LETTER
Special Section on Foundations of Computer Science — Algorithm, Theory of Computation, and their Applications —

Exact Exponential Algorithm for Distance-3 Independent Set Problem

Katsuhisa YAMANAKA*(a), Member, Shogo KAWARAGI(b), Nonmember, and Takashi HIRAYAMA(c), Member

SUMMARY Let $G = (V, E)$ be an unweighted simple graph. A distance-$d$ independent set is a subset $I \subseteq V$ such that $\text{dist}(u, v) \geq d$ for any two vertices $u, v$ in $I$, where $\text{dist}(u, v)$ is the distance between $u$ and $v$. Then, maximum distance-$d$ independent set problem requires to compute the size of a distance-$d$ independent set with the maximum number of vertices. Even for a fixed integer $d \geq 3$, this problem is NP-hard. In this paper, we design an exact exponential algorithm that calculates the size of a maximum distance-3 independent set in $O(1.4143^n)$ time.

key words: exact exponential algorithm, independent set, distance-$d$ independent set, maximum distance-$d$ independent set

1. Introduction

Let $G = (V, E)$ be an unweighted simple graph with the vertex set $V$ and the edge set $E$. We denote by $n$ the numbers of vertices in $G$. An independent set of $G$ is a subset $I \subseteq V$ of vertices such that $[u, v] \notin E$ holds for all $u, v \in I$. Maximum Independent Set problem (MaxIS for short) asks to calculate the size of an independent set of $G$ with the maximum number of vertices. This problem is one of the most fundamental and important problems in theoretical computer science and is a classic NP-hard problem.

In this paper, we deal with a generalization of an independent set. A distance between two vertices $u, v$ of $G$, denoted by $\text{dist}(u, v)$, is the number of edges on a shortest path between them. For an integer $d \geq 2$, a distance-$d$ independent set is a subset $I \subseteq V$ such that $\text{dist}(u, v) \geq d$ for any two vertices $u, v \in I$. Then, Maximum Distance-$d$ Independent Set problem (MaxDdIS for short) requires to compute the size of a distance-$d$ independent set with the maximum number of vertices. Examples of distance-3 independent sets are shown in Fig. 1.

Fig. 1 (a) A distance-3 independent set and (b) a maximum distance-3 independent set of the graph.

When $d = 2$, MaxDdIS is equivalent to MaxIS. Hence, it is obvious that MaxDdIS is NP-hard. Furthermore, for a fixed integer $d \geq 3$, MaxDdIS is NP-hard [1]. Hence, it seems to be hard to give a polynomial-time algorithm for MaxDdIS. As existing results, approximation algorithms for MaxD3IS are presented by Eto et al. [2], [3].

In this paper, we focus on exact exponential algorithms for MaxD3IS. For MaxIS, there are a lot of existing exact exponential algorithms. However, to the best of our knowledge, our algorithm is the first exact exponential algorithm for the problem. Our algorithm calculates the size of a maximum distance-3 independent set of a graph in $O(1.4143^n)$ time.

2. Algorithm and Its Running Time Analysis

Let $G = (V, E)$ be a graph. We denote by $N(v) = \{u \mid [v, u] \in E\}$ the set of the neighbors of $v$. We define $N^d(v) = \{u \mid \text{dist}(v, u) \leq d \text{ and } u \neq v\}$.

To solve MaxDdIS, we define a restricted variant of MaxDdIS. Suppose that we are given a graph $G = (V, E)$, a distance-$d$ independent set $I$ of $G$, and a subset $X \subseteq V$, where $I \cap X = \emptyset$, the problem RedMaxDdIS asks for the size of a maximum distance-$d$ independent set of $G$ including $I$ and excluding $X$. If we set $I = \emptyset$ and $X = \emptyset$, then RedMaxDdIS is equivalent to MaxDdIS. We say that $(G, I, X)$ is an instance of RedMaxDdIS. Let denote by $\alpha(G, I, X)$ the size of a maximum distance-$d$ independent set of $G$ including $I$ and excluding $X$. We denote by $F = V \setminus (I \cup X)$ and we say that a vertex in $F$ is free.

Now, let us assume that $d = 3$. Let $(G, I, X)$ be an instance of RedMaxD3IS. First, we give the following simple reduction.

Reduction D3IS. 1. Add all the vertices in $N^2(I)$ into $X$.

Let $X'$ be the subset obtained by applying Reduction D3IS. 1. Obviously, $\alpha(G, I, X) = \alpha(G, I, X')$ holds.

The second reduction is as follows.

Reduction D3IS. 2. Remove all the vertices of degree-1 in $X$.

