

A new algorithm for the Maximum-weight Planar Subgraph Problem

Paulo A. G. Katakí, Márcia R. Cappelle, Les R. Foulds, Humberto J. Longo

Instituto de Informática – Universidade Federal de Goiás

Alameda Palmeiras, Quadra D, Campus Samambaia, 74001-970, Goiânia, GO, Brasil

{pauloaugustogomes,marcia,lesfoulds,longo}@inf.ufg.br

RESUMO

Algoritmos para o problema de identificar um subgrafo planar de peso máximo de um determinado grafo G com pesos nas arestas são relevantes em uma ampla variedade de áreas de aplicações. Propomos um novo algoritmo heurístico melhoria de busca local para esse problema $\mathcal{NP}$ -difícil, que se baseia em uma transformação em tempo polinomial para o conhecido problema de subgrafo gerador conexo no grafo dual de G . Os testes realizados com o algoritmo proposto, com instâncias numéricas geradas sinteticamente e da literatura, mostraram que o algoritmo geralmente realizado pelo menos bem os métodos heurísticos bem estabelecidos anteriormente para o problema.

PALAVRAS CHAVE. Subgrafo planar, peso nas arestas, máximo, heurística, testes computacionais.

TAG – Teoria e Algoritmos em Grafos, MH – Metaheurísticas, OC – Otimização Combinatória.

ABSTRACT

Algorithms for the problem of identifying a maximum edge-weight planar subgraph of a given edge-weighted graph G are relevant in a wide variety of application areas. We propose a new local search improvement heuristic algorithm for this $\mathcal{NP}$ -hard problem that is based on a polynomial-time transformation to the well-known connected spanning subgraph problem in the dual graph of G . When implemented on synthetically generated numerical instances and those from the literature, the algorithm usually performed at least as well as earlier, well-established heuristic methods for the problem.

KEYWORDS. Subgraph, planar, edge weight, maximum, heuristic, computational experience.

TAG – Theory and Algorithms in Graphs, MH – Metaheuristics, OC – Combinatorial Optimization.

1. Introduction

The analysis of various systems of discrete assets is often facilitated by depicting such a system as a graph where vertices generally represent system assets (such as stocks, facility activities, electrical junctions, proteins or social groups) and edges represent the relationships between them [Massara et al., 2017]. It is often useful to visualise the data by drawing the graph in the plane with edges intersecting only at their endpoints. The edges are commonly nonnegatively weighted due to asset correlations. Often, the best balance between the retention of asset relationship information and the clarity of the graph drawing involves solving a particular combinatorial optimisation model,

the Maximum-weight Planar Subgraph Problem (MWSP), the subject of the present paper, that is now defined. Some of the necessary graph-theoretic notation and terminology is explained later in the present section. Let $G = (V(G), E(G), w)$ be a finite, simple, undirected, edge-weighted graph of order n , with vertex set $V(G)$, edge set $E(G)$ and nonnegative edge weights w . The Maximum-weight Planar Subgraph Problem (MWSP) involves searching G for a planar, spanning subgraph $G' = (V(G), E(G'))$ with $E(G') \subseteq E(G)$, having the highest sum of edge weights $s(G') = \sum_{e \in E(G')} w(e)$ among all such subgraphs of G . This instance of the MWSP is denoted by $MWSP(G, w)$ and $s(G')$ is termed the *value* of G' when it is viewed as a solution to $MWSP(G, w)$.

The MWSP has many important applications including (i) the analysis of financial data, where the vertices represent stocks or shares and the edges, their correlations [Massara et al., 2017], (ii) as a subproblem of the facility (plant) layout problem, where the vertices represent the activities of the facility and the edges, the adjacencies between them [Ahmadi-Javid et al., 2015], (iii) integrated circuit design, where the vertices represent electrical elements and the edges, the physical connections between them [Lengauer, 2012], (iv) systems biology, where the vertices represent proteins and the edges, protein interactions in a metabolic network [Song et al., 2012] and (v) social systems, where the vertices represent social agents (e.g. individuals, groups or companies) and the edges, social interactions [Easley and Kleinberg, 2010].

Some of the graph-theoretic concepts to be used are now defined and discussed. Further details are available, for instance, in the book by Bondy and Murty [2010]. A graph is termed *k-regular* if each of its vertices is of degree k , a given positive integer. A 3-regular graph is termed *cubic*. An internal edge of a cycle C in a graph is termed a *chord* of C . A *planar* graph G is a graph that can be embedded in the plane without any edge crossings, G is a *maximal* planar graph if it is planar and adding any missing edge to it renders it no longer planar, and G is a *plane* graph if it is an actual drawing in the plane without any edge crossings. The regions of a plane graph are termed *faces*. If the graph is maximal planar its faces are here termed *triangles*. Planarity ensures easy visualisation of the main asset relationships by enabling the graph to be drawn without edges crossing. Note that G is not necessarily complete. However, if G does happen to be complete, because the edge weights are all assumed to be nonnegative, there will be an optimal solution to any numerical instance of the MWSP that is a maximal planar subgraph. In any case, we can confine the search for MWSP optimal solutions to the planar spanning subgraphs of G .

A combinatorial problem relevant to the present study is the Connected Spanning Subgraph Problem (CSSP) that involves searching a given graph H for a connected, spanning, k -regular subgraph of H , that is denoted by $CSSP(H, k)$. Cheah and Corneil [1990] showed that the edge-weight maximising version of the CSSP is $\mathcal{NP}$ -hard for $k \geq 3$. As $CSSP(H, 3)$ can be polynomially reduced to $MWSP(G, w)$ when H is the dual of G , the MWSP is also $\mathcal{NP}$ -hard. The special case where $w(e) = c, \forall e \in E(G)$, degenerates to finding a planar subgraph with $3n - 6$ edges, which can be solved in linear time ([Jünger and Mutzel, 1993]).

