A Bisection Approach to Subcubic Maximum Induced Matching *

Gordon Hoi[1], Sanjay Jain[2], Ammar Fathin Sabili[2] and Frank Stephan[23]

[1] School of Informatics and IT, Temasek Polytechnic
21 Tampines Ave 1, Singapore 529757, Republic of Singapore
hoickg@gmail.com
[2] School of Computing, National University of Singapore
13 Computing Drive, Singapore 117417, Republic of Singapore
sanjay@comp.nus.edu.sg, ammar@comp.nus.edu.sg, fstephan@comp.nus.edu.sg
[3] Department of Mathematics, National University of Singapore
10 Lower Kent Ridge Road, Singapore 119076, Republic of Singapore

Abstract. In this paper, we present a faster exact algorithm which solves the Maximum Induced Matching problem for subcubic graphs. Here let n be the overall number of vertices and k be the number of those vertices of degree 3 where all neighbours have also at least degree 2. Then the runtime is at most $O(1.2335^k) \cdot Poly(n)$, giving an FPT bound for the time used by the algorithm; the algorithm uses the result of Monien and Preis combined with a bound obtained by applying the measure and conquer technique where the number k replaces n as the measure used; note that $k \leq n$.

Keywords. Branch and Bound, Exponential Time Algorithms, Graph Theory, Measure and Conquer, Fixed Parameter Tractable Problems.

1 Introduction

In this paper we will be dealing with only undirected graphs. Given a graph $G = (V, E)$, where V is its set of vertices and E is its set of edges. We let n denote the number of vertices in the graph G. For $U \subseteq V$, we let $\mathrm{N}(U)$ denote the set of neighbours of U in the graph G, that is $\{e \in V : (\exists e' \in U)[(e, e') \in E]\}$. We let $E(U)$ denote the set of edges in E with both endpoints in U.

For a graph $G = (V, E)$ and a subset $S \subseteq V$, the induced graph $G[S] = (S, E(S))$, has as set of edges all those members $(e, e') \in E$ where e, e' are both in S. S is said to be induced matching if every vertex in S has exactly one neighbour in S. The Maximum Induced Matching problem asks for the largest possible S such that S is an induced matching. This problem has a number of practical applications, such as in VLSI design, network flow problems [9] and risk-free marriages [18]. The Maximum Induced Matching (MIM) problem is much

* G. Hoi (as RF), S. Jain (as Co-PI) and F. Stephan (as PI) were supported in part by the Singapore Ministry of Education Academic Research Fund Tier 2 grant MOE2019-T2-2-121 / R146-000-304-112. Additionally S. Jain was supported by NUS grant E-252-00-0021-01.

harder than the similarly defined Maximum Matching problem where one asks only for a subset $F \subseteq E$ such that F is a matching (i.e., no vertex is an endpoint of two edges in F). While the MIM problem is **NP**-hard, the second problem can be solved in polynomial time [21]. Gupta, Raman and Saurabh [11] provided two subsequent algorithms to solve the MIM problem in time $O(1.6957^n)$ and $O(1.4786^n)$, respectively. Chang, Chen and Hung [2] improved the running time to $O(1.4658^n)$ and subsequently they improved it to $O(1.4321^n)$. Finally, Xiao and Tan [20] provided further improvements, running in time $O(1.4231^n)$ and $O(1.3752^n)$ where the last algorithm uses exponential space in contrast to the former ones which use polynomial space. The value $O(1.4231^n)$ is also the state of the art for the special case of graphs with maximum degree 4 [20, Lemma 11] with polynomial space; however, Xiao and Tan [20] did not comment on the complexity of solving the MIM problem for graphs with maximum degree 3 for polynomial space.

In this paper, we study a special case of the MIM problem that is limited to subcubic graphs, where each vertex has at most three neighbours. Prior algorithms that were used to solve the MIM problem on subcubic graphs were relying on applying the fastest Maximum Independent Set algorithm [19] on the line graph of G^2 ($L(G^2)$) to obtain a running time of $O(1.3139^n)$, using polynomial space.

We present a faster exact algorithm to solve the MIM problem for subcubic graphs in polynomial space and $O(1.2335^n)$ time; this paper supersedes the technical report [12] by solving the MIM problem for subcubic graphs faster than the previous $O(1.2630^n)$ and with a simpler algorithm. To achieve this, we design a branch and bound algorithm in conjunction with the Measure and Conquer technique [4] and the bisection result of Monien and Preis (stated below); though Monien and Preis also provided bounds for graphs of degree 4, the bisection width of that subcase is so large that the approach chosen here cannot be adjusted to higher degrees in a competative way.

If we allow for the use of exponential space to solve this problem, then the general pathwidth bounds of Fomin and Høie [5] imply that the algorithm runs in time $O(r^n)$ for every $r > 3^{1/6}$ and thus the problem can be solved in time $O(1.2010^n)$; Kumar and Kumar [16, Theorems 2 and 4] provide their own way to this and other results. Our algorithm will not match this exponential space bound but provides with time $O(1.2335^n)$ the best polynomial space algorithm so far.

The interested reader will find more information on exponential time algorithms in the textbooks of Gaspers [7] and of Fomin and Kratsch [6]. Families of graphs for which maximum induced matching problem is **NP**-hard has been studied in great detail by the following references [1, 3, 9, 10, 13, 14].

2 Definitions and Preliminaries

Now we proceed more formally to introduce the notations used in this paper.

Definition 1. A subcubic graph is a graph where each vertex has at most three neighbours. For any subcubic graph, we call a vertex W_1-vertex if it has degree three and all its neighbours have degree at least two. Otherwise, the vertex is called W_0-vertex.

For the naming of vertices in the present work, we use the following conventions: vertices are usually denoted by a, b, c, d, e where a is of degree one, b of degree two, c of degree three, d of degree two or three and e of degree zero to three. Furthermore, subscripts and superscripts can be used to have more names.

