Búsqueda de vecindad variable para el problema de localización de instalaciones sin capacidad

XVI Congreso Español de Metaheurísticas, Algoritmos Evolutivos y Bioinspirados (MAEB)

Lucas Martín-García (lucas.martin@urjc.es)
Isaac Lozano-Osorio (isaac.lozano@urjc.es)
J. Manuel Colmenar (josemanuel.colmenar@urjc.es)
Belén Melián-Batista (mbmelian@ull.es)

Universidad Rey Juan Carlos

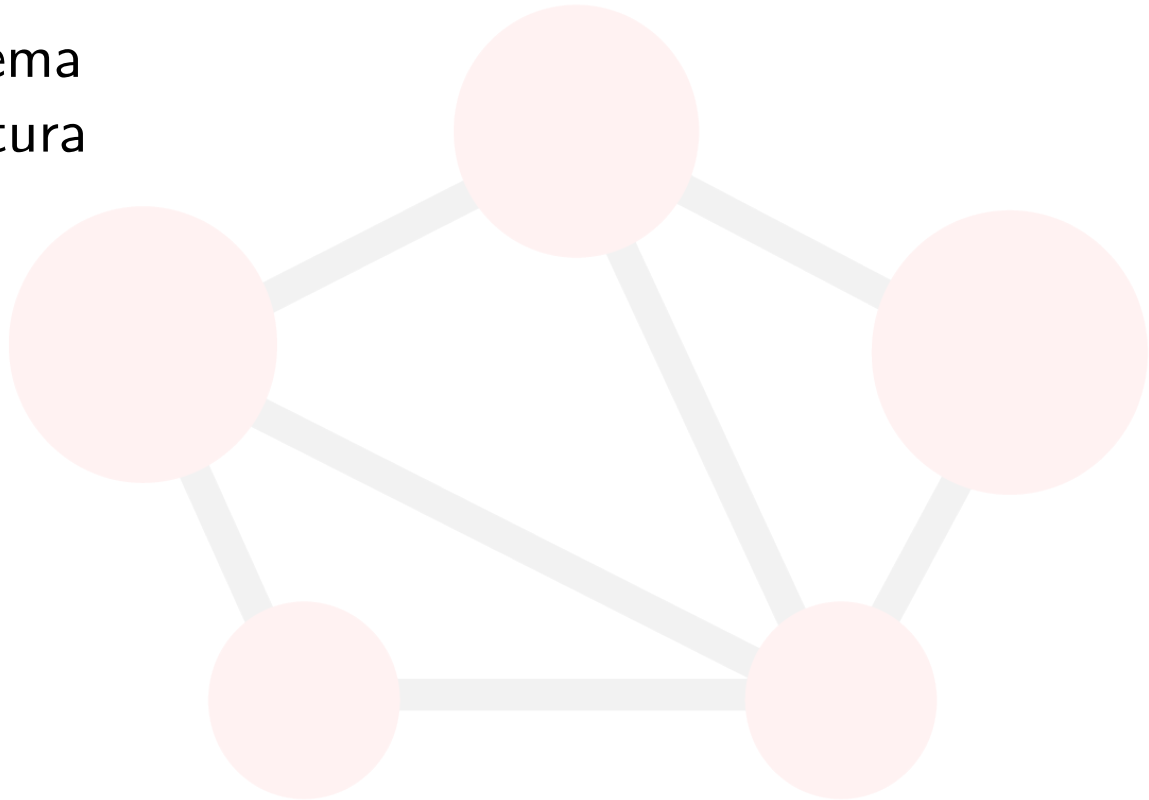
grafo RESEARCH

Universidad de La Laguna

MINISTERIO DE CIENCIA E INNOVACIÓN

AGENCIA ESTATAL DE INVESTIGACIÓN

Índice

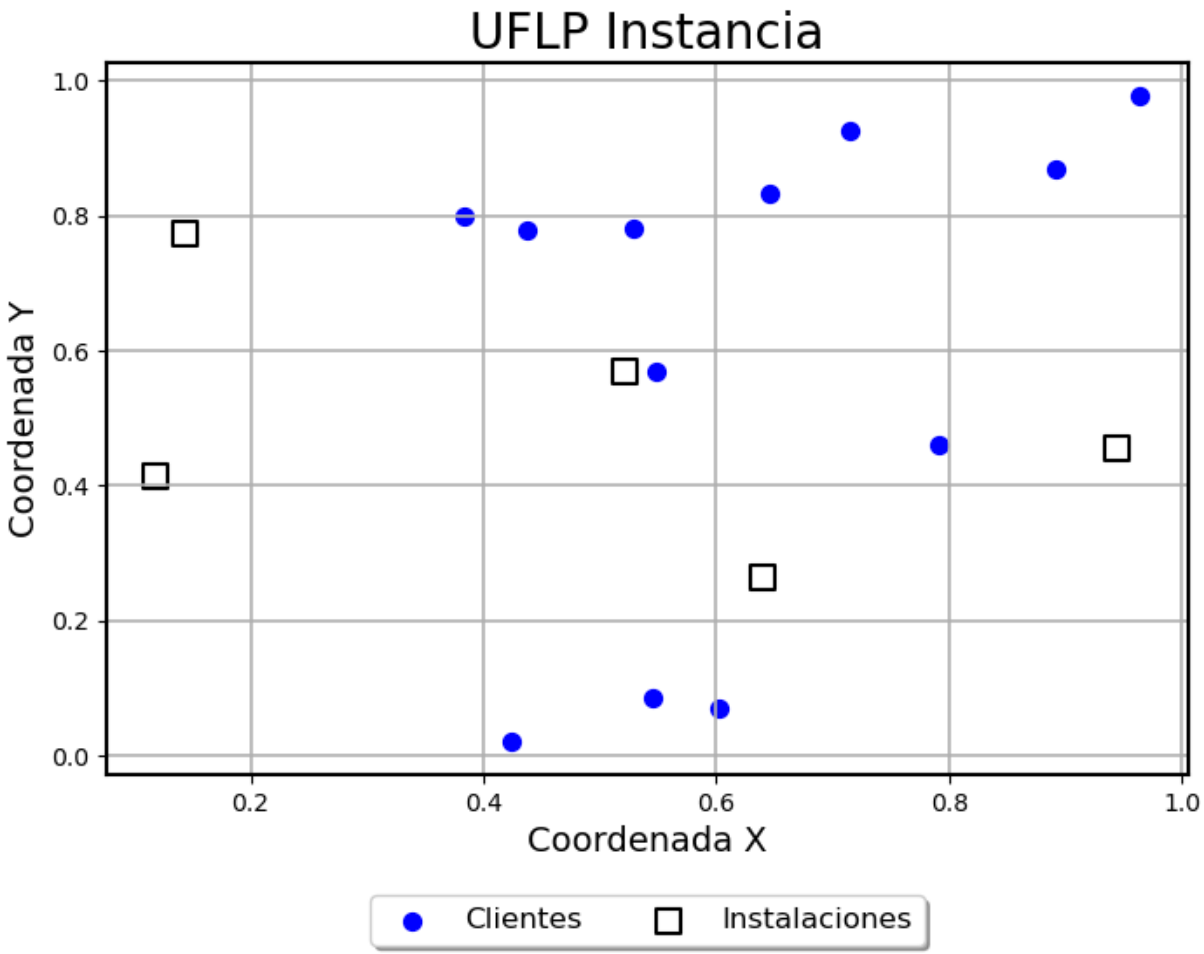


Introducción

- El problema **UFLP** (*Uncapacitated Facility Location Problem*) pertenece a la familia de problemas de **ubicación de instalaciones** (*Facility Location Problems*).
- Busca **minimizar** la suma del **coste de abrir instalaciones** más el **coste de asignar clientes** a estas instalaciones.
- Aplicaciones del mundo real:
  - Asignación de recursos y configuración de redes.
  - Logística y transporte.

