Collective Graph Exploration Parameterized by Vertex Cover

Siddharth Gupta
BITS Pilani, Goa Campus, India

Guy Sa'ar
Ben Gurion University of the Negev, Beersheba, Israel

Meirav Zehavi
Ben Gurion University of the Negev, Beersheba, Israel

Abstract

We initiate the study of the parameterized complexity of the Collective Graph Exploration (CGE) problem. In CGE, the input consists of an undirected connected graph G and a collection of k robots, initially placed at the same vertex r of G, and each one of them has an energy budget of B. The objective is to decide whether G can be *explored* by the k robots in B time steps, i.e., there exist k closed walks in G, one corresponding to each robot, such that every edge is covered by at least one walk, every walk starts and ends at the vertex r, and the maximum length of any walk is at most B. Unfortunately, this problem is NP-hard even on trees [Fraigniaud *et al.*, 2006]. Further, we prove that the problem remains W[1]-hard parameterized by k even for trees of treedepth 3. Due to the para-NP-hardness of the problem parameterized by treedepth, and motivated by real-world scenarios, we study the parameterized complexity of the problem parameterized by the vertex cover number (vc) of the graph, and prove that the problem is fixed-parameter tractable (FPT) parameterized by vc. Additionally, we study the optimization version of CGE, where we want to optimize B, and design an approximation algorithm with an additive approximation factor of $O(\mathsf{vc})$.

2012 ACM Subject Classification Theory of computation → Fixed parameter tractability; Theory of computation → Approximation algorithms analysis

Keywords and phrases Collective Graph Exploration, Parameterized Complexity, Approximation Algorithm, Vertex Cover, Treedepth

Digital Object Identifier 10.4230/LIPIcs.IPEC.2023.22

Related Version *Full Version*: https://arxiv.org/abs/2310.05480 [13]

Funding *Siddharth Gupta*: Supported by Engineering and Physical Sciences Research Council (EPSRC) grant EP/V007793/1.
Guy Sa'ar: Supported in part by the Israeli Smart Transportation Research Center and by the Lynne and William Frankel Center for Computing Science at Ben-Gurion University.
Meirav Zehavi: Supported by the European Research Council (ERC) grant titled PARAPATH.

1 Introduction

Collective Graph Exploration (CGE) is a well-studied problem in computer science and robotics, with various real-world applications such as network management and fault reporting, pickup and delivery services, searching a network, and so on. The problem is formulated as follows: given a set of robots (or agents) that are initially located at a vertex of an undirected graph, the objective is to explore the graph as quickly as possible and return to the initial vertex. A graph is *explored* if each of its edges is visited by at least one robot. In each time step, every robot may move along an edge that is incident to the vertex it is placed

at. The total time taken by a robot is the number of edges it traverses. The exploration time is the maximum time taken by any robot. In many real-world scenarios, the robots have limited energy resources, which motivates the minimization of the exploration time [6].

The CGE problem can be studied in two settings: *offline* and *online*. In the offline setting, the graph is known to the robots beforehand, while in the online setting, the graph is unknown and revealed incrementally as the robots explore it. While CGE has received considerable attention in the online setting, much less is known in the offline setting (Section 1.1). Furthermore, most of the existing results in the offline setting are restricted to trees. Therefore, in this paper, we investigate the CGE problem in the offline setting for general graphs, and present some approximation and parameterized algorithms with respect to the vertex cover number of the graph.

1.1 Related Works

As previously mentioned, the CGE problem is extensively studied in the online setting, where the input graph is unknown. As we study the problem in the offline setting in this paper, we only give a brief overview of the results in the online setting, followed by the results in the offline setting.

Recall that, in the online setting, the graph is unknown to the robots and the edges are revealed to a robot once the robot reaches a vertex incident to the edge. The usual approach to analyze any online algorithm is to compute its *competitive ratio*, which is the worst-case ratio between the cost of the online and the optimal offline algorithm. Therefore, the first algorithms for CGE focused on the competitive ratios of the algorithms. In [11], an algorithm for CGE for trees with competitive ratio $O(\frac{k}{\log k})$ was given. Later in [14], it was shown that this competitive ratio is tight. Another line of work studied the competitive ratio as a function of the vertices and the depth of the input tree [4, 7, 8, 10, 14, 19]. We refer the interested readers to a recent paper by Cosson *et al.* [5] and the references within for an in-depth discussion about the results in the online setting.