Definition 2. Given a subcubic graph (V, E), we also denote by W_1 (respectively W_0) the set of all the W_1-vertices (respectively W_0-vertices) in the graph. We say that there is a *meta-edge* between two W_1-vertices c_1 and c_2 if there is a sequence (possibly empty) of W_0-vertices $d_1, d_2, \ldots, d_t$ such that the graph has the edges $(c_1, d_1), (d_t, c_2)$ and (d_i, d_{i+1}) for i with $1 \leq i < t$ (in case t is 0, then there is an edge between c_1 and c_2). The meta-edges between c_1 and c_2 are denoted by $[c_1, c_2]$, $[c_1, c_2]'$ and $[c_1, c_2]''$ where the primed versions are only used if there are several such meta-edges and one needs to distinguish them.

The (multi) graph of all W_1-vertices and meta-edges between these vertices is called the meta-graph associated to (V, E). Note that the meta-graph might be a "multi-graph", as there maybe multiple meta-edges between two vertices in the meta-graph.

Sometimes, we also consider the meta-edge (as in above) as the sequence of edges (c_1, d_1), (d_1, d_2), $\ldots$, (d_t, c_2) and denote/write it as $c_1 - d_1 - d_2 \ldots d_t - c_2$. We also sometimes say that the meta-edge $[c_1, c_2]$ contains/consists of the edges $(c_1, d_1), (d_1, d_2), \ldots, (d_t, c_2)$ or contains the vertices $c_1, d_1, \ldots, d_t, c_2$.

If they exist, $[c_1, c_2]$, $[c_1, c_2]'$ and $[c_1, c_2]''$ have only the end vertices c_1 and c_2 in common. We note that there are at most three meta-edges between c_1, c_2 and if there are three, then c_1 and c_2 are the only W_1-vertices in their component of the graph.

If S is a subset of V then S defines an induced matching of G if and only if every vertex in S has exactly one neighbour in S. For the ease of notations, we also say that an edge is in S if and only if it is in E and both its endpoints are in S. As the matching is induced, the endpoints of distinct edges in S are not neighbours in (V, E), that is, not connected by an edge in E.

Note that given a graph G, we can find its W_1-vertices and the meta-edges between them in polynomial time; meta-edges are also called paths and, more precisely, they are paths whose endpoints are W_1-vertices and where no inner node of the path is a W_1-vertex. As we can easily detect a degree one neighbour of a degree three W_0-vertex, one can easily find all the meta-edges starting from any W_1-vertex (to some W_1-vertex, including possibly itself). There are at most three meta-edges between two W_1-vertices and the latter happens if and only if these W_1-vertices are only connected to each other through meta-edges.

Furthermore, Monien and Preis showed the following.

Theorem 3 (Monien and Preis [17]). *For every $\varepsilon > 0$, there is a number k_ε such that for all subcubic graphs with at least $n \geq k_\varepsilon$ vertices, a polynomial time* *algorithm finds a partition of the graph into two halves, with each half having at least $n/2 - 1/2$ vertices, such that there are at most $(1/6 + \varepsilon) \cdot n$ edges between the two halves. This set of edges is called the "bisection" of the graph. For graphs with less than k_ε vertices, the algorithm returns the optimal bisection by using table look-up.*

In our earlier technical report [12, Corollary 2], we used this fact to give the corresponding algorithm for meta-graphs where they originally run the algorithm for possible double-meta-edges between two W_1-vertices to be considered as single meta-edges and then adjust the result such that for each case of original double-meta-edges between two W_1-vertices they choose one of the neighbouring single meta-edges to replace it. Thus one W_1-vertex may be moved from one half of the bisection into the other. The W_1-vertex chosen to be moved is the one which makes the partition more balanced, that is, which keeps the difference in the number of W_1-vertices between the two sides of the bisection bounded by two. Thus by relaxing the permitted size difference to two instead of one (as in the original result of Monien and Preis), one can get that the bisection of the meta-graph does not have double-meta-edges.

The following proposition is straightforward.

Proposition 4 (Subset Principle). *Suppose $G = (V, E)$ is a graph whose MIM we need to find and that we are considering two alternatives for MIM,*

- **(a)** *An induced matching S union with MIM of induced subgraph from $U \subseteq V$, where $(S \cup N(S)) \cap U = \emptyset$ and*
- **(b)** *An induced matching S' union with MIM of induced subgraph from $U' \subseteq V$, where $(S' \cup N(S')) \cap U' = \emptyset$.*

Suppose further that above U, U', S, S' satisfy $U \subseteq U'$ and $|S| \leq |S'|$.

Then, one can ignore the alternative MIM computed by (a) above as it is bounded in size by the MIM obtained in (b).

Note that we use the above subset principle to cut down on some alternative expansions S, S' of a given preliminary matching (with associated remaining graph U, U' respectively).

Lemma 5. *Suppose we are considering the MIM problem for a (sub) graph $(U, E(U))$. Suppose $Y \subseteq X \subseteq U$ such that there are no edges of the form (e, e'), with $e \in X - Y$ and $e' \in U - X$. Suppose M is an MIM of U.*

Suppose $S \subseteq X$ such that each member of S has exactly one neighbour in S in the graph $(U, E(U))$. Suppose that,

- **(i)** *$(Y - M) \cap S = \emptyset$, and*
- **(ii)** *$|E(S)| - |E(M \cap X)| \geq$ number of vertices in $M \cap Y$ which have edges in $E(M)$ to vertices outside X.*

Then S is a part of some MIM in $(U, E(U))$.

Proof. Suppose M' be the set of vertices in $M - X$ which have an edge to a vertex in X. Let $S' = M' \cup (M \cap X)$. Note that $|S| \geq |S'|$. Also, $U - X - N(S)$ is a superset of $U - X - N(X \cup M')$. Lemma follows using the subset principle (Proposition 4). $\square$

Intuitively, in the above Lemma, think of Y as "boundary of X", i.e., the only vertices in Y which have edges to vertices outside X.

Definition 6 (Simplification Rule SR). Consider a graph (U, E) in which we want to find an MIM S. Let e be a vertex with a neighbour d of degree at least two such that all other neighbours of d are of degree one (there may be one or two such neighbours, say a and perhaps a'). Then using the subset principle, it can be assumed without loss of generality that $d, a \in S$. Thus we can remove the vertices e, d, a, a' from further consideration for finding remaining part of MIM.