Introducción

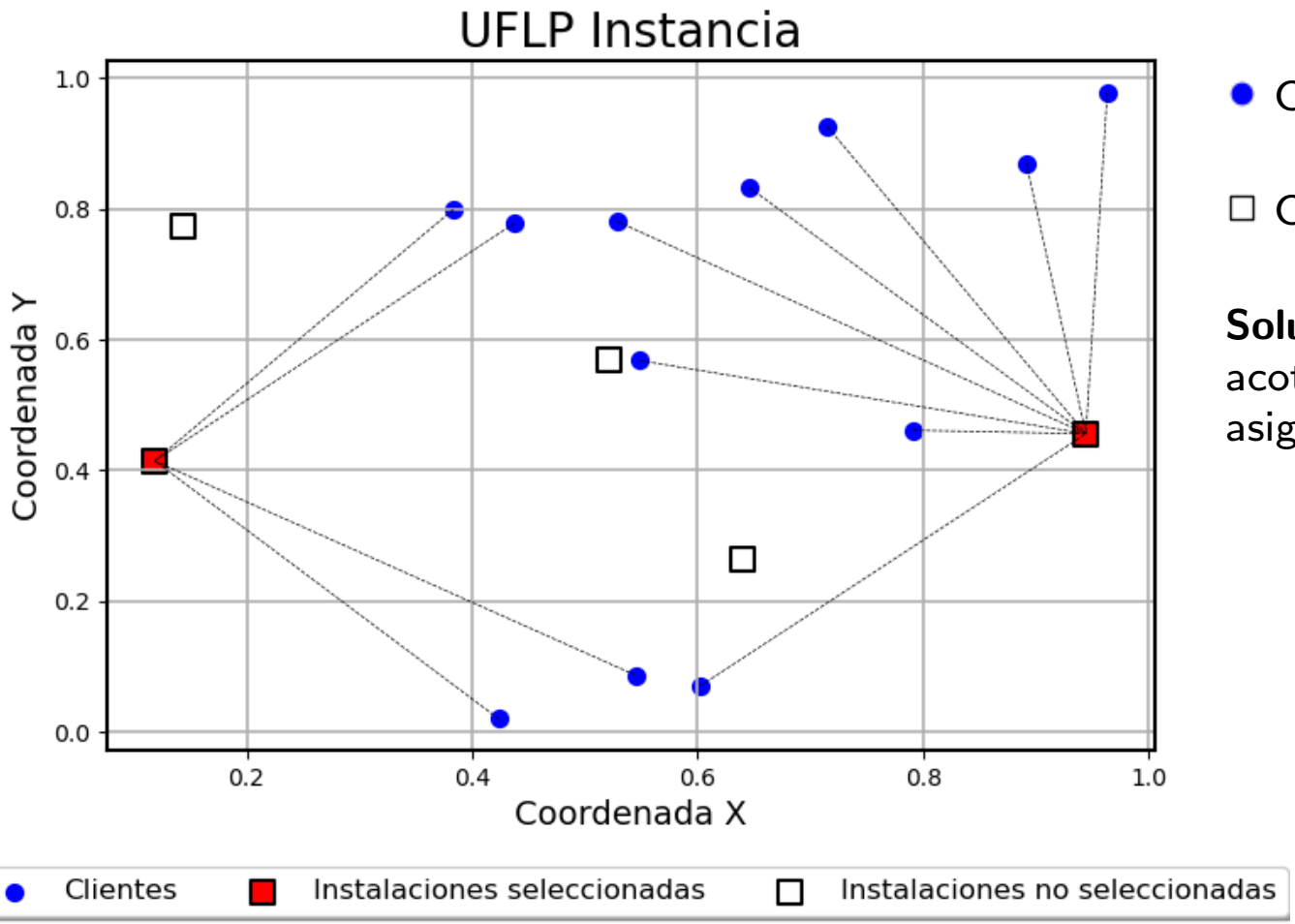
Representación de la solución



- Conjunto de **clientes**
- Conjunto de **instalaciones**

Introducción

Representación de la solución

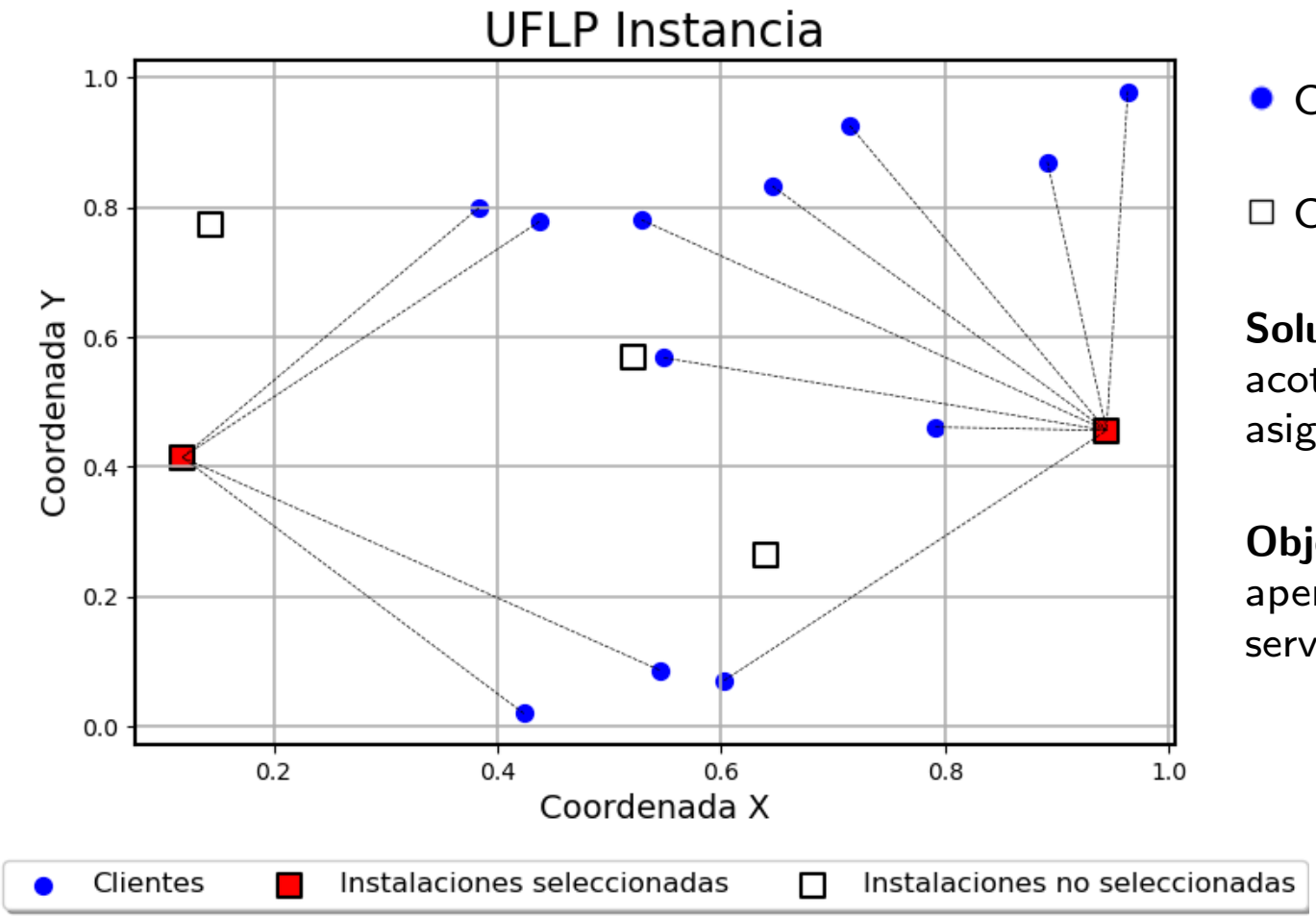


- Conjunto de **clientes**
- Conjunto de **instalaciones**

Solución: abrir un número no acotado de instalaciones y asignar los clientes.

Introducción

Representación de la solución



