

A GRASP heuristic for the maximum-weight planar subgraph problem

Rafael Castro, Wellington Martins, Les Foulds, Humberto Longo

Instituto de Informática, Universidade Federal de Goiás

Alameda Palmeiras, Quadra D, Campus Samambaia, CEP 74690-900, Goiânia, Goiás, Brazil

rcastro@discente.ufg.br, {wmartins, lesfoulds, longo}@ufg.br

ABSTRACT

Algorithms for the maximum-weight planar subgraph problem (MWSP) are relevant in a wide variety of applications, ranging from the analysis of financial data, facility layout, integrated circuit design, systems biology, social systems, among others. The MWSP is an NP-hard problem and it is primarily addressed using heuristic algorithms, commonly utilising construction or improvement approaches. In this article, we present a customised GRASP (Greedy Randomized Adaptive Search Procedure) meta-heuristic designed to encompass both of these approaches. During the construction phase, multiple feasible solutions are generated in parallel, whereas in the improvement phase, the most optimal feasible solution obtained is subjected to further enhancement through an equivalent dual 3-regular graph. Experimental validation demonstrates that the proposed algorithm outperforms established heuristic methods, achieving superior solution quality with improved running time performance.

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

MH – Meta-heuristics, TAG – Graph Theory and Algorithms.

1. Introduction

The analysis of various systems of discrete assets is often facilitated by representing such a system as a graph G with its vertices representing system assets (such as stocks, facility activities, electrical junctions, proteins or social groups) and its edges representing the relationships between the assets [Massara et al., 2017]. The edges are commonly assigned mostly nonnegative weights that represent the fact that most of the asset correlations are positive. When the system has a significant number of assets any drawing of its associated graph, being complete, is difficult to interpret. In this case, the best balance between the retention of asset relationship information and the clarity of the graph drawing involves choosing all of the vertices (assets) and only as many edges (asset relationships) as possible such that the resulting subgraph can be drawn in the plane with edges intersecting only at their endpoints. As will be seen among the graph-related definitions given in the next section, the chosen subgraph is termed *planar*. The (planar) subgraph chosen to be drawn is one that has the maximum total edge weight among all planar subgraphs. This involves solving a particular combinatorial optimisation problem, the Maximum-weight Planar Subgraph Problem (MWSP). The MWSP, which is well known to be $\mathcal{NP}$ -hard [Jünger and Mutzel, 1993], is the subject of the present paper and is formally defined in the next section.

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].

As the MWSP is an $\mathcal{NP}$ -hard problem most of the solution approaches to it are heuristic algorithms. There are two main themes for such approaches: construction methods and improvement methods. Construction methods are commonly used when no feasible solution is available and a final solution is built up from an initial subgraph of G having few vertices, usually a K_4 , the complete graph with four vertices, which is termed a *seed*. Vertices (and their incident edges) are added sequentially to the initial subgraph, usually in a greedy fashion according to the additional edge weight induced, to achieve a final solution. Conversely, improvement methods are used to enhance a given feasible solution, usually by edge exchanges. The simple operations involving vertices and edges, used in both approaches, are commonly referred to as *local* or *topological moves*. These concepts will be made more clear in the following sections.

The objective of the present paper is to present a customised GRASP meta-heuristic for the MWSP. The main parts of this new GRASP are based on recently-reported local search construction and improvement heuristic algorithms for the MWSP. A redesigned version of the construction algorithm by Silva et al. [2022], exploring fine grained parallelism, does the work of generating feasible solutions (the GRASP greedy random adaptive phase). A customisation of the improvement heuristic algorithm by Kataki et al. [2020] does the job of finding minimal local solutions (the GRASP local search phase). To the best of our knowledge, this work is the first that aims to use GRASP for solving MWSP instances. The closest approach we could find was the use of GRASP to solve the problem of finding the maximum planar subgraph of an unweighted graph (graph planarisation problem), by Resende and Ribeiro [1997].

The remainder of this work is organised as follows. Preliminary definitions, notation used throughout the text and some instances of topological moves are presented in Section 2. The proposed method is in Section 3, with a detailed explanation of the constructive phase, and how the method functions as a whole. The present paper is rounded out in Section 4 with a conclusion that summarises evidence that the proposed algorithm outperforms the earlier algorithms.

2. Preliminaries

Some of the graph-theoretic concepts to be used are now defined and discussed. Further details are available, for instance, in the books [Foulds, 1992; Bondy and Murty, 2010]. Let $G = (V(G), E(G), w)$ be a simple, undirected, edge-weighted graph of finite order n , with vertex set $V(G)$, edge set $E(G)$ and edge weight function $w : E \rightarrow \mathbb{R}^+ \cup \{0\}$. G is termed *k-regular* if each of its vertices has the same degree $k \in \mathbb{N}$. If G is a 3-regular graph, it is termed *cubic graph*. An internal edge of a cycle C of G is termed a *chord* of C . Graph G is termed *planar* if it can be embedded in the plane in such a way that its edges intersect only at their endpoints. Of the many existing characterisations of graph planarity, the one most related to theme of the present article was proposed by Whitney [1932]. G is termed *maximal planar* if it is planar and adding any missing edge to it renders it no longer planar. If G is planar, a *plane graph* of G is an actual drawing of G in the plane such that the edges of G intersect only at their endpoints. The regions of a plane graph are termed *faces*. If the graph is maximal planar its faces are here termed *triangles*. G is termed *complete* if every pair of its vertices are directly connected by an edge. A subgraph of G is termed *spanning* if it contains all the vertices of G .