The above situation is known in the literature. It is very similar to the pending edge elimination of Xiao and Tan [20, Lemma 5] and implied by the Simplification Rule S3 of our earlier technical report [12]. One can adjust the notions of Xiao and Tan to cover this case fully by saying that d has pending edges if and only if d has neighbours of degree 1 and furthermore all neighbours of d of degree 2 or more form a clique – note that a single neighbour of degree at least 2 forms always a clique of size 1. In this situation, we can assume by arguments of Xiao and Tan [20, Lemma 5] that there is a maximum induced matching of the graph which contains an edge between d and one of its degree 1 neighbours. For completeness we give a proof below for soundness of the simplification rule.

Proposition 7. *Simplification Rule SR is sound.*

Proof. Suppose that S is an MIM. The case that $e \notin S$ is trivial. We now consider the case that e is in S. Then either d must be in S or some other vertex adjacent to e must be in S, along with e. Without loss of generality, we can assume that $\{e, d\} \subseteq S$ as the proof for either case is similar. Let $U = V - S - N(S) - \{a, a'\}$, since after the removal of S and $N(S)$ from V, the vertices a and a' have 0 degree, and can thus be removed from consideration.

Now, consider $S' = S \cup \{d, a\} - \{e\}$. Let $U' = V - S' - N(S') - \{a'\}$, as the vertex a' becomes a 0 degree vertex after the removal of S and $N(S)$, and can thus be removed from consideration.

We have that $U \subseteq U'$ and $|S| \leq |S'|$, and thus from the Subset Principle given earlier, we can assume without loss of generality that $\{d, a\}$ is in MIM. $\square$

Proposition 8. *(a) If a subcubic graph (component) is acyclic, then an MIM of the graph can be found in polynomial time in the number of vertices.*

(b) If a subcubic graph consists only of W_0-vertices, then an MIM of the graph can be found in time polynomial in the number of vertices.

(c) If a subcubic graph consists of at most a constant κ number of W_1-vertices, then an MIM of the graph can be found in time polynomial in the number of vertices (though it may be exponential in κ).

Proof. These three items are consequences of the fact that a graph with at most κ W_1-vertices has path width at most $(1/6+\varepsilon)\kappa+O(1)$ for any $\varepsilon>0$ and therefore the MIM problem can be solved in polynomial time for fixed κ where the usage of both the computation time and the space is exponential in κ [5, 16]. For the sake of completeness, we give here the full proofs.

(a) Note that repeated use of Simplification Rule SR will leave the graph to be having only disconnected edges. Thus, we can find an MIM in polynomial time in the number of vertices.

(b) Without loss of generality assume that the graph is connected (otherwise consider each component separately). If the graph is acyclic, then part (a) gives us the result. If the graph has a cycle, then one can pick an edge (d, d') from it and branch the three cases that both d, d' are in MIM, or d is not in MIM or d' is not in MIM (both d, d' not being in MIM is covered by both of the last two cases). This gives at most three cases and after that one can use part (a) to solve the MIM problem in polynomial time for the graph.

(c) As the number of W_1-vertices is bounded by a constant κ, we can consider for each W_1-vertex d one of its neighbours d' and branch as in (b) for the cases that both d, d' are in MIM or d is not in MIM or d' is not in MIM. This gives a branching algorithm of complexity $O(3^\kappa)\cdot Poly(n)$. We assume that $\kappa\geq 2$ in order to also handle the pathological case that there are two W_1-vertices connected by three meta-edges – $\kappa\geq 2$ is sufficient as the graph component containing these two W_1-vertices is then isolated and does neither contain a further W_1-vertex nor a further meta-edge, see Definition 2. $\square$

Note that for any δ with $0<\delta<1$, one can find $\varepsilon>0$ and corresponding κ such that for all $m\geq\kappa$, $[\frac{m}{2}-1]/[m(1/6+\varepsilon)]\geq 3-\delta$. We will take δ to be small enough value.

3 Overview of the Algorithm

Intuitively, we will be constructing a branching tree, where the root of the tree represents the original graph G with n vertices whose MIM we need to find. Nodes in the branching tree are of the form (G, U, S, B), where U is a subset of the vertices of G (the node then represents finding an MIM for the induced subgraph of G on the vertices U), S denotes a matching in G (not necessarily maximum) such that the vertices in S and their neighbours in G do not belong to U. B is a set of meta-edges between two partitions (initially obtained using Monien Pries algorithm) which are remaining to process when we are in the branching phase (more on this below). We implicitly assume that the two partitions associated with the bisection edges B are also kept (we omit mentioning them explicitly for ease of notation).

At a particular node (G, U, S, B) of the branching tree, the algorithm will first do some simplification of the problem (using the Simplification Rule SR mentioned above), and then, if needed, do branching. The branching would only be in the "branching phases", where initially a bisection is obtained using the

Monien Preis method for bisecting and then the bisection meta-edges are removed one by one for each branching as we go down the branching tree (some of the meta-edges may get automatically removed due to the simplification rule).

It will always be the case that the MIM problem for the subgraph of G at a node (G, U, S, B) of the branching tree can be solved in polynomial time (in n) using the solutions of the MIM problem for the subgraphs at its children. Furthermore, the children of the nodes in the branching tree can be obtained in polynomial time (in n) from the subgraph represented by the node. Thus the overall complexity of finding an MIM of the graph G is within a polynomial factor of the number of leaves in the branching tree. Thus, we wish to bound the number of leaves in the branching tree.

Let μ denote our measure of complexity for a graph/subgraph (see more details in the next section). Then we let $T(\mu)$ denote the maximum number of leaf nodes generated by the algorithm when we have μ as the parameter for the input problem. Since the search tree is only generated by applying a branching rule, it suffices to consider the number of leaf nodes generated by these rules (as simplification rules take time only polynomial in n, the number of vertices in the original graph). To do this, we employ techniques in [15]. Suppose a branching rule has $r \geq 2$ children, with $t_1, t_2, \ldots, t_r$ reduction in the complexity measure for these children (compared to the measure of the subgraph at the parent). Then, any function $T(\mu)$ which satisfies $T(\mu) \geq T(\mu - t_1) + T(\mu - t_2) + \ldots T(\mu - t_r)$, with appropriate base cases, would satisfy the bounds for the branching rule. To solve the above linear recurrence, one can model this as $x^{-t_1} + x^{-t_2} + \ldots + x^{-t_r} = 1$. Let β be the root of this recurrence, where $\beta \geq 1$. Then any $T(\mu) = (\beta')^{\mu}$, with $\beta' \geq \beta$ would satisfy the recurrence for this branching rule. In addition, we denote the branching factor $\tau(t_1, t_2, \ldots, t_r)$ as β. Tuple $(t_1, t_2, \ldots, t_r)$ is also known as the branching vector [6]. If there are k branching rules, with branching factors $\beta_1, \ldots, \beta_k$, in the branch and bound algorithm, then the overall complexity of the algorithm can be seen as the largest branching factor among all k branching rules; i.e. $z = max\{\beta_1, \beta_2, \ldots, \beta_k\}$, and therefore the time complexity of the algorithm is bounded above by $O(z^{\mu}) \cdot Poly(n)$.