We now discuss the results in the offline setting. In [1], it was shown that the CGE problem for edge-weighted trees is NP-hard even for two robots. In [2, 18], an $(2 - 2/(k + 1))$-approximation was given for the optimization version of CGE for edge-weighted trees where we want to optimize B. In [11], the NP-hardness was shown for CGE for unweighted trees as well. In [9], a 2-approximation was given for the optimization version of CGE for unweighted trees where we want to optimize B. In the same paper, it was shown that the optimization version of the problem for unweighted trees is XP parameterized by the number of robots.

1.2 Our Contribution and Methods

In this paper, we initiate the study of the CGE problem for general unweighted graphs in the offline setting and obtain the following three results. We first prove that CGE is FPT parameterized by vc, where vc is the vertex cover number of the input graph. Specifically, we prove the following theorem.

▶ **Theorem 1.1.** CGE *is in FPT parameterized by* $\mathsf{vc}(G)$*, where* G *is the input graph.*

We then study the optimization version of CGE where we want to optimize B and design an approximation algorithm with an additive approximation factor of $O(\mathsf{vc})$. Specifically, we prove the following theorem.

▶ **Theorem 1.2.** *There exists an approximation algorithm for* CGE *that runs in time* $\mathcal{O}((|V(G)| + |E(G)|) \cdot k)$*, and returns a solution with an additive approximation of* $8 \cdot \mathsf{vc}(G)$*, where* G *is the input graph and* k *is the number of robots.*

Finally, we show a border of (in-)tractability by proving that CGE is W[1]-hard parametrized by k, even for trees of treedepth 3. Specifically, we prove the following theorem.

▶ **Theorem 1.3.** CGE *is W[1]-hard with respect to* k *even on trees whose treedepth is bounded by* 3.

We first give an equivalent formulation of CGE based on Eulerian cycles (see Lemma 3.4). We obtain the FPT result by using Integer Linear Programming (ILP). By exploiting the properties of vertex cover and the conditions given by our formulation, we show that a potential solution can be encoded by a set of variables whose size is bounded by a function of vertex cover.

To design the approximation algorithm, we give a greedy algorithm that satisfies the conditions given by our formulation. Again, by exploiting the properties of vertex cover, we show that we can satisfy the conditions of our formulation by making optimal decisions at the independent set vertices and using approximation only at the vertex cover vertices.

To prove the W-hardness, we give a reduction from a variant of Bin Packing, called Exact Bin Packing (defined in Section 2). We first prove that Exact Bin Packing is W[1]-hard even when the input is given in unary. We then give a reduction from this problem to CGE to obtain our result. Due to lack of space, several concepts and proofs are deferred to the full version of this paper [13].

1.3 Choice of Parameter

As mentioned in the previous section, we proved that CGE is W[1]-hard parameterized by k even on trees of treedepth 3. This implies that we cannot get an FPT algorithm parameterized by treedepth and k even on trees, unless FPT = W[1]. Thus, we study the problem parameterized by the vertex cover number of the input graph, a slightly weaker parameter than the treedepth.

Our choice of parameter is also inspired by several practical applications. For instance, consider a delivery network of a large company. The company has a few major distributors that receive the products from the company and can exchange them among themselves. There are also many minor distributors that obtain the products only from the major ones, as this is more cost-effective. The company employs k delivery persons who are responsible for delivering the products to all the distributors. The delivery persons have to start and end their routes at the company location. Since each delivery person has a maximum working time limit, the company wants to minimize the maximum delivery time among them. This problem can be modeled as an instance of CGE by constructing a graph G that has a vertex for the company and for each distributor and has an edge between every pair of vertices that correspond to locations that can be reached by a delivery person. The k robots represent the k delivery persons and are placed at the vertex corresponding to the company. Clearly, G has a small vertex cover, as the number of major distributors is much smaller than the total number of distributors.

For another real-world example where the vertex cover is small, suppose we want to cover all the streets of the city as fast as possible using k agents that start and end at a specific street. The city has a few long streets and many short streets that connect to them. This situation is common in many urban areas. We can represent this problem as an instance

of CGE by creating a graph G that has a vertex for each street and an edge between two vertices if the corresponding streets are adjacent. The k robots correspond to the k agents. Clearly, G has a small vertex cover, as the number of long streets is much smaller than the total number of streets.

