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

MAEB 2.1: Sesión Especial: Scatter Search y Path Relinking

Organizadores:

Anna Martínez-Gavara y

Jesús Sanchez-Oro Calvo

GRASP with Path Relinking for the Constrained Incremental Graph Drawing Problem

A. Martínez-Gavara
Dept. Estadística i Inv. Operativa
Universitat de València
València, España
gavara@uv.es

A. Napoletano
Dept. Mathematics and Applications
University of Napoli Federico II
Napoli, Italy
antonio.napoletano2@unina.it

P. Festa
Dept. Mathematics and Applications
University of Napoli Federico II
Napoli, Italy
paola.festa@unina.it

T. Pastore
Dept. Mathematics and Applications
University of Napoli Federico II
Napoli, Italy
tommaso.pastore@unina.it

R. Martí
Dept. Estadśtica i Inv. Operativa
Universitat de València
València, España
rafael.marti@uv.es

***Abstract*—Graph Drawing is a well-established area in Computer Science, with applications from scheduling to software diagrams. The main quality desired for drawings is readability, and edge crossing minimization is a well-recognized method for a good representation of them. This work focuses in incremental graph drawing problems, with the aim to minimize the number of edge crossings while satisfying some constraints to preserve the absolute position of the vertices in previous drawings. We propose a mathematical model and a GRASP (Greedy Randomized Adaptive Search Procedure) with PR (Path Relinking) methodology to solve this problem. Finally, we compare our methodology with `CPLEX` and heuristic solutions obtained by a well-known black-box solver, `LocalSolver`.**

***Index Terms*—heuristics, graph drawing, path relinking**

I. Introduction

Graph Drawing is a relevant topic of research in areas such as workflow visualization, database modeling, bioinfomatics and decision diagrams. The difficulty lies in getting readable informations from systems represented by graphs. See, for example, Kaufmann and Wagner [4] and Di Battista [1] for a throughly survey in graph drawing. The edge-crossing minimization criterion is one of the most common to obtain a readable drawing. The problem of minimizing the number of crossings is NP-complete.

In this work we consider hierarchical graphs, this is not a limitation since there exists several methodologies to convert any directed acyclic graph (DAG) into a layered graph. The Hierarchical Directed Acyclical Graph (HDAG) representation is done by setting all the vertices in layers, and all the edges pointing in the same direction. The most well-known method to obtain a good representation of a graph is the Sugiyama's procedure [9], which has become a standard in the field.

We focus on minimizing the number of edge crossings in incremental graph drawings. The goal is to preserve the mental map of the user over successive drawings. In this way, we consider a new model adding constraints on both the relative ([7]) and the absolute position of the original vertices ([1] in the context of orthogonal graphs).

In this work, we propose a new mathematical programing formulation and a GRASP (Greedy Randomized Adaptive Search Procedure) with PR (Path Relinking) method. We adapt the well-known `Local Solver` black-box optimizer to solve this problem. Finally, we compare our heuristic with `CPLEX` and `Local Solver`, we have verified that `GRASP+PR` is able to obtain a fraction of the optimal solutions.

This work has been partially supported by the Spanish Ministerio de Economía y Competitividad with grant ref. TIN2015-65460-C02.

II. Mathematical Programing Model

A hierarchical graph $H = (G, p, L)$ is defined as a graph $G = (V, E)$ where V and E represent the sets of vertices and edges, respectively, p is the number of layers, and $L : V \rightarrow \{1, 2, \ldots, p\}$ is a function that indicates the layer where a vertex $v \in V$ resides. Let V^t be the set of vertices in layer t, and let E^t be the set of edges from V^t to V^{t+1}, with $n_t = |V^t|$.

The problem of minimizing edge-crossing is well-known in graph drawing [1] and, in the context of hierarchical graphs, may be formulated as the problem of finding the optimal ordering in each layer. A drawing D of a hierarchical graph H is the pair (H, Π) where Π is the set $\{\pi^1, \ldots, \pi^p\}$, with π^t establishing the ordering of the vertices in the layer t. We define as $C(D)$ the total number of edge crossings in drawing D.

From an original hierarchical graph $H = (G, p, L)$ and its drawing $D = (H, \Pi_0)$, we can consider the addition of some vertices $\hat{V}$ and edges $\hat{E}$ obtaining an incremental graph $IH = (IV, IE, p, L)$, where $IV = V \cup \hat{V}$ and $IE = E \cup \hat{E}$, p is the number of layers and m_t is the number of vertices in the incremental graph in layer t.

The goal in the Incremental Graph Drawing Problem (IGDP) is to minimize the number of edge crossings of an incremental drawing $ID = (IH, \Pi)$, while conserving the same relative position between the original vertices as in the original drawing D. The mathematical programing formulation of IGDP is based on a linear integer formulation for the Multi-Layer Drawing Problem proposed by Jünger et al. in [3], with a new set of constraints that preserves the relative position of the original vertices. As the authors in [3], we define the binary variables x_{ik}^t, where x_{ik}^t take the value 1 if vertex i precedes vertex k and 0 otherwise, where i, k are vertices that reside in the same layer t, $t = 1, \ldots, p$. Let $\pi_0^t(i)$ be the original position of the vertex i. The new constraints are:

$$x_{ij}^t + x_{ji}^t = 1, \quad \forall\, i, j \in V^t \ : 1 \leq i < j \leq m_t \qquad (1)$$

$$x_{ij}^t = 1, \quad \forall\, i, j \in V^t \ : \ \pi_0^t(i) < \pi_0^t(j). \qquad (2)$$

Finally, we add the requirement that the positions of the original vertices has to be set close enough to their positions in the original drawing. The maximum distance slack between the original position of a vertex and the new one, is represented by K. The mathematical model for the Constrained Incremental Graph Drawing Problem (C-IGDP) consists in the IGDP model adding the following set of constraints:

$$\max\{1, \pi_0^t(i) - K\} \leq \pi^t(i) \leq \min\{\pi_0^t(i) + K, m_t\}, \ \forall\ i \in V, \qquad (3)$$

where $\pi_0^t(i)$ and $\pi^t(i)$ represent the position of the vertex i, original and the new one where i can be assigned, respectively.

III. Greedy Randomized Adaptive Search Procedure

In this work, we propose a GRASP metaheuristic. The method iteratively performs two phases, constructive and improvement. The first phase employs a greedy function $g(v)$ that measures the minimum number of edge crossings $g(v, q)$ when vertex v is inserted in position q in its layer, for all positions q.

Initially, the construction phase starts with the original drawing, and iteratively, a vertex v is selected from the Restricted Candidate List (RCL) and it is placed in the position in which the number of crossings is minimum. The RCL consists of all unselected vertices (CL) with the number of edge crossings lower than or equal to a threshold τ:

$$\tau = \min_{v \in CL} g(v) + \alpha \left(\max_{v \in CL} g(v) - \min_{v \in CL} g(v) \right). \qquad (4)$$

That is $RCL = \{v \in CL : g(v) \leq \tau\}$. The performance of this method depends on the parameter α which balances greediness and randomness. Note that during the construction phase all the original vertices should maintain the restriction $\max\{1, \pi_0(v) - K\}$ and $\min\{\pi_0(v) + K, m_{L(v)}\}$. The pseudocode of construction phase is described in Fig. 1.

Once a solution ID is obtained, we apply our improvement method which consists in two neighborhoods $N_0(ID)$ and $N_1(ID)$. First, the method explores $N_0(ID)$, which consists in swapping the positions of two incremental vertices. For a incremental vertex v, from the first until the last incremental vertex in a layer, our procedure searches the best swap move with all the other incremental vertices, and performs it if it reduces the number of edge crossings. The algorithm scans all the swap moves from layer 1 to p until no further improvement is possible. Since only incremental vertices are involved, all the moves are feasible. Henceforth, we call this local search phase as *Swap*.

```
1  ID ← D;
2  CL ← IV \ V;
3  forall v ∈ CL do
4  |  compute g(v) and τ;
5  forall v ∈ CL do
6  |  if g(v) ≤ τ then
7  |  |  RCL ← RCL ∪ {v};
8  while ID is not complete do
9  |  v* ←select_node_randomly(RCL);
10 |  ID ← add_node_to_solution(v*);
11 |  recompute g and τ;
12 |  rebuild RCL;
13 return ID
```

Fig. 1. Constructive phase for the C-IGDP.

Then, the local search method resorts to a second phase, called *Insertion*, based on neighborhood $N_1(ID)$. In a given layer, the method scans all incremental vertices (starting from the first one) and explores all its feasible insertions in previous positions (a move is feasible if the position of the original vertices is within the limits of (3)). As in the case of $N_0(ID)$, the local search performs sweeps from layer 1 to p until no improvement is possible.

Our improvement procedure ends when *Swap* and *Insertion* phases are performed and a local optimum is found.

IV. Path Relinking

In this work, we propose a hybridization of GRASP with forward Path Relinking (PR). PR was originally proposed in the context of Tabu Search [2] to integrate search diversification and intensification. Later, Laguna and Martí in [6] adapted this technique as an intensification phase for GRASP. The method generates intermediate solutions in the path to connect high-quality solutions, which could be better or more diverse than the solutions being connected.