4 Measure and Conquer Algorithm

Assign weight 0 to each of the vertices in W_0, and weight 1 to each of the vertices in W_1. The measure μ for a graph is defined as the sum of the weights of all the vertices in the graph. In other words, μ for a graph is the number of vertices in W_1. By definition, $\mu \leq n$. Therefore, any algorithm solving in time $O(z^{\mu})$ also runs in time $O(z^n)$ since $O(z^{\mu}) \subseteq O(z^n)$, for any $z > 1$.

We now state the main result of this work. Note that the proof implies the easier formula that the MIM of a subcubic graph with n vertices can be solved in $O(1.2335^n)$ time; the $Poly(n)$ is absorbed by the uprounding of the branching factors to 1.2335.

Theorem 9. *The MIM problem of a subcubic graph with n vertices out of which k are W_1-vertices can be solved in $O(1.2335^k) \cdot Poly(n)$ steps.*

Here also the constant κ from above is absorbed as a constant into the O-expression and κ is a constant independent of k. Next we give the algorithm.

Algorithm $MIM(G, U, S, B)$

Invariant: $G = (V, E)$ is the original graph. S consists of a matching in G such that in the original graph G, none of the vertices in S and none of their neighbours are in $U \subseteq V$. Thus, if S is a part of some maximum induced matching of G, then the union of S and any maximum induced matching of $(U, E(U))$ will give a maximum induced matching of G.

B if non-empty, denotes that the algorithm is in a branching phase, where the set of meta-edges in B bisects the W_1-vertices of U into two (nearly equal) parts. We will be branching and removing the meta-edges in the bisection B one by one.

1. Simplify the graph by setting $(G, U, S, B) = \text{Simplify}(G, U, S, B)$.
   The procedure Simplify() is given below this algorithm.
2. If $B = \emptyset$ and $(U, E(U))$ is the disjoint union of nonempty graphs (U_1, E_1) and (U_2, E_2) with no edge between U_1 and U_2 in (V, E) then solve both subgraphs independently by using $MIM(G, U_1, S, \emptyset)$ and $MIM(G, U_2, S, \emptyset)$ and return the union of the two matchings as the answer.
3. If $B = \emptyset$ and the induced subgraph over U has at most κ number of W_1-vertices, then compute an MIM $S1$ for the induced subgraph over U and return $S \cup S1$.
4. If $B = \emptyset$ and $(U, E(U))$ has more than κ W_1-vertices, then do a bisection based on the Monien Preis algorithm obtaining a bisection meta-edge set B' and return $MIM(G, U, S, B')$. This starts a branching phase.
5. If $B \neq \emptyset$ (we are in a branching phase), then pick one meta-edge $[c_1, c_2]$ in B where c_1, c_2 denote its endpoints.
   5.1. Suppose the bisection partitions are P_1 and P_2. If the component in U connected to $[c_1, c_2]$ on one side, say P_1 without loss of generality, consists of at most two W_1-vertices, then:
   Transfer this component in P_1 connected to $[c_1, c_2]$ to P_2. This move causes some meta-edges $B' \subseteq B$ to be from P_2 to P_2. Then return $(G, U, S, B - B')$ (i.e., the meta-edges in B' are removed). Note that this gives only one child in the branching tree. Note that at least one meta-edge $[c_1, c_2]$ was deleted.
   5.2. If the meta-edge $[c_1, c_2]$ contains a W_0-vertex with degree three, then pick one such vertex c, and let a be its neighbouring vertex of degree one. Then branch based on whether c will be in the MIM or not:
   – $MIM(G, U - \{c, a\}, S, B - \{[c_1, c_2]\})$ and
   – $MIM(G, U - \{c, a\} - \mathrm{N}(c), S \cup \{c, a\}, B - \{[c_1, c_2]\})$.
   The algorithm then returns the better answer among these two branches.
   Note that, if in the second item above one considers that the neighbour of c used in the matching is not a but some other neighbour d of c, then not only c and $\mathrm{N}(c)$ but also all $\mathrm{N}(d)$ have to be removed from U, and while $\mathrm{N}(a) = \{c\}$, $\mathrm{N}(\{c, d\})$ is a proper superset

of $\mathrm{N}(\{c,a\})$. Thus by the subset principle only the two branchings above need to be considered and the above two branches are exhaustive for the current case in which the meta-edge $[c_1,c_2]$ contains a W_0-vertex of degree three.

Note that the simplification process (step 2) in the two children will then remove all the vertices in the meta-edge (and maybe more), and thus the meta-edge is deleted; similar comment applies for each of the cases below.

5.3. If the meta-edge $[c_1,c_2]$ contains no W_0-vertices with degree three, then all the W_0-vertices in the meta-edge are of degree two. Suppose the meta-edge is c_1-b_1-b_2-…-b_t-c_2 where c_1,c_2 are the two bordering W_1-vertices and all vertices $b_1,\ldots,b_t$ have degree 2. For the ease of notation, we use b_0,b_{t+1} as aliases for c_1,c_2 in formulas for S_0,S_1,S_2, in 5.3.3 below, though these vertices do not have degree 2.

5.3.1 If $t=0$, then branch based on whether both c_1,c_2 are in the MIM, or c_1 is not in the MIM or c_2 is not in the MIM (there is some overlap, when both c_1 and c_2 are not in the MIM but this is ok, as we are taking the best of the cases).

