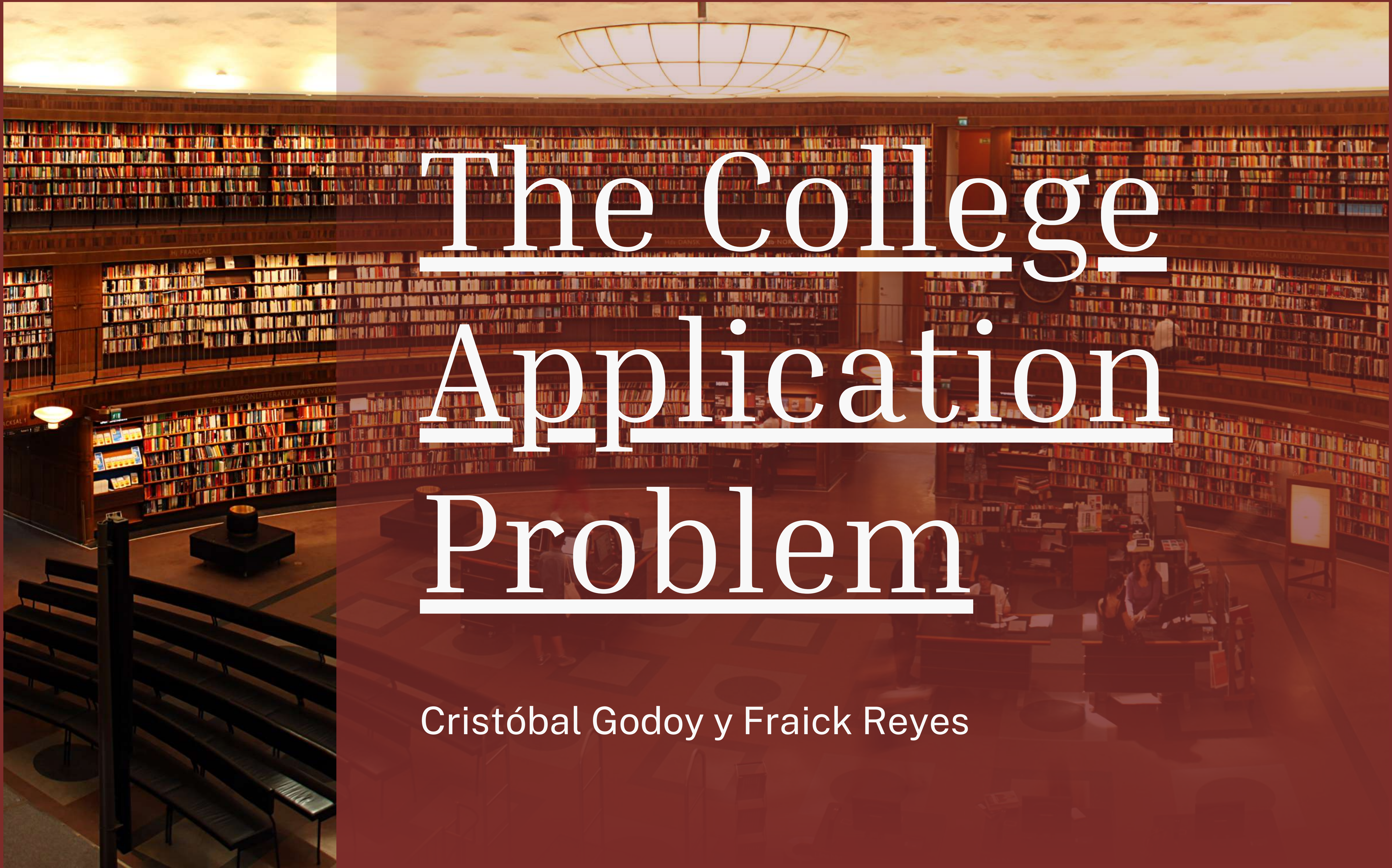


The College Application Problem

Cristóbal Godoy y Fraick Reyes

Introducción del problema

$$\begin{aligned} \text{maximize} \quad & v(\mathcal{X}) = \mathbb{E}[max\{t_j Z_j \;:\; j \in \mathcal{X}\}] \\ \text{subject to} \quad & \mathcal{X} \subseteq \mathcal{C}, \quad \sum_{j \in \mathcal{X}} g_j \leq H \end{aligned}$$

Donde $\mathcal{C} = \{1 \ldots m\}$ es una lista de índices que representan las universidades a las que vamos a postular, $H > 0$ es el presupuesto; para $j = 1 \ldots m$, $g_j, t_j > 0$ son el costo y la utilidad de postular a la universidad j respectivamente y $Z_j \sim Ber(f_j)$.