The objective of the present paper is to present a local search improvement heuristic algorithm for the MWSP (Section 4), based on an MWSP-to-CSSP polynomial-time transformation (Section 2). The proposed algorithm is designed to attempt to improve a given solution to a particular MWSP numerical instance by increasing its total edge weight. The solution is first transformed into an instance of an equivalent CSSP instance, which is solved heuristically and finally, the CSSP solution is converted back into a hopefully improved solution to the original MWSP instance. Some results that underpin the proposed algorithm are given in the Section 3. Computational experience in implementing the heuristic is reported in the Section 5. The present paper is rounded out

with a summary in the final section.

2. The MWSP-to-CSSP transformation

If a graph G has subgraphs (i) $B = (V(B), E(B))$, an arbitrary graph and (ii) $C = (V(C), E(C))$, a cycle; B is said to be *enclosed within* C if $V(B) \subsetneq V(C)$. Suppose G' is a planar subgraph of G , and has the 3-cycle $\langle a, b, c \rangle$ as a face. Figures 1(a) to 1(c) show the only three possibilities of the 3-cycle $\langle a, b, c \rangle$ being enclosed within 4-, 5- and 6-cycles of G' , having one, two and three other adjacent triangles, respectively.

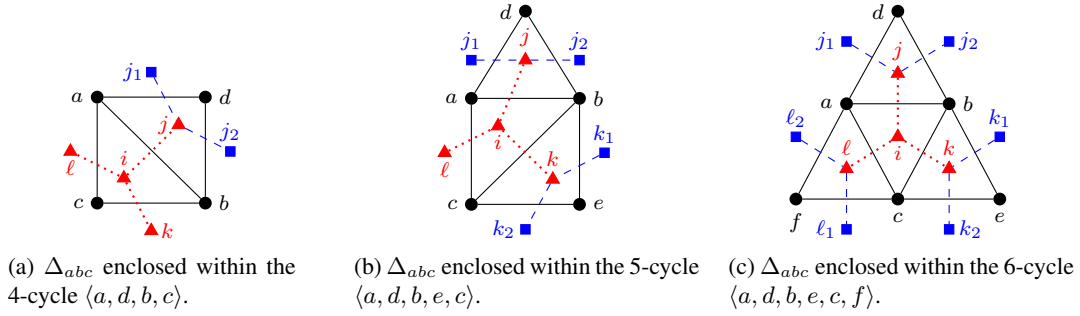


Figure 1: Examples of a triangle Δ_{abc} being enclosed within 4-, 5- and 6-cycles in G' and the corresponding star gadgets in the dual graph H' .

In preparation for the description of the proposed MWSP algorithm, we recall some well-known results concerning cycles in graphs [Kreweras, 1972]. Suppose now, G' is a spanning, maximal planar subgraph of K_n , the complete graph of order $n \geq 4$, and C is a p -cycle in K_n :

- the maximum number of chords enclosed within the p -cycle C in K_n is $C(p) = \binom{p}{2} - p = \frac{p^2-3p}{2}$;
- clearly, there is a combination of $\binom{C(p)}{p-3}$ (of order $O(n^6)$) of such chords when they are taken three at a time;
- the number of combinations of such chords that may constitute a planar partition of C in G' is the Catalan number $C_{p-2} = \frac{(2p-4)!}{(p-2)!(p-1)!}$;
- K_n has $\binom{n}{3} = \frac{n^3-3n^2+2n}{6}$ 3-cycles and as G' is maximal planar, it has $2n-4$ triangles among those 3-cycles.

The above concepts for $p = 4, 5, 6$ are illustrated in Figure 1 and are discussed in the next section. Let the simple, planar graph $G = (V(G), E(G))$ be a plane graph. The dual $G^* = (V(G^*), E(G^*))$ of G is a new plane graph having a vertex in $V(G^*)$ for each face in G and edges in $E(G^*)$ that correspond to the edges in $E(G)$ as follows. If edge $e \in E(G)$ separates two faces f_1 and f_2 of G , then the corresponding dual edge $e^* \in E(G^*)$ is an edge joining the vertices in G^* that correspond to f_1 and f_2 . Although a dual is sometimes called a *combinatorial dual* or an *algebraic dual*, they are equivalent for a planar graph, and just the term *dual* is adopted here.

We now show how any numerical instance of the MWSP can be transformed into an instance of the CSSP. Any planar subgraph of a given edge-weighted graph $G = (V(G), E(G), w)$, is termed a *feasible* solution to $MWSP(G, w)$. Let K_n^w be a complete graph of order n with nonnegative edge weights defined by an edge weight function w . Naturally, $MWSP(K_n, w)$ is an instance of the MWSP. Furthermore, let $G' = (V(G'), E(G'))$ be a maximal planar, spanning subgraph of K_n^w , i.e., a feasible solution to $MWSP(K_n, w)$ with $V(G') = V(K_n^w)$ and $E(G') \subseteq E(K_n^w)$. Assume further that G' is a plane graph. (As any feasible solution to the MWSP is planar, it can be represented as a plane graph.) Let $v = |V(G')|$, $e = |E(G')|$ and $f = |T(G')|$, where

$T(G')$ denotes the set of triangles in G' , including the exterior region. Then, $v = n$ and $e = 3n - 6$. By Euler's formula, $v - e + f = 2$ and $f = 2n - 4$ ([Bondy and Murty, 2010]).