That is consider three children,
- $MIM(G,U-\{c_1,c_2\}-\mathrm{N}(c_1)-\mathrm{N}(c_2),S\cup\{c_1,c_2\},B-\{[c_1,c_2]\})$,
- $MIM(G,U-\{c_1\},\ S,\ B-\{[c_1,c_2]\})$,
- $MIM(G,U-\{c_2\},\ S,\ B-\{[c_1,c_2]\})$

and return the best of the three answers.

5.3.2 If $t=1$, then branch based on whether (c_1,b_1) is an edge in the MIM, or (b_1,c_2) is an edge in the MIM, or b_1 is not in the MIM.

That is, consider three children:
- $MIM(G,U-\{c_1,b_1\}-\mathrm{N}(c_1)-\mathrm{N}(b_1),S\cup\{c_1,b_1\},B-\{[c_1,c_2]\})$,
- $MIM(G,U-\{b_1,c_2\}-\mathrm{N}(b_1)-\mathrm{N}(c_2),S\cup\{b_1,c_2\},B-\{[c_1,c_2]\})$,
- $MIM(G,U-\{b_1\},S,B-\{[c_1,c_2]\})$,

and return the best of the three answers.

5.3.3 If $t\geq 2$, for $\ell\in\{0,1,2\}$, let $S_\ell=\{b_{3h+\ell},b_{3h+\ell+1}:h\geq 0$ and $3h+\ell\leq t\}$, where we take $b_0=c_1$ and $b_{t+1}=c_2$. Branch based on putting S_0 or S_1 or S_2 in S, and appropriately removing the corresponding vertices and their neighbours.

That is, for $\ell\in\{0,1,2\}$, let
$U_\ell=\bigcup\{\{b_{3h+\ell},b_{3h+\ell+1}\}\cup\mathrm{N}(b_{3h+\ell})\cup\mathrm{N}(b_{3h+\ell+1}):3h+\ell\leq t,\ell\in\{0,1,2\},h\geq 0\}$
and consider the three children:
$MIM(G,U-U_\ell,S\cup S_\ell,B-\{[c_1,c_2]\})$ for $\ell\in\{0,1,2\}$.
The algorithm then returns the best answer among these three branches.

End Algorithm MIM

Function Simplify(G,U,S,B).

While at least one of the following three if-conditions applies to the graph $(U,E(U))$ do the following three steps.

1. If there is a W_0-vertex c_1 in U such that (a) c_1 was a W_1-vertex before the previous step of the algorithm and was part of a meta-edge in B, and (b) there is a meta-edge $[c_2, c_3]$ in U which has c_1 as a vertex in it, and c_2, c_3 are in different partitions of the bisection B. Then, place $[c_2, c_3]$ in B. Intuitively, above replaces an old meta-edge in B, if c_2, c_3 above are in the different partitions of the bisection.
2. If there is a meta-edge $[c_1, c_2]$ in B such that some of the vertices on this meta-edge are not in U then remove this meta-edge from B.
3. If there is a vertex d of degree at least two which has a neighbour e of arbitrary degree and all other neighbours of d (say a and perhaps a') have degree 1 in U then update $S = S \cup \{a, d\}$ and $U = U - \{d, e, a, a'\}$. (Here $a' = a$ in the case that the degree of d is 2.)

End While

Return the updated (G, U, S, B).

End Function Simplify

5 Verification of the Properties of the Algorithm

Now we consider the analysis of the algorithm. We shall prove that the cases are exhaustive, show that the algorithm preserves the optimality of the solution, the correctness of the algorithm and, finally, establish the overall runtime of the algorithm.

Proposition 10. *The case-distinction of the algorithm is exhaustive.*

Proof. In Step 1 of the algorithm, we do some cleaning via the simplify algorithm. The aim is to (i) update the meta-edges in case some of the vertices in previous step have been converted from degree 3 to degree 2 vertices or some of the meta-edges no longer exist (see steps 1 and 2 of procedure Simplify), and (ii) handle the case when there is a vertex d of degree at least two with exactly one neighbour e of degree 2 or more (in which case we can without loss of generality, using the Simplification Rule SR, assume that the MIM has the edge from d to one of the other neighbours of d).

In Step 2, we deal with disconnected components of the graph. Hence, after this line, we can assume that our graph is a connected graph.

In Step 3, if there are at most κ number of W_1-vertices, then we solve the problem directly. After which, we can assume that any given instance has more than κ number of W_1-vertices.

In Step 4, we apply the Monien Preis algorithm to obtain a bisection of meta-edges B and begin the branching phase.

In Step 5, we deal with the different cases of branching based on a selected meta-edge. Step 5.1 deals with the case that some component in one the partitions P_1 and P_2 has no more than three W_1-vertices. If the selected meta-edge contains a degree three W_0-vertex, then this case is handled by Step 5.2. Otherwise, the case is handled by Step 5.3, where we consider the cases of the number

$t+1$ of edges in the meta-edge being $t=0$ (Step 5.3.1), $t=1$ (Step 5.3.2) and $t \geq 2$ (Step 5.3.3).

Therefore, we have covered all cases. □

Next, we show that algorithm preserves the optimality of the solution.

Proposition 11. *The algorithm generates a maximum induced matching.*

Proof. Note that the Simplification Rule SR clearly preserves optimality of the solution. Step 2 clearly preserves optimality as it works on different components of the graph separately. Step 3 explicitly computes optimal answer. Step 4 and Step 5.1 preserve optimality as the subgraph does not change. Step 5.2 preserves optimality as if c is in an MIM, then by the subset principle we can assume that (c, a) is in the MIM. Step 5.3.1 preserves optimality as we consider all possible cases of c_1, c_2 being in the MIM, where if both c_1, c_2 are in the MIM, then it must be via edge (c_1, c_2) being in the matching. Step 5.3.2 preserves optimality as we have considered all possible cases for (c_1, b_1) and (b_1, c_2) being in the matching. Now we consider Step 5.3.3.