- Conjunto de **clientes**
- Conjunto de **instalaciones**

Solución: abrir un número no acotado de instalaciones y asignar los clientes.

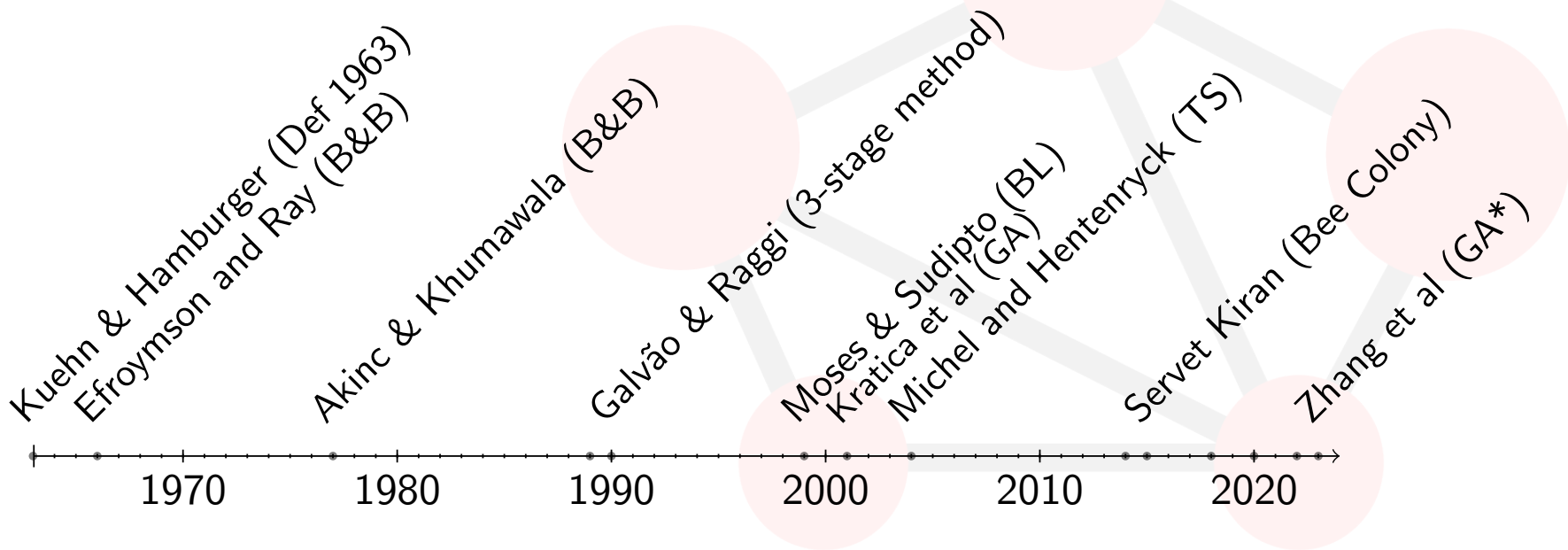
Objetivo: minimizar coste de apertura más coste de servicio.

$$\sum_{i=1}^{m}\sum_{j=1}^{n} d_{ij}y_{ij} + \sum_{j=1}^{n} g_j x_j$$

Introducción

Revisión de la literatura

Se ha demostrado ser *NP*-Difícil (Karp et al 1972).