Let Δ_{abc} denote the triangle of $T(G')$ that has $a, b, c \in V(G')$ as its vertices and let $H = (V(H), E(H), \varphi)$ be the edge-weighted dual of G' , which is also assumed to be a plane graph. Hence, $V(H) = \{u_{abc} \mid \Delta_{abc} \in T(G')\}$ and $E(H) = \{\{u_{abc}, u_{abd}\} \mid \Delta_{abc}, \Delta_{abd} \in T(G') \text{ having edge } \{a, b\} \in E(G') \text{ in common}\}$. The weight of the edge $\{u_{abc}, u_{abd}\}$ is defined as $\varphi(\{u_{abc}, u_{abd}\}) = w(\{a, b\})$. Let $v_H = |V(H)|$, $e_H = |E(H)|$ and $f_H = |F(H)|$, where $F(H)$ denotes the set of faces of H . Then, $v_H = 2n - 4$ and $e_H = 3n - 6$. Once again, by Euler's formula, $f_H = n$. When $n \geq 4$, each triangle $t \in T(G')$ has exactly three other neighbouring triangles adjacent to it and each has a distinct edge in common with t . Thus, the vertex in H corresponding to t is adjacent to exactly three other vertices in H and hence, H is a cubic graph. If G' is a given plane graph, the plane graph H always exists and is easy to find from G' .

3. Some preliminary results

Lemma 1. *Let G be a maximal planar graph of order at least four. Any triangle in G is enclosed within a p -cycle in G , where $4 \leq p \leq 6$.*

Proof. Suppose Δ_{abc} is a triangle of a maximal planar graph G . Due to the nature of maximal planarity, Δ_{abc} must have a distinct edge in common with q other triangles in G , where $1 \leq q \leq 3$. Thus, the vertices a, b and c and the vertices of the other triangles induce a cycle with 4, 5 or 6 vertices, respectively. $\square$

Throughout the rest of this article, let G' be a given, maximal planar graph that has a dual graph H that is planar, connected, spanning and cubic. Consider a subgraph of H that is induced by a single vertex $i \in V(H)$, and its neighbouring vertices $j, k, \ell \in V(H)$. The subgraph is a star graph with hub i (of degree three) which is termed a *star gadget* and is denoted by $i \star jk\ell$. We define a *star move* to be the particular operation of replacing a star gadget in H by a distinct star gadget. The operation is equivalent to replacing a triangle t , in G' that is enclosed within a 4-, 5- or 6-cycle C , by another triangle enclosed within C . Figure 1 illustrates these three cases of triangle enclosure. The star move does not affect any of the triangles outside C .

Referring to Figure 1, suppose a star move $i \star jk\ell$, is applied in H , where $\{j, j_1\}, \{j, j_2\}, \{k, k_1\}, \{k, k_2\}, \{\ell, \ell_1\}, \{\ell, \ell_2\} \in E(H)$. With the exception of vertex i , let $A = \{j_1, j_2, k_1, k_2, \ell_1, \ell_2\}$ be the set of the vertices in H adjacent to j, k and ℓ . With the removal of the star gadget $i \star jk\ell$, the degrees of the vertices in A are all decreased by unity. The insertion of a new star gadget $i' \star j'k'\ell'$, connects each of the vertices j', k' and ℓ' to two of the vertices of A in a way that restores the degrees of all the vertices in A to three. Hence, the new graph resulting from this operation is another planar, cubic graph, different from H .

The case where t is relocated within a cycle different from C has already been investigated ([Foulds and Robinson, 1979]). A subgraph generated by a star move is termed *valid* if it is simple (i.e., without self-loops nor parallel edges), planar, connected, spanning and cubic. A star move is termed *valid* if it generates a valid subgraph. A subgraph generated by a star move, or a star move itself, is termed *invalid* if it is not valid. Let $\text{CSSP}(H, k)$ denote the instance of the CSSP based on a given connected graph H for some positive integer k . Any connected, spanning subgraph of a given graph H is termed a *feasible* solution to $\text{CSSP}(H, k)$. Also, let S_H be the set of graphs that can be generated by a finite sequence of star moves applied to H , including the null sequence that corresponds to H itself. The following result is discussed further in the next section:

Result 1. *G' is a feasible solution to $\text{MWSP}(G, w)$ if, and only if, H' is a feasible solution to $\text{CSSP}(H, 3)$, where H is the dual of G' and $H' \in S_H$.*

If the feasible solutions G' and H' are related in the way just described, their edges are matched and have identical weights. Thus, it follows that the values of these two solutions (the total sums of their edge weights) are equal. Result 1 is of computational convenience as it is often possible improve the value of a solution to the MWSPS by working with dual graphs in the CSSP environment, as detailed in the description of the proposed algorithm in Section 4. Such improvements for the numerical instances of various previously proposed MWSPS problems are presented in Section 5.

Figure 1 illustrates some examples of the graphs G' and H' . For instance, Figure 1(c) depicts the triangle Δ_{abc} , and its neighboring triangles Δ_{abd} , Δ_{bce} and Δ_{acf} (all in G) and the corresponding dual vertices $i = u_{abc}$, $j = u_{abd}$, $k = u_{bce}$ and $\ell = u_{acf}$ in $V(H')$. For example, the edge $\{i, j\} = \{u_{abc}, u_{abd}\} \in E(H')$ corresponds to edge $\{a, b\} \in E(G')$ (a member of both Δ_{abc} and Δ_{abd}). Also, Δ_{abd} , Δ_{bce} and Δ_{acf} each have a distinct edge of Δ_{abc} and two additional, distinct edges of $E(G')$. The counterpart in H' is the star gadget $i \star j k \ell$ comprising a central vertex $i = u_{abc}$ and three other vertices adjacent to that vertex: $j = u_{abd}$, $k = u_{bce}$ and $\ell = u_{acf}$, as highlighted in the figure. Next, the cases of the inclusion of triangles in 4-, 5- and 6-cycles in G' and their counterparts in H are analysed.