The edges in the meta-edge $[c_1, c_2]$ are $(c_1, b_1), (b_1, b_2), \ldots, (b_t, c_2)$. Consider some MIM M of the subgraph U. For the case analysis below, we will use Lemma 5 with $X = \{c_1, b_1, \ldots, b_t, c_2\}$, and $Y = \{c_1, c_2\}$ (except for case 3.1.2 where X, Y are explicitly defined differently), to claim that one of S_1, S_2, S_3 is contained in some MIM for U, and thus optimality is preserved in step 5.3.3.

We now consider the following cases.

Case 1: Both c_1 and c_2 are not in M.

Case 1.1: $t+1 = 3h$. In this case by Lemma 5, there is an MIM of U containing S_1 as $E(S_1)$ gives maximum number of edges from $[c_1, c_2]$ without using c_1, c_2.

Case 1.2: $t+1 = 3h+1$. In this case $E(M)$ can have at most h edges from $[c_1, c_2]$, and both $E(S_1)$ and $E(S_2)$ achieve this without using either of c_1, c_2. Thus, by Lemma 5, there is an MIM of U containing S_1 or S_2.

Case 1.3: If $t+1 = 3h+2$, then $E(M)$ has at most h edges from $[c_1, c_2]$. $E(S_2)$ achieves h edges from $[c_1, c_2]$ without using either of c_1, c_2. Thus, by Lemma 5, there is an MIM of U containing S_2.

Case 2: Only one of c_1, c_2 is in M. Assume without loss of generality that c_1 is in M. Then, (b_1, b_2) and (b_t, c_2) are not in $E(M)$.

Case 2.1.1: $t+1 = 3h$ for some h and $(c_1, b_1) \in E(M)$. Now, $E(S_0)$ achieves maximum number of edges in $[c_1, c_2]$ without using c_2. Thus, by Lemma 5, there is an MIM of U containing S_2.

Case 2.1.2: $t+1 = 3h$ for some h and $E(M)$ does not contain (c_1, b_1). In this case $E(M)$ can have at most $h-1$ edges from $[c_1, c_2]$. Now, $E(S_1)$ has h edges from $[c_1, c_2]$ and does not contain c_2. Thus, by Lemma 5, there is an MIM of U containing S_1 (as the edge in M from c_1 can be dropped due to extra edge in $E(S_1)$).

Case 2.2: $t+1 = 3h+1$, for some h. In this case $E(M)$ can have at most h edges from $[c_1, c_2]$. $E(S_2)$ has h edges, and S_2 does not contain c_1, c_2, b_1. Thus, there is an MIM of U containing S_2 as we can replace the edges of $E(M)$ having at least one end point in $b_1, \ldots, b_t$ by edges of $E(S_2)$.

Case 2.3.1: $t+1 = 3h+2$ for some h and $(c_1, b_1) \in E(M)$. In this case $E(S_0)$ has at least the same number of edges from $[c_1, c_2]$ as $E(M)$ has from $[c_1, c_2]$. Also, S_0 does not contain c_2. By Lemma 5, there is an MIM of U containing S_0.

Case 2.3.2: $t+1 = 3h+2$ for some h and $(c_1, b_1) \notin E(M)$. In this case $E(M)$ can have at most h of the edges from $[c_1, c_2]$. $E(S_2)$ has h edges from $[c_1, c_2]$ and S_2 does not contain c_1, c_2, b_1, b_t. Thus, there is an MIM of U containing S_2 as we can replace the edges of $E(M)$ having at least one endpoint in $b_1, \ldots, b_t$ by edges of $E(S_2)$.

Case 3: Both c_1, c_2 are in M.

In this case (b_1, b_2) and (b_{t-1}, b_t) are not in $E(M)$.

Case 3.1.1: $t+1 = 3h$ for some h and (c_1, b_1) is in $E(M)$. In this case $E(S_0)$ contains at least the same number of edges from $[c_1, c_2]$ as $E(M)$, and $c_2, b_t \notin S_0$. By Lemma 5, there is an MIM of U containing S_0.

Case 3.1.2: $t + 1 = 3h$ and (c_1, b_1) is not in $E(M)$. In this case if M has h edges from $[c_1, c_2]$, then it could be only be by M containing S_2. Otherwise, $E(M)$ has at most $h - 1$ edges from $[c_1, c_2]$. $E(S_2)$ has h edges from $[c_1, c_2]$ and it does not contain c_1, b_1. Thus, using Lemma 5 with $X = \{b_1, b_2, \ldots, b_t, c_2\}$ and $Y = \{b_1, c_2\}$, we have that there is an MIM containing S_2.

Case 3.2.1: $t+1 = 3h+1$ for some h and $E(M)$ has at least one of (c_1, b_1) or (b_t, c_2). In case $E(M)$ has $h+1$ edges from $[c_1, c_2]$, then S_0 is contained in $E(M)$. In case $E(M)$ has at most h edges from $[c_1, c_2]$, then as $E(S_0)$ has $h + 1$ edges and S_0 contains both c_1, c_2, by Lemma 5, we have that some MIM contains S_0.

Case 3.2.2: $t + 1 = 3h + 1$ for some h and $E(M)$ does not contain any of (c_1, b_1) or (b_t, c_2). In this case $E(M)$ has at most $h - 1$ edges from $[c_1, c_2]$. As $E(S_0)$ has $h + 1$ edges, using Lemma 5, we have that some MIM contains S_0.

Case 3.3.1: $t + 1 = 3h + 2$ for some h and $E(M)$ has at most h edges from $[c_1, c_2]$. Then, $E(S_2)$ has h edges from $[c_1, c_2]$ and S_2 does not contain c_1, c_2, b_1, b_t. Thus, there is an MIM of U containing S_2 as we can replace the edges of $E(M)$ having at least one endpoint in $b_1, \ldots, b_t$ by edges of $E(S_2)$.

Case 3.3.2: $t + 1 = 3h + 2$ for some h and $E(M)$ has $h + 1$ edges from $[c_1, c_2]$. Then, $E(M)$ must contain at least one of (c_1, b_1) or (b_t, c_2). In these cases, consider $E(S_0)$ and $E(S_1)$ respectively, which both contain $h + 1$ edges from $[c_1, c_2]$ and do not contain c_2 and c_1 respectively. Thus, using Lemma 5, we have that some MIM contains S_0 or S_1 respectively in this case.

Thus, step 5.3.3 preserves optimality. □

Note that it follows from Propositions 10 and 11 that the algorithm is correct.