$$\begin{aligned} \text{maximize} \quad & v(\mathcal{X}) = \sum_{j \in \mathcal{X}} \Big(f_j t_j \prod_{\substack{i \in \mathcal{X}: \\ i > j}} (1 - f_i) \Big) \\ \text{subject to} \quad & \mathcal{X} \subseteq \mathcal{C}, \quad \sum_{j \in \mathcal{X}} g_j \leq H \end{aligned}$$

Simulated Annealing

Input: Utility values $t(0,\infty)^m$, admissions probabilities $f \in (0,1]^m$, application costs $g \in (0,\infty)^m$, budget $H \in (0,\infty)^m$.

Parameters: Initial temperature $T \geq 0$, temperature reduction parameter $r \in (0,1]$, number of iterations N.

Add schools to $\mathcal{X}$ in descending order by $f_j t_j / g_j$ until the budget is exhausted;
for $i = 1 \ldots N$ **do**
    $\mathcal{X}_n \leftarrow \text{copy}(\mathcal{X})$;
    Add random schools from $\mathcal{C} \setminus \mathcal{X}$ to $\mathcal{X}_n$ until $\mathcal{X}_n$ infeasible;
    Remove random schools in $\mathcal{X}$ from $\mathcal{X}_n$ until feasibility restored;
    $\Delta = v(\mathcal{X}_n) - v(\mathcal{X})$;
    if $\Delta \geq 0$ **then** $\mathcal{X} \leftarrow \mathcal{X}_n$ **else** $\mathcal{X} \leftarrow \mathcal{X}_n$ with probability $\exp(\Delta/T)$;
    $T \leftarrow rT$;
end
return *the best* $\mathcal{X}$ *found*

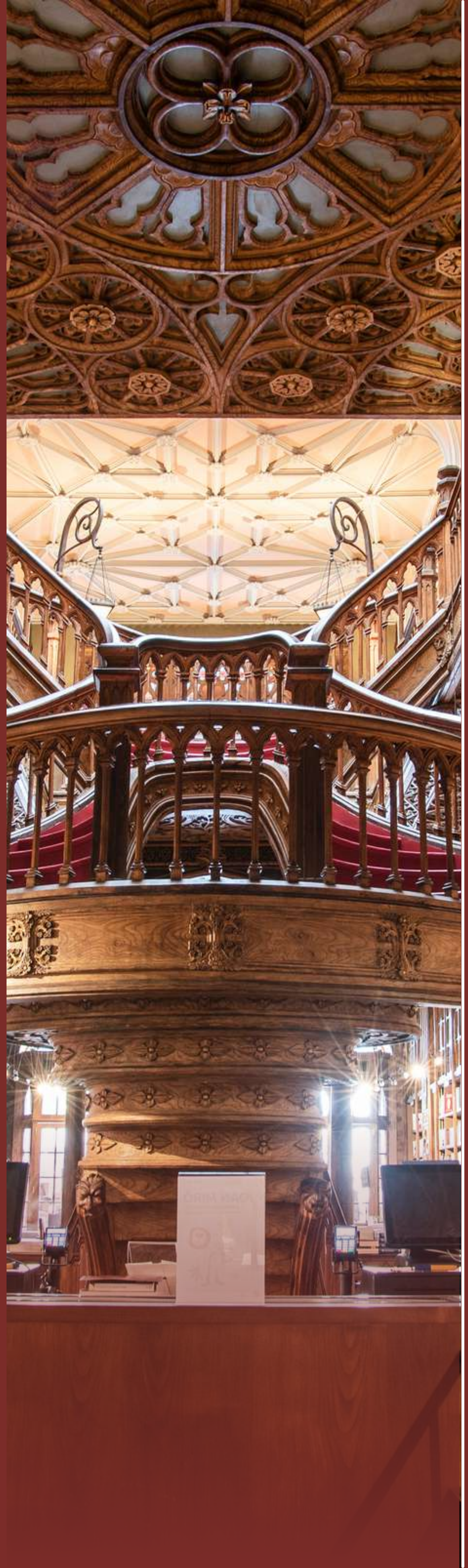
Generación de datos

Para obtener datos generamos $(t_j)_j \sim Exp(10)$ aproximando al entero superior más cercano. Para obtener correlación negativa entre t_j y f_j, definimos $f_j = 1/(t_j + 10Q_j)$, donde $Q_j \sim Unif([0,1))$. Los $(g_j)_j \sim Unif(\{5,\ldots,10\})$ y definimos $H = \lfloor \frac{1}{2} \sum g_j \rfloor$.