The MWSP involves searching G for a planar, spanning subgraph $G' = (V(G), E(G'), F(G'))$, 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 . $F(G')$ denotes the set of faces of G' , but will sometimes be omitted from the definition of G' for the sake of simplicity. An edge of $E(G')$ with endpoints $x, y \in V(G)$ is denoted by xy and a face of $F(G')$ with vertices $x, y, z \in V(G)$ is denoted by xyz . An 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 condition of planarity ensures easy visualisation of any feasible solution G' to $MWSP(G, w)$ to be drawn in the plane, with the edges in $E(G')$ intersecting only at their endpoints. Note that G is not necessarily complete. However, if G is complete, since the edge weights of G are all assumed to be nonnegative, there will always be an optimal solution of the MWSP corresponding to G that is a maximal planar subgraph of G .

2.1. Some topological moves

This section presents some fundamental techniques that allow the construction of planar subgraphs vertex-by-vertex or the improvement of an existing one through edge replacements. These simple operations are commonly referred to as *local* or *topological moves* and have the common characteristic of invariably transforming a given maximal planar graph (MPG) into a new one. Applying an arbitrary number of these moves to a given MPG maintains maximal planarity in each of the generated graphs. In the following descriptions of topological moves, let $G = (V(G), E(G), w)$ be a simple weighted graph, $G' = (V(G'), E(G'), F(G'))$ a maximal planar subgraph of G and $\mathcal{S} = V(G) \setminus V(G')$ the subset of vertices of G not yet inserted into G' .

The *Face Dimpling* (FD) construction move $m_{FD}(G', u, abc)$, with $u \in \mathcal{S}$ and $abc \in F(G')$, inserts u within the face abc and connects u to the three vertices of the face [Foulds and Robinson, 1978, 1979]. This move yields a new maximal planar subgraph $G'' = (V(G''), E(G''), F(G''))$, with $V(G'') = V(G') \cup \{u\}$, $E(G'') = E(G') \cup \{ua, uc, ub\}$, $F(G'') = F(G') \setminus \{abc\} \cup \{uab, uac, ubc\}$, as illustrated in Figures 1(a) and 1(b). The value of G'' is $s(G'') = s(G') + \gamma(m_{FD}(G', u, abc))$, where:

$$\gamma(m_{FD}(G', u, abc)) = w(ua) + w(ub) + w(uc), \quad ua, ub, uc \in E(G). \quad (1)$$

Given a vertex $u \in \mathcal{S}$, the set of valid FD moves using u still available to be applied to G' is

$$FD(G', u) = \{m_{FD}(G', u, abc) \mid abc \in F(G')\}, \quad u \in \mathcal{S}. \quad (2)$$

Thus, for each vertex $u \in \mathcal{S}$, the FD move that maximises the value $s(G'')$ of the obtained graph G'' is defined as:

$$max_{FD}(G', u) = \arg \max_{m \in FD(G', u)} \gamma(m), \quad u \in \mathcal{S}. \quad (3)$$

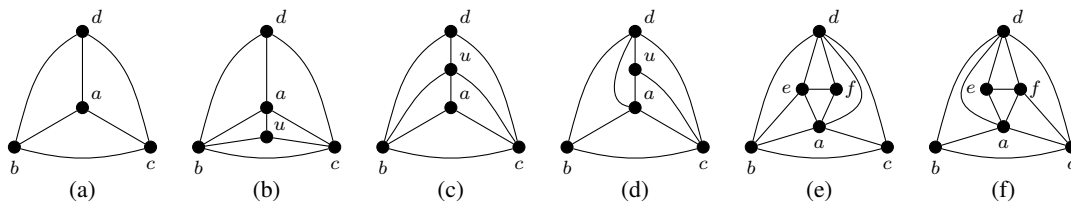


Figure 1: The face dimpling ((a) $\rightarrow$ (b)), edge dimpling ((a) $\rightarrow$ (c)), edge flip ((c) $\rightarrow$ (d)) and edge super flip ((e) $\rightarrow$ (f)) topological moves.

The *Edge Dimpling* (ED) move replaces an existing edge by a new vertex and connects it to the other four vertices that were already members of the two faces that shared the removed edge [Alexander, 1930]. A move $m_{ED}(G', u, ad)$ firstly removes the edge $ad \in E(G')$, which is common to faces abd and acd say, of G' , inducing the face $abcd$. Next, the new vertex $u \in \mathcal{S}$ is

inserted within the face $abcd$. Finally, it connects u to the vertices a, b, c, d . Therefore, the original faces abd and acd no longer exist and new faces bud , aub , auc and cud come into being. The move is illustrated in Figures 1(a) and 1(c). The gain $\gamma(m_{ED}(G', u, ad))$ obtained by applying an ED move, the set $ED(G', u)$ of valid ED moves and the move $max_{ED}(G', u)$ that maximises the value $s(G'')$ is defined along the lines of (1), (2) and (3), respectively.

An *Edge Flip* (EF) improvement move $m_{EF}(G', ab, cd)$ consists of replacing a chord ab of a 4-cycle C in the MPG G' by the other chord cd of C . The move can be performed only if the resulting graph is simple and results in a distinct MPG. Figures 1(c) and 1(d) illustrate the edge flip move. The first figure depicts an original MPG G' , and the second figure shows the MPG G'' obtained after exchanging the edge bu for the edge ad . As $ad \notin E(G')$, the edge bu can be replaced by ad to transform G into G' . The gain $\gamma(m_{EF}(G', ab, cd))$ obtained by applying an ED move, the set $EF(G')$ of valid EF moves and the move max_{EF} that maximises the value $s(G'')$ is also defined along the lines of (1), (2) and (3), respectively.