Firstly, suppose a triangle $t = \Delta_{abc}$ is enclosed within a 4-cycle $C_4 = \langle a, d, b, c \rangle$, of G' , as shown in Figure 1(a). Replacing t by another triangle of G' , while preserving C_4 , can be achieved by replacing a single edge of t by the other diagonal of C_4 . For example, edge $\{a, b\}$ of t can be replaced by edge $\{c, d\}$, if $\{c, d\}$ is not already in $E(G')$. This case corresponds to the edge-cut move proposed by Pesch et al. [1999]. (The case when $\{c, d\} \in E(G')$ is covered in [Foulds and Robinson, 1979]. Furthermore, it has been shown that a finite sequence of edge replacement operations (as just described) can be found to transform G' into a maximal planar graph with the same vertex set that is of maximum weight [Ning, 1987].

Let vertices $i, i', j, j', j_1, j_2, k, \ell \in V(H')$ have adjacencies as shown in Figure 2. The corresponding star move in H' (the dual of G') involves replacing the star gadget in H' , $i \star j k \ell$ say, by a new star gadget in H' that preserves two of the vertices j, k and ℓ . Referring to Figure 2(a), if the star gadget $i \star j k \ell$ is extracted from H' , there is only one possible subgraph in G' where Δ_{abc} is replaced by another triangle, such that its enclosing cycle C_4 is preserved, as shown in Figures 2(b) and 2(c). The figures also show the only two possibilities for the replacement of the star gadget $i \star j k \ell$. Only one edge of Δ_{abc} in G' is affected by the star move and the vertices in H' corresponding to the triangles outside C_4 in G' are not affected by the move.

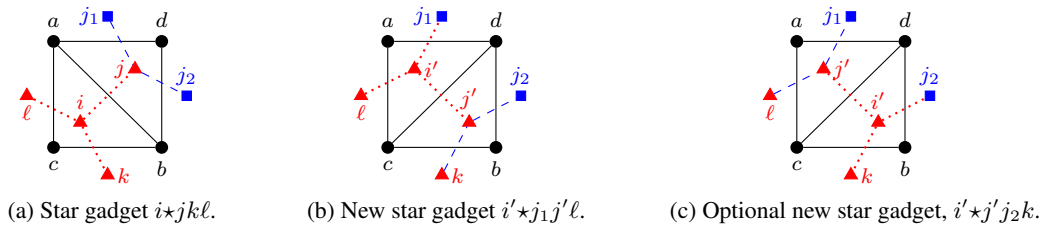


Figure 2: Examples of star gadget replacements in H' based on a 4-cycle in G' .

Secondly, suppose a triangle $t = \Delta_{abc}$ is enclosed within a 5-cycle $C_5 = \langle a, d, b, e, c \rangle$ of G' as shown in Figure 1(b). Replacing t by another triangle of G' , while preserving C_5 , can be achieved by replacing one (or two) of the edges of t by one (or two) new edges (not yet present in G'). This case corresponds to the partner-rotation move proposed by Pesch et al. [1999]. The analysis of the case where one or both of the new edges already members of G' is similar to that for

the analogous case for the 4-cycle case above and is not included here.

Replacing $i \star jkl$ of H' by another *valid* star gadget, or not making a replacement at all, leads to five possible subgraphs that preserve C_5 in G' . The equivalent replacement of the star gadget $i \star jkl$ of H' in Figure 1(b) by a new valid star gadget preserves either one or two of the vertices j, k and ℓ , and also their adjacent vertices j_1, j_2, k_1 and k_2 , shown in the figure, depending on which triangle in G' corresponds to the central vertex i , of the star gadget. Following the figure, it assumed that t corresponds to i . The cases where either Δ_{abd} or Δ_{acf} correspond to i follow from the previous 4-cycle case. Simplified versions of these subgraphs are shown in Figure 3. The case shown in Figure 3(b) can be constructed from the case in Figure 3(a) by extracting the 2-path $\langle a, b, c \rangle$ (along with the edges $\{a, b\}$ and $\{b, c\}$) and inserting the 2-path $\langle d, e, a \rangle$. In fact, any case shown in Figures 3(a) to 3(e) can be generated from any other case by replacing a 2-path enclosed within C_5 by another 2-path enclosed within C_5 . At most two edges of t are affected by the star move and the vertices in H' corresponding to the triangles outside C_5 are not affected.

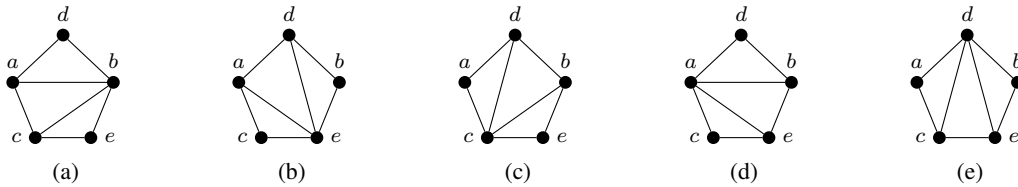


Figure 3: The planar triangulations of the vertex-labelled 5-cycle.

Thirdly, suppose a triangle $t = \Delta_{abc}$ is enclosed within a 6-cycle $C_6 = \langle a, d, b, e, c, f \rangle$ of G' , as shown in Figure 1(c). Replacing t by another triangle of G' , while preserving C_6 , can be achieved by replacing either one, two or three of the edges of t by the same number of new edges (not being members of $E(G')$). The analysis of the case when at least one of the incoming edges is already a member $E(G')$ can be based on the similar 4- and 5-cycle cases above and is left to the reader. The corresponding replacement of the star gadget $i \star jkl$ of H' in Figure 1(c) preserves at most two of the vertices j, k and ℓ .