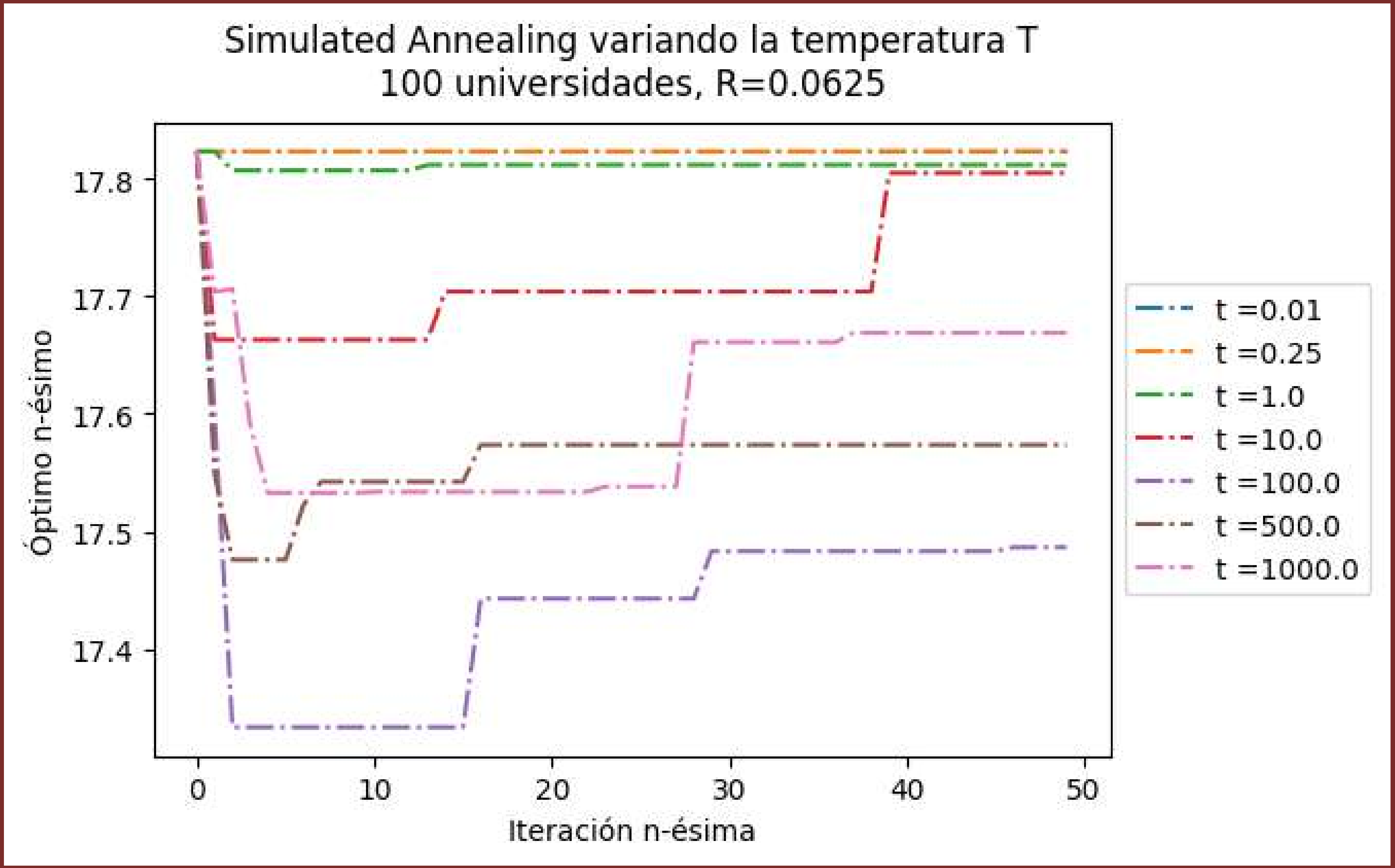
Un ejemplo de generar 10 universidades es:

```
Utilidades: [ 6.  6. 27. 19.  3.  4.  6.  3.  2. 15.]
Costos: [ 6  6 10  5 10  9  6 10 10 10]
Presupuesto: 41
Parámetro Bernoulli: [0.09565584 0.09565584 0.03179231 0.04263639 0.13415356 0.11828518
 0.09565584 0.13415356 0.1549392  0.05140293]
```

Para lo siguiente fijamos m = 500 universidades.