A further improvement move is the *Edge Super Flip* (SF). Suppose the MPG G' contains a 4-cycle $C = \langle u, v, x, y \rangle$ with vertices $u, v, x, y \in V(G')$, a chord $vy \in E(G')$ and faces $uvy, vxy \in F(G')$. If the chord ux is also contained in G' , then it is not possible to apply the edge flip move to vy . The *edge super flip* move is designed to cover one case of this situation. That is, by replacing the chord of vy by a particular new edge (not yet in $E(G')$), thus creating a distinct MPG. Figures 1(e) and 1(f) illustrate the SF move. The first figure depicts an original MPG G' , and the second one the MPG G'' , obtained by a particular SF move. Firstly, following Figure 1(e), edge be is removed, and abe and bde that were 3-cycles in G' disappear and are no longer faces of $F(G)$, making $C_1 = \langle a, b, d, e \rangle$ a quadrilateral face of the resulting graph. Secondly, following Figure 1(f), ad is “flipped” into C_1 to become its new chord. The removal of ad from the 4-cycle $C_2 = \langle a, c, d, f \rangle$ leaves that cycle without a chord. The “potential new chord” of C_2 , the edge cf , cannot already be present in G and this edge is added to the graph. As $cf \notin E(G')$, the SF move is valid and be is replaced by cf to transform G' into G'' , which is also an MPG. Once again, the profit $\gamma(m_{SF}(G', ab, cd))$ obtained by applying an SF move, the set $SF(G')$ of valid SF moves and the move max_{SF} that maximises the value $s(G'')$ is also defined along the lines of (1), (2) and (3), respectively.

[Pesch et al., 1999] interpreted some topological improvement moves as the act of rotating a particular substructure of vertices and edges within a graph. Following this idea, [Kataki et al., 2020] extended the concept and proposed some new improvement moves. These basic operations can be interpreted as the rotation of a chosen subgraph within a given cycle. A subset of these new moves extract a specific subgraph that is enclosed within a 6-cycle and then inserts it again in the cycle in a different way, yielding a distinct maximal planar graph.

As an example of the concepts discussed in the previous paragraph, let t be any combination of three edges as chords enclosed within a vertex-labelled 6-cycle $C_6 = \langle a, d, b, e, c, f \rangle$ of G . Although there are 84 possible combinations of three edges as chords enclosed within a vertex-labelled 6-cycle, only the 14 cases shown in Figure 2 define distinct enclosed planar subgraphs.

Thus, with the new rotational topological moves proposed by [Kataki et al., 2020], a new maximal planar graph can be obtained by taking the structure formed by t and rotating it clockwise or counterclockwise within C_6 . Depending on the initial chord structure assumed, the new rotational improvement move is termed either *face*-, *3-path*- or *claw*-rotation (cases 2(a) $\leftrightarrow$ 2(b), 2(c) $\leftrightarrow$ 2(d) $\leftrightarrow$ 2(e) or 2(f) $\leftrightarrow$ 2(g) $\leftrightarrow$ 2(h), and 2(i) $\leftrightarrow \dots \leftrightarrow$ 2(n), respectively).

Furthermore, starting with any of the cases in Figure 2 and removing its combination t , of three chords from C_6 and inserting some other planar combination t' say, of three chords, always leads to one of the other cases in Figure 2. Thus, the search for a subgraph that increases the total

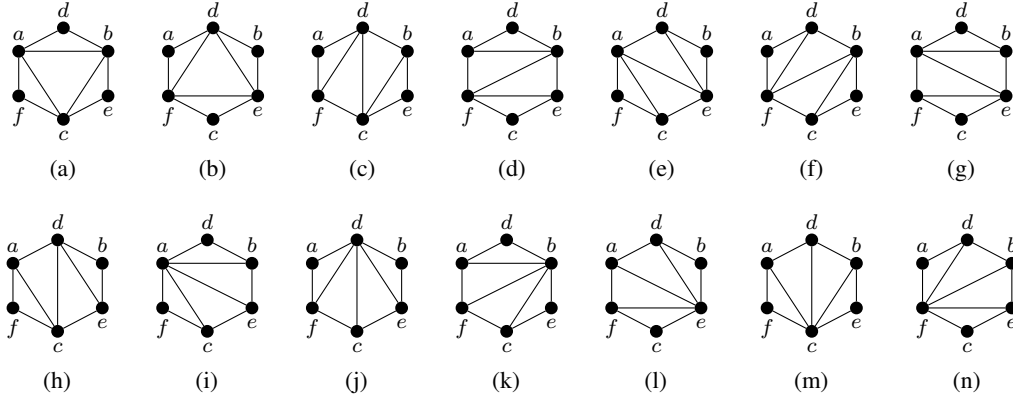


Figure 2: The planar triangulations of the vertex-labelled 6-cycle.

weight of a solution for an instance $MWPS(P, w)$ can be limited to only those that can be included in 6 cycles. The increase in the value $s(G')$ of a new maximal planar subgraph G' obtained by any such operation increases, relatively to $s(G)$, is equal to the difference between the sums of the weights of the new edges inserted and edges removed. However, managing all 6 cycles in G and its embedded subgraphs can become an arbitrarily difficult task as the cardinality of $V(G)$ grows. Moreover, transforming a structure formed by three chords embedded in a particular 6-cycle into some other valid structure embedded in the same 6-cycle, can also entail a large number of simple topological moves before an improved feasible solution is identified.

Fortunately, recognising 6-cycles and replacing structures included in them become simple tasks, of $\mathcal{O}(1)$ time complexity, when these operations are performed on the dual graph of G . This possibility was explored by [Kataki et al., 2020] who developed a topological move termed the *Star Move*, which is described in Subsection 2.2 below. This move in the dual graph of G is part of a general solution procedure for $MWPS(P, w)$ that switches back and forth between G and its dual H . Such switches between a primal problem and its dual problem are common in optimisation, such as in the primal-dual algorithm for linear programming.