There are 84 possible combinations of three edges as chords enclosed within a vertex-labelled 6-cycle. However, there are only 14 distinct *planar* subgraphs that can be enclosed within C_6 , as shown in Figure 4. Any plane subgraph G_1 say, can be obtained from some other subgraph G_2 say, by taking the structure formed by the edges of G_2 that are enclosed within C_6 and rotating it either clockwise or anticlockwise. In these particular cases, all three edges of t in G' are affected by the star move in H' . In general, p edges of t are affected by the star move, where $1 \leq p \leq 3$, and the vertices in H' corresponding to triangles outside C_6 in H' are not affected.

As can be seen in Figure 4, the subgraphs have been grouped into four blocks (B1–B4). The purpose of this division is to avoid working with invalid star moves or those that generate repeated subgraphs, as explained further in Section 4. Examples of the rotating structures are given below:

- B1: The triangle $\langle a, b, c \rangle$ in Figure 4(a) and its rotation, the triangle $\langle d, e, f \rangle$ in Figure 4(b).
- B2: The 3-path $\langle f, d, c, b \rangle$ in Figure 4(c) and its rotations, the 3-paths $\langle a, b, f, e \rangle$ in Figure 4(d) and $\langle d, e, a, c \rangle$ in Figure 4(e).
- B3: The 3-path $\langle d, f, b, c \rangle$ in Figure 4(f) and its rotations, the 3-paths $\langle b, a, e, f \rangle$ in Figure 4(g) and $\langle e, d, c, a \rangle$ in Figure 4(h), being reflections of the graphs in the previous block.
- B4: The tree $\langle a, b, e, c, \rangle$ rooted at a in Figure 4(i) and its rotations, the tree $\langle d, e, c, f \rangle$ rooted at d in Figure 4(j), the tree $\langle b, c, f, a \rangle$ rooted at b in Figure 4(k), the tree $\langle e, f, a, d \rangle$ rooted at f in Figure 4(l).

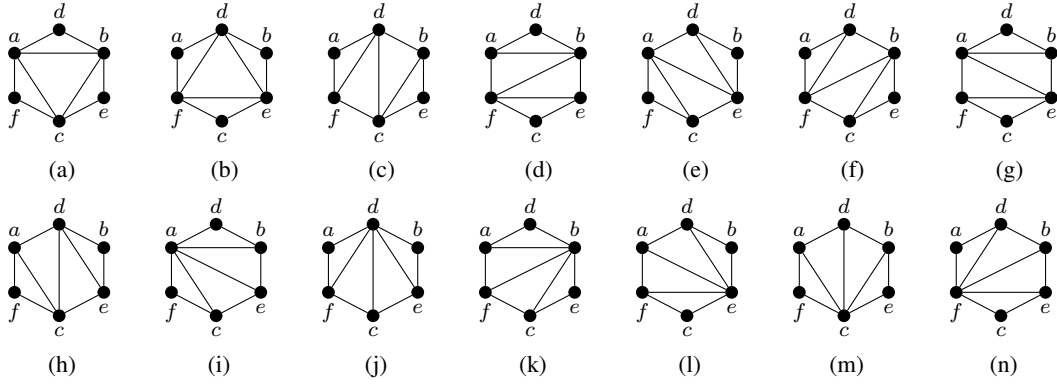


Figure 4: The planar triangulations of the vertex-labelled 6-cycle.

Figure 4(l), the tree $\langle c, a, d, b \rangle$ rooted at c in Figure 4(m) and the tree $\langle f, d, b, e \rangle$ rooted at f in Figure 4(n).

Still regarding the enclosing cycles in G' , the two cases of enclosing a triangle t within a 4-cycle are covered by those of enclosing t within a 5-cycle. The two cases shown in Figure 2 are the same as those shown in Figures 3(a) and 3(c) (neglecting vertex e). The cases of enclosing t within 5-cycles are covered by those of enclosing t within 6-cycles. Thus, the cases shown in Figures 3(a)–3(e) are the same as those shown in Figures 4(a), 4(e), 4(m), 4(i) and 4(h), respectively, when vertex f is neglected. Thus, it is necessary to consider only the star moves in H' that correspond to triangles enclosed within 6-cycles in G' . This fact is used to guide the choice of which star move to apply when implementing the proposed MWPSp heuristic algorithm, as described in the next section.

4. The proposed MWPSp algorithm: The Cubic Heuristic

The proposed local search MWPSp heuristic algorithm, called the *Cubic Heuristic* ($\mathcal{CH}$), is based on the results described in the previous section. In particular, it uses the fact that a feasible solution to an instance of the MWPSp can be modified by selecting an enclosing 6-cycle and replacing the chords of that cycle. The algorithm always preserves planarity while walking through neighbouring solutions, being graphs generated by various edge replacements. This fact eliminates the necessity of using reported graph planarity testing routines that, although linear, are still costly.

The proposed algorithm works on a cubic graph H' which is the dual of a given plane maximal planar subgraph G' , being a solution to an instance of $MWPSp(G, w)$. The proposed algorithm outputs a new plane maximal planar subgraph G'' of G , the dual of the new cubic graph H'' obtained with star moves executed in H' . The basic idea is that it is usually easier to identify a star gadget in H' than to identify a possible rotating structure embedded within a 6-cycle in G' . Thus, throughout the execution of the $\mathcal{CH}$ heuristic, star gadgets are sequentially identified in H' and the appropriate star move is performed. The equivalent in G is a sequence of maximal planar subgraphs that span $V(G)$. However, only the final maximal planar graph identified from G' is actually constructed. The $\mathcal{CH}$ heuristic algorithm comprises two parts, with their pseudocodes given as Algorithms 1 and 2 below. The overall time complexity of $\mathcal{CH}$ is $\mathcal{O}(n^2)$.