2 Preliminaries

For $k \in \mathbb{N}$, let $[k]$ denote the set $\{1, 2, \ldots, k\}$. For a multigraph G, we denote the set of vertices of G and the multiset of edges of G by $V(G)$ and $E(G)$, respectively. For $u \in V(G)$, the *set of neighbors* of u in G is $\mathsf{N}_G(u) = \{v \in V \mid \{u, v\} \in E(G)\}$. When G is clear from the context, we refer to $\mathsf{N}_G(u)$ as $\mathsf{N}(u)$. The *multiset of neighbors* of u in G is the multiset $\widehat{\mathsf{N}}_G(u) = \{v \in V \mid \{u, v\} \in E(G)\}$ (with repetition). When G is clear from the context, we refer to $\widehat{\mathsf{N}}_G(u)$ as $\widehat{\mathsf{N}}(u)$. The *degree* of u in G is $|\widehat{\mathsf{N}}_G(u)|$ (including repetitions). Let $\widehat{E}$ be a multiset with elements from $E(G)$. Let $\mathsf{Graph}(\widehat{E})$ denote the multigraph $(V', \widehat{E})$, where $V' = \{u \mid \{u, v\} \in \widehat{E}\}$. A multigraph H is a *submultigraph* of a multigraph G if $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$. Let $V' \subseteq V(G)$. We denote the submultigraph induced by V' by $G[V']$, that is, $V(G[V']) = V'$ and $E(G[V']) = \{\{u, v\} \in E(G) \mid u, v \in V'\}$. Let $U \subseteq V(G)$. Let $G \setminus U$ denote the subgraph $G[V(G) \setminus U]$ of G.

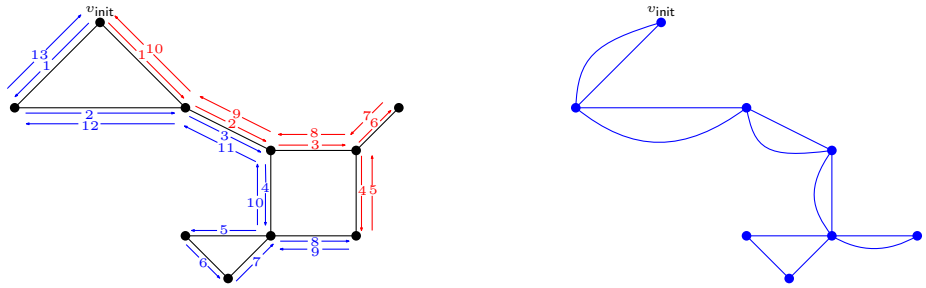
An *Eulerian cycle* in a multigraph $\widehat{G}$ is a cycle that visits every edge in $E(\widehat{G})$ exactly once. A *vertex cover* of G is $V' \subseteq V(G)$ such that for every $\{u, v\} \in E(G)$, at least one among u and v is in V'. The *vertex cover number* of G is $\mathsf{vc}(G) = \mathsf{min}\{|V'| \mid V' \text{ is a vertex cover of } G\}$. When G is clear from context, we refer to $\mathsf{vc}(G)$ as vc. A *path* P in G is $(v_0, \ldots, v_\ell)$, where (i) for every $0 \leq i \leq \ell$, $v_i \in V(G)$, and (ii) for every $0 \leq i \leq \ell - 1$, $\{v_i, v_{i+1}\} \in E(G)$ (we allow repeating vertices). The *length* of a path $P = (v_0, \ldots, v_\ell)$, denoted by $|P|$, is the number of edges in P (including repetitions), that is, ℓ. The set of vertices of P is $V(P) = \{v_0, \ldots, v_{\ell-1}\}$. The multiset of edges of P is $E(P) = \{\{v_i, v_{i+1}\} \mid 0 \leq i \leq \ell - 1\}$ (including repetitions). A *cycle* C in G is a path $(v_0, \ldots, v_\ell)$ such that $v_0 = v_\ell$. A *simple cycle* is a cycle $C = (v_0, \ldots, v_\ell)$ such that for every $0 \leq i < j \leq \ell - 1$, $v_i \neq v_j$. An *isomorphism* of a multigraph G into a multigraph G' is a bijection $\alpha : V(G) \to V(G')$, such that $\{u, v\}$ appears in $E(G)$ ℓ times if and only if $\{\alpha(u), \alpha(v)\}$ appears in $E(G')$ ℓ times, for an $\ell \in \mathbb{N}$. For a multiset A, we denote by 2^A the *power set* of A, that is, $2^A = \{B \mid B \subseteq A\}$. Let A and B be two multisets. Let $A \setminus B$ be the multiset $D \subseteq A$ such that every $d \in A$ appears exactly $\mathsf{max}\{0, d_A - d_B\}$ times in D, where d_A and d_B are the numbers of times d appears in A and B, respectively. A *permutation* of a multiset A is a bijection $\mathsf{Permut}_A : A \to [|A|]$.