Introducción

Formulación matemática

$$\text{Minimizar} \quad \sum_{i=1}^{m}\sum_{j=1}^{n} d_{ij}y_{ij} + \sum_{j=1}^{n} g_j x_j \tag{1}$$

Sujeto a

$$\sum_{j=1}^{n} y_{ij} = 1, \quad i = 1, 2, \ldots, m \tag{2}$$

$$y_{ij} \leq x_j, \quad i = 1, 2, \ldots, m, \quad j = 1, 2, \ldots, n \tag{3}$$

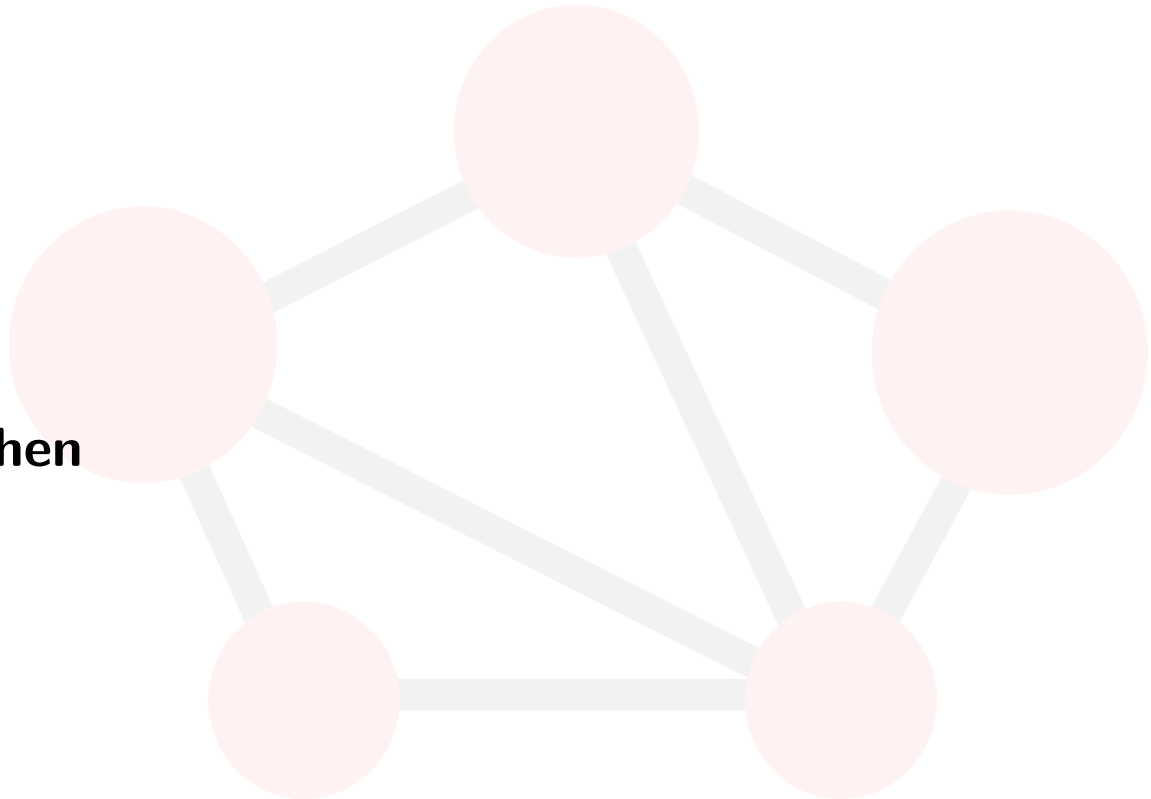
$$y_{ij}, x_j \in \{0, 1\}, \quad i = 1, 2, \ldots, m, \quad j = 1, 2, \ldots, n \tag{4}$$

Implementado en Gurobi.

BVNS

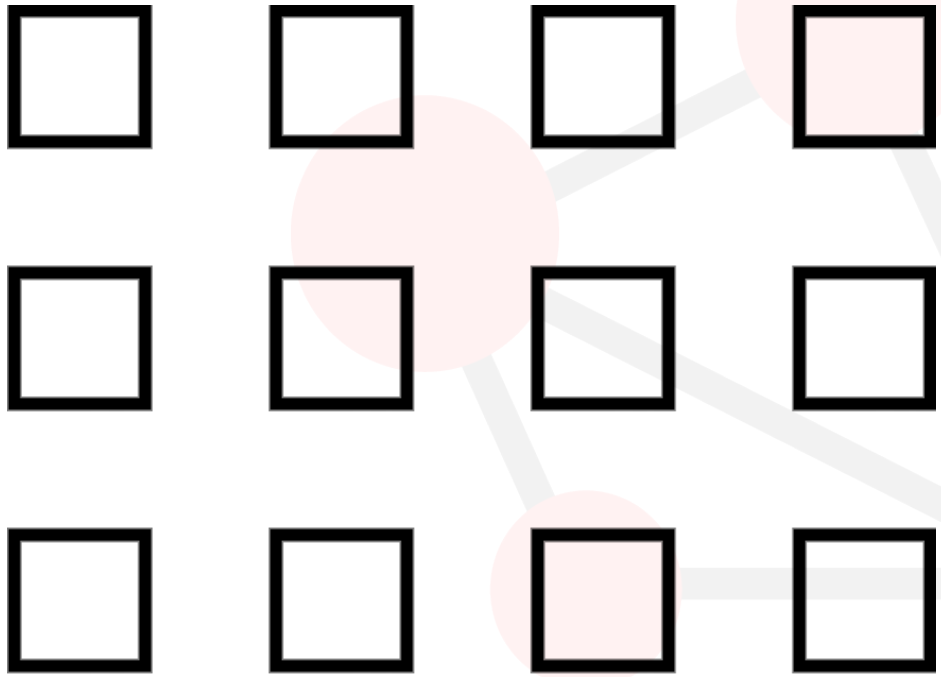
Algoritmo 1 BVNS(k_{max},N)

```
1: S ← Constructivo(N)
2: S* ← S
3: k ← 0
4: while k < k_max do
5:     S ← Shake(S,k)
6:     S ← Mejora(S)
7:     if F(S) < F(S*) then
8:         S* ← S
9:         k ← 1
10:    else
11:        k ← k + 1
12:    end if
13: end while
14: return S*
```