Algorithm 1 receives as input a maximal planar graph G' (a feasible solution to an instance $MWPSp(G, w)$) and outputs a further maximal planar graph that is hopefully, of higher weight than G' . It does this by first building the cubic graph H that is the dual of G' (Step 1). The time complexity of this step is $\mathcal{O}(n)$, and this can be achieved by using a face traversal algorithm [Boost

C++ Libraries, 2020]. The algorithm then iterates for a pre-specified number of times, attempting to increase the value of the current dual graph at each iteration, while preserving its planarity.

Algorithm 1: CubicHeuristic(G' , k , $nbIterations$)

Input: Maximal planar edge-weighted graph G' ; ... k and number of iterations $nbIterations$.

Output: Maximal planar edge-weighted graph G'' .

```

1   $H \leftarrow \text{buildCubicGraph}( G' );$ 
2   $bestGraph \leftarrow H;$ 
3   $bagGraphs \leftarrow \emptyset;$ 
4  for  $iteration \leftarrow 1$  to  $nbIterations$  do
5       $neighbours \leftarrow \text{getKNeighbours}( H, k );$ 
6       $bestNeighbour \leftarrow \text{getBestNeighbour}( neighbours, bagGraphs );$ 
7       $\text{addNeighboursToBag}( bagGraphs, neighbours );$ 
8      if  $\text{isValid}(bestNeighbour)$  then
9           $H \leftarrow bestNeighbour;$ 
10     else
11          $H \leftarrow \text{getBestGraph}( bagGraphs );$ 
12          $\text{delete}( bagGraphs, H );$ 
13      $\text{visit}( H );$ 
14     if  $(\text{getCost}( H ) > \text{getCost}( bestGraph ))$  then
15          $bestGraph \leftarrow H;$ 
16  $G'' \leftarrow \text{buildMaximalPlanar}(bestGraph);$ 
17 return  $G''$ .
```

The core of Algorithm 1 is the construction of new cubic graphs that can be obtained from H by star moves (neighbours of H). To do this, the routine “getKNeighbours()” (Step 5) identifies all the neighbours of the current graph H that can be generated by a single star move and takes the k ’s with the highest edge-weights sums, that is done in $\mathcal{O}(n^2)$ time, as explained latter. The routine “getBestNeighbour()”, at Step 6, identifies the heaviest of the chosen k neighbors that is not yet in the “bagGraphs” set. The complexity of this step is $\mathcal{O}(\lg(|bagGraphs|)) = \mathcal{O}(1)$ (where $\lg(n)$ denotes $\log_2(n)$), since the set size $|bagGraphs|$ is a constant. These chosen k neighbours of H are stored in the $bagGraphs$ set routine “addNeighboursToBag()” at Step 7, which takes $\mathcal{O}(n \lg(|bagGraphs|))$ time, since the number of possible neighbours is at most $14 \cdot (2n - 4)$ (see Section 3).

Step 8 is used to check if the current H best neighbour has itself at least one neighbour that is yet not a member of the $bagGraphs$ set. If this is the case, it will be the graph H in the next iteration (Step 9). Otherwise, the graph of maximum value in the set $bagGraphs$ is used (routine “getBestGraph()” at Step 11), that is excluded from the $bagGraphs$ set (Step 12) so as not to be chosen again later. This operation can be done in $\mathcal{O}(\lg(|bagGraphs|))$ time when a balanced binary search tree is used to store the $bagGraphs$ set. After this part, H is inserted into the set of visited graphs (Step 13, in time complexity $\mathcal{O}(n)$) and a check is made to see if the value $s(H)$ of the current graph H is higher than that of the heaviest one found so far (Steps 14–15, in time complexity $\mathcal{O}(1)$). At the end of the number of pre-specified iterations, the best (heaviest) cubic graph found is converted back to its maximal planar dual graph (routine “buildMaximalPlanar()” at Step 16, in time complexity $\mathcal{O}(n)$).

Algorithm 2 receives as input a cubic graph H and a positive integer k , and outputs a set containing the k heaviest neighbours of H . It does this by first calling the routine “buildStars()”, in Step 1, to construct the set S of all star gadgets in H . The time complexity of this routine is $\mathcal{O}(n^2)$, since it is necessary to go through all the $2n - 4$ vertices of H , (that correspond to the faces of G') and check their adjacency. Then, the set *starNeighbours* of pairs of star gadgets is constructed. This set contains all pairs formed by an original star gadget and an additional star gadget that could replace it after a star move operation on H (Steps 4–7). The time complexity for this is $\mathcal{O}(n)$, since the set S has size $\mathcal{O}(n)$ and each star gadget in S has at most 14 neighbours. Next, the members in the set *starNeighbours* are ordered according to the star gadget improvement factor, i.e., the difference between the weight of an original star that is the input to a star move operation and the weight of the precedent star gadget (Step 8 that has time complexity $\mathcal{O}(n \lg n)$).