Let ID^i and ID^g be the initial and guiding incremental drawings, respectively. A path relinking move consists of replacing an entire layer from the initial solution ID^i with a layer from ID^g. Fig. 2 illustrates the path between two solutions ID^i and ID^g with 3 layers and 6 vertices. The three first intermediate solutions ID^1, ID^2 and ID^3 are generated from ID^i, exchanging one of its layer with the layers of ID^g. The process continues from the best solution found, in this case solution ID^1 with $C(ID^1) = 2$, in case of ties the algorithm choose one of the solutions at random. The algorithm finds two solutions, ID^4 and ID^5, that are better

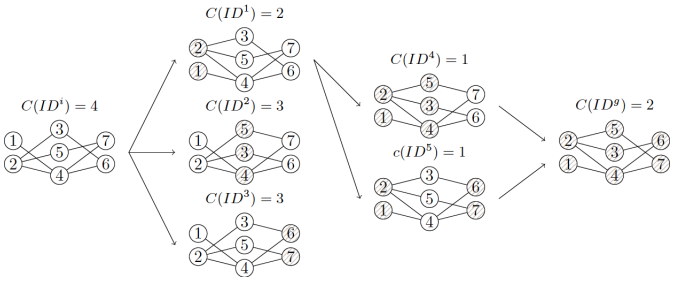

Fig. 2. Illustration of PR.

than ID^i and ID^g, both with number of edge crossings equal to 1.

The PR explores the path between all pairs of solutions in a set, called *Elite Set* (ES), which is constructed with $m = |ES|$ solutions generated with our procedure, and has the property of being as good as diverse as possible. Initially, the ES contains the m first solutions obtained with our GRASP. In each iteration, this set is updated with the generated solution ID^* if ID^* is better than the best solution in ES or sufficiently different from the other elite solutions. Diversity is measured as the number of positions that are not occupied by the same vertices divided by the number of vertices in the graph (with a distance larger than a parameter γ), while the quality is evaluated by the objective function. The algorithm ends when no new solutions are admitted to ES.

V. Experimental results

This section describes the experimental analysis to study the effectiveness and efficiency of the procedures presented above. In particular, we consider the following methods: GRASP (construction phase + local search), and GRASP+PR where we couple GRASP with Path Relinking. We also compare our heuristics with the solution obtained by running the integer linear programing proposed above with LocalSolver, and CPLEX.

The algorithms are implemented in C++, and the experiments have performed in an Intel Corei7-4020HQ CPU @2.60Ghz x 8. The set of instances consists of 240 files with 2, 6, 13 and 20 as the number of layers and the graph density varying in the range $\{0.065, 0.175, 0.3\}$. This set is generated in line with previous graph drawing papers [8] and [5], and it is available on-line in the web-page http://www.optsicom.es.

To avoid over-training, we consider 24 representative instances from the set of 240 to fine tune the algorithmic parameters. Note that, an important element is the value of the slack, K, between the original position of the vertices and the feasible ones. In this work, we consider low values of this parameter, $K = 1, 2$ and 3. Then, for each instance in the testing set, we have at most three different instances, one for each value of K (if the number of incremental vertices in one layer is less than K, then this value cannot be considered). To sum up, we have 609 instances in our *testing set*, and 62 instances in the *training set*.

These preliminary experimentation is devoted to fine tune the parameters in our GRASP+PR heuristic. First, the value of the parameter α in the construction phase which controls the balance between the diversity and the quality of the solution. We test three different values of α: 0.25, 0.5, and 0.75, and we also include a variant, labelled as $random$ that, for each construction, the value of α is selected by random in the range $[0, 1]$. The best performance is achieved with this $random$ variant. The last parameters to be set are the elite set size and the distance parameter in the PR procedure. We set these values to $|ES| = 3$ and $\gamma = 0.2$. The selection of the values of the parameters in these preliminary experiments is a trade-off between quality and computing time, for this reason, we do not reproduce the tables.

Table I shows the comparison between GRASP and GRASP+PR with CPLEX and LocalSolver over the entire set of 609 instances, classified by size. We execute our heuristics for 100 iterations, LocalSolver is run with a time limit of 20 seconds on the instances with 2 layers, and 60, 150 and 300 seconds on those with 6, 13 and 20 layers, respectively. Finally, we configure CPLEX to run for a maximum of 1800 seconds. This table shows for each procedure, the average number of crossings ($\bar{C}$), the average percent deviation from the best solution found ($\%\ dev$), the average percentage deviation between the heuristic solution value and the CPLEX best solution value ($\%\ gap$), the number of best solutions ($Bests$), the number of optimum solutions (Opt), and the CPU-time in seconds required to execute the method ($Time$).

TABLE I
Comparison on entire benchmark set according to instance size

Procedures	$\overline{C}$	$\%\ gap$	$\%\ dev$	$Bests$	Opt	$Time$
2 Layers ($32 \leq n \leq 96$), 171 instances						
CPLEX	2408.50	-	0.00	171	171	0.77
GRASP	2409.19	0.14	0.14	161	161	1.11
GRASP+PR	2408.92	0.14	0.14	164	164	1.18
LocalSolver	2785.47	17.01	17.01	12	12	20.11
6 Layers ($48 \leq n \leq 288$), 159 instances						
CPLEX	9995.70	-	0.01	157	157	56.24
GRASP	9997.43	0.13	0.14	81	81	5.53
GRASP+PR	9994.32	0.08	0.08	100	98	5.39
LocalSolver	11024.89	16.59	16.6	0	0	62.43
13 Layers ($104 \leq n \leq 611$), 141 instances						
CPLEX	23469.05	-	0.21	131	128	273.77
GRASP	23319.41	0.13	0.33	29	27	15.22
GRASP+PR	23305.22	0.02	0.22	54	44	15.34
LocalSolver	25530.24	15.95	16.16	0	0	162.06
20 Layers ($120 \leq n \leq 960$), 138 instances						
CPLEX	37918.20	-	0.38	118	116	383.73
GRASP	37522.42	0.03	0.39	21	20	25.77
GRASP+PR	37495.44	-0.12	0.24	49	29	28.39
LocalSolver	40954.07	14.30	14.68	0	0	328.77

Table I shows that, as expected, GRASP+PR obtains better results than GRASP. It is worth mentioning that CPLEX is able to obtain the optimal solution in 572 out of the 609 instances, and, with similar CPU time as our heuristics, it is able to obtain all the exact solutions for the small instances. However, in large instances, our heuristics are able to obtain similar quality

results as CPLEX with lower CPU time. Note that GRASP + PR is able to obtain a $\%\ gap$ of -0.12 which means that on average beats CPLEX. LocalSolver obtains the largest deviations in both $\%\ dev$ and $\%\ gap$, even if it is executed for longer CPU times than the competing heuristics.

VI. Conclusions

In this work, we propose a new mathematical model to tackle the Multilayer Incremental Graph Drawing Problem. We consider new constraints (by means of a paramenter K) to preserve key characteristics when updating an existing drawing. We develop a GRASP algorithm combined with a Path Relinking to obtain high quality solutions in the long term. We compare our heuristic with general solvers as CPLEX, which is able to obtain optimal solutions in 572 out of the 609 instances, and LocalSolver. GRASP+PR is competitive with them, obtaining similar quality solutions as CPLEX in smaller times on large instances, and outperforming LocalSolver.

References

[1] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition, 1998.

[2] F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publishers, Norwell, MA, USA, 1997.

[3] M. Jünger, E. K. Lee, P. Mutzel, and T. Odenthal. A polyhedral approach to the multi-layer crossing minimization problem. In *International Symposium on Graph Drawing*, pages 13–24. Springer, 1997.

[4] M. Kaufmann and D. Wagner, editors. *Drawing Graphs: Methods and Models*. Springer-Verlag, London, UK, UK, 2001.

[5] M. Laguna, R. Martí and V. Valls. Arc crossing minimization in hierarchical digraphs with tabu search. *Computers and Operations Research*, 24:1175–1186, 1997.

[6] M. Laguna and R. Martí. Grasp and path relinking for 2-layer straight line crossing minimization. *INFORMS Journal on Computing*, 11:44–52, 1999.

[7] R. Martí, A. Martínez-Gavara, J. Snchez-Oro, and A. Duarte. Tabu search for the dynamic bipartite drawing problem. *Computers & Operations Research*, 91:1–12, 2018.

[8] J. Sánchez-Oro, A. Martínez-Gavara, M. Laguna, A. Duarte, and A. Martí. Variable neighborhood scatter search for the incremental graph drawing problem. *Computational Optimization and Applications*, 68:775–797, 2017.

[9] K. Sugiyama, S. Tagawa, and M. Toda. Methods for visual understanding of hierarchical system structures. *IEEE Trans. Syst. Man, Cybern.*, 11:109–125, 1981.