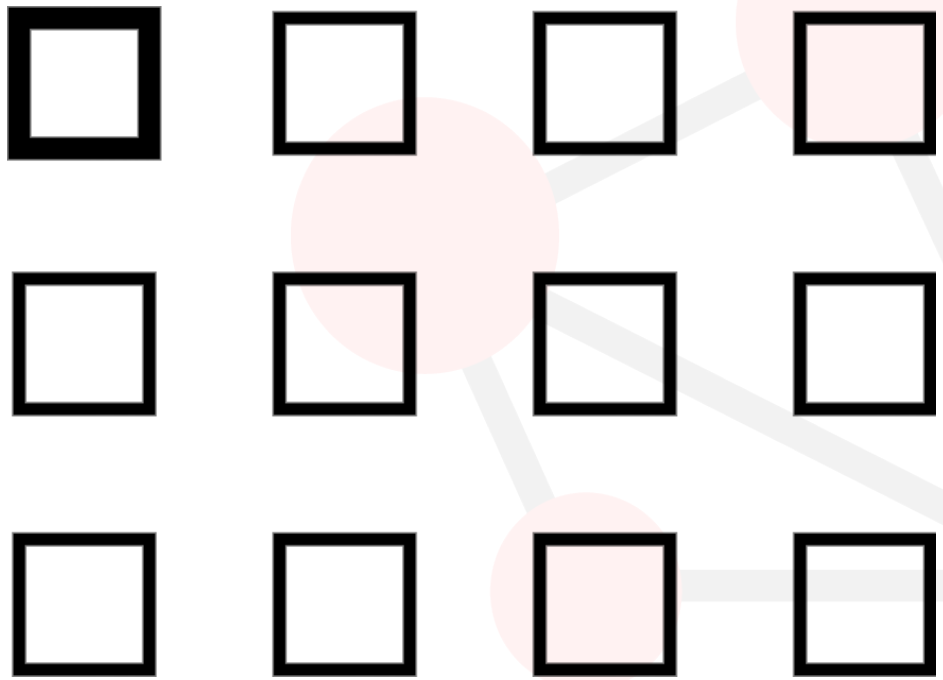
Constructivo

Constructivo aleatorio



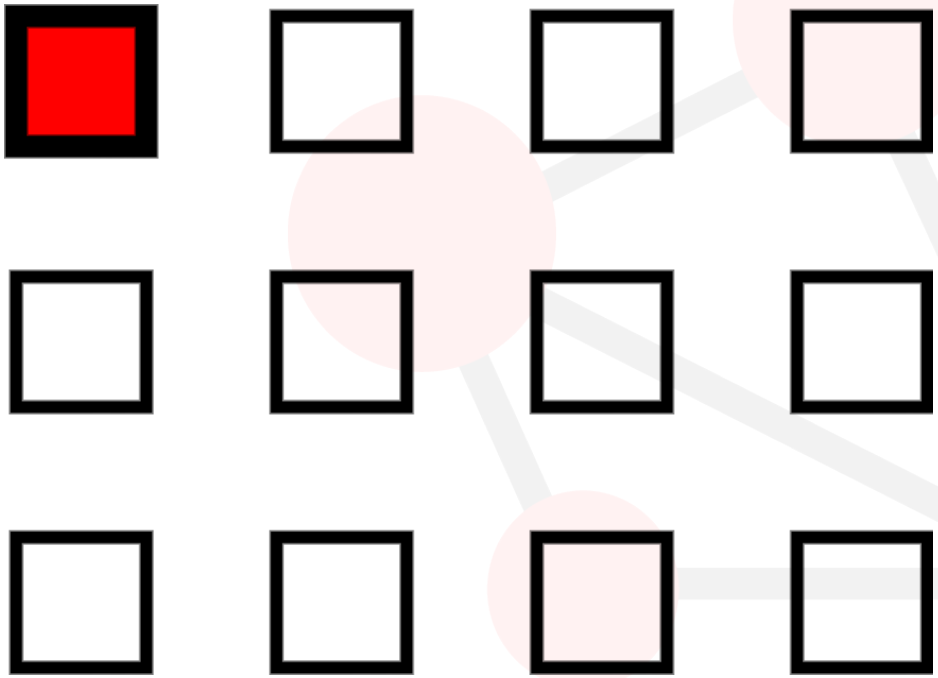
Constructivo

Constructivo aleatorio



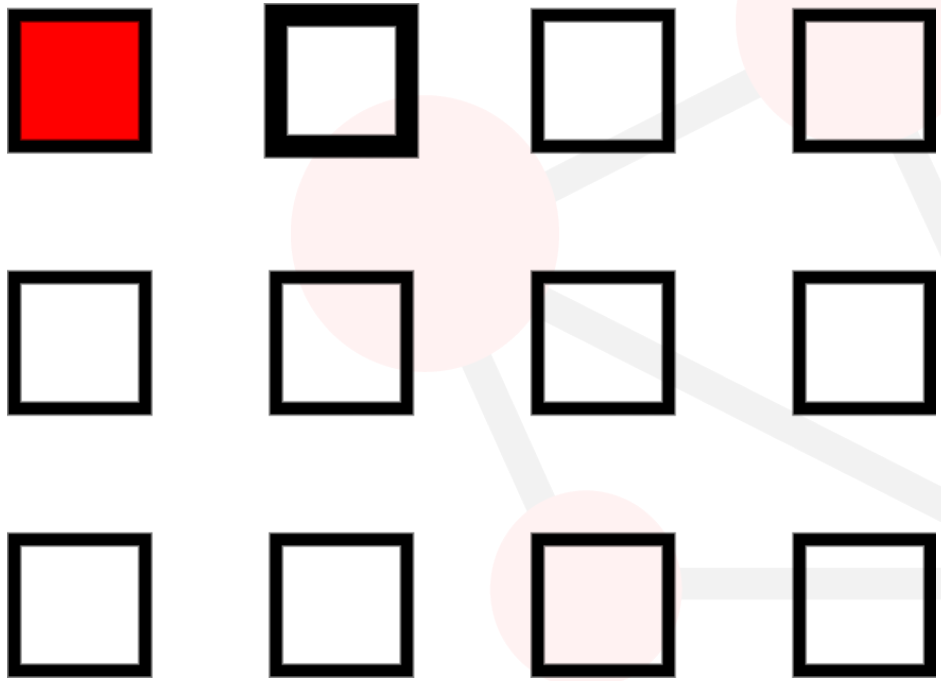
Constructivo

Constructivo aleatorio



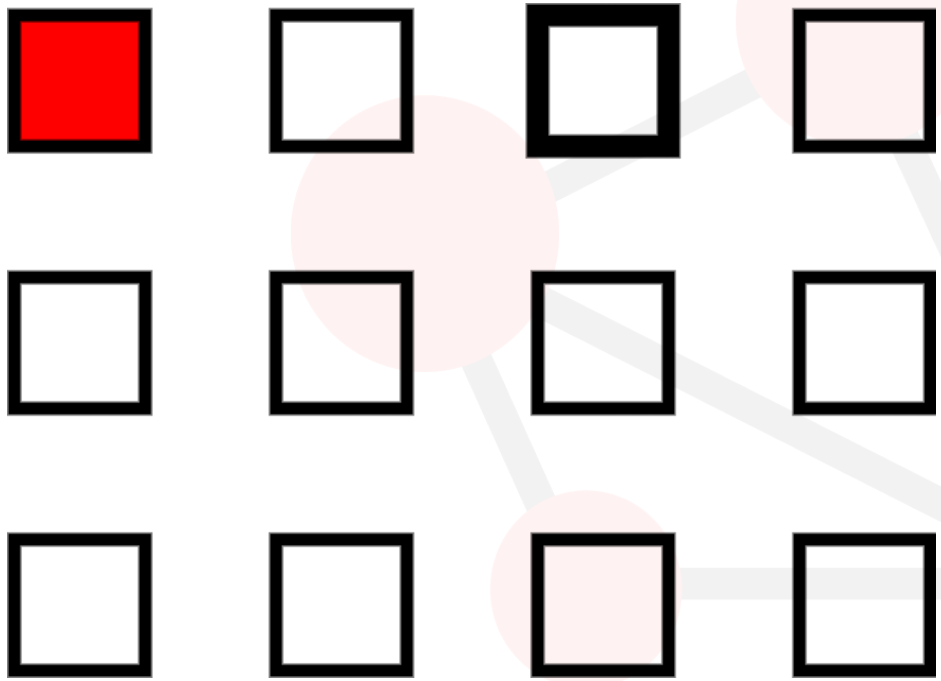
Constructivo

Constructivo aleatorio



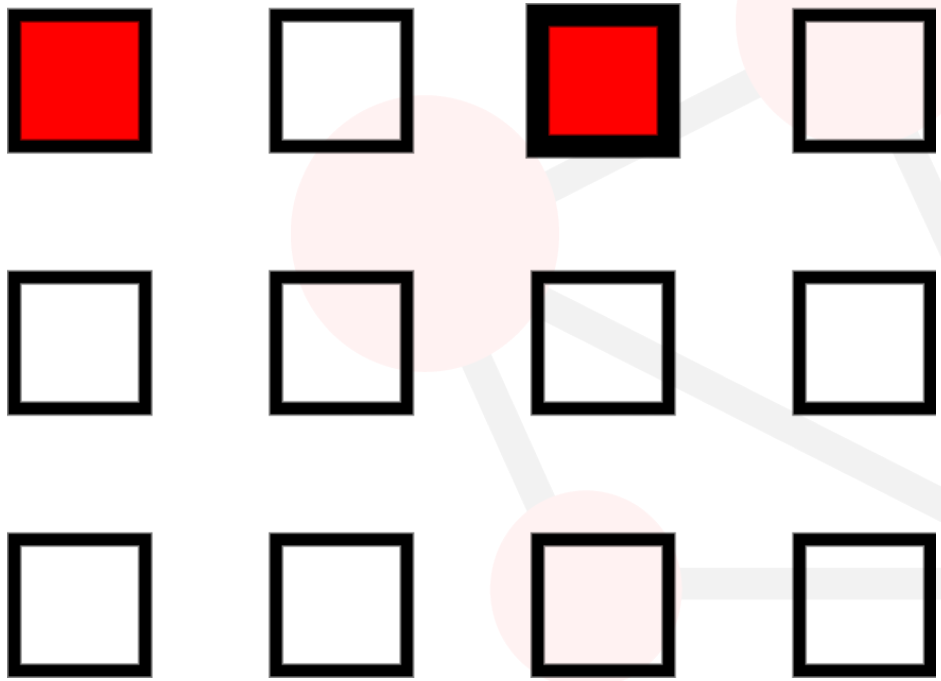
Constructivo

Constructivo aleatorio



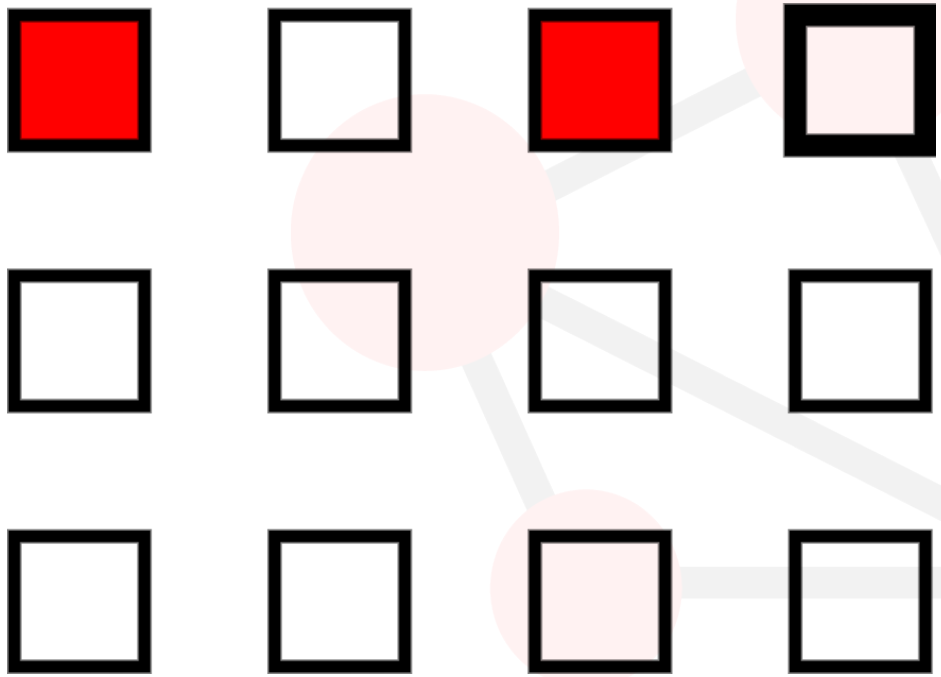
Constructivo

Constructivo aleatorio



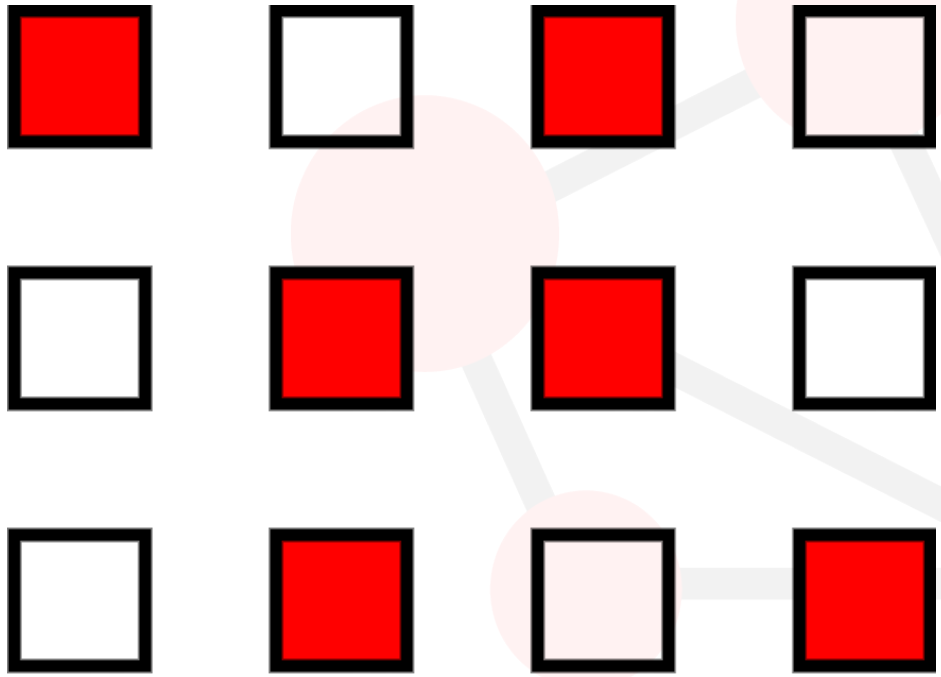
Constructivo

Constructivo aleatorio



Constructivo

Constructivo aleatorio



Método de mejora

Algoritmo 2 `Mejora(S)`

1: $mejora \leftarrow$ `Verdadero`
2: **while** *mejora* **do**
3: $mejora \leftarrow$ `Falso`
4: $x \leftarrow \arg\min_{i \in S} \mathcal{F}(S \setminus \{i\})$
5: $S' \leftarrow S \setminus \{x\}$
6: **if** $\mathcal{F}(S') < \mathcal{F}(S)$ **then**
7: $mejora \leftarrow$ `Verdadero`
8: $S \leftarrow S'$
9: **end if**
10: **end while**
11: **return** S

Shake

Algoritmo 3 `Shake(`S`,`k`)`

1: $C \leftarrow \{ a_i : a_i \in A \wedge a_i \notin S\}$
2: $C' \leftarrow$ `SeleccionAleatoria(C,k)`
3: $S \leftarrow S \cup C'$
4: **return** S

Resultados

- Lenguajes de programación: **Java 21**, **Python 3.8** y **Gurobi 11**.