Let $(G', I, X')$ be the instance obtained by applying Reduction D3IS. 2 to $(G, I, X)$. Then, $\alpha(G, I, X) = \alpha(G', I, X')$ holds, because any distance-3 independent set of $(G, I, X)$ is
also a distance-3 independent set of \((G', I, X')\). The third reduction is as follows.

**Reduction D3IS. 3.** Remove every edge between two vertices in \(X\).

Now, we prove that Reduction D3IS. 3 is correct.

**Lemma 1:** Let \((G, I, X)\) be an instance of ResMaxD3IS, and let \((G', I, X)\) be an instance obtained by applying Reduction D3IS. 3. Then, \(\alpha(G, I, X) = \alpha(G', I, X)\) holds.

**Proof.** Since edges are removed from \(G\), clearly \(\alpha(G, I, X) \leq \alpha(G', I, X)\) holds. We assume for a contradiction that \(\alpha(G, I, X) < \alpha(G', I, X)\) holds. Let \(I'_M\) be a maximum distance-3 independent set of \((G', I, X)\). Namely, \(|I'_M| = \alpha(G', I, X)\) holds. Let \(e = (u, v)\) be a removed edge in the reduction. Recall that \(u, v \in X\). If \(I'_M\) has no vertex in \(N(u) \cap F\) and \(I'_M\) has no vertex in \(N(v) \cap F\), then \(I'_M\) is also a distance-3 independent set of \((G, I, X)\). Assume that \(I'_M\) includes a vertex \(x \in N(u) \cap F\). Again \(I'_M\) is also a distance-3 independent set of \((G, I, X)\), because, for every vertex \(y\) in \(N(v)\), the length of any path from \(x\) to \(y\) along \(e\) is 3. Hence, \(\alpha(G, I, X) \geq |I'_M| = \alpha(G', I, X)\), which is a contradiction. \(\square\)

The last reduction is as follows.

**Reduction D3IS. 4.** Remove all the isolated vertices in \(X\).

Let \((G', I, X')\) be the instance obtained by applying Reduction D3IS. 4. Clearly, we have \(\alpha(G, I, X) = \alpha(G', I, X')\).

Let \((G', I', X')\) be the instance obtained by exhaustively applying Reduction D3IS. 1–4 for a given instance \((G, I, X)\) (actually \(I' = I\) holds, however, we use \(I'\) to unify the notations). In \((G', I', X')\), we can observe that every vertex in \(I'\) is an isolated vertex in \(G'\) and the other connected components consist of free vertices and the vertices in \(X'\). For a vertex \(v\), we denote by \(f(v)\) and \#\(f(v)\) the set and number of the free vertices in \(N^2(v)\). A connected component \(C\) is cyclic if \(C\) includes four or more free vertices and \#\(f(v)\) = 2 for every free vertex \(v\) in \(C\). In what follows, we show that ResMaxD3IS is polynomial-time solvable on cyclic connected components. This is a key observation of our algorithm.

Let \(C\) be a cyclic connected component in \(G'\). We define a cyclic order \((v_1, v_2, \ldots, v_k)\) among the free vertices in \(C\), as follows. Let \(v\) be any free vertex in \(C\), and let \(u, w\) be the two free vertices in \(N^2(v)\). Then, it can be observed that \(u\) and \(w\) are non-adjacent, because if \(u\) and \(w\) are adjacent, \(C\) includes only three free vertices \(v, u,\) and \(w\) (recall that \#\(f(v)\) = 2 for every free vertex \(v\) in \(C\)). Now, we choose \(v\) as \(v_1\) and \(u\) as \(v_2\) (we can choose \(u\) or \(w\) arbitrary). Next, let \(x\) be a free vertex in \(N^2(u)\) except \(v\). Then, we choose \(x\) as \(v_3\). We repeat the same process. This process assigns an order to each free vertex in \(C\) and ends up with \(w\). We call the obtained order a cyclic order of the free vertices in \(C\). In the cyclic order \((v_1, v_2, \ldots, v_k)\), we denote by \(\text{succ}(v_i)\) the successor of \(v_i\) for each \(i = 1, 2, \ldots, k\) in \(C\). Note that \(v_1 = \text{succ}(v_k)\). We also denote \(\text{succ}(\text{succ}(v_i))\) by \(\text{succ}^2(v_i)\).