Detección de puntos débiles en redes utilizando GRASP y Path Relinking

Sergio Pérez-Peló
Dept. of Computer Sciences
Universidad Rey Juan Carlos
C/Tulipán, S/N, 28933, Móstoles, Spain
Email: sergio.perez.pelo@urjc.es

Jesús Sánchez-Oro
Dept. of Computer Sciences
Universidad Rey Juan Carlos
C/Tulipán, S/N, 28933, Móstoles, Spain
Email: jesus.sanchezoro@urjc.es

Abraham Duarte
Dept. of Computer Sciences
Universidad Rey Juan Carlos
C/Tulipán, S/N, 28933, Móstoles, Spain
Email: abraham.duarte@urjc.es

***Resumen*—El avance que se ha producido en la tecnología en los últimos años ha hecho que sea imprescindible disponer de herramientas que permitan monitorizar y controlar las debilidades que aparecen en redes. Debido al carácter dinámico de las redes, continuamente se conectan y desaparecen dispositivos de las mismas, siendo necesario reevaluar las debilidades que han aparecido tras cada modificación. La evaluación de estas debilidades tiene como objetivo proporcionar información de utilidad al responsable de la red para conocer qué nodos de la misma es necesario reforzar. Los continuos cambios crean la necesidad de disponer de herramientas que evalúen los puntos críticos en tiempos reducidos. En este trabajo se presenta un algoritmo metaheurístico basado en la metodología GRASP (*Greedy Randomized Adaptive Search Procedure*) para detectar puntos débiles en redes. Además, se propone un método de combinación basado en *Path Relinking* para mejorar la calidad de las soluciones generadas. Los resultados experimentales muestran la calidad de la propuesta comparándola con el mejor algoritmo encontrado en el estado del arte.**

***Index Terms*—α-separator problem, GRASP, Path Relinking, nodos críticos**

I. Introducción

En los últimos años la ciberseguridad se ha convertido en uno de los campos más relevantes para todo tipo de usuarios: desde empresas e instituciones hasta usuarios individuales. El incremento del número de ataques a diferentes redes, así como de la importancia de la privacidad en Internet, han creado la necesidad de tener redes más seguras, confiables y robustas. Un ciberataque a una compañía que conlleve pérdida de información personal de sus clientes puede resultar en un importante daño económico y social [1]. Además, los ataques de Denegación de Servicio (del inglés *Denial of Service, DoS*) y ataques de Denegación de Servicio distribuídos (del inglés *Distributed Denial of Service, DDoS*) han ido siendo cada vez más comunes, debido a que un ataque exitoso puede resultar en la deshabilitación de un servicio de un proveedor de Internet, por ejemplo. Además, si otros servicios dependen a su vez del servicio atacado, podría producirse un fallo en cascada, afectando a un mayor número de clientes [2].

Es importante identificar cuáles son los nodos más importantes en una red. Ésta es una cuestión de interés para ambas partes de un ciberataque: el atacante y el defensor. El primero estará interesado en deshabilitar dichos nodos con el objetivo de hacer la red más vulnerable, mientras que el segundo estará interesado en reforzar esos nodos importantes con medidas de seguridad más robustas. Además, ambas partes quieren consumir la mínima cantidad de recursos posibles para minimizar los costes derivados del ataque / defensa: por un lado, el atacante está interesado en ocasionar el máximo daño posible a la red empleando la mínima cantidad de recursos posible; por otro lado, el defensor querrá reforzar la red minimizando los costes de mantenimiento y seguridad. Por lo tanto, es interesante para ambas partes identificar cuáles son los puntos más débiles en la red.

Definimos una red como un grafo $G = (V, E)$, donde V es el conjunto de vértices ($|V| = n$) y E es el conjunto de aristas ($|E| = m$). Un vértice $v \in V$ representa un nodo de la red, mientras que una arista $(v_1, v_2) \in E$, con $v_1, v_2 \in V$ indica que hay una conexión en la red entre los vértices v_1 y v_2. Definimos un separador de una red como un conjunto de vértices $S \subseteq V$ cuya eliminación supone la separación de la red en dos o más componentes conexas. Formalmente,

$$V \setminus S = C_1 \cup C_2 \ldots \cup C_p$$
$$\forall\ (u, v) \in E^\star\ \exists\ C_i\ :\ u, v \in C_i$$

donde $E^\star = \{(u, v) \in E\ :\ u, v \notin S\}$.

Este trabajo se centra en encontrar un conjunto de vértices mínimo $S^\star$ que permita separar la red G en componentes conexas de tamaño menor que $\alpha \cdot n$. En términos matemáticos,

$$S^\star \leftarrow \arg\min_{S \in \mathbb{S}} |S| \quad : \quad \max_{C_i \in V \setminus S} |C_i| \leq \alpha \cdot n$$

Cabe mencionar que el número de componentes conexas resultantes no es relevante para este problema. La restricción actual del problema del separador α (α-SP) es que el número de vértices en cada componente conexa debe ser menor o igual que $\alpha \cdot n$. Este problema es $\mathcal{NP}$-completo para redes de topología general cuando se consideran $\alpha \geq \frac{2}{3}$ [3]. Se han propuesto algunos algoritmos en tiempo polinómico cuando la topología de la red es un árbol o un ciclo [4]. Sin embargo, estos algorimos requieren tener un conocimiento previo de la topología de la red, lo que no es usual en problemas de la vida real.

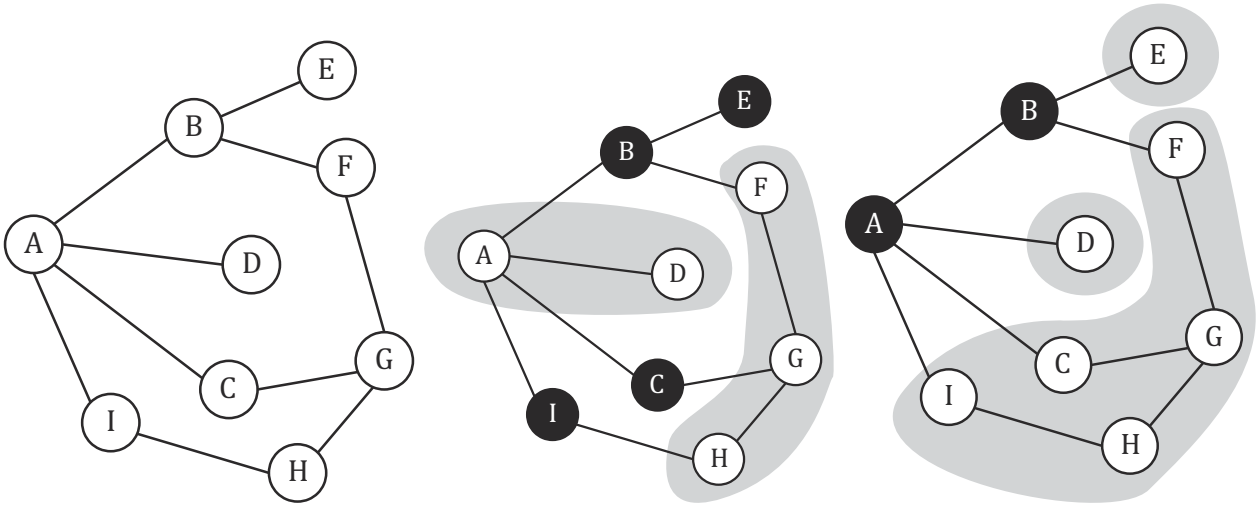

Figura 1: 1(a) Ejemplo de un grafo derivado de una red, 1(b) una solución factible con 4 nodos en el separador (`B,C,E`, y `I`), y 1(c) una solución mejor con 2 nodos en el separador (`A` y `B`)

Este problema ha sido abordado tanto desde la perspectiva exacta como la heurística. En concreto, se han presentado algoritmos que resuelven el problema en un tiempo polinómico para topologías como árboles o ciclos, así como un algoritmo voraz con una relación de aproximación de $\alpha \cdot n + 1$ [4]. Adicionalmente, un algoritmo heurístico para estudiar los separadores y los Sistemas Autónomos de Internet se propuso en [5]. Dependiendo del valor de α, el problema del separador α puede relacionarse con otros problemas bien conocidos. En particular, cuando $\alpha = \frac{1}{n}$, es equivalente al *minimum vertex cover problem*, y cuando $\alpha = \frac{2}{n}$ es análogo al *minimum dissociation set problem*. Por lo tanto, el problema del seaparador α es una generalización de estos problemas, que son también $\mathcal{NP}$-completos [6].

II. Propuesta algorítmica

Greedy Randomized Adaptive Search Procedure (GRASP), que en castellano se podría traducir como *procedimientos de búsqueda voraz, aleatorizados y adaptativos*, es un procedimiento multi-arranque que fue originalmente propuesto en [7] pero no se definió formalmente hasta [8]. Los procedimientos multi-arranque son un conjunto de procedimientos que intentan evitar quedar atrapados en óptimos locales reiniciando el procedimiento desde otro punto de inicio del espacio de búsqueda, lo que permite aumentar la diversificación de la estrategia de búsqueda. Esta metaheurística se ha utilizado recientemente con éxito en varios problemas de optimización [9]–[11].

Cada iteración en GRASP tiene dos fases bien diferenciadas: construcción y mejora. La primera tiene como objetivo generar una solución inicial de calidad para el problema, mientras que la segunda trata de mejorar la solución inicial a través de algún método de optimización local [12]. Tradicionalmente el optimizador de la segunda fase suele ser un procedimiento de búsqueda local sencillo, pero es común combinar esta metaheurística con procedimientos de búsqueda más complejos, como pueden ser Tabu Search [13] o Variable Neighborhood Search [14], entre otros, lo que prueba la versatilidad de la metodología GRASP.

