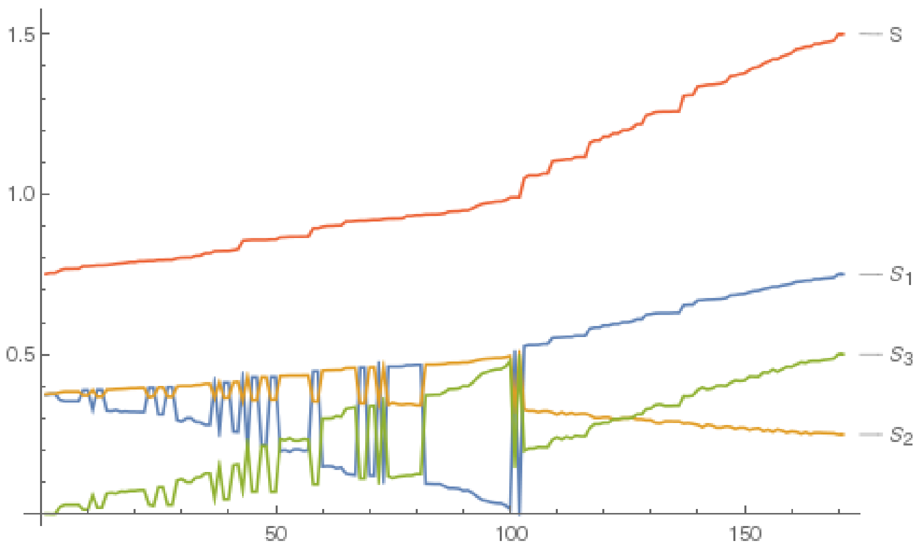


Figura 12: Sensibilidades en función del índice n que ordena las soluciones en orden creciente de sensibilidad total.

Para el escenario 1, el número de soluciones es 333, siendo el tiempo de cómputo de 60 segundos. Por razones de espacio a continuación mostramos sólo las mejores diez soluciones.

n	R_1	R_2	R_3	C_4	C_5	ϵ_G	ϵ_ω	ϵ_Q	S
1	1870	5620	1400	1.6e-07	2.0e-08	0.2	0.3	0.1	0.7506
2	18700	56200	14000	1.6e-08	2.0e-09	0.2	0.3	0.1	0.7506
3	24900	75000	18700	1.2e-08	1.5e-09	0.4	0.2	0.1	0.7509
4	2490	7500	1870	1.2e-07	1.5e-08	0.4	0.2	0.1	0.7509
5	2550	7680	1820	1.2e-07	1.5e-08	0.4	0.3	0.2	0.7570
6	25500	76800	18200	1.2e-08	1.5e-09	0.4	0.3	0.2	0.7570
7	1960	5900	1330	1.6e-07	2.0e-08	0.4	0.4	0.2	0.7632
8	19600	59000	13300	1.6e-08	2.0e-09	0.3	0.4	0.2	0.7632
9	2430	7320	1910	1.2e-07	1.5e-08	0.4	0.3	0.2	0.7680
10	24300	73200	19100	1.2e-08	1.5e-09	0.4	0.3	0.2	0.7680

A.2. Óptimos de Pareto

A continuación mostramos los óptimos de Pareto del escenario 2, que resultan ser 128 en total. El índice n corresponde a la solución respectiva del apéndice A.1. Para cada solución se muestran las cantidades $S_i = |S_{R_i}^{Q}|$ utilizadas para evaluar el criterio de dominancia (14) entre soluciones.