2.2. The star move on a cubic graph

Let $H = (V(H), E(H))$ be a connected cubic planar graph. [Kataki et al., 2020] defined a *Star Gadget* to be a subgraph of H that is induced by a vertex $i \in V(H)$ and three of its neighbouring vertices $j, k, \ell \in V(H)$ (i.e. a star graph, with hub i of degree 3). The authors denoted such a subgraph as $i \star jk\ell$. The *Star Move* was defined as the operation of replacing a star gadget $i \star jk\ell$ in H by another star gadget $i' \star j'k'\ell'$ (eventually $j' = j, k' = k$ or $\ell' = \ell$). The graph generated from H by a star move is called *valid* if it is also a connected cubic planar graph and *invalid* if it does not meet such requirements. A star move is said to be valid if it generates a valid graph.

When H is the dual graph of a connected maximal planar graph $G = (V(G), E(G))$, then H is connected, cubic and planar [Ore, 1967]. A star move operation on graph H is equivalent to the replacement in G of at most four triangular faces t_1, t_2, t_3 and t_4 say, included within a 6-cycle C_6 say, by other (triangular) faces included within the same cycle C_6 . In general, p edges of a triangle t in G are affected by a star move in the dual graph H , where $1 \leq p \leq 3$. The new planar subgraphs that can be obtained by this process of replacing triangles within C_6 are exactly the same as those that can be achieved by replacing one to three chords of C_6 , as explained in the previous section. Vertices in H corresponding to triangles outside the C_6 in G , as well as these faces, are not affected by the star move operation. However, there can be 2, 3 or 4 faces included in C_6 that are affected.

The notions discussed in the previous paragraph are illustrated in Figure 3, where vertices a, b, c, d, e and f of fragments of primal graphs G are coloured black, and their duals H , are coloured red and blue. Consider the star gadget $i \star j k \ell$ in H and the triangles $t_1 = \Delta_{abc}$, $t_2 = \Delta_{abd}$, $t_3 = \Delta_{acf}$ and $t_4 = \Delta_{bce}$ included within the 6-cycle $\langle a, d, b, e, c, f, a \rangle$ in G , as illustrated in Figure 3(a). After a star move with the star gadget $i \star j k \ell$ is applied to graph H , the result in graph G is the replacement of two to four of the faces t_1 to t_4 by new faces, depending on the choice of the new star gadget to be inserted in H , as described below:

Figure 3(b): Replacement of faces t_1 and t_3 with the new faces $t'_1 = \Delta_{abf}$ and $t'_3 = \Delta_{bcf}$ (new star gadget $i' \star j' k' \ell'_1$);

Figure 3(c): Replacement of faces t_1 , t_2 and t_3 with the new faces $t'_1 = \Delta_{adf}$, $t'_2 = \Delta_{bcd}$ and $t'_3 = \Delta_{cdf}$ (new star gadget $i' \star j' k' \ell'_2$);

Figure 3(d): Replacement of faces t_1 to t_4 with the new faces $t'_1 = \Delta_{adf}$, $t'_2 = \Delta_{bde}$, $t'_3 = \Delta_{cef}$ and $t'_4 = \Delta_{def}$ (new star gadget $i' \star j' k' \ell'_3$).

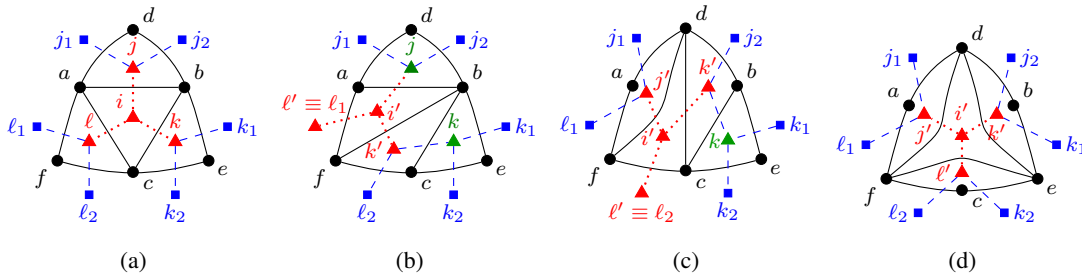


Figure 3: Illustrations of the equivalence between triangle replacement in a primal maximal planar graph and the application of a star move in its dual.

3. The DimFlipCubic Grasp

GRASP (Greedy Randomised Adaptive Search Procedure), proposed in [Feo and Resende, 1989, 1995], is a multi-start meta-heuristic that has been successfully applied to a wide range of combinatorial optimisation problems. Its behaviour can be summarised in two main blocks: (i) generating feasible solutions (a greedy randomised adaptive method) and (ii) finding local minima (a local search phase).

The first phase has three main parts: (i) a greedy algorithm, which always makes locally optimal choices in the expectation that will lead to a globally optimal solution; (ii) an adaptive function that guides the greedy algorithm on selecting the element for the solution been built (adaptive in the sense to be dynamic, in contrast to the static basic approach that assigns fixed scores to the elements before starting a solution construction); and (iii) a probabilistic selection scheme, which does the choices of the elements to be added to a partial solution among the options available in a Restricted Candidate List (RCL). In general, the most commonly used criteria for constructing this list are: based on cardinality (CB), which takes into account a fixed number of elements based on the order of local increment in the solution; or value-based (VB), which brings each element closer to a certain percentage of the one with the highest local optimal increase.