II-A. Criterio voraz

La fase de construcción necesita definir un criterio voraz $g(v)$ que evalúe, para cada vértice $v \in V$, la relevancia de insertar dicho vértice en la solución. Para el problema que estamos tratando, α-SP, el criterio voraz debe indicar cuáles de los vértices que aún no han sido incluidos en la solución son nodos críticos en la red.

En este trabajo se propone la utilización de criterios utilizados en el área del análisis de redes sociales, de manera que se tratará de identificar qué vértices serían los más importantes de la red si ésta se tratara de una red social. En el análisis de redes sociales, un vértice es importante si mantiene a un gran número de usuarios de la red conectados entre sí [15], por lo que la analogía con el problema del α-SP es directa: un vértice es importante en la red si su desaparición hace que los dispositivos conectados a él dejen de estar conectados.

Existen numerosas métricas para evaluar la importancia de un usuario en una red social, siendo el cálculo de la mayoría de ellas muy costoso computacionalmente. Esto es debido a que varias de las métricas, como pueden ser *Betweenness* [16] o *PageRank* [17] necesitan conocer todos los caminos más cortos que pasan entre cada par de usuarios de la red. En este trabajo se utilizará la métrica denominada *closeness* [18], la cual considera que un vértice es importante si es alcanzable (i.e., se puede llegar a él a través de muchos caminos) por un elevado número de vértices de la red. Más formalmente, dada una red conexa G, el valor del *closeness* de un vértice se calcula como la suma de la longitud de las rutas más cortas entre el nodo en cuestión y el resto de nodos en el grafo.

Cabe destacar que la evaluación de esta métrica solo necesita conocer el camino más corto entre cada par de vértices, por lo que basta con realizar un recorrido en anchura (si la red no es ponderada) o aplicar el algoritmo de Dijkstra (si la red es ponderada). La idea que subyace bajo esta métrica es que, cuanto más central es un nodo, más cerca está de todos los demás nodos.

El valor del *closeness* de un nodo $v \in V$, definido como $C(v)$ se evalúa como el cálculo inverso de la suma de las distancias desde ese nodo v hasta el resto de nodos de la red $u \in V \setminus \{v\}$. En términos matemáticos,

$$C(v) \leftarrow \frac{1}{\sum_{u \in V \setminus S} d(v,u)}$$

donde $d(v,u)$ es la distancia entre los nodos v y u. En este trabajo, se considera que la distancia entre dos vértices $u, v \in V$ será la longitud del camino más corto que conecta u con v. Cabe destacar que esta métrica solo tiene sentido en componentes conexas, ya que de otra manera no sería posible encontrar un camino entre los vértices u y v.

II-B. Fase de construcción

El procedimiento de construcción parte de una solución vacía e iterativamente añade vértices a dicha solución hasta que ésta es factible. En el caso del α-SP, inicialmente ningún vértice ha sido añadido al separador, por lo que es necesario añadir vértices al mismo hasta cumplir la restricción de que la red se encuentra dividida en componentes conexas de tamaño inferior a $\alpha \cdot n$. El Algoritmo 1 muestra el pseudocódigo del método de generación de soluciones propuesto.

Algorithm 1 *Construir*(G, β)

1: $v \leftarrow Random(V)$
2: $S \leftarrow \{v\}$
3: $CL \leftarrow V \setminus \{v\}$
4: **while not** $esFactible(S, G)$ **do**
5: $\quad g_{\text{mín}} \leftarrow \text{mín}_{v \in CL}\, C(v)$
6: $\quad g_{\text{máx}} \leftarrow \text{máx}_{v \in CL}\, C(v)$
7: $\quad \mu \leftarrow g_{\text{máx}} - \beta \cdot (g_{\text{máx}} - g_{\text{mín}})$
8: $\quad RCL \leftarrow \{v \in CL \ : \ C(v) \geq \mu\}$
9: $\quad v \leftarrow Random(RCL)$
10: $\quad S \leftarrow S \cup \{v\}$
11: $\quad CL \leftarrow CL \setminus \{v\}$
12: **end while**
13: **return** S

El procedimiento comienza eligiendo al azar el primer vértice que va a ser incluido en la solución (pasos 1-2). La elección aleatoria de este primer elemento permite aumentar la diversidad de las soluciones generadas en la fase de construcción. A continuación, se crea la lista de candidatos *CL* con todos los vértices de la red salvo el elegido inicialmente (paso 3). El método de construcción entonces añade un nuevo vértice a la solución hasta que ésta satisfaga las restricciones del problema (pasos 4-12). La función *esFactible* tiene como objetivo comprobar que todas las componentes conexas de la red tienen un tamaño inferior a $\alpha \cdot n$ si se eliminan los nodos incluidos en la solución parcial S.

Tras ello, se calculan el mínimo y el máximo valor (pasos 5 y 6, respectivamente) para la función voraz elegida. En este caso, se utiliza el valor de *closeness* descrito en la Sección II-A. Estos dos valores se utilizan para calcular un umbral μ (paso 7) que limitará los vértices que entran a formar parte de la lista de candidatos restringida *RCL* (paso 8). El valor del umbral dependerá de un parámetro β (con $0 \leq \beta \leq 1$) que controla la aleatoriedad o voracidad del método. Por un lado, si $\beta = 0$, entonces $\mu = g_{\text{máx}}$ y, por tanto, solo se considerarán en la *RCL* aquellos vértices con el máximo valor de *closeness*, resultando en un método completamente voraz. Por otra parte, si $\beta = 1$, entonces $\mu = g_{\text{mín}}$, formando parte de la *RCL* todos los vértices de la *CL*, convirtiéndose así en un método totalmente aleatorio. De esta forma, cuanto mayor sea el valor del parámetro β, más aleatorio será el método, y viceversa. La Sección III discutirá cómo afecta el valor de este parámetro a la calidad de las soluciones obtenidas.

II-C. Fase de mejora

Las soluciones generadas en la fase de construcción se han construido mediante un procedimiento aleatorizado (a través del parámetro β), por lo que, en general, es posible aplicar un método de optimización local para mejorar la calidad de la solución. Aunque es posible utilizar cualquier tipo de optimizador en esta fase, el problema tratado requiere de la obtención de soluciones en períodos cortos de tiempo, por lo que se han descartado optimizadores complejos, optando por un procedimiento de búsqueda local tradicional.

Los procedimientos de búsqueda local tienen como objetivo encontrar el óptimo local de una solución con respecto a una vecindad predefinida. En primer lugar, es necesario definir qué vecindad va a explorar el procedimiento. En este trabajo se considerará la vecindad 2×1, la cuál contiene a todas las soluciones alcanzables desde una solución inicial S mediante la eliminación de dos vértices que estaban originalmente en la solución y la inserción de un nuevo vértice en la misma. Cabe destacar que cualquier movimiento dentro de esta vecindad que produzca una solución factible habrá mejorado el valor de la función objetivo, ya que habrá reducido el conjunto de vértices que necesitan ser eliminados de la red en una unidad. Más formalmente, la vecindad se define como:

$$N(v,S) \leftarrow \{S' \ : \ S' \leftarrow S \setminus \{u,v\} \cup \{w\}, \\ \forall\ u,v \in S \wedge\ \forall\ w \in V \setminus S\}$$

Existen dos métodos tradicionales para explorar la vecindad definida: *First Improvement* y *Best Improvement*. El primero recorre todas las soluciones de la vecindad, realizando siempre el primer movimiento que conduzca a una solución de mayor calidad, comenzando la búsqueda de nuevo desde la nueva solución. Sin embargo, *Best Improvement* primero explora todas las soluciones existentes en la vecindad, realizando el movimiento que lleve a la solución de mayor calidad.

En este trabajo se considerará únicamente la variante *First Improvement* debido a dos motivos principales. En primer lugar, *First Improvement* requiere de un menor coste computacional ya que no recorre la vecindad completa para realizar un movimiento y, como se ha mencionado en otras ocasiones, en este problema se requiere un tiempo de cómputo reducido. Por otra parte, dada la vecindad 2×1 definida, cualquier movimiento que lleve a una solución factible producirá una solución cuyo valor de la función objetivo se habrá mejorado en una unidad, sea cual sea el movimiento que se realice. Por tanto, para esta vecindad no es útil comprobar todos los posibles movimientos disponibles.

Teniendo en cuenta lo mencionado anteriormente, el procedimiento de búsqueda local propuesto para el problema del α-SP consiste en recorrer de manera aleatoria la vecindad de cada una de las soluciones generadas en la fase de construcción, parando cuando no se encuentre ninguna solución de mayor calidad en la misma, y devolviendo la mejor solución encontrada en la búsqueda.

II-D. Path Relinking

Path Relinking (PR) es un método de combinación de soluciones propuesto originalmente como una metodología para integrar estrategias de intensificación y diversificación en el contexto de la búsqueda tabú [19]. El comportamiento de PR se basa en explorar las trayectorias que conectan soluciones de alta calidad, con el objetivo de generar soluciones intermedias que, eventualmente, pueden ser mejores que las soluciones originales utilizadas para construir la trayectoria. Laguna y Martí [20] adaptaron *Path Relinking* en el contexto de GRASP como un método destinado a aumentar la intensificación. El algoritmo de *Path Relinking* opera sobre un conjunto de soluciones, llamado *Elite Set* (ES), que se encuentra ordenado siguiendo un criterio descendiente de acuerdo al valor de un criterio de calidad predefinido de cada solución. En este trabajo, consideramos como criterio de calidad el valor de la función objetivo de la solución. En concreto, el *Elite Set* se compone de las soluciones de mayor calidad generadas mediante el algoritmo GRASP. Este diseño generalmente se denomina estático [21], ya que en primer lugar se aplica GRASP para construir el conjunto de élite y posteriormente se aplica *Path Relinking* para explorar trayectorias entre todos los pares de soluciones en el ES. Por otra parte, el diseño dinámico considera la actualización del *Elite Set* cada vez que se explora una trayectoria.