n	S_1	S_2	S_3	n	S_1	S_2	S_3
1	0.3739	0.3754	0.0015	71	0.1210	0.4600	0.3390
2	0.3759	0.3770	0.0011	73	0.1137	0.4612	0.3475
3	0.3759	0.3770	0.0011	74	0.4619	0.3425	0.1193
4	0.3576	0.3808	0.0233	76	0.4620	0.3443	0.1178
5	0.3534	0.3834	0.0300	80	0.4672	0.3410	0.1262
6	0.3534	0.3834	0.0300	81	0.4672	0.3410	0.1262
9	0.3876	0.3722	0.0154	82	0.0960	0.4684	0.3724
10	0.3876	0.3722	0.0154	83	0.0954	0.4684	0.3730
11	0.3289	0.3882	0.0592	84	0.0947	0.4684	0.3738
12	0.3889	0.3676	0.0213	86	0.0882	0.4700	0.3818
13	0.3889	0.3676	0.0213	87	0.0817	0.4728	0.3911
14	0.3241	0.3895	0.0654	89	0.0804	0.4735	0.3931
15	0.3227	0.3901	0.0674	90	0.0742	0.4748	0.4006
16	0.3261	0.3913	0.0652	91	0.0750	0.4750	0.4000
17	0.3204	0.3920	0.0716	92	0.0668	0.4780	0.4112
18	0.3218	0.3927	0.0709	93	0.0556	0.4817	0.4261
19	0.3203	0.3941	0.0738	94	0.0451	0.4846	0.4395
20	0.3203	0.3941	0.0738	95	0.0420	0.4862	0.4442
21	0.3198	0.3953	0.0756	96	0.0380	0.4871	0.4491
22	0.3198	0.3953	0.0756	97	0.0353	0.4884	0.4531
23	0.3959	0.3654	0.0305	98	0.0347	0.4886	0.4539
25	0.3120	0.3971	0.0851	99	0.0291	0.4904	0.4613
26	0.3120	0.3971	0.0851	100	0.0157	0.4947	0.4790
27	0.3976	0.3675	0.0301	101	0.4950	0.3317	0.1634
28	0.3976	0.3675	0.0301	102	0.0144	0.4951	0.4807
29	0.2941	0.4000	0.1060	103	0.5253	0.3270	0.1982
30	0.2903	0.4008	0.1105	104	0.5294	0.3235	0.2059
31	0.3003	0.4010	0.1007	105	0.5294	0.3235	0.2059
33	0.2889	0.4037	0.1148	106	0.5296	0.3191	0.2105
34	0.2877	0.4041	0.1164	107	0.5323	0.3226	0.2097
35	0.2794	0.4080	0.1286	108	0.5323	0.3226	0.2097
36	0.2794	0.4080	0.1286	109	0.5514	0.3182	0.2332
37	0.4111	0.3630	0.0482	110	0.5526	0.3116	0.2410
38	0.2604	0.4112	0.1509	112	0.5544	0.3115	0.2429
39	0.4115	0.3642	0.0473	113	0.5545	0.3166	0.2380
40	0.4115	0.3642	0.0473	117	0.5804	0.3100	0.2704
41	0.2587	0.4128	0.1540	118	0.5839	0.3012	0.2827
42	0.2576	0.4141	0.1566	122	0.5949	0.2975	0.2975
43	0.4274	0.3547	0.0726	123	0.5949	0.2975	0.2975

44	0.2102	0.4282	0.2180	127	0.6094	0.2969	0.3125
45	0.4286	0.3571	0.0714	128	0.6094	0.2969	0.3125
46	0.4286	0.3571	0.0714	129	0.6231	0.2944	0.3287
47	0.2156	0.4287	0.2131	130	0.6253	0.2941	0.3312
48	0.2156	0.4287	0.2131	131	0.6282	0.2864	0.3418
49	0.4293	0.3586	0.0707	132	0.6289	0.2851	0.3437
50	0.4293	0.3586	0.0707	133	0.6290	0.2903	0.3387
51	0.1973	0.4326	0.2353	134	0.6290	0.2903	0.3387
52	0.2012	0.4329	0.2317	135	0.6295	0.2902	0.3393
53	0.1954	0.4336	0.2382	136	0.6295	0.2902	0.3393
54	0.2030	0.4336	0.2306	137	0.6537	0.2845	0.3693
55	0.2030	0.4336	0.2306	139	0.6552	0.2833	0.3719
56	0.1985	0.4338	0.2353	140	0.6683	0.2797	0.3886
57	0.2005	0.4339	0.2334	141	0.6691	0.2725	0.3966
58	0.4467	0.3527	0.0939	143	0.6711	0.2780	0.3931
59	0.4467	0.3527	0.0939	144	0.6721	0.2709	0.4012
60	0.1498	0.4490	0.2991	148	0.6856	0.2669	0.4187
63	0.1440	0.4510	0.3070	149	0.6877	0.2708	0.4169
64	0.1463	0.4512	0.3049	150	0.6889	0.2704	0.4186
65	0.1277	0.4574	0.3298	151	0.6942	0.2637	0.4306
66	0.1249	0.4575	0.3326	154	0.7040	0.2603	0.4437
67	0.1229	0.4582	0.3354	156	0.7107	0.2581	0.4526
68	0.4588	0.3471	0.1118	159	0.7175	0.2558	0.4617
69	0.4588	0.3471	0.1118	163	0.7276	0.2523	0.4752
70	0.1218	0.4594	0.3376	165	0.7305	0.2513	0.4791