▶ **Definition 2.1 (v_{init}-Robot Cycle).** *Let G be a graph, let $v_{\mathsf{init}} \in V(G)$. A* v_{init}-robot cycle *is a cycle $\mathsf{RC} = (v_0 = v_{\mathsf{init}}, v_1, v_2, \ldots, v_\ell = v_{\mathsf{init}})$ in G for some ℓ.*

When v_{init} is clear from the context, we refer to a v_{init}-robot cycle as a robot cycle.

▶ **Definition 2.2 (Solution).** *Let G be a graph, $v_{\mathsf{init}} \in V(G)$ and $k \in \mathbb{N}$. A* solution *for $(G, v_{\mathsf{init}}, k)$ is a set of k v_{init}-robot cycles $\{\mathsf{RC}_1, \ldots, \mathsf{RC}_k\}$ with $E(G) \subseteq E(\mathsf{RC}_1) \cup E(\mathsf{RC}_2) \cup \cdots \cup E(\mathsf{RC}_k)$. Its* value *is $\mathsf{val}(\{\mathsf{RC}_1, \ldots, \mathsf{RC}_k\}) = \mathsf{max}\{|E(\mathsf{RC}_1)|, |E(\mathsf{RC}_2)|, \ldots, |E(\mathsf{RC}_k)|\}$ (see Figure 1a for an illustration).*

▶ **Definition 2.3 (Collective Graph Exploration with k Agents).** *The* Collective Graph Exploration *(CGE) problem with k agents is: given a connected graph G, $v_{\mathsf{init}} \in V(G)$ and $k \in \mathbb{N}$, find the minimum B such that there exists a solution $\{\mathsf{RC}_1, \ldots, \mathsf{RC}_k\}$ where $\mathsf{val}(\{\mathsf{RC}_1, \ldots, \mathsf{RC}_k\}) = B$.*



(a) (b)

Figure 1 (a) An illustration of a graph G (drawn in black) and a solution for $(G, v_{\text{init}}, k = 2)$. The 2 robot cycles are shown by red and blue edges where the edge labels show the order in which the edges were covered by the respective robots. (b) The Robot Cycle-Graph for the robot cycle drawn in blue.

▶ **Definition 2.4 (Collective Graph Exploration with k Agents and Budget B).** *The* Collective Graph Exploration *(*CGE*) problem with k agents and budget B is: given a connected graph G, $v_{\text{init}} \in V(G)$ and $k, B \in \mathbb{N}$, find a solution $\{\mathsf{RC}_1, \ldots, \mathsf{RC}_k\}$ where $\mathsf{val}(\{\mathsf{RC}_1, \ldots, \mathsf{RC}_k\}) \leq B$, if such a solution exists; otherwise, return "no-instance".*

▶ **Definition 2.5 (Bin Packing).** *The* Bin Packing *problem is: given a finite set I of items, a size $s(i) \in \mathbb{N}$ for each $i \in I$, a positive integer B called bin capacity and a positive integer k, decide whether there is a partition of I into disjoint sets $I_1, \ldots, I_k$ such that for every $1 \leq j \leq k$, $\sum_{i \in I_j} s(i) \leq B$.*

▶ **Definition 2.6 (Exact Bin Packing).** *The* Exact Bin Packing *problem is: given a finite set I of items, a size $s(i) \in \mathbb{N}$ for each $i \in I$, a positive integer B called bin capacity and a positive integer k such that $\sum_{i \in I} s(i) = B \cdot k$, decide whether there is a partition of I into disjoint sets $I_1, \ldots, I_k$ such that for every $1 \leq j \leq k$, $\sum_{i \in I_j} s(i) = B$.*

▶ **Definition 2.7 (Integer Linear Programming).** *In the* Integer Linear Programming Feasibility *(ILP) problem, the input consists of t variables $x_1, x_2, \ldots, x_t$ and a set of m inequalities of the following form:*

$$\begin{aligned} a_{1,1}x_1 + a_{1,2}x_1 + \cdots + a_{1,p}x_t &\leq b_1 \\ a_{2,1}x_1 + a_{2,2}x_2 + \cdots + a_{2,p}x_t &\leq b_2 \\ \vdots \quad\quad \vdots \quad\quad\quad\quad \vdots \quad\quad \vdots & \\ a_{m,1}x_1 + a_{m,2}x_2 + \cdots + a_{m,p}x_t &\leq b_m \end{aligned}$$

where all coefficients $a_{i,j}$ and b_i are required to integers. The task is to check whether there exist integer values for every variable x_i so that all inequalities are satisfiable.

▶ **Theorem 2.8** ([