Dadas una solución inicial, S_i y una solución guía S_g, *Path Relinking* crea una trayectoria de soluciones intermedias desde S_i hasta S_g a través de la inserción en S_i de características pertenecientes a S_g que aún no se encuentran en S_i. En el contexto de α-SP, se propone la eliminación de un vértice v que se encuentra en la solución que se está explorando en la trayectoria S' pero no está incluido en la solución guía, a la vez que se inserta un elemento u que no se encuentra en S' pero sí en S_g. Formalmente, se define entonces el movimiento *swap* de la siguiente manera:

$$Swap(S', v, u) \leftarrow S' \setminus \{v\} \cup \{u\} : \ v \in S' \setminus S_g, \ u \in S_g \setminus S'$$

En cada iteración, *Path Relinking* realiza un movimiento de *swap* que genera una nueva solución intermedia en la trayectoria. Cabe destacar que este movimiento produce, con alta probabilidad, una solución no factible, por lo que será necesario reparar la solución intermedia generada tras un movimiento $Swap(S', v, u)$. Para ello, tras el movimiento, se insertarán de manera aleatoria tantos nodos en la solución como sean necesarios para que la solución vuelva a ser factible.

Existen numerosas variantes de *Path Relinking* que difieren en cómo se construye la trayectoria. Tradicionalmente se consideran dos variantes de *Path Relinking*: *Greedy Path Relinking* (GPR) y *Random Path Relinking* (RPR). GPR, en cada iteración, genera una solución intermedia por cada elemento incluido en S_g que no está incluido en la solución intermedia S', continuando la trayectoria por la mejor solución intermedia de entre las generadas. Por otra parte, RPR selecciona aleatoriamente un elemento de entre los disponibles en S_g que no están incluidos en S' y genera una única solución intermedia con un movimiento *swap* donde se incluya dicho vértice. Mientras que GPR se centra en la intensificación, RPR tiene como objetivo la diversificación. De nuevo, la exploración exhaustiva de la trayectoria de GPR hace que no sea un método adecuado para el problema α-SP, debido al elevado coste computacional. En su lugar, se considera la variante RPR, menos exigente computacionalmente al explorar una única solución en cada etapa de la trayectoria. Para compensar la falta de intensificación en esta etapa, se aplicará el procedimiento de mejora descrito en la Sección II-C a la mejor solución encontrada en la trayectoria.

En este trabajo el *Elite Set* se compone de todas las soluciones generadas en la fase de construcción y mejora. Es importante aclarar que no se permite la entrada de soluciones ya existentes en el *Elite Set*, por lo que todas las soluciones combinadas serán siempre diferentes entre sí.

III. Resultados computacionales

Esta sección está dedicada a analizar el rendimiento de los algoritmos propuestos y comparar los resultados obtenidos con el mejor método previo encontrado en el estado del arte. Los algoritmos se han desarrollado en Java 9 y todos los experimentos se han llevado a cabo en un Intel Core 2 Duo 2,66 GHz con 4 GB de RAM.

El conjunto de instancias utilizadas en esta experimentación se ha generado utilizando el mismo generador de grafos propuesto en el mejor trabajo previo [22]. Específicamente, los grafos se generan utilizando el modelo Erdös Rényi [23], en el que cada nuevo nodo insertado tiene la misma probabilidad de estar conectado a un nodo ya existente en el grafo. Se ha generado un conjunto de 28 instancias de 100 vértices de diferente densidad (de 200 a 500 aristas).

La experimentación se encuentra dividida en dos partes diferentes: experimentación preliminar y experimentación final. La

primera está diseñada para ajustar el único parámetro del que requiere GRASP, β, que controla la aleatoriedad del método, mientras que la segunda se dedica a analizar el rendimiento de la mejor variante de GRASP junto con Path Relinking en comparación con el mejor algoritmo anterior encontrado en el estado del arte.

Todos los experimentos proporcionan las siguientes métricas: F.O., el valor de la función objetivo promedio; Tiempo (s), el tiempo promedio de cómputo medido en segundos; Desv.(%), la desviación porcentual promedio con respecto a la mejor solución encontrada en el experimento; y #Mejores, el número de veces que un algoritmo alcanza la mejor solución del experimento.

El experimento preliminar considerará un subconjunto de 20 instancias para evitar el sobreajuste del algoritmo. Para ajustar el valor del parámetro, se ha ejecutado la fase de construcción y mejora para generar un total de 100 soluciones por cada instancia, devolviendo la mejor solución encontrada, con $\beta = \{0{,}25, 0{,}50, 0{,}75, RND\}$, donde el valor *RND* indica que se obtiene un valor de β aleatorio en cada construcción. La Tabla I muestra los resultados obtenidos para cada valor.

Cuadro I: Rendimiento de GRASP (100 fases de construcción y mejora) con diferentes valores de β

β	F.O.	Tiempo (s)	Desv. (%)	#Mejores
0.25	29.75	18.37	2.31	11
0.50	29.90	18.92	2.91	12
0.75	31.65	18.81	7.09	7
RND	29.80	18.99	3.02	10

Como se puede observar, el valor de β que obtiene los mejores resultados es $\beta = 0{,}25$, que se corresponde con la variante que proporciona una menor aleatoriedad a la construcción. Este resultado parece indicar que el criterio de *closeness* utilizado es muy preciso para identificar los vértices críticos de la red. Por lo tanto, la diversificación se incluirá en mayor medida con la inclusión del método de combinación *Path Relinking*.

El experimento final está diseñado para comparar los resultados obtenidos incluyendo el método de combinación *Path Relinking* respecto al mejor algoritmo encontrado en el estado del arte, esta vez ejecutado sobre el total de instancias disponibles. El mejor algoritmo previo está basado en *Random Walks* combinado con *Markov Chains*. Debido a no haber recibido respuesta por parte de los autores previos, los resultados proporcionados son los obtenidos tras reimplementar detalladamente el método previo [24]. Los resultados se muestran en la tabla II.

Como se puede observar, el mejor algoritmo resulta ser *Path Relinking*, obteniendo el mejor resultado en todas las instancias del conjunto de datos utilizado. Comparando los resultados frente a GRASP, podemos concluir que *Path Relinking* es capaz de mejorar los resultados previos, quedando GRASP a una desviación del 7.42 %, a cambio de un mayor coste computacional (18.53 frente a 275.67 segundos). Aún con este incremento en el tiempo de cómputo, sigue requiriendo menos

Cuadro II: Comparativa de los resultados obtenidos por el algoritmo previo (RW) frente a los algoritmos GRASP y *Path Relinking* propuestos en este trabajo.

	F.O.	Tiempo (s)	Desv. (%)	#Mejores
GRASP	31.93	18.53	7.42 %	3
PR	29.43	275.67	0.00 %	28
RW	46.46	721.58	37.81 %	0

de la mitad del tiempo necesitado por el mejor método previo. Cabe destacar que, incluso GRASP de manera aislada, sin considerar *Path Relinking*, obtiene mejores resultados en todas las métricas que el método previo.

Para confirmar que existen diferencias estadísticamente significativas entre los métodos propuestos, se ha llevado a cabo el test no paramétrico de Wilcoxon. El p-valor menor que 0.00001 demuestra que sí que existen diferencias entre los métodos, resultando el siguiente ranking: PR (1.05), GRASP (1.95), y RW (3.00). Además, se ha llevado a cabo el test de Wilcoxon con signos para comparar algoritmos por pares. La comparativa de PR frente a RW ha resultado en un p-valor menor que 0.00001, al igual que el análisis entre GRASP y PR. Analizando estos resultados, podemos concluir que *Path Relinking* emerge como el mejor algoritmo del estado del arte para el problema del α-SP.

IV. Conclusiones

En este trabajo se ha presentado un algoritmo basado en GRASP para la detección de puntos críticos en redes. Además, las soluciones generadas se combinan utilizando *Path Relinking* para mejorar la calidad de las soluciones obtenidas. Los resultados obtenidos superan a las mejores soluciones encontradas en el estado del arte, todo ello respaldado por test estadísticos. El algoritmo GRASP es capaz de encontrar soluciones de calidad en cortos períodos de tiempo (menos de 20 segundos en promedio), mientras que *Path Relinking* es un procedimiento más costoso computacionalmente pero capaz de mejorar significativamente las soluciones encontradas por GRASP. Ambos métodos obtienen mejores soluciones que el método previo, basado en *Random Walks* combinado con modelos de Markov. La solución propuesta emerge como el mejor método en el estado del arte, pudiendo obtener soluciones de alta calidad en tiempo real con GRASP y mejorarlas posteriormente con *Path Relinking* en aquellas situaciones en las que el tiempo no sea crítico.

Agradecimientos

Este trabajo ha sido parcialmente financiado por el Ministerio de Economía y Competitividad, ref. TIN2015-65460-C2-2-P.