Algorithm 2: getKNeighbours(H, k)

Input: A cubic graph H and an integer k

Output: Set containing the k heaviest neighbours of H .

```

1   $S \leftarrow \text{buildStars}( H );$ 
2   $\text{starsNeighbours} \leftarrow \emptyset;$ 
3   $\text{bestNeighbours} \leftarrow \emptyset;$ 
4  foreach  $s \in S$  do
5       $C \leftarrow \text{generatePossibleStars}( s, H );$ 
6      foreach  $c \in C$  do
7           $\text{starsNeighbours} \leftarrow \text{starsNeighbours} \cup \{s, c\};$ 
8   $\text{sort}( \text{starsNeighbours} );$ 
9  foreach  $\{star, newStar\} \in \text{starsNeighbours}$  do
10     if ( $\text{isNotVisited}( newStar )$  and ( $|\text{bestNeighbours}| < k$ )) then
11          $H' \leftarrow \text{buildGraph}( H, star, newStar );$ 
12          $\text{bestNeighbours} \leftarrow \text{bestNeighbours} \cup \{H'\};$ 
13          $\text{visit}( newStar );$ 
14     if  $\text{gain}( star, newStar ) < 0$  and  $\text{isNotVisited}( newStar )$  then
15          $\text{visit}( newStar );$ 
16 return  $\text{bestNeighbours};$ 
```

The main part of Algorithm 2 is the loop that goes from Step 9 to 15, which iterates over the pairs of star gadgets that can generate a new neighbour of the graph H , with time complexity $\mathcal{O}(n^2)$. As the *starNeighbours* set is ordered by the gain of replacing one star gadget by another, naturally, the first graphs generated by this approach will be the heaviest. Step 10 is used to check whether the *newStar* gadget (which may replace the *star* gadget in H) has not yet been visited. That is, the step is used to check if the $star \leftrightarrow newStar$ exchange has not yet been performed. As the enclosing 6-cycle of each star gadget is unique, this verification considers only the *newStar* gadget. The step also checks if the set cardinality of *bestNeighbours* is at most k . When these two conditions are met, a new graph H' is built by replacing the star gadget *star* by the *newStar* gadget (Step 11). Then, H' is inserted in the set *bestNeighbours* (Step 12) and the *newStar* gadget is marked by the “visit()” routine (Step 13) as one that has already been analysed.

In order to depart from a locally optimal solution, it is sometimes productive to allow a $star \leftrightarrow newStar$ exchange that decreases the value $s(H)$ of a graph H . If *newStar* has not yet

been visited, it may possibly be visited, leading to a temporary decrease in total weight. Steps 14 – 15 take this possibility into account.

5. Computational results

The experimental work was conducted on a computer running Ubuntu 18.04.3 LTS 64-bits, with 40 Intel CPU Xeon E5-2660 v3 2.60GHz and 128GB of ECC RAM. The CPU code was compiled with the GCC 4.8.5, O3 optimisation flag set and the code targeted the native architecture. The experiments used various synthetically generated MWSP instances $MWSP(K_n, w)$, $n = 10, 15, \dots, 100$; with $w(e), e \in E(K_n)$ being randomly generated in the range $[0, 199]$, as proposed by Coelho et al. [2020]. The instances were heuristically solved by the so-called *face-dimpling* (FD) and *face-edge dimpling* (FED) algorithms ([Coelho et al., 2020]) that currently appear to be among the best-performing heuristic methods for the MWSP. In order to evaluate the performance of the $\mathcal{CH}$ algorithm, it was used to attempt to improve the solutions just mentioned.

Table 1 displays the results of the experiments. The numbers in the second column are upper bounds on the values of the optimal solutions, obtained by rounding up solutions values calculated with computer codes that implement an exact MWSP algorithm and some bounding techniques proposed by Ahmadi-Javid et al. [2015]. The numbers in bold represent the values of optimal solutions, i.e., are sharp bounds. The columns labelled FD and FED display values of solutions to the instances and their percentage gaps from the upper bound. The values were obtained by Coelho et al. [2020] by using the face dimpling and face-edge dimpling algorithms, respectively. The columns (“ $\mathcal{CH}$ over FD ” and “ $\mathcal{CH}$ over FED ”) list the solution values obtained by using the $\mathcal{CH}$ algorithm to improve the solutions produced by FD and FED , respectively.

n	Upper	FD		$\mathcal{CH}$ over FD		FED		$\mathcal{CH}$ over FED	
	Bound	Solution	Gap (%)	Solution	Gap (%)	Solution	Gap (%)	Solution	Gap (%)
10	2107	2107	0.000	—	—	2107	0.000	—	—
15	5772	5743	0.502	5768	0.069	5772	0.000	—	—
20	8434	8061	4.423	8429	0.059	8321	1.340	8434	0.000
25	11594	11291	2.613	11528	0.569	11431	1.406	11594	0.000
30	13770	13435	2.433	13714	0.407	13644	0.915	13744	0.189
35	17177	16833	2.003	16919	1.502	17028	0.867	17128	0.285
40	19862	19232	3.172	19632	1.158	19454	2.054	19605	1.294
45	22384	21486	4.014	21794	2.636	21669	3.197	21755	2.810
50	25763	24847	3.557	25054	2.752	25126	2.474	25162	2.333
55	28536	27461	3.768	27661	3.066	27771	2.681	27937	2.099
60	31272	30085	3.799	30283	3.163	30432	2.689	30576	2.226
65	34226	32889	3.908	32939	3.760	33119	3.236	33190	3.027
70	37493	36103	3.708	36415	2.875	36410	2.889	36569	2.464
75	40130	38600	3.814	38723	3.506	38953	2.935	39022	2.761
80	43058	41497	3.626	41589	3.412	41702	3.150	41900	2.689
85	45701	43943	3.847	44277	3.116	44268	3.136	44407	2.831
90	48828	46977	3.792	47061	3.619	47301	3.128	47313	3.103
95	51626	49673	3.783	49812	3.514	49961	3.225	50019	3.113
100	54875	53037	3.350	53100	3.235	53377	2.730	53388	2.710
Average gaps (%)			3.164		2.357		2.213		1.996

Table 1: Experimental results showing the improvements of the $\mathcal{CH}$ algorithm over the face dimpling (FD) and face-edge dimpling (FED) methods.