A.3. Versión Continua con R_1 fijado

Resultados obtenidos para la versión continua logarítmica con R_1 fijo. Los valores de R_1 que no aparecen en la tabla son aquellos para los cuales no se encontró solución.

R_1	R_2	R_3	C_4	C_5	ϵ_G	ϵ_ω	ϵ_Q
1000	3000	999	2.63e-07	3.22e-08	7.90e-10	2.48e-04	2.47e-04
1100	3300	999	2.49e-07	3.08e-08	4.14e-11	6.77e-05	9.57e-05
1200	3600	999	2.38e-07	2.96e-08	1.75e-07	1.23e-03	1.41e-03
1300	3900	999	2.28e-07	2.85e-08	9.42e-10	5.32e-05	1.50e-04
1500	4500	1125	2.00e-07	2.50e-08	8.76e-10	2.35e-04	3.65e-04
1600	4800	1200	1.88e-07	2.34e-08	1.26e-10	3.33e-05	1.57e-04
1800	5400	1350	1.67e-07	2.08e-08	1.60e-11	7.85e-05	2.63e-05
2000	6000	1500	1.50e-07	1.88e-08	7.73e-10	4.45e-05	5.27e-05
2200	6600	1650	1.36e-07	1.71e-08	1.88e-10	2.47e-04	2.82e-04
2400	7200	1800	1.25e-07	1.56e-08	1.24e-10	5.49e-05	3.37e-05
2700	8100	2025	1.11e-07	1.39e-08	4.30e-10	1.48e-04	8.27e-06
3000	9000	2250	1.00e-07	1.25e-08	8.06e-11	1.16e-04	1.93e-04
3300	9900	2475	9.09e-08	1.14e-08	2.80e-10	8.98e-05	1.94e-04