Proposition 12. *The running time of the algorithm is* $O(1.2335^k) \cdot Poly(n)$.

Proof. Note that the general idea of the algorithm works by repeatedly applying the Monien Preis algorithm to get a bisection and we branch on the meta-edges until the component has small enough number of W_1-vertices for us to apply Step 3 directly. Recall that the leaves of the branching tree are small enough sub-graphs of the graph G. We bound the overall runtime of the algorithm by bounding the number of leaves in the branching tree. For this, we bound the number of leaves $R(\mu, e)$, of the branching tree whose associated subgraph has a fixed vertex e of the graph G (here μ denotes the complexity measure, number of W_1-vertices in the graph). Thus, $\sum_{e \in V} R(\mu, e)$ will bound the total number of leaves. As mentioned earlier, the complexity of the algorithm is bounded by $Poly(n)$ times the number of leaves in the branching tree. As the number of vertices in V can be absorbed in $Poly(n)$, it is thus sufficient to bound $R(\mu, e)$, for each vertex e.

Now we explain the details. As mentioned above, branching happens only in phases, where initially using the Monien Preis method, the graph is partitioned in two parts (nearly equal), where the number of bisection edges is at most $m(1/6+\varepsilon)$, where m is the number of W_1-vertices in the corresponding subgraph.

For bounding $R(\mu, e)$, for arbitrary vertex e, we will consider only those branches of the computation tree whose subgraph contains e. In particular, if there are two disjoint components of the graph, only one will produce a sub-computation tree which contributes to $R(\mu, e)$. This can be used to calculate the branching factor of rules removing a bisection meta-edge. Only one side of the bisection will eventually contain e and therefore all W_1-vertices on the other side of the bisection can be counted as removed; these vertices will be counted in an amortised way per cutting of an meta-edge in the bisection.

Thus, in the analysis below, we only need to worry about the W_1-vertices from the partition corresponding to the vertex e as above (plus some of the W_1-vertices which were moved over from the other partion due to step 5.1 above). As there are at most $m(1/6 + \varepsilon)$ number of bisection edges and each partition consists initially of at least $m/2-1$ number of W_1-vertices, we can allocate credit of $(m/2-1)/(m(1/6+\varepsilon)) \geq 3-\delta$ for each deletion of bisection meta-edge, i.e., for each of the branching steps in step 5 of the algorithm. We call this AuxCredit in the analysis below. Note that this also means that for the other reductions in the number of W_1-vertices, we should consider only reductions due to W_1-vertices in the same partition side as e, since the reductions due to other partition has already been taken into account due to AuxCredit, and thus cannot be counted twice.

Now our aim is to inductively bound $R(s, e)$, the number of leaves of the subtree starting with s W_1-vertices which contain the vertex e among others.

We will show that $R(s, e) \leq p^s$, where $p = 1.2335$ is the above chosen strict upper bound of all the "branching factors" that are obtained in the cases below. Here the choice of 1.2335 as the upper bound follows the conventions, as usually only the first four digits after the decimal dot are considered, but any other upper bound would also do.

Suppose that at any node (with s W_1-vertices in the subgraph associated with it) in the branching tree we have reduction in number of W_1-vertices for its r children respectively as $\alpha_1, \alpha_2, \ldots, \alpha_r$ (where each α_i is positive) after the simplification process (step 1) in that child; we do this "after the simplification" for ease of our calculations. Note that this is fine as the simplification process does not do any branching. Note also that for bounding the value $R(s, e)$ when computing the reductions in $\alpha_1, \alpha_2, \ldots$, we need to consider only the reduction from "the same part in the bisection partition to which vertex e belongs", as we have already considered the vertices from the other side in AuxCredit; the subtrees produced by other branches will be counted in other terms $R(n, e')$.

Let $\gamma > 1$ be such that

$$\gamma^s = \sum_{i:1 \le i \le r} \gamma^{s-\alpha_i}$$

where this γ is represented as $\tau(\alpha_1, \alpha_2, \ldots, \alpha_r)$. Then, using any $\beta \ge \gamma$ would give a valid bound for $R(s, e)$ with respect to "this branching" in the branching tree.

Now, step 5.1 above has only one branch and is therefore a reduction rule. This will only take polynomial time in n.

In step 5.2, the reduction on each side of the partition reduces at least one W_1-vertex after the simplification process (the W_1-vertices at the end of the meta-edge $[c_1, c_2]$) and thus the reduction in the number of W_1-vertices is at least $4 - \delta$ (taking into account AuxCredit mentioned above) on each side, for each of the two cases. Thus, the branching factor is at least $\tau(4 - \delta, 4 - \delta)$.

In step 5.3, first note that following:

If the vertex c_1 (respectively c_2) is removed from the children subgraph, then either it causes another bisection edge to be deleted, or it causes at least 2 other W_1-vertices to be removed or become W_0-vertices in the same partition side as c_1 (respectively c_2) after the simplification process in the child nodes. To see this, consider the following exhaustive cases. If the two meta-edges (different from $[c_1, c_2]$) originating from c_1 both lead to a W_1-vertex c, then there must be a meta-edge from c to another W_1-vertex c'. After simplification process, as c_1 is removed, c is also removed and c' either gets removed or becomes a W_0-vertex. If the two meta-edges (different from $[c_1, c_2]$) originating from c_1 lead to two different W_1-vertices c and c', then after simplification process both c and c' either get removed or become W_0-vertices. The only remaining case is when the meta-edge from c_1 leads to itself, which is already considered in step 5.1. Thus, in each of the above possibilities, when c_1 (respectively c_2) gets removed from child subgraph, we get at least two additional decrease of W_1-vertices on the same side of the partition or a further reduction $3 - \delta$ due to AuxCredit for the additional deletion of bisection edge (in addition to c_1 being removed). Thus, besides the deletion of the bisection edge from c_1, there is at least an additional reduction of three W_1-vertices in this case.

Now we consider each of the substeps in step 5.3.