Now, let us have observations for two consecutive free vertices in the cyclic order. For a vertex \(v_i\) and \(\text{succ}(v_i)\), they are adjacent (Fig. 2 (a)) or have one or more common adjacent vertices in \(X\) (Fig. 2 (b)). They may be adjacent and have one or more common adjacent vertices in \(X\) (Fig. 2 (c)). We say that a pair \((v_i, \text{succ}(v_i))\) is adjacent if \(v_i\) and \(\text{succ}(v_i)\) are adjacent and is non-adjacent if \(v_i\) and \(\text{succ}(v_i)\) are not adjacent. Note that, if \((v_i, \text{succ}(v_i))\) is non-adjacent, \(v_i\) and \(\text{succ}(v_i)\) have one or more common adjacent vertices in \(X\), that is \(\text{dist}(v_i, \text{succ}(v_i)) = 2\). We have the following lemma.

**Lemma 2:** Let \(C\) be a cyclic connected component, and let \((v_1, v_2, \ldots, v_k)\) be a cyclic order of the free vertices in \(C\). Then, if a pair \((v_i, \text{succ}(v_i))\) is adjacent for some \(i, 1 \leq i \leq k\), then \((\text{succ}(v_i), \text{succ}^2(v_i))\) is non-adjacent.

**Proof.** Assume for a contradiction that \((\text{succ}(v_i), \text{succ}^2(v_i))\) is adjacent. Then, \(f(v_i) = \{\text{succ}(v_i), \text{succ}^2(v_i)\}\), \(f(\text{succ}(v_i)) = \{v_i, \text{succ}^2(v_i)\}\), and, \(f(\text{succ}^2(v_i)) = \{v_i, \text{succ}(v_i)\}\). This implies that \(C\) includes only 3 free vertices, which contradicts the definition of cyclic connected components. \(\square\)

From the above lemma, two adjacent pairs do not appear consecutively in a cyclic order. Hence, we immediately have the following corollary.

**Corollary 1:** Let \(C\) be a cyclic connected component, and let \((v_1, v_2, \ldots, v_k)\) be a cyclic order of the free vertices in \(C\). Then, \(\text{dist}(v_i, \text{succ}^2(v_i)) \geq 3\) holds.

Now, we are ready to prove the key lemma below.

**Lemma 3:** Let \(C\) be a cyclic connected component, and let \(k\) be the number of free vertices in \(C\). Then, the size of a maximum distance-3 independent set of \(C\) is \([k/2]\).

**Proof.** First, we show that one can construct a distance-3 independent set \(I_C\) with \(|I_C| = [k/2]\). Let \(S = \langle v_1, v_2, \ldots, v_k \rangle\) be a cyclic order of the free vertices in \(C\). From Corollary 1, \(\text{dist}(v_i, \text{succ}^2(v_i)) \geq 3\) holds for each \(i\). Hence, we can observe that the set \(\{v_{2i-1} | i = 1, 2, \ldots, [k/2]\}\) is a distance-3 independent set of \(C\).

Next, we prove that the size \([k/2]\) is the maximum. Let us assume for a contradiction that \(I'_C\) is a distance-3 independent set of \(C\) and \(|I'_C| > [k/2]\). Then, \(I'_C\) contains two consecutive free vertices on \(S\). Hence \(I'_C\) is not a distance-3 independent set of \(C\). \(\square\)

Now, we present our algorithm in **Algorithm 1** and **Algorithm 2**. **Algorithm 1** is the main routine ResMaxD3IS and **Algorithm 2** is the subroutine Helper. We are given an instance \((G, I, X)\), ResMaxD3IS \((G, I, X)\) returns the size of a maximum distance-3 independent set of \(G\) including \(I\) and excluding \(X\). Note that ResMaxD3IS \((G, \emptyset, \emptyset)\) returns the size of a maximum distance-3 independent set of \(G\). The
Algorithm 1: ResMaxD3IS(G, I, X)
1 begin
2 Repeat to apply Reduction D3IS. 1–4 exhaustively, and let (G’, I’, X’) be the obtained instance. Let F’ be the set of
3 free vertices of (G’, I’, X’).
4 if F’ = 0 then
5 return \[f]’\]
6 if \exists v \in F’ with \#f(v) = 0 then
7 return ResMaxD3IS(G’, I’ ∪ \{v\}, X’)
8 if \exists v \in F with \#f(v) = 1 then
9 return max [ResMaxD3IS(G’, I’ ∪ \{v\}, X’ ∪ \{u\}),
10 ResMaxD3IS(G’, I’ ∪ \{u\}, X’ ∪ N2(u))]
11 if \exists v \in F’ with \#f(v) ≥ 3 then
12 return max [ResMaxD3IS(G’, I’ ∪ \{v\}, X’ ∪ N2(v)),
13 ResMaxD3IS(G’, I’ ∪ \{v\}, X’ ∪ \{u\})]
14 if \forall v \in F’ with \#f(v) = 2 then
15 Let C1, C2, ..., Ck be the connected components of G’
16 each of which contains free vertices and vertices in X’.
17 return \[f]’ | \sum_{i=2}^{k} \text{helper}(C_i, I’ ∩ V(C_i), X’ ∩ V(C_i))
18 \{C_i \} is the set of the vertices in */