Referencias

[1] G. Andersson, P. Donalek, R. Farmer, N. Hatziargyriou, I. Kamwa, P. Kundur, N. Martins, J. Paserba, P. Pourbeik, J. Sanchez Gasca, R. Schulz, A. Stankovic, C. Taylor, and V. Vittal, "Causes of the 2003 major grid blackouts in North America and Europe, and recommended means to improve system dynamic performance," *IEEE Transactions on Power Systems*, vol. 20, no. 4, pp. 1922–1928, 2005.

[2] P. Crucitti, V. Latora, and M. Marchiori, "Model for cascading failures in complex networks," *Phys. Rev. E*, vol. 69, p. 045104, 2004.
[3] U. Feige and M. Mahdian, "Finding small balanced separators." in *STOC*, J. M. Kleinberg, Ed. ACM, 2006, pp. 375–384.
[4] M. Mohamed-Sidi, "K-Separator Problem. (Problème de k-Sèparateur)." Ph.D. dissertation, Telecom & Management SudParis, Èvry, Essonne, France, 2014.
[5] M. Wachs, C. Grothoff, and R. Thurimella, "Partitioning the Internet." in *CRiSIS*, F. Martinelli, J.-L. Lanet, W. M. Fitzgerald, and S. N. Foley, Eds. IEEE Computer Society, 2012, pp. 1–8.
[6] M. Garey and D. Johnson, *Computers and Intractability - A guide to the Theory of NP-Completeness*. San Fransisco: Freeman, 1979.
[7] T. A. Feo and M. G. C. Resende, "A probabilistic heuristic for a computationally difficult set covering problem," *Operations Research Letters*, vol. 8, no. 2, pp. 67 – 71, 1989.
[8] T. A. Feo, M. G. C. Resende, and S. H. Smith, "A greedy randomized adaptive search procedure for maximum independent set." *Operations Research*, vol. 42, no. 5, pp. 860–878, 1994.
[9] A. Duarte, J. Sánchez-Oro, M. G. C. Resende, F. Glover, and R. Martí, "Greedy randomized adaptive search procedure with exterior path relinking for differential dispersion minimization." *Inf. Sci.*, vol. 296, pp. 46–60, 2015.
[10] J. D. Quintana, J. Sánchez-Oro, and A. Duarte, "Efficient greedy randomized adaptive search procedure for the generalized regenerator location problem." *Int. J. Comput. Intell. Syst.*, vol. 9, no. 6, pp. 1016–1027, 2016.
[11] J. Sánchez-Oro, M. Laguna, R. Martí, and A. Duarte, "Scatter search for the bandpass problem," *Journal of Global Optimization*, vol. 66, no. 4, pp. 769–790, 2016.
[12] F. W. Glover and G. A. Kochenberger, *Handbook of metaheuristics*. Springer Science & Business Media, 2006, vol. 57.
[13] F. Glover and M. Laguna, "Tabu search," in *Handbook of combinatorial optimization*. Springer, 2013, pp. 3261–3362.
[14] P. Hansen and N. Mladenović, "Variable neighborhood search," in *Search methodologies*. Springer, 2014, pp. 313–337.
[15] J. Scott, *Social Network Analysis: A Handbook*. Sage Publications, 2000.
[16] U. Brandes, "A faster algorithm for betweenness centrality," *Journal of mathematical sociology*, vol. 25, no. 2, pp. 163–177, 2001.
[17] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web." Stanford InfoLab, Tech. Rep., 1999.
[18] M. E. Newman, "The mathematics of networks," *The new palgrave encyclopedia of economics*, vol. 2, no. 2008, pp. 1–12, 2008.
[19] F. Glover, *Tabu search and adaptive memory programming – advances, applications, and challenges*. Dordrecht, Massachusetts, USA: Kluwer Academic Publishers, 1996.
[20] M. Laguna and R. Martí, "Grasp and path relinking for 2-layer straight line crossing minimization." *INFORMS Journal on Computing*, vol. 11, no. 1, pp. 44–52, 1999.
[21] F. Glover, M. Laguna, and R. Martí, "Fundamentals of scatter search and path relinking," *Control and cybernetics*, vol. 29, no. 3, pp. 653–684, 2000.
[22] J. Lee, J. Kwak, H. W. Lee, and N. B. Shroff, "Finding minimum node separators: A markov chain monte carlo method," in *DRCN 2017 - Design of Reliable Communication Networks; 13th International Conference*, March 2017, pp. 1–8.
[23] P. Erdős and A. Rényi, "On random graphs," *Publicationes Mathematicae*, vol. 6, p. 290, 1959.
[24] B. Hassibi, M. Hansen, A. Dimakis, H. Alshamary, and W. Xu, "Optimized markov chain monte carlo for signal detection in mimo systems: An analysis of the stationary distribution and mixing time," *Signal Processing, IEEE Transactions on*, vol. 62, no. 17, pp. 4436–4450, 2014.

Where facility centers should be located?

A.D. López-Sánchez
Department of Economics, Quantitative Methods and Economic History
Pablo de Olavide University
Sevilla, Spain
adlopsan@upo.es

J. Sánchez-Oro
Department of Computer Science
Rey Juan Carlos University
Madrid, Spain
jesus.sanchezoro@urjc.es

A.G. Hernández-Díaz
Department of Economics, Quantitative Methods and Economic History
Pablo de Olavide University
Sevilla, Spain
agarher@upo.es

M. Laguna
Leeds School of Business
University of Colorado
Boulder, Colorado
laguna@colorado.edu

***Abstract*—Facility location problems aim to determine the best position to place facility centers. This task is not easy since many objectives may be taken into consideration to choose the best possibility and the locations can be substantially different. This paper presents a Scatter Search algorithm with Path Relinking to decide where facility centers should be located depending on the objective functions under consideration. The objectives considered in this study are to minimize the distance between demand points and their nearest facilities, to maximize the number of demand points covered by a facility center and to minimize the maximum distance between demand points and their nearest facilities. Then, this problem is addressed as a three-objective optimization problem and the algorithm proposed is compared against other competitors obtaining promising results according to three different performance metrics.**

***Index Terms*—Scatter Search, Path Relinking, Multi-objective optimization problems, Facility location problems**

I. Introduction

It is not an easy task to locate facility centers since many objectives may be taken into consideration to choose the best possibility. Depending on the objective and the constraints imposed by the decision-maker, the facility centers may be located in different places. Hence, the best location may be substantially different depending on the objective function considered for optimization. Specifically, this work is focused on three common objective functions: f_1, with the aim of minimizing the average distance between demands and their nearest facilities; f_2, focused on covering the maximum number of demand points; and f_3, which minimizes the maximum distance between demand points and their nearest facilities. But what happens when the decision-maker wants to optimize all the objectives at the same time? In such a case, the location problem becomes a multi-objective facility location problem (mo-FLP) in which the objectives are usually in conflict. That is, there is not a single solution that simultaneously optimizes all objectives, or, in other words, the value of one objective function cannot be improved without deteriorating the value of at least another objective function. Those solutions are known as efficient solutions, non-dominated solutions or Pareto optimal solutions.

The mo-FLP that optimizes f_1, f_2, and f_3, was recently studied by Karatas and Yakici, see [10]. The authors developed an hybrid algorithm, named ITER-FLOC, that combines branch & bound techniques and iterative goal programming. In particular, authors proposed a different formulation for each considered objective. Initially, lower and upper bounds for each objective are evaluated. Then, for each iteration of the ITER-FLOC algorithm, the location models are solved, verifying if the termination criterion has been achieved. If so, the algorithm ends, returning the Pareto front constructed. Otherwise, the lower and upper bounds of each objective are updated. Prior to executing the next iteration, the location models are updated with additional constraints.

They are able to generate the Pareto optimal solutions with high level of diversity and cardinality. However, the drawbacks are the requirement of preference information since goal programming is considered an *a priori* method and its slowness because it an exact algorithm that needs to be solve multiple times (one for each considered goal). Additionally, the method requires from several input parameters that can difficult the scalability of the algorithm for new datasets.

Here, the mo-FLP is addressed using a Scatter Search algorithm combined with Path Relinking (SSPR). The output of the algorithm is the approximation of the Pareto front containing efficient solutions. A variety of optimization problems has been solved and the computational results indicate that the Scatter Search algorithm is able to find the Pareto set in a simple run within short computational time.

This work is structured as follows. Section II describes the problem. Section III gives details of the Scatter Search algorithm with Path Relinking implemented to solve the problem under consideration. Section IV presents the computational results. Finally, Section V summarizes the paper and discusses

J. Sánchez-Oro is supported by the Spanish Ministry of "Economía y Competitividad", Grant Refs. TIN2015-65460-C2-2-P. A.D. López-Sánchez and A.G. Hernández-Díaz acknowledge support from the Spanish Ministry of Science and Innovation through Project ECO2016-76567-C4-1-R.

future work.

II. Location Problem Definition

Let $I = \{1 \dots m\}$ be the set of available locations to host a facility center and $J = \{1 \dots n\}$ the set of demand points that requires to be covered by a facility center. Each demand point $i \in I$ has an associated weight w_i that represents the cost of satisfying its necessity with a facility center. Additionally, let $d_{ij} \geq 0$ be the distance between a candidate location $i \in I$ and a demand point $j \in J$. The distance is evaluated as the length of the shortest path that connects x with y. A solution S for a facility location problem is then represented as the set of p candidate locations to host a facility center, and the objective of the problem is to find a solution with the optimum value with respect to one or more objective function.