The gaps between the upper bounds on the optimal solution values and the the solution values generated by both the FD and FED algorithms were decreased by the $\mathcal{CH}$ algorithm. These

measures are given as percentages in the columns labelled as “Gap” in Table 1. For the collection of instances with known optimal solutions ($10 \leq n \leq 40$), the average gap was reduced by almost 2% (1%) compared to the *FD* (*FED*) solutions. The *CH* algorithm had better performance when it was applied to the solutions generated by the *FED* algorithm, also finding optimal solutions in the cases where $n = 20$ and $n = 25$. However, for $n = 40$ and $n = 45$, applying *CH* to the solutions provided by *FD* as seeds produced superior results, compared to those provided by *FED* as seeds. The last line of the table shows the average gaps obtained with each approach.

In order to gauge the relative solution quality of the *CH* method, the 100-vertex, WMPSP instance of Tumminello et al. [2005] was used. An upper bound on the solution value of the instance was calculated using the bounding techniques proposed by Ahmadi-Javid et al. [2015]. The instance was solved by (i) the PMFG method of Tumminello et al. [2005], (ii) the TMFG method of Massara et al. [2017], (iii), (iv) the *FD* and *FED* methods of Coelho et al. [2020] and (v), (vi) the *CH* method with the seed solution being the *FD* and *FED* solutions, respectively. The results are shown in Table 2. The PMFG, TMFG, *FD*, *FED* and Upper-Bound results shown in the table were originally obtained by Coelho et al. [2020]. The lines “*CH* over *FD*” and “*CH* over *FED*” display the solution values generated by *CH* and are original to the present paper. As can be seen in the table, the solution values produced by the *CH* over *FED*, *CH* over *FD*, *FED*, *FD*, *TMFG* and *PMFG* algorithms are of decreasing value, with *CH* over *FED* being the best-performed method and *PMFG* being the worst-performed.

Method	Solution value	Gap (%)
Upper-Bound	121.835	—
<i>CH</i> over <i>FED</i>	116.964	3.998
<i>CH</i> over <i>FD</i>	116.623	4.278
<i>FED</i>	116.897	4.053
<i>FD</i>	116.553	4.335
<i>TMFG</i>	116.224	4.605
<i>PMFG</i>	115.568	5.144

Table 2: Solution values for the 100-vertex instance [Tumminello et al., 2005].

6. Summary

A new polynomial-time improvement search heuristic algorithm is proposed for the maximum edge-weight planar subgraph problem (MWSP). The algorithm transforms a given MWSP instance (G, w) into an equivalent instance of the connected spanning subgraph problem (CSSP) in the dual of G . An approximate solution technique is applied to the CSSP instance to identify a feasible solution to it that is then converted back to a solution to the original MWSP instance. The proposed algorithm was applied to various numerical instances, nineteen constructed synthetically and one 100-vertex instance from the literature. The solutions obtained from the instances created synthetically are on average within 2% of optimality and at most 3.1% in the worst case. The algorithm managed to improve the best solution found so far for the instance from the literature and, for the first time, its optimality gap is now below 4%.

References

- Ahmadi-Javid, A., Ardestani-Jaafari, A., Foulds, L. R., Hojabri, H., and Farahani, R. Z. (2015). An algorithm and upper bounds for the weighted maximal planar graph problem. *Journal of the Operational Research Society*, 66(8):1399–1412.

- Bondy, J.-A. and Murty, U. S. R. (2010). *Graph Theory*, volume 244 of *Graduate Texts in Mathematics*. Springer-Verlag, London.
- Boost C++ Libraries (2020). Planar Face Traversal. Version 1.73.0. https://www.boost.org/doc/libs/1_51_0/libs/graph/doc/planar_face_traversal.html.
- Cheah, F. and Corneil, D. G. (1990). The complexity of regular subgraph recognition. *Discrete Applied Mathematics*, 27(1–2):59–68.
- Coelho, V. S., Foulds, L. R., Longo, H. J., and Martins, W. S. (2020). Parallelised heuristics for the maximum-weight planar subgraph problem. Unpublished paper.
- Easley, D. and Kleinberg, J. (2010). *Networks, Crowds, and Markets: Reasoning About a Highly Connected World*. Cambridge University Press, Cambridge, UK.
- Foulds, L. R. and Robinson, D. F. (1979). Construction properties of combinatorial deltahedra. *Discrete Applied Mathematics*, 1(1):75–87.
- Jünger, M. and Mutzel, P. (1993). Solving the maximum weight planar subgraph problem by branch and cut. In *Proc. third conference of integer programming and combinatorial optimization (IPCO 3)*, pages 479–492.
- Kreweras, G. (1972). Sur les partitions non croisees d’un cycle. *Discrete Mathematics*, 1(4):333–350.
- Lengauer, T. (2012). *Combinatorial algorithms for integrated circuit layout*. Springer Science & Business Media.
- Massara, G. P., Di Matteo, T., and Aste, T. (2017). Network filtering for big data: Triangulated maximally filtered graph. *Journal of Complex Networks*, 5(2):1–18.
- Ning, Q. (1987). On a conjecture of Foulds and Robinson about deltahedra. *Discrete Applied Mathematics*, 18(3):305–308.
- Pesch, E., Glover, F., Bartsch, T., Salewski, F., and Osman, I. (1999). Efficient facility layout planning in a maximally planar graph model. *International Journal of Production Research*, 37(2):263–283.
- Song, W.-M., Di Matteo, T., and Aste, T. (2012). Hierarchical information clustering by means of topologically embedded graphs. *PLOS ONE*, 7(3):1–14.
- Tumminello, M., Aste, T., Di Matteo, T., and Mantegna, R. N. (2005). A tool for filtering information in complex systems. *Proceedings of the National Academy of Sciences*, 102(30):10421–10426.