- Características del servidor: AMD EPYC 7643 de 32 núcleos con 32 GB de memoria RAM.
- Instancias: 15 CAP (66-1100 nodos) 87 K90 (502-914 nodos) 700 G700 (3006-3906 nodos).
- Métricas:
  - $\mathcal{F}$: valor de la función objetivo.
  - **Tiempo (s)**: tiempo de ejecución en segundos.
  - **GAP. (%)**: desviación al valor óptimo.
  - **# Opt**: número total de instancias en las que el algoritmo es capaz de alcanzar la solución óptima.

Resultados preliminares

Búsqueda Local	K	$\mathcal{F}$	Tiempo (s)	GAP. (%)	# Opt
First improvement	0.1	3 201 593.58	11.66	0.8096	25
	0.2	3 191 440.99	13.90	0.6698	29
	0.3	3 185 544.26	17.43	0.6613	30
	0.4	3 187 689.18	22.45	0.6555	31
	0.5	3 177 980.34	30.48	0.5416	39
Best improvement	0.1	3 176 264.31	**2.95**	0.1385	55
	0.2	3 172 301.69	5.89	0.0470	80
	0.3	3 174 707.54	10.02	0.0619	77
	0.4	**3 170 593.02**	15.94	**0.0355**	87
	0.5	3 171 236.10	23.22	0.0385	**89**

Tabla 1: Resultados del algoritmo BVNS con diferentes métodos de mejora y valores de K sobre un 20% del conjunto de datos.

Resultados finales

Instancias	Algoritmo	$\mathcal{F}$	Tiempo (s)	GAP. (%)	# Opt
Cap	EGTOA (Zhang)	3 502 545.59	258.931	0.0000	15
	BVNS	3 509 998.78	0.027	0.0823	12
	Modelo Exacto	3 502 545.59	0.802	0.0000	15
K90	EGTOA (Zhang)	3 069 045.61	1 405.238	12.1286	0