The customisation of the GRASP method proposed here for solving the MWSP, called *DimFlipCubic Grasp*, has two main components: (i) an inner constructive method, termed *DimFlip*, which aims to build feasible solutions; and (ii) a local search heuristic, strongly based on the *Cubic Heuristic* of Kataki et al. [2020], applied as the local search phase of the method with the aim of finding globally optimal solutions. Therefore, after building a feasible solution for the MWSP, the next step is to apply a local search to it.

The first phase of the proposed *DimFlipCubic Grasp* consists of building a feasible solution for the MWSP instance, which is done by the *DimFlip* heuristic. This heuristic returns random solutions, as it chooses the construction move based on an RCL list. It is also adaptive, as this list is updated at every step based on current moves. Its main aspect is to be very fast in building a new maximal solution when implemented to exploit fine-grain parallelism, as was done in this work.

Section 3.1 describes the *DimFlip* method and Section 3.2 highlights some details of the local search phase performed by *Cubic Heuristic* and provides a high level pseudo-code of the proposed customised GRASP.

3.1. Generating feasible solutions – the *DimFlip* method

The *DimFlip* method is essentially a constructive method of maximal planar solutions, but it is associated with an improvement routine called between its constructive steps. This routine tries to change the current solution, without inserting a new vertex, in order to increase its value. This method uses the *face* and *edge dimpling* moves in its construction stages and also the *edge* and *super edge flip* move in its improvement stages.

DimFlip heuristic receives as input a graph $G = (V(G), E(G))$ of an MWSP instance and a set $\mathcal{C}$ of *seeds* (subgraphs K_4 of G). The solution building process starts from a *seed* in $\mathcal{C}$ and, successively, apply *face* and *edge dimpling* moves to grow the current partial solution. In order to improve the performance of the method it is used a cache memory, applying a similar approach as done at Massara et al. [2017], with the difference that stores, for each vertex $v \in V(G)$ still not inserted in the current solution, not only the best face, but also the edge dimpling movement associated with v . At each iteration, those moves that improve the current solution by a certain minimum percentage of the current best move are also taken into account and inserted in an RCL list. The next move to be applied is chosen, uniformly and randomly, from this list. All this process is done in parallel for each seed in $\mathcal{C}$.

A pseudo-code of the whole method is provided in Algorithm 1, which in addition to G and $\mathcal{C}$ also receives as input parameters α (used in the *value-based* RCL list construction mechanism), π (used to randomly decide whether the improving super edge flipping local search method shall be applied to the built solution) and q (the quantity of iterations to be performed of the local search just mentioned). At step 1 the set $\mathcal{R}$, of subsets of edges used in each solution built in parallel, is initialised as empty. In steps 3 through 6 are performed the pre-processing tasks of selecting a seed and constructing the auxiliary sets $\mathcal{S}$ of available vertices to be “dimpled”, $\mathcal{E}$ of edges and $\mathcal{F}$ of faces. In steps 8 through 13, a solution is constructed from the selected seed. Firstly, in step 8, the RCL list is built, then, in step 9, the move mv to be executed in step 10 is chosen, uniformly and randomly, from the RCL list. Next, in step 11, the sets $\mathcal{S}$, $\mathcal{E}$ and $\mathcal{F}$ are updated accordingly. Each of these three last steps has time complexity of $\mathcal{O}(1)$. In steps 12 and 13, with probability π , a local search method is applied to the newly constructed solution. A pseudo-code of this improvement method is provided in Algorithm 2. In the 14 step, the edges of the built solution are stored in $\mathcal{R}$. Finally, in step 15, the best solution, that is, with the largest sum of edge weights, is retrieved from $\mathcal{R}$ and returned in step 16.

The process of building the RCL list used in Algorithm 1 (steps 8 and 9) is based on some definitions provided in Section 2.1. Given that $\gamma(m)$ is the gain from applying a move m , let $m_{\max}(G')$ be the FD or the ED move, using any vertex $u \in \mathcal{S}$, that maximises the value $s(G'')$ of the resulting graph G'' . Then,

$$\gamma(m_{\max}(G')) \geq \max_{u \in \mathcal{S}} \{ \gamma(\max_{FD}(G', u)), \gamma(\max_{ED}(G', u)) \}. \quad (4)$$

Therefore, the RCL list associated to the current maximal planar subgraph G' is defined as:

$$RCL(G') = \bigcup_{u \in \mathcal{S}} \{m \in \{max_{FD}(G', u), max_{ED}(G', u)\} \mid \gamma(m) \geq (1.0 - \alpha) \cdot \gamma(m_{\max}(G'))\}, \quad (5)$$

where α is the parameter introduced previously for the *VB* mechanism. The time complexity of building the RCL list based on (5) is $\mathcal{O}(|\mathcal{S}| \cdot (|\mathcal{F}| + |\mathcal{E}|))$, where $\mathcal{F}$ and $\mathcal{E}$ are the sets of faces and edges, respectively, in the current solution.

The local improvement method used simultaneously with the construction of the solutions (Algorithm 2) iterates at most q times trying to find moves, among those available in the RCL list, that improve the value of the current partial solution. Step 2 searches for such movements, with time complexity $\mathcal{O}(|\mathcal{E}|)$. All other steps of the Algorithm 2 have $\mathcal{O}(1)$ time complexity. Therefore, the time complexity of the entire algorithm is $\mathcal{O}(q \cdot |\mathcal{E}|)$.