In step 5.3.1, clearly, c_1 (respectively c_2) is removed from at least 2 of the three children nodes. Thus, the branching factor is at least $\tau(4 - \delta, 6 - \delta, 6 - \delta)$. Here $3 - \delta$ in each child is due to AuxCredit for the bisection edge to c_1 (respectively c_2), 1 in each child is due to c_1, c_2 becoming W_0-vertices or being removed from children, furthermore 2 in at least two of the children is due to two additional W_1-vertices being removed or becoming W_0-vertices as mentioned above or additional $3 - \delta$ AuxCredit due to another additional bisection edge being deleted.

In step 5.3.2, similarly, c_1 (respectively c_2) is removed in at least two of the children as both are neighbours of b_1. Thus, branching factor is at least $\tau(4-\delta, 6-\delta, 6-\delta)$ as in the case of step 5.3.1.

In step 5.3.3, note that c_1 is removed from at least two of the children subgraph as c_1 is a neighbour of b_1 which is in S_0 and S_1. Similarly for c_2 (though which of S_0, S_1, S_2 is used depends on $t \mod 3$). Thus, branching factor is at least $\tau(4-\delta, 6-\delta, 6-\delta)$ as in the case of step 5.3.1.

Now $\tau(4,4) < 1.1893$ and $\tau(4,6,6) < 1.2335$. Thus, as the worst branching factor is strictly below $p = 1.2335$, the value δ can be chosen to be small enough due to the gap above in the inequality ($\delta = 10^{-6}$ works). ☐

6 Conclusion

The present work investigates the complexity of the MIM problem on subcubic graphs. For graphs of degree at most four, the best known polynomial space algorithm has the same bound as the overall best polynomial space algorithm using time $O(1.4231^n)$ [20] and any improvement to the special case of degree 4 would result in an improvement of the algorithms for the general problem as well. However, the case of graphs with maximum degree 3 has in general only been solved by invoking the maximum independent set of line graphs, a method which gives a $O(1.3139^n)$ time and polynomial space algorithm. The current work gives an improved use of the bisection method of Monien and Preis and obtains a polynomial space algorithm running in time $O(1.2335^n)$. Here, by adding a polynomial multiplicative factor $Poly(n)$, 1.2335^n can be replaced by 1.2335^k, where k is the number of degree three vertices in the graph all of whose neighbours have degree at least two, giving us an FPT algorithm.

References

1. Kathie Cameron. *Induced matchings.* Discrete Applied Mathematics 24, no. 1-3 (1989): 97-102.
2. Maw-Shang Chang, Li-Hsuan Chen and Ling-Ju Hung. *Moderately exponential time algorithms for the maximum induced matching problem.* Optimization Letters 9, no. 5 (2015): 981-998.
3. William Duckworth, David F. Manlove and Michele Zito. *On the approximability of the maximum induced matching problem.* Journal of Discrete Algorithms 3, no. 1 (2005): 79-91.
4. Fedor V. Fomin, Fabrizio Grandoni and Dieter Kratsch. *A measure & conquer approach for the analysis of exact algorithms.* Journal of the ACM (JACM) 56, no. 5 (2009): 1-32.
5. Fedor V. Fomin and Kjartan Høie. *Pathwidth of cubic graphs and exact algorithms.* Information Processing Letters 97, no. 5 (2006): 191-196.
6. Fedor V. Fomin and Dieter Kratsch. *Exact Exponential Algorithms.* Texts in Theoretical Computer Science. An EATCS Series. Springer, Berlin, Heidelberg, 2010.

7. Serge Gaspers. *Exponential Time Algorithms: Structures, Measures, and Bounds.* 216 pages, VDM Verlag Dr. Müller, 2010.
8. Serge Gaspers and Gregory B. Sorkin. *Separate, measure and conquer: faster polynomial-space algorithms for Max 2-CSP and counting dominating sets.* ACM Transactions on Algorithms (TALG), 13 no. 4 (2017):44:1-36.
9. Martin Charles Golumbic and Renu C. Laskar. *Irredundancy in circular arc graphs.* Discrete Applied Mathematics 44, no. 1-3 (1993): 79-89.
10. Martin Charles Golumbic and Moshe Lewenstein. *New results on induced matchings.* Discrete Applied Mathematics 101, no. 1-3 (2000): 157-165.
11. Sushmita Gupta, Venkatesh Raman and Saket Saurabh. *Maximum r-regular induced subgraph problem: Fast exponential algorithms and combinatorial bounds.* SIAM Journal on Discrete Mathematics 26, no. 4 (2012): 1758-1780.
12. Gordon Hoi, Ammar Fathin Sabili and Frank Stephan. An algorithm for finding maximum induced matching in subcubic graphs. Technical report on http://www.arxiv.org/abs/2201.03220 (2022).
13. C. W. Ko and F. Bruce Shepherd. *Bipartite domination and simultaneous matroid covers.* SIAM Journal on Discrete Mathematics 16, no. 4 (2003): 517-523.
14. Daniel Kobler and Udi Rotics. *Finding maximum induced matchings in subclasses of claw-free and P 5-free graphs, and in graphs with matching and induced matching of equal maximum size.* Algorithmica 37, no. 4 (2003): 327-346.
15. Oliver Kullmann. *New methods for 3-SAT decision and worst-case analysis.* Theoretical Computer Science, 223 no. 1-2 (1999):1-72.
16. Akash Kumar and Mithilesh Kumar. *Deletion to Induced Matching.* Technical Report on https://arxiv.org/abs/2008.09660 (2020).
17. Burkhard Monien and Robert Preis. *Upper bounds on the bisection width of 3- and 4-regular graphs.* Journal of Discrete Algorithms 4 no. 3 (2006): 475-498.
18. Larry J. Stockmeyer and Vijay V. Vazirani *NP-completeness of some generalizations of the maximum matching problem.* Information Processing Letters 15, no. 1 (1982): 14-19.
19. Mingyu Xiao and Hiroshi Nagamochi *Exact algorithms for maximum independent set.* Information and Computation 255 (2017): 126-146.
20. Mingyu Xiao and Huan Tan. *Exact algorithms for maximum induced matching.* Information and Computation 256 (2017): 196-211.
21. Silvio Micali and Vijay V. Vazirani. *An $O(V^{1/2}E)$ algoithm for finding maximum matching in general graphs.* In 21st Annual Symposium on Foundations of Computer Science (sfcs 1980), pp. 17-27, IEEE 1980.