	BVNS	2 768 151.54	0.321	0.0891	28
	Modelo Exacto	2 765 298.57	2.642	0.0000	87
G700	EGTOA (Zhang)	27 202 096.73	3 597.980	174.8163	23
	BVNS	2 947 040.14	16.881	0.0246	435
	Modelo Exacto	2 943 809.00	62.861	0.0000	700
Agregado	EGTOA (Zhang)	24 140 913.79	1 236.663	153.8985	38
	BVNS	2 938 163.67	**14.769**	0.0327	475
	Modelo Exacto	**2 934 894.58**	55.168	**0.0000**	**802**

Tabla 2: Resultados finales de los algoritmos EGTOA, BVNS y Exacto.

Conclusiones

El algoritmo **BVNS** obtiene soluciones competitivas con el estado del arte con un coste computacional reducido.

Trabajo futuro:

- Ampliar el estado del arte.
- Incorporar técnicas de aprendizaje automático para guiar la exploración del espacio de soluciones.
- Explorar aprendizaje por refuerzo para obtener información de las instancias durante la búsqueda de soluciones.

Agradecimientos

Este trabajo ha sido financiado gracias a:

- **Ministerio de Ciencia e Innovación** con ref. PID2021-126605NB-I00 y PID2021-125709OA-C22 financiado por MCIN /AEI /10.13039/501100011033 and by ERDF A way of making Europe.
- **CIRMA**: "This work has been supported by Comunidad Autónoma de Madrid, CIRMA-CM Project (TEC-2024/COM-404)".
- **Cátedra ENIA**: Ministerio para la Transformación Digital y de la Función Pública (Ref. TSI-100930-2023-3).