Regarding Algorithm 1, since the cardinalities of the sets $|\mathcal{S}|$, $|\mathcal{E}|$ and $|\mathcal{F}|$ are bounded by n , i.e., $|\mathcal{S}|, |\mathcal{E}|, |\mathcal{F}| \in \mathcal{O}(n)$, the time complexity of steps 3 to 13 is $\mathcal{O}(n^3 + Q \cdot q \cdot n)$, where Q is the total number of calls to routine *SuperEdgeFlip()* (step 13 of Algorithm 2) when building a single-seed solution. Since $Q \leq n$, in the worst case the time complexity of all these steps together is $\mathcal{O}(n^3 + q \cdot n^2)$. These construction steps are executed in parallel for each seed in $\mathcal{C}$, and the heaviest solution obtained is returned. Thus, the overall method has time complexity $\mathcal{O}(|\mathcal{C}| \cdot (n^3 + q \cdot n^2)/p)$, where p is the total number of processing elements.

3.2. The local search phase and the overall method

Kataki et al. [2020] presented a heuristic local search algorithm for the MWSPSP based on a polynomial transformation of the MWSPSP to the weighted version of the Connected Spanning Subgraph Problem (CSSP) [Cheah and Corneil, 1990]. They proposed first transforming a feasible solution of an instance of MWSPSP into an equivalent dual 3-regular graph, then applying a custom local search to this graph, and finally transforming the final 3-regular graph back into a new feasible solution for the original instance of MWSPSP. A detailed description of this process and of the heuristic itself, denoted *Cubic Heuristic* by those author, are also available at [Kataki, 2021].

The custom Grasp algorithm listed in the Algorithm 3 pseudo-code, denoted *DimFlipCubic Grasp* receives as input parameters a graph G , the set $\mathcal{C}$ of seeds in G , the quantity g of *grasp* iterations (steps 2 for 5 of Algorithm 2), the amount r of *cubic* iterations together with the amount k of neighbours to be expanded in each iteration of the *Cubic Heuristic*, and the parameters α, π and q explained in the previous section. This custom Grasp uses the *Cubic Heuristic* in the local search phase with the aim of finding globally optimal solutions. In each iteration of the algorithm, a maximal planar subgraph solution is constructed using the *DimFlip* heuristic (step 3). In the next step, the *Cubic Heuristic* is applied to the solution constructed by the *DimFlip* heuristic. If this solution outperforms the current best solution, it is stored as *bestSolution* (step 5). Steps 3 through 5 have time complexity $\mathcal{O}(|\mathcal{C}| \cdot (n^3 + q \cdot n^2)/p + n^2 \cdot r)$. Thus, the final time complexity of the entire method is $\mathcal{O}(g \cdot (|\mathcal{C}| \cdot (n^3 + q \cdot n^2)/p + n^2 \cdot r))$.

4. Computational experiments

The computational experiments carried out with the heuristics described in Section 3 were tested using an Intel(R) Xeon(R) CPU E5-2620 @ 2.00GHz with 32GB of RAM and a NVIDIA GeForce RTX 3060 GPU with 3.584 CUDA Cores and 12GB of RAM. The developed C++ code was compiled with NVIDIA (R) Cuda compiler driver 12.1, using `-std=c++17`, the `compute_86` architecture, with the code targeting the native architecture and the `O3` optimisation flag set. All experiments were carried out with the computer dedicated only to running the tests with the chosen data instances.

Algorithm 1: DimFlip($G, \mathcal{C}, \alpha, \pi, q$)

Input: Graph $G = (V, E)$, set $\mathcal{C}$ of seeds in G , parameters $\alpha \in \mathbb{R}$, $\pi \in \mathbb{R}$ and $q \in \mathbb{N}$.

Output: Maximal planar subgraph G' , filtered from G .

```

1  $\mathcal{R} \leftarrow \emptyset$ 
2 for  $k \leftarrow 1$  to  $|\mathcal{C}|$  do in parallel
3    $\mathcal{C}_k \leftarrow \{v_1^k, v_2^k, v_3^k, v_4^k\} \in \mathcal{C}$ ;
4    $\mathcal{S} \leftarrow V(G) - \mathcal{C}_k$ ;
5    $\mathcal{E} \leftarrow \{\{v_1^k, v_2^k\}, \{v_1^k, v_3^k\}, \{v_1^k, v_4^k\}, \{v_2^k, v_3^k\}, \{v_2^k, v_4^k\}, \{v_3^k, v_4^k\}\}$ ;
6    $\mathcal{F} \leftarrow \{\{v_1^k, v_2^k, v_3^k\}, \{v_1^k, v_2^k, v_4^k\}, \{v_1^k, v_3^k, v_4^k\}, \{v_2^k, v_3^k, v_4^k\}\}$ ;
7   while ( $\mathcal{S} \neq \emptyset$ ) do
8      $RCL(\mathcal{S}, \mathcal{E}, \mathcal{F}) \leftarrow \text{ConstructRCL}(\alpha)$ ;
9      $m \leftarrow \text{SelectMove}(RCL(\mathcal{S}, \mathcal{E}, \mathcal{F}))$ ;
10    Apply move  $m$ ;
11    Update sets  $\mathcal{S}$ ,  $\mathcal{E}$  and  $\mathcal{F}$ ;
12    if ( $\text{ApplySuperEdge}(\pi)$ ) then
13      Call  $\text{SuperEdgeFlip}(q)$ ;
14   $\mathcal{R} \leftarrow \mathcal{R} \cup \{\mathcal{E}\}$ 

15  $\mathcal{E}' \leftarrow \arg \max_{\mathcal{E}'' \in \mathcal{R}} s(V(G), \mathcal{E}'')$ ;
16 return  $G' = (V(G), \mathcal{E}')$ 

```

Algorithm 2: SuperEdgeFlip(q)

Input: Quantity $q \in \mathbb{N}$ of iterations.

```

1 for  $k \leftarrow 1$  to  $q$  do
2    $m_{best} \leftarrow \text{BestEdgeFlip}()$ ;
3   if ( $GA(m_{best}) > 0$ ) then
4     apply  $m_{best}$  move;
5   else
6     stop.