This work is focused on optimizing three different objective functions simultaneously, becoming a multi-objective problem. The first considered objective function, f_1, is focused on minimizing the average weighted distance between the demand points and their nearest facility center, which can be found in the literature as the p-Median Problem (pMP). Given a solution S, the evaluation of f_1 is formally defined as:

$$f_1(S) \leftarrow \frac{1}{n} \sum_{j \in J} w_j \cdot d_{i^\star j}, \text{ where } i^\star \leftarrow \arg\min_{i \in S} d_{ij}$$

The second objective function, f_2, tries to maximize the number of demand points whose necessity is satisfied by the selected candidate locations, which is usually referenced as the Maximal Coverage Location Problem (MCLP). A demand point is covered by a facility center if the distance between them is smaller or equal than a predefined threshold r. More formally,

$$f_2(S) \leftarrow \left| \left\{ j \in J \ : \ d_{i^\star j} \leq r, \text{ where } i^\star \leftarrow \arg\min_{i \in S} d_{ij} \right\} \right|$$

The third objective function, f_3, is intended to minimize the maximum distance between the demand points and the facility centers, which result in the p-Center Problem (pCP). This objective function is formally defined as:

$$f_3(S) \leftarrow \max_{j \in J} d_{i^\star j}, \text{ where } i^\star \leftarrow \arg\min_{i \in S} d_{ij}$$

Some of the first studies dealing the previous location problems were proposed by [2], [8], [9].

III. Scatter Search algorithm with Path Relinking

Scatter Search (SS), first proposed by Glover, see [5], is a metaheuristic framework which generates, maintains, and transforms a reference set of solutions, *RefSet*. It has been successfully applied to a large variety of optimization problems [12], [13]. Figure 1 depicts the general scheme for Scatter Search.

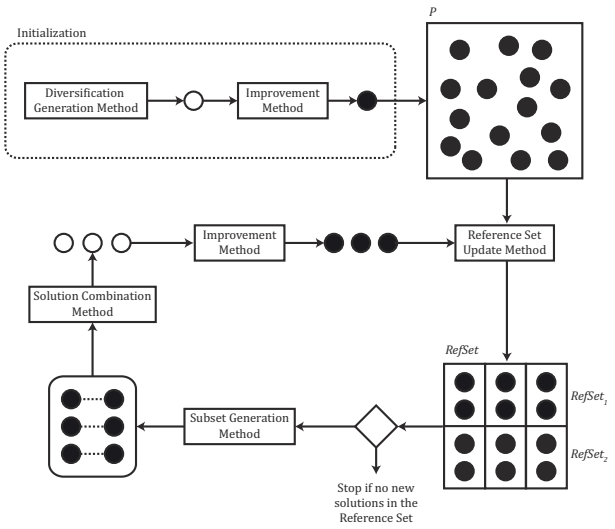

Fig. 1. Scatter Search algorithm.

The method firstly generates an initial population P. These solutions are generated using a diversification generation method (Section III-A). Then, a local optimum for each solution is found by an improvement method (Section III-B). It is important to remark that there are not repeated solutions in the initial population P.

Traditional implementations of Scatter Search constructs the reference set *RefSet* with the $\beta/2$ best solutions of set P in terms of objective function value. This work adapt this criterion for a multi-objective approach, by dividing the *RefSet* in three subsets, $RefSet_1$, $RefSet_2$, and $RefSet_3$, with $|RefSet_1| = |RefSet_2| = RefSet_3 = \beta$. In particular, $RefSet_1$ initially contains the best $\beta/2$ solutions with respect to f_1, while $RefSet_2$ and $RefSet_3$ contains the best $\beta/2$ ones with respect to f_2 and f_3, respectively. The remaining $\beta/2$ solutions for each *RefSet* are selected as the most diverse solutions among the remaining solutions in P. Diversity between a solution S and the *RefSet* under construction is measured as the minimum distance between S and every solution in the *RefSet*, considering the distance between two solutions as the number of candidate locations that differs in both solutions.

Once the *RefSet* has been created, Scatter Search selects the subsets of reference solutions that will be combined with the aim of finding new trial solutions. The most common implementation of the subset generation method consists of combining all pairs of solutions included in the *RefSet*. Considering that we maintain three *RefSet*, this implementation would be rather computationally demanding. Instead, we generate all pairs of solutions (S, S') such that S and S' belong to different *RefSet* (i.e., $S \in$ *RefSet*$_i$, $S' \in$ *RefSet*$_j$, with $1 \leq i, j \leq 3 \wedge i \neq j$).

The solution combination method (Section III-C) is designed to combine all the subsets generated in the previous step in order to generate new solutions that become candidate for entering in the *RefSet*. Regarding that the moFLP considers

three different objective functions, a local optimum is found with respect to each one of the by using the improvement method. Then, each improved solution is evaluated for being included in its corresponding *RefSet*. In particular, a solution enters in the *RefSet* if it is better than the worst solution already in the *RefSet*. It is worth mentioning that the size of the *RefSet* remains constant throughout the whole process. Therefore, the new solution must replace another one. Specifically, the new solution replaces the most similar solution already in the *RefSet* that presents a quality smaller or equal than it.

Scatter Search iterates until a stopping criterion is met. The algorithm proposed in this work stops when it has not been possible to include new solutions in any of the *RefSet*, returning the set of efficient solutions found during the search.

A. Diversification Generation Method

The diversification generation method (DGM) is designed for creating an initial set of solutions P which will become the source for creating the initial *RefSet*. On the one hand, P should contain high quality solutions in order to guide the search through promising regions of the search space. On the other hand, solutions in P must be diverse enough to provide different solutions for the combination stage, thus diversifying the search.

We propose three different constructive methods in this work, DGM_1, DGM_2, and DGM_3 each one of them focused on generating promising solutions for f_1, f_2, and f_3, respectively. With the aim of increasing the diversity of the set of solutions constructed, we propose a Greedy Randomized Adaptive Search Procedure (GRASP). GRASP is a multi-start methodology originally proposed by [4], that is conformed by two stages: construction and improvement. The former consists of a greedy, randomized, and adaptive construction of a solution while the latter is designed for finding a local optimum with respect to a predefined neighborhood.

The diversification generation method proposed in this work follows a traditional GRASP construction scheme. The method initially creates a candidate list *CL* with all the candidate locations available to host a facility. Then, each candidate location is evaluated with a greedy function that estimates the relevance of locating a facility in that candidate. For this problem, we propose a different greedy function for each objective function considered f_1, f_2, f_3. In particular, the greedy function value for each candidate location is calculated as the corresponding objective value (f_1, f_2, and f_3, respectively) if the location is assigned to host a facility. After evaluating all the candidates, the method calculates a threshold μ as follows:

$$\mu \leftarrow g_{\min} + \alpha \cdot (g_{\max} - g_{\min})$$

This threshold is used for constructing the restricted candidate list *RCL*, that contains the most promising candidates to host a facility. In particular, the *RCL* is conformed with all the candidate locations whose greedy function value is smaller or equal than threshold μ. For each iteration, a random element is selected from the *RCL* to host the next facility. Notice that $\alpha \in [0, 1]$ is a parameter of the method that controls the randomness of the constructive procedure. On the one hand, if $\alpha = 0$ then the *RCL* would contain the candidates with the minimum greedy function value, being a totally greedy procedure. On the other hand, when $\alpha = 1$, the *RCL* contains all the candidate locations in *CL*, becoming a random procedure. Then, it is interesting to find a balance between diversification and intensification by varying the value of the α parameter. A new candidate is selected in each iteration following this strategy until p candidate locations already host a facility.

B. Improvement Method

The diversification generation method is designed to produce not only high quality solutions, but also diverse ones. The increase in the diversity of the solutions generated usually implies a decrease in the quality of those solutions. Therefore, it is interesting to use a local improving method designed to find a local optimum with respect to a previously defined neighborhood.

Regarding the algorithmic proposal for the moFLP, the solutions included in the set of initial solutions P can be further improved using a local optimizer. Scatter Search is a versatile methodology than allows using different types of optimizers, from local search methods to complete metaheuristics like Tabu Search or VNS, among others (see [REF AUTOCITA] for some successfull application of complete metaheuristics in the improving phase). For this problem, we propose a local search method designed to improve the quality of the initial solutions and of those resulting from the combination method. The computational effort required to evaluate the solutions makes the use of complete metaheuristics not suitable for the problem under consideration.

Prior to define the local improvement method, it is necessary to present the neighborhood of solutions considered by the method. We define the neighborhood $N(S)$ as all the solutions that can be reached from an initial solution S by performing a single *interchange* move, which consists of removing a selected location and replacing it with any non-selected facility location. More formally,

$$N(S) \leftarrow \{S' \ : \ S' \leftarrow S \setminus \{v\} \cup \{u\} \ \forall \ v \in S \ \forall \ u \in V \setminus S\}$$

Having defined $N(S)$, the local search method proposed visits all neighbor solutions in a random order and replaces the incumbent solution with the first neighbor solution with a better objective function value. It is worth mentioning that the local search proposed follows a first improvement approach in order to reduce the computational effort required to apply it. Specifically, if an interchange move results in an improvement with respect to the objective function being optimized, the move is performed, restarting the search with the new best solution. This strategy reduces the complexity of the search since the opposite strategy, best improvement, requires the complete exploration of the neighborhood in each iteration to select the best solution in the neighborhood.