Algorithm 2: Helper(G, I, X)
1 begin
2 if G has 3 or less free vertices then
3 return \alpha(G, I, X) using an exhaustive search.
4 else /* G is a cyclic connected component. */
5 return \alpha(G, I, X)
6 end

algorithm is a branching algorithm. The algorithm repeats to
either select or discard each vertex in V. The set of selected
vertices is a distance-3 independent set of G and denoted by
I. The set of the discarded vertices is denoted by X.

First, the algorithm applies Reduction D3IS. Let (G’, I’, X’) be the obtained instance. Let I_M(G’, I’, X’) be a maximum distance-3 independent set for
(G’, I’, X’). Let us assume that there is a free vertex v with
\#f(v) = 0 in (G’, I’, X’). Then, v is always included in
I_M(G’, I’, X’). In this case, we always select v and recursiv-
ely call ResMaxD3IS(G’, I’ ∪ \{v\}, X’). Next, let us assume
that there is a free vertex v with \#f(v) = 1. Let u be the free
vertex in f(v). If we select v, then u is discarded into X. On
the other hand, if we discard v, we have to select u. Because
otherwise, v and u are both discarded, then we cannot ob-
tain a maximum distance-3 independent set for (G’, I’, X’).
In this case, the branching vector is (2, 2), since the number
of free vertices decreases by 2 in each case. Then, its
branching factor is \tau(2, 2) < 1.4143. Intuitively, this means
that the number of nodes in a search tree is bounded above
by \tau(2, 2)^n. (See [4] for the formal definitions of branch-
ning vectors and branching factors.) Next, let us assume that
there is a free vertex v with \#f(v) ≥ 3. If we select v, then v
is selected and at least 3 vertices in f(v) are discarded. If we
discard v, then v is inserted into X’. Hence, the branching
vector is (4, 1) and its branching factor is \tau(4, 1) < 1.3803.
Finally, let us assume that every vertex v holds \#f(v) = 2. In
this case, one can compute \alpha(G’, I’, X’) in polynomial time.
For each connected component C of G’, we compute the
size of a maximum distance-3 independent set of C using
Helper procedure. Let n_C be the number of free vertices in C.
Helper procedure calculates the size of a maximum
distance-3 independent set of C in an exhaustive manner if
n_C ≤ 3 holds. Otherwise, since C is a cyclic connected com-
ponent, the size of its maximum distance-3 independent set of C is [n_C/2] from Lemma 3. Hence, in this case, one can obtain \alpha(G’, I’, X’) in polynomial time.

Consequently, the worst case branching factor is
\tau(2, 2) < 1.4143. Therefore, we obtain the main theorem
below.

Theorem 1: One can solve MaxD3IS in O(1.4143^n) time.

By slightly modifying the algorithm, we also have the
following corollary.

Corollary 2: One can find a maximum distance-3 indepen-
dent set in O(1.4143^n) time.

3. Conclusions

We have designed an algorithm that calculates the size of a
maximum distance-3 independent set of a given graph in
O(1.4143^n) time, where n is the number of vertices. By
slightly modifying the algorithm, we can construct a max-
imum distance-3 independent set in O(1.4143^n) time.
Our algorithm uses a basic technique, called branching, for
designing exact algorithms. Our future work includes design-
ing more efficient exact algorithms using other techniques.
For example, we may use the measure and conquer tech-
nique.

References

[1] H. Eto, F. Guo, and E. Miyano, “Distance-d independent set prob-
lems for bipartite and chordal graphs,” J. Comb. Optim., vol.27, no.1,
pp.88–99, 2014.
[2] H. Eto, T. Ito, Z. Liu, and E. Miyano, “Approximability of the dis-
tance independent set problem on regular graphs and planar graphs,”
Proc. 10th International Conference on Combinatorial Optimization
and Applications, Lecture Notes in Computer Science, vol.10043,
pp.270–284, 2016.
[3] H. Eto, T. Ito, Z. Liu, and E. Miyano, “Approximation algorithm for
the distance-3 independent set problem on cubic graphs,” Proc. 11th
International Conference and Workshops on Algorithms and Compu-
tation, Lecture Notes in Computer Science, vol.10167, pp.228–240,
2017.
[4] F.V. Fomin and D. Kratsch, Exact Exponential Algorithms, Texts in
Theoretical Computer Science, An EATCS Series, Springer, 2010.