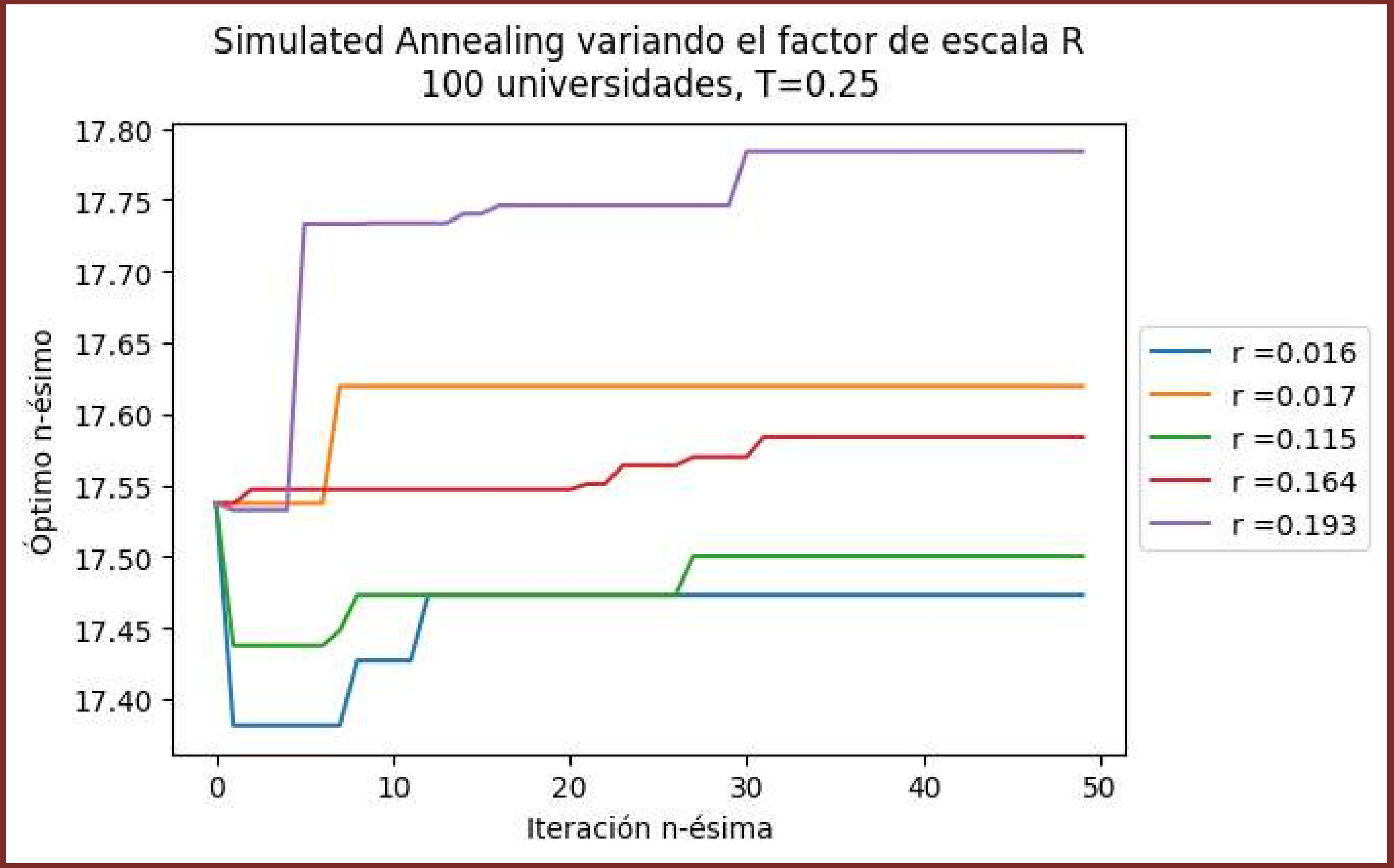


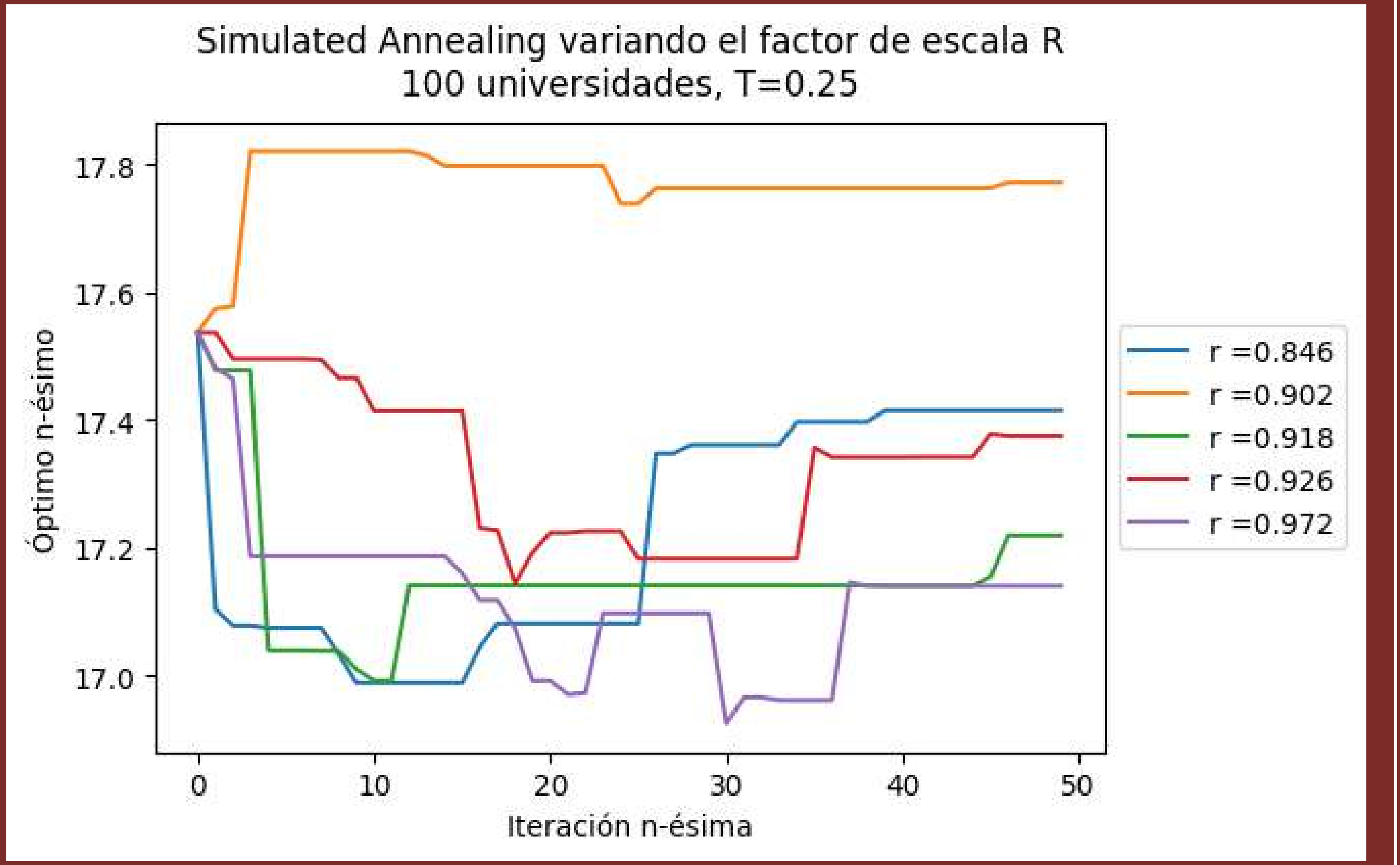
Análisis del Algoritmo