In the framework of Scatter Search, the improvement method is applied in two different stages, see Figure 1. First of all, it is used for locally improving the solutions that are included in the initial population. It is worth mentioning that the set P is divided into three subsets (one for each objective function), so the local search method improves each solution with respect to the objective function considered for its construction.

The local search method is also applied to those solutions resulting from the combination stage. As it is described in Section III-C, the combination stage does not produce local optimum with respect to any neighborhood, so the resulting solutions can be further improved with a local optimizer. Specifically, for each solution derived from the combination stage, three local optima are found, one for each objective function of the moFLP, respectively. Then, each improved solution is evaluated to be included in the *RefSet*.

C. Solution Combination Method

The solution combination method is responsible for generating new solutions in each iteration of the Scatter Search algorithm by combining two or more solutions that are already in the *RefSet*. The combination can be performed following different strategies, from genetic operators to the generation of paths between solutions.

In this work we propose using Path Relinking (PR) [6], [7] as a combination method, which has been successfully applied in several recent works [1], [3]. Given an initial and a guiding solution, s_i and s_g, respectively, PR constructs a path of solutions that starts in s_i and finishes in s_g. The objective of PR is iteratively transform the initial solution into the guiding one. The transformation is achieved by adding attributes of the guiding solution into the initial one while removing those attributes of the initial solution that are not present in the guiding one, stopping when s_i becomes s_g.

There exist several strategies for combining solutions in the context of Path Relinking: Random Path Relinking, Greedy Path Relinking, or Greedy Randomized Path Relinking, among others. Most of the greedy variants require the exploration of all the alternative solutions in each step of the path, in order to select the most promising solution to continue the path. However, these strategies are usually very time consuming, increasing the required computing time to execute the algorithm. We have selected the Random Path Relinking variant in order to accelerate the proposed algorithm, thus increasing the diversification of the search.

Starting from the initial solution s_i, Random Path Relinking generates a random solution in the neighborhood defined in Section III-B that inserts a new candidate location which is already in the guiding solution, removing one of the candidate locations currently in s_i that does not belongs to s_g. Notice that after a certain number of iterations, the initial solution would become the guiding one, since in each iteration the initial solution will have an additional candidate location in common with s_g.

IV. Computational Results

In this section the numerical results are shown in order to prove the superiority of the Scatter Search algorithm with Path Relinking in comparison to the algorithm proposed by Karatas and Yakici in [10].

Since the original instances are not available, we request the code from the previous work in order to have a fair comparison. Additionally, 20 new instances of sizes were generated following the instructions of the previous work. The instances are divided into three sets depending on their size: small, those with 20 candidate locations and 50 demand points; medium, with 50 candidate locations and 100 demand points; and large, with 200 candidate locations and 400 demand points. Table I summarizes the following parameters of the instances generated:

- m: number of candidate locations to host a facility
- n: number of demand points
- p: number of candidate locations that must be selected
- r: radius in which a facility is covering a demand point

We refer the reader to [10] for a more detailed description on the instance structure.

TABLE I
Parameters setting for the instances.

Parameter	Small	Medium	Large
m	20	50	200
n	50	100	400
p	5	10	15
r	20	15	10

All the algorithms proposed in this work have been implemented using Java 8 and the experiments were performed on an Intel Core i7 920 (2.67 GHz) with 8 GB RAM. It is worth mentioning that the previous algorithm has been also executed in the same computer in order to have a fair comparison.

Table II shows average results of both the Scatter Search with Path Relinking algorithm proposed (SSPR) and the best previous method (ITER-FLOC) the 20 instances. Regarding the multi-objective nature of the problem under consideration, we have considered using the following metrics: coverage, C; hypervolume, HV; epsilon indicator, Eps; and CPU time. Results in Table II shows the superiority of the SSPR algorithm.

If we focus on the coverage metric, it can be hold that the proportion of solutions covered by the SSPR algorithm is larger than the proportion of solution covered by the ITER-FLOC algorithm. Furthermore, the SSPR scales better thatn the ITER-FLOC algorithm, as it can be seen with the increase of the solutions covered by SSPR when increasing the size of the instance, achieving a 100% of coverage when analyzing the set of large instances.

Regarding the hypervolume (larger values are better) and the epsilon indicator (smaller values are better), we can conclude that SSPR consistently obtains better results thatn the ITER-FLOC algorithm. Again, the larger the instance set, the better

the results of SSPR when compared against ITER-FLOC. This behavior suggests that SSPR is a more adequate algorithm for real-life problems with a large number of candidate locations and demand points.

Finally, if we consider the computing time, we can see that ITER-FLOC is equivalent or even faster when considering small or medium instances, but the performance does not scale good with the size of the instance. Therefore, when solving the largest instances the proposed SSPR algorithm is considerably faster than the ITER-FLOC approach.

TABLE II
AVERAGE RESULTS.

	C(SSPR,ITER-FLOC)	C(ITER-FLOC,SSPR)
Small	**0.61**	0.17
Medium	**0.82**	0.00
Large	**1.00**	0.00
	HV(SSPR)	HV(ITER-FLOC)
Small	**0.28**	0.07
Medium	**0.52**	0.12
Large	**0.81**	0.20
	Eps(SSPR)	Eps(ITER-FLOC)
Small	**0.44**	0.78
Medium	**0.14**	0.70
Large	**0.00**	0.50
	CPU(SSPR)	CPU(ITER-FLOC)
Small	**6.41**	7.25
Medium	72.66	**19.74**
Large	**1920.53**	5920.34

V. Conclusions

A population-based metaheuristic with a method for combining solutions have been proposed for a multi-objective facility location problem (mo-FLP) which considers three different objectives of interest in real-life problems: to minimize the average distance between demands and their nearest facilities, to maximize the total number of demand points covered, and to minimize the maximum distance between demand points and their nearest facilities.

The computational experiments shows how Scatter Search with Path Relinking is a suitable algorithm for solving large scale instances, performing better in both quality and computing time than the best previous algorithm found in the state of the art. The experiments have been performed in the same computer in order to have comparable results, concluding that SSPR outperforms the best previous method considering all the metrics presented.

Acknowledgment

J. Sánchez-Oro is supported by the Spanish Ministry of "Economía y Competitividad", Grant Refs. TIN2015-65460-C2-2-P and TIN2014-54806-R.

A.D. López-Sánchez and A.G. Hernández-Díaz acknowledge support from the Spanish Ministry of Science and Innovation through Project ECO2016-76567-C4-1-R.

References

[1] V. Campos, R. Martí, J. Sánchez-Oro, A. Duarte, "GRASP with path relinking for the orienteering problem", Journal of the Operational Research Society, vol. 65, pp. 1800–1813, 2014.
[2] R. Church, and C.R. Velle, "The maximal covering location problem," Papers in Regional Science, vol. 32, pp. 101–118,1974.
[3] A. Duarte, J. Sánchez-Oro, M.G.C. Resende, F. Glover, R. Martí, "Greedy randomized adaptive search procedure with exterior path relinking for differential dispersion minimization", Information Sciences, vol. 296, pp. 46–60, 2015.
[4] T. Feo, M.G.C. Resende. "A probabilistic heuristic for a computationally difficult set covering problem", Operations Research Letters, vol. 8, pp. 67–71, 1989.
[5] F. Glover, "Heuristics for integer programming using surrogate constraints," Decision Sciences, vol. 8, pp. 156–166, 1977.
[6] F. Glover"Tabu search and adaptive memory programming – advances, applications, and challenges," Kluwer Academic Publishers, 1996.
[7] F. Glover, and M. Laguna, "Tabu Search," Kluwer Academic Publishers, 1997.
[8] S.L. Hakimi, "Optimum Locations of Switching Centers and the Absolute Centers and Medians of a Graph," Operations Research, vol. 12, pp. 450–459, 1964.
[9] S.L. Hakimi, "Optimum Distribution of Switching Centers in a Communication Network and Some Related Graph Theoretic Problems," Operations Research, vol. 13, pp. 462–475, 1965.
[10] M. Karatas, and E. Yakc, "An iterative solution approach to a multi-objective facility location problem," Applied Soft Computing, vol. 62, pp. 272–287, 2018.
[11] R. Martí, M. Laguna, and F. Glover. "Principles of scatter search," European Journal of Operational Research, vol. 169, pp. 359–372, 2006.
[12] J. Sánchez-Oro, M. Laguna, R. Martí, and A. Duarte. "Scatter Search for the Bandpass Problem", Journal of Global Optimization, vol. 66(4), pp. 769–790, 2016
[13] J. Sánchez-Oro, A. Martínez-Gavara, M. Laguna, A. Duarte, and R. Martí. "Variable neighborhood scatter search for the incremental graph drawing problem", Computational Optimization and Applications, vol. 68(3), pp. 775–797, 2017
[14] J.J. Sylvester, "A Question in the Geometry of Situation," Quarterly Journal of Pure and Applied Mathematics, vol. 1, 1857.