```

Algorithm 3: DimFlipCubic($G, \mathcal{C}, \alpha, \pi, q, g, r, k$)

Input: Graph $G = (V, E)$, set $\mathcal{C}$ of seeds in G , parameters $g, r, k, q \in \mathbb{N}$ and $\alpha, \pi \in \mathbb{R}$.

```

1  $bestSol \leftarrow \emptyset$ ;
2 for  $k \leftarrow 1$  to  $g$  do
3    $mwpsspSol \leftarrow \text{DimFlip}(G, \mathcal{C}, \alpha, \pi, q)$ ;
4    $\text{CubicHeuristic}(mwpsspSol, k, r)$ ;
5   Update  $bestSol$ ;
6 return  $bestSol$ .

```

The computational experiments used synthetically generated instances ($MWSPS(K_n, w)$, $n = 10, 25, \dots, 100$), originally proposed by de Sousa Coelho [2018], with weights $w(e)$, $e \in E(K_n)$, randomly generated in the range $[0, 199]$. These instances were generated in a way that are especially difficult to be solved optimally. It was also used a 100-vertex instance by Tumminello et al. [2005], which is based on an empirical study of practical financial data derived from the US stock market, with vertices and edges representing stocks and given correlation coefficients between them, respectively.

It was tested both the *DimFlip* heuristic alone and the *DimFlipCubic Grasp*, i.e Algorithms 1 and 3, respectively. Both heuristics were executed 30 times with each instance, using the distance between the results obtained and the available upper bounds (percentage gap) as a comparison metric. These gaps are presented with a 95% confidence interval. Processing times, in seconds, include operations carried out on the CPU and on the GPU kernel. The input parameter $\mathcal{C}$ (seed set) of the *DimFlip* heuristic (step 3 of Algorithm 3) was fixed as the set of all possible tetrahedra in the input graph G . For the test on the *DimFlip* alone, the set of seeds was generated with the same filtering method used at *Restricted Seeds* heuristic by Silva et al. [2023], using y and z as 95.

Table 1 contains some results obtained with both heuristics described in Section 3, using eighteen of the instances generated by de Sousa Coelho [2018] and a 100-vertex instance of Tumminello et al. [2005] (the highlighted row in the table). To evaluate the performance of the proposed heuristics, their results are compared with the known best upper bounds for each instance (values from the second column of Table 1, found with the exact algorithm proposed by Kataki [2021]) The values in bold have been proven to be optimal and the non-optimal results were obtained by rounding down the values of the solutions calculated by using an exact algorithm for the MWSPS and some bounding techniques proposed by Ahmadi-Javid et al. [2015]. The average gaps presented in the last row of the table are only related to the results obtained for the instances by de Sousa Coelho [2018] ($n = 15, 20, \dots, 100$).

Table 1: Results obtained with *DimFlip* and *DimFlipCubic Grasp* heuristics with instances of de Sousa Coelho [2018] and Tumminello et al. [2005].

Instance	Upper Bound	DimFlip			DimFlipCubic Grasp			
		Kernel (s)	Value (μ)	gap range (%)	Time (s)	Kernel (s)	Value (μ)	gap range (%)
15	5772	0.012	5765.00	[0.01,0.23]	929.36	0.149	5772.0	[0.00, 0.00]
20	8434	0.024	8293.47	[1.35,1.98]	1708.33	0.311	8434.0	[0.00, 0.00]
25	11594	0.036	11373.90	[1.74,2.06]	2539.05	1.010	11583.9	[0.05, 0.12]
30	13770	0.054	13526.10	[1.65,1.89]	3188.22	2.869	13761.4	[0.03, 0.09]
35	17177	0.075	16868.60	[1.56,2.03]	3802.38	8.855	17140.3	[0.17, 0.26]
40	19862	0.112	19285.30	[2.73,3.07]	5165.42	20.600	19718.8	[0.58, 0.87]
45	22384	0.154	21550.13	[3.68,3.77]	5940.59	50.142	21878.9	[2.13, 2.38]
50	25763	0.191	24930.77	[3.16,3.30]	7653.45	86.769	25263.9	[1.88, 1.99]
55	28536	0.256	27499.67	[3.53,3.73]	8520.99	200.314	27912.5	[2.10, 2.27]
60	31272	0.319	30142.67	[3.50,3.73]	10308.05	349.863	30577.6	[2.16, 2.28]
65	34226	0.423	32895.67	[3.79,3.98]	12571.60	714.127	33409.1	[2.32, 2.46]
70	37493	0.465	36145.53	[3.51,3.68]	13957.40	400.668	36604.8	[2.30, 2.44]
75	40130	0.629	38617.80	[3.68,3.86]	16106.51	1959.566	39112.4	[2.46, 2.61]
80	43058	0.737	41356.37	[3.68,3.78]	16911.54	2898.401	41891.2	[2.64, 2.78]
85	45701	0.904	43965.40	[3.72,3.87]	20829.83	4740.410	44495.0	[2.58, 2.70]
90	48828	1.003	46980.33	[3.69,3.88]	23531.87	5903.532	47438.4	[2.80, 2.89]
95	51626	1.331	49669.93	[3.72,3.85]	30179.26	10956.065	50204.7	[2.70, 2.81]
100	54875	1.516	52900.47	[3.50,3.70]	36145.57	14933.693	53457.1	[2.52, 2.64]
Tumminello	121835	1.490	116657.92	[4.22,4.27]	33615.09	14844.75	116985.7	[3.97, 3.99]
Average gaps (%):				[2.91, 3.15]	[1.63, 1.76]			