Análisis del Algoritmo

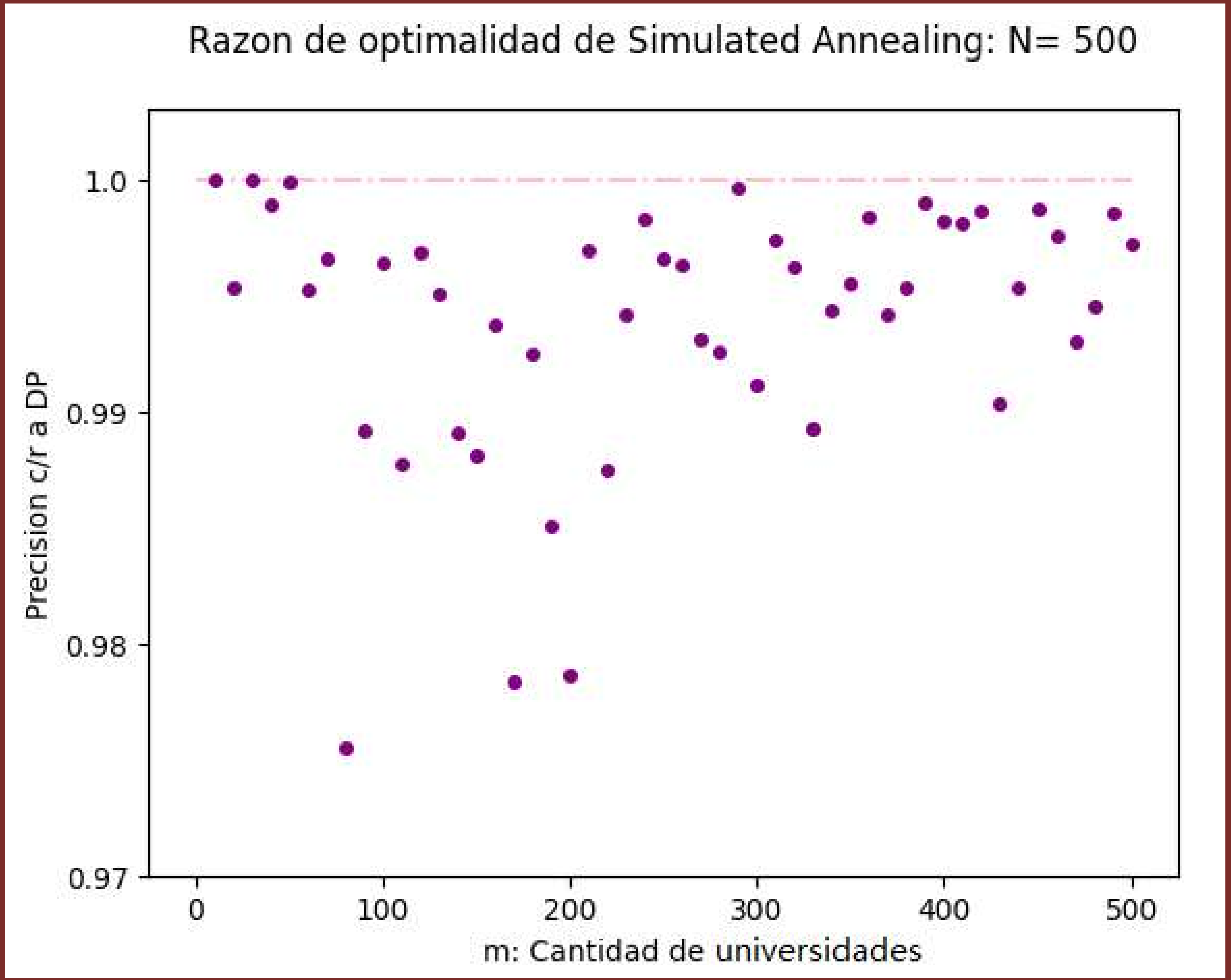
Variando factor de escala con temperatura T = 1/4:





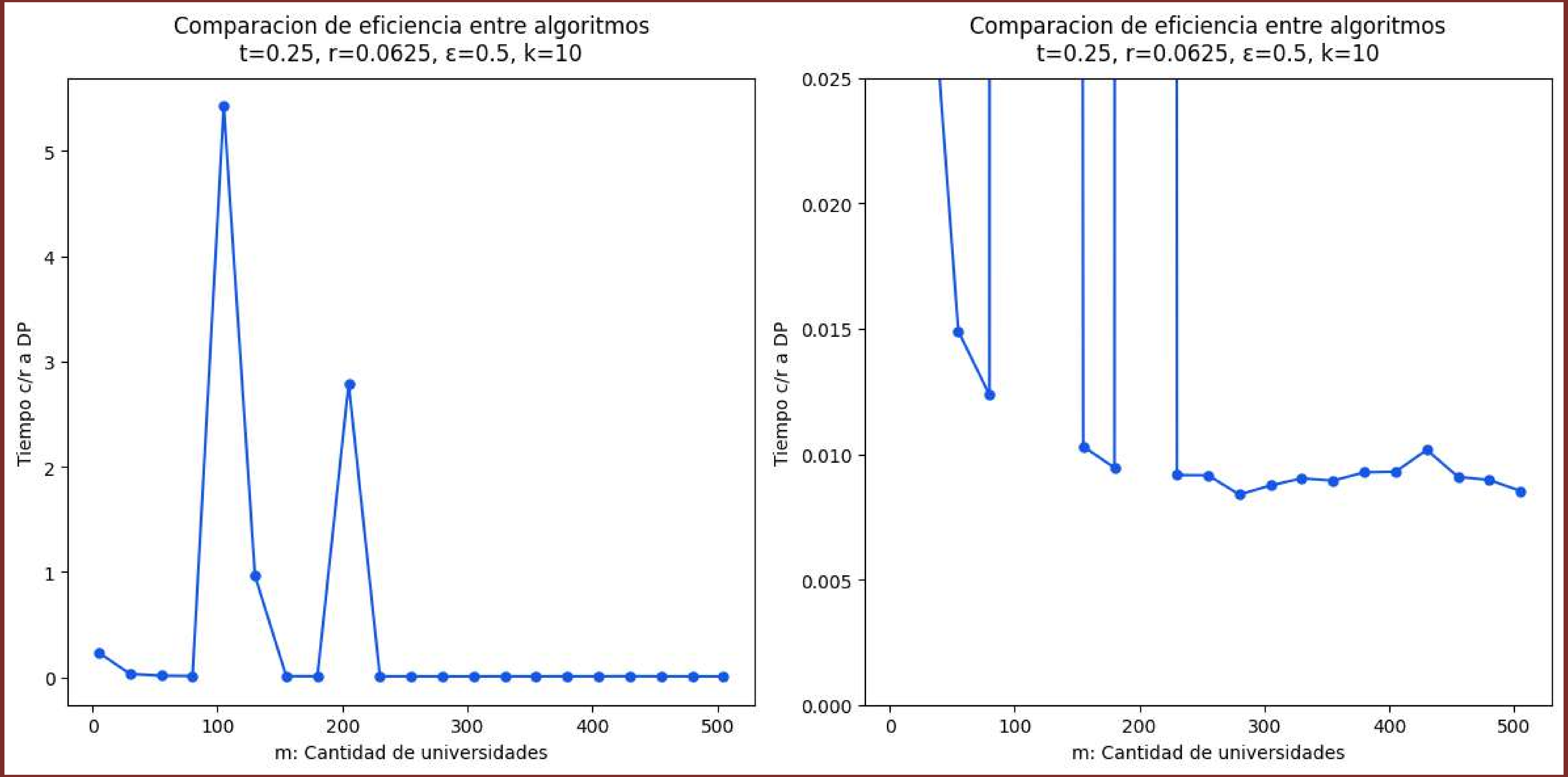
Comparación con Programación Dinámica

Test de eficacia:



Comparación con Programación Dinámica

Test de eficiencia:



Variante de Función Objetivo

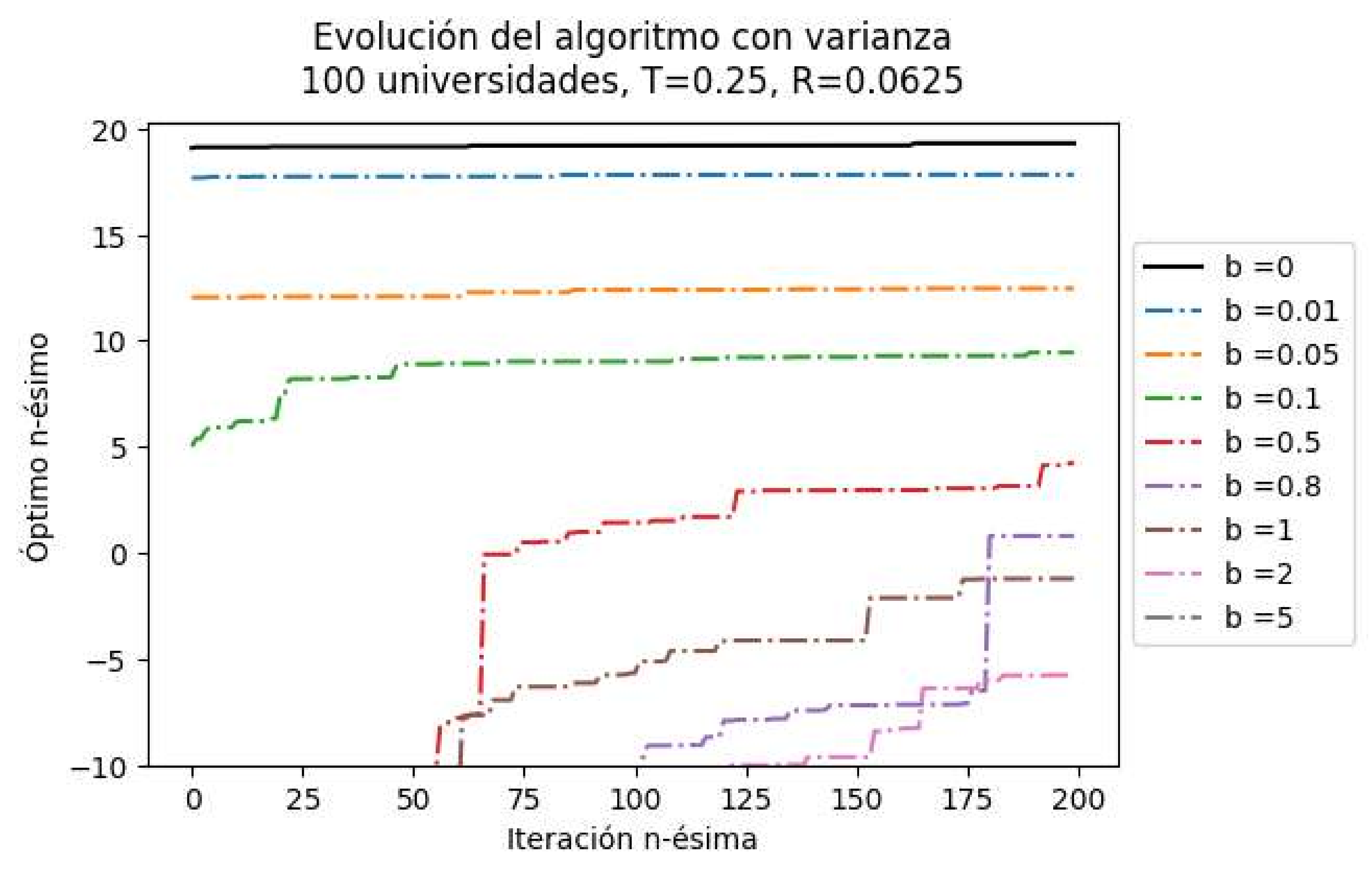
Podemos modificar la función objetivo para incorporar una penalización por la varianza $\beta \geq 0$, con lo siguiente:

$$v_\beta(\mathcal{X}) = \mathrm{E}[X] - \beta \operatorname{Var}(X) = v(\mathcal{X}; \tau) + \beta v(\mathcal{X}; t)^2$$

Con $\tau_j = t_j - \beta t_j^2$.

A continuación estudiaremos el comportamiento de las soluciones variando este parametro de penalización.

Variante de Función Objetivo



Variante de Función Objetivo

Quisimos ver si cambiaban las soluciones óptimas dada la penalización, mostramos un caso con 10 universidades:

```
===========================================================================
f= [0.15  0.131 0.161 0.074 0.082 0.051 0.056 0.04  0.025 0.021]
b= 0:  [0. 0. 0. 0. 0. 1. 1. 1. 1. 1.]
b= 0.01 : [0. 0. 0. 0. 1. 1. 1. 0. 1. 1.]
b= 0.05 : [0. 0. 1. 1. 1. 1. 1. 0. 0. 0.]
b= 0.1 : [0. 1. 1. 1. 1. 1. 0. 0. 0. 0.]
b= 0.5 : [1. 1. 1. 1. 0. 1. 0. 0. 0. 0.]
b= 0.8 : [1. 1. 1. 1. 0. 1. 0. 0. 0. 0.]
b= 1 : [1. 1. 1. 1. 0. 1. 0. 0. 0. 0.]
b= 2 : [1. 1. 1. 1. 0. 1. 0. 0. 0. 0.]
b= 5 : [1. 1. 1. 1. 0. 1. 0. 0. 0. 0.]
===========================================================================
```

Conclusiones

01

Eficacia

Simulated Annealing converge el óptimo salvo un error menor al 3% y la gran mayoría de veces menor al 1% .

02

Eficiencia

Simulated Annealing suele converger considerablemente más rápido que el algoritmo de programación dinámica para gran cantidad de universidades.

03

Variante

Tomar en cuenta el riesgo a postular puede afectar bastante en la solución óptima si la penalización es suficientemente grande.

Referencias:

[1] Max Kapur, Sung-Pil Hong (2022); The College Application Problem; arXiv:2205.01869

Extra

Algoritmo Programación Dinámica:

$$V[j,h] = \max\{V[j-1,h], (1-f_j)V[j-1,h-g_j] + f_j t_j\}$$

Algorithm 3: Dynamic program for Ellis's problem with integral application costs.

Input: Utility values $t \in (0,\infty)^m$, admissions probabilities $f \in (0,1]^m$, application costs $g \in \mathbb{N}^m$, budget $H \in \mathbb{N}$.

Index schools in ascending order by t;
Fill a lookup table with the values of $V[j,h]$;
$h \leftarrow H$;
$\mathcal{X} \leftarrow \emptyset$;
for $j = m, m-1, \ldots, 1$ **do**
    if $V[j-1,h] < V[j,h]$ **then**
        $\mathcal{X} \leftarrow \mathcal{X} \cup \{j\}$;
        $h \leftarrow h - g_j$;
    end
end
return $\mathcal{X}$