3600	10800	2700	8.34e-08	1.04e-08	7.70e-12	5.72e-07	4.10e-05
3900	11700	2925	7.70e-08	9.62e-09	8.45e-12	1.19e-05	8.73e-06
4300	12900	3225	6.98e-08	8.72e-09	4.74e-12	2.78e-06	2.55e-05
4700	14100	3525	6.39e-08	7.98e-09	1.00e-09	1.03e-04	9.60e-06
5100	15300	3825	5.88e-08	7.36e-09	4.12e-11	2.82e-05	7.46e-05
5600	16800	4200	5.36e-08	6.70e-09	4.52e-10	8.18e-05	3.80e-05
6200	18600	4650	4.84e-08	6.05e-09	1.67e-12	4.72e-06	1.32e-05
6800	20400	5100	4.41e-08	5.52e-09	1.13e-10	3.23e-05	1.62e-04
7500	22500	5625	4.00e-08	5.00e-09	1.53e-10	1.95e-04	4.87e-05
8200	24600	6150	3.66e-08	4.57e-09	4.12e-10	7.57e-05	2.17e-05
9100	27300	6825	3.30e-08	4.12e-09	1.91e-12	4.14e-05	1.37e-06
10000	30000	7500	3.00e-08	3.75e-09	4.20e-10	6.68e-05	2.55e-04
11000	33000	8250	2.73e-08	3.41e-09	9.36e-12	1.17e-06	2.66e-06
12000	36000	9000	2.50e-08	3.13e-09	1.12e-09	1.14e-04	9.85e-05
13000	39000	9750	2.31e-08	2.89e-09	3.87e-11	6.52e-05	8.28e-05
15000	45000	11250	2.00e-08	2.50e-09	2.90e-10	2.19e-05	9.84e-05
16000	48000	12000	1.88e-08	2.34e-09	3.63e-08	4.43e-04	6.98e-05
18000	54000	13500	1.67e-08	2.08e-09	1.09e-09	1.78e-05	1.07e-04
20000	60000	15000	1.50e-08	1.88e-09	4.46e-10	2.51e-05	8.23e-05
22000	66000	16500	1.36e-08	1.71e-09	1.87e-11	1.09e-04	1.29e-05
24000	72000	18000	1.25e-08	1.56e-09	1.61e-09	4.54e-05	5.33e-06
27000	81000	20250	1.11e-08	1.39e-09	2.51e-11	2.66e-05	2.75e-05
30000	90000	22500	1.00e-08	1.25e-09	5.07e-11	4.60e-05	9.33e-05
33000	99000	24750	9.09e-09	1.14e-09	2.09e-10	2.45e-05	6.53e-05
36000	108000	27000	8.34e-09	1.04e-09	5.61e-11	7.99e-06	1.31e-06
39000	117000	29250	7.70e-09	1.00e-09	1.68e-07	1.93e+00	1.92e+00

Referencias

1. Corral, C.: Designing RC active filters with standard-component values. Electronic Design Network Magazine, 141-154 (2000)
2. Lovay, M., Romero, E., Peretti, M.: Diseño de Filtros Activos Robustos usando Algoritmos Genéticos. SII 2015 4° Simposio Argentino de Informática Industrial.
3. Dimopoulos, H.: Analog Electronics Filters: Theory, Design and Synthesis. Springer (2012)
4. Raut, R., Swamy, M. N. S.: Modern Analog Filter Analysis and Design: A Practical Approach. Wiley-VCH (2010).
5. Solnon, C.: Ant Colony Optimization and Constraint Programming. (Wiley-ISTE 2010)
6. Socha, K., Dorigo, M.: Ant colony optimization for continuous domains. European Journal of Operational Research 185 (2008)
7. Dorigo, M.: Optimization, Learning and Natural Algorithms (en italiano). Ph.D. thesis, Dipartimento di Elettronica, Politecnico di Milano, Italy (1992).
8. Dorigo, M., Maniezzo, V., Colorni, A.: Ant System: Optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics – Part B 26 (1), 29–41. (1996)

9. Stützle, T., Dorigo, M.: ACO algorithms for the traveling salesman problem. En: Miettinen, K., Mäkelä, M.M., Neittaanmäki, P., Périaux, J. (Eds.), Evolutionary Algorithms in Engineering and Computer Science. John Wiley and Sons, Chichester, UK, pp. 163–183 (1999)
10. Dorigo, M., Stützle, T.: Ant Colony Optimization. MIT Press, Cambridge, MA (2004)
11. Guntsch, M., Middendorf, M.: A population based approach for ACO. Applications of Evolutionary Computing, Proceedings of EvoWorkshops 2002: EvoCOP, EvoIASP, EvoSTim, vol. 2279 of LNCS, pp. 71-80 (2002)
12. Freuder E., Wallace R.: Partial constraint satisfaction. Artificial Intelligence, vol. 58, pp. 21–70 (1992)
13. Shiex T., Fargier H., Verfaillie G.: Valued constraint satisfaction problems: hard and easy problems. International Joint Conference on Artificial Intelligence (IJCAI),MIT Press, Cambridge, Etats- Unis, p. 631–637, (1995)
14. Goss, S., Aron, S., Deneubourg, J.-L., Pasteels, J.: Self-organized shortcuts in the Argentine ant. Naturwissenschaften 76, 579–581 (1989)