Búsqueda de vecindad variable para el problema de localización de instalaciones sin capacidad

XVI Congreso Español de Metaheurísticas, Algoritmos Evolutivos y Bioinspirados (MAEB)

Lucas Martín-García (lucas.martin@urjc.es)
Isaac Lozano-Osorio (isaac.lozano@urjc.es)
J. Manuel Colmenar (josemanuel.colmenar@urjc.es)
Belén Melián-Batista (mbmelian@ull.es)

Universidad Rey Juan Carlos

grafo RESEARCH

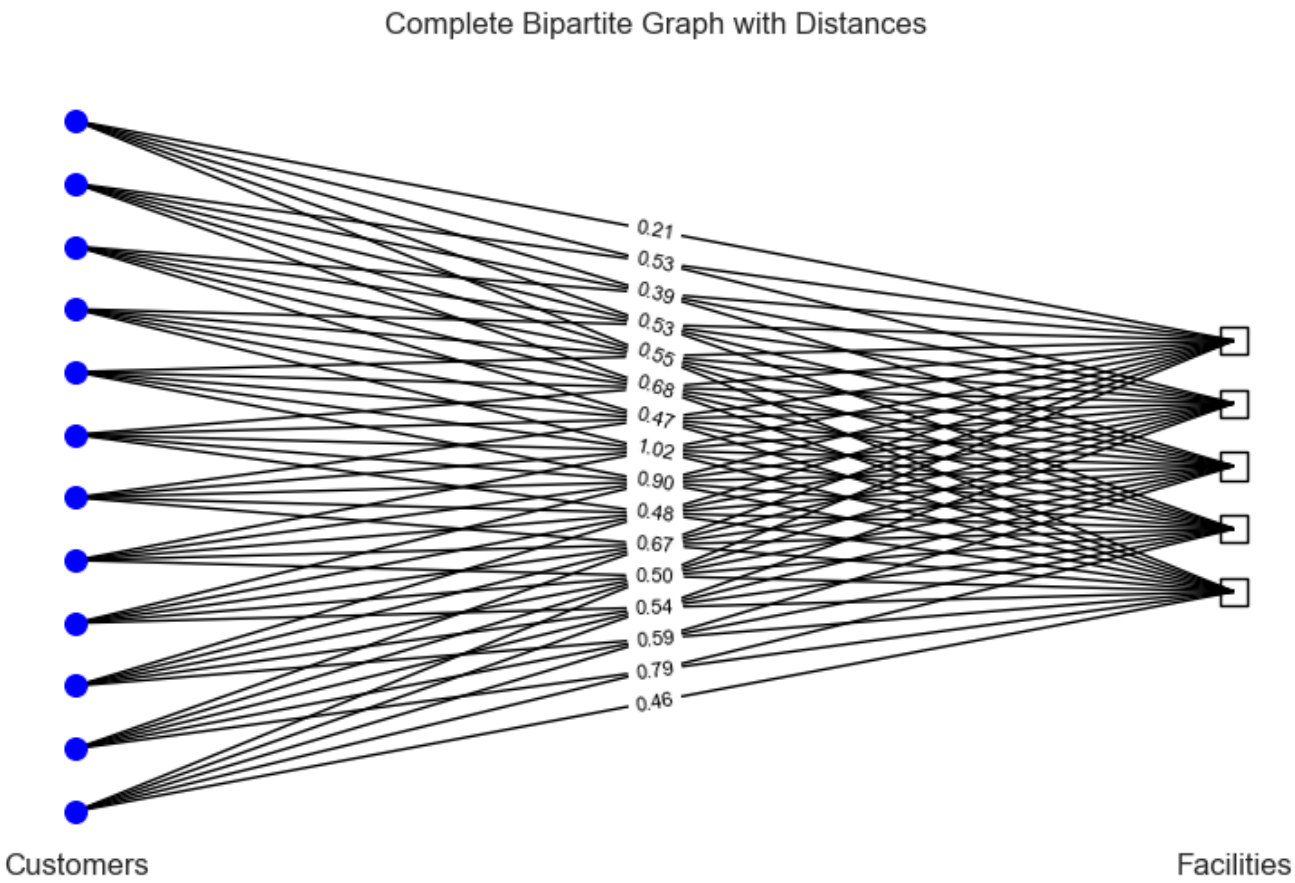
Universidad de La Laguna

MINISTERIO DE CIENCIA E INNOVACIÓN

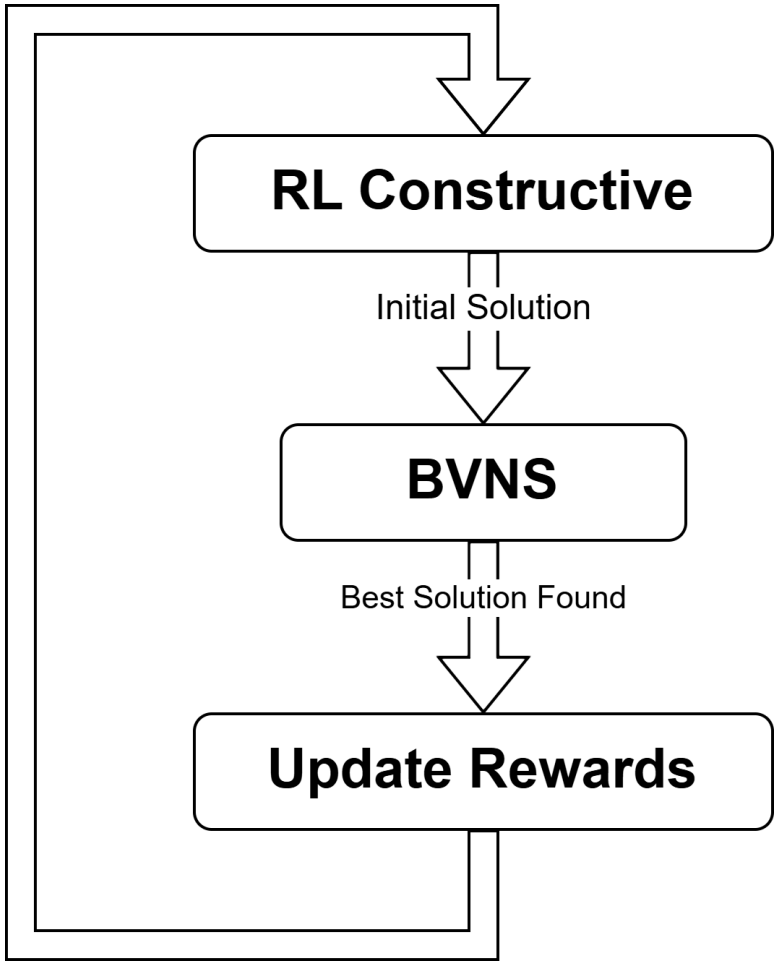
AGENCIA ESTATAL DE INVESTIGACIÓN

Resultados

Estructura de la instancia



Reinforcement Learning Constructive



Reward Update

$$R_{k+1}(i) = R_k(i) + \Delta R(i)$$

where for $i = 1, \ldots, N$:

$$\Delta R(i) = \begin{cases} +1/N & \text{if } i \in S^* \\ -1/N & \text{if } i \notin S^* \end{cases}$$

Results

GAP Convergence