From the third through the fifth columns are summarised the results obtained with the *DimFlip* heuristic: column labelled “Kernel(s)” contains the average time, in seconds, of the thirty running times of the heuristic with each instance, column “Value(μ)” shows the average values of the thirty obtained solutions and the fifth column contains the 95% confidence gap interval. The sixth to ninth columns summarise the results obtained with the *DimFlipCubic GRASP*: column labelled “Time(s)” shows the average total time (in seconds) spent solving each instance (both CPU and GPU), column “Kernel(s)” contains the average time spent by the GPU, the next three columns are equivalent to those described for the *Dimflip* heuristic. The heuristics parameters were set as follows: $\alpha = 0.05$, $\pi = \frac{1}{3}$, $q = 1000$, $g = 10$ and $r = 5000$. Comparing these results we have a improvement from other well known heuristics. For the instances by de Sousa Coelho [2018], the TMFG Tumminello et al. [2005] has an average gap of 5.24%, *FD* Coelho et al. [2016] an average gap of 3.34%, the *CH + FED* Kataki et al. [2020] an average gap of 1.89%, the restricted seeds Silva et al. [2023] got a gap interval, with 95% of confidence, equals to [2.98%, 3.16%], and the *RS + CH* Silva et al. [2022] a gap interval equals to [1.73%, 2.13%].

5. Conclusions

In this paper, we presented a customised GRASP meta-heuristic for the maximum-weight planar subgraph problem. Our approach features a novel construction phase that leverages parallelism and a cache-like data structure, along with the use of an equivalent dual 3-regular graph during the improvement phase. This solution was applied to several synthetically created numerical instances and to a 100-vertex instance from the literature. The proposed GRASP meta-heuristic proved capable of finding solutions that, on average, are in the range of 1.63% to a maximum of 1.76% away from the optimal solution. These results are encouraging, particularly given that the GRASP method, being heuristic, does not guarantee an optimal solution. For future work, we plan to further investigate how parameter choices influence the quality of the approximated results.

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.
- Alexander, J. W. (1930). The combinatorial theory of complexes. *Annals of Mathematics*, 31: 292–320.
- Bondy, J.-A. and Murty, U. S. R. (2010). *Graph Theory*, volume 244 of *Graduate Texts in Mathematics*. Springer-Verlag, London.
- Cheah, F. and Corneil, D. G. (1990). The complexity of regular subgraph recognition. *Discrete Applied Mathematics*, 27(1–2):59–68.
- Coelho, V. S., Martins, W. S., Foulds, L. R., Dias, E. S., Castonguay, D., and Longo, H. J. (2016). Uma proposta de solução aproximada para o problema do subgrafo planar de peso máximo. In *XVII Simpósio em Sistemas Computacionais de Alto Desempenho (WSCAD 2016)*, p. 49–60, Porto Alegre, RS, Brasil. SBC.
- de Sousa Coelho, V. (2018). An applied study using parallelism for the maximum weight planar subgraph problem. Master's thesis, Universidade Federal de Goiás, Goiânia, Goiás, Brazil. Portuguese version available at <http://repositorio.bc.ufg.br/tede/handle/tede/8503>.
- Easley, D. and Kleinberg, J. (2010). *Networks, Crowds, and Markets: Reasoning About a Highly Connected World*. Cambridge University Press, Cambridge, UK.
- Feo, T. A. and Resende, M. G. C. (1989). A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, 8(2):67–71.
- Feo, T. A. and Resende, M. G. C. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6(2):109–133.
- Foulds, L. R. (1992). *Graph Theory Applications*. Universitext. Springer-Verlag, New York, 1 edition.
- Foulds, L. R. and Robinson, D. F. (1978). Graph theoretic heuristics for the plant layout problem. *The International Journal of Production Research*, 16(1):27–37.

- 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)*, p. 479–492.
- Kataki, P. A. G. (2021). Novos algoritmos heurístico e exato para o MWSP. Master's thesis, Instituto de Informática, Universidade Federal de Goiás, Goiânia-GO, Brazil.
- Kataki, P. A. G., Cappelle, M. R., Foulds, L. R., and Longo, H. J. (2020). A new algorithm for the Maximum-weight Planar Subgraph Problem. In *Anais do LII Simpósio Brasileiro de Pesquisa Operacional*, p. 1–12, João Pessoa, PB, Brazil.
- 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):161–178.
- Ore, Ø. (1967). *The four-color problem*. Pure and applied mathematics. Academic Press.
- 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.
- Resende, M. G. C. and Ribeiro, C. C. (1997). A GRASP for graph planarization. *Networks*, 29(3):173–189.
- Silva, R., Coelho, V., Martins, W., Longo, H., and Foulds, L. (2023). A constructive parallel heuristic approach for the mwpsp. In *Anais do XXIV Simposio em Sistemas Computacionais de Alto Desempenho*, p. 289–300, Porto Alegre, RS, Brasil. SBC. URL <https://sol.sbc.org.br/index.php/sscad/article/view/26528>.
- Silva, R. C., Kataki, P. A. G., R., F. L., and Longo, H. J. (2022). A constructive and improving heuristic algorithm for the maximum weight planar subgraph problem. In *Proceedings of the LIV Simpósio Brasileiro de Pesquisa Operacional, SBPO 2022*, p. 1–12, Juiz de Fora – MG, Brazil.
- 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.
- Whitney, H. (1932). Non-separable and planar graphs. *Transactions of the American Mathematical Society*, 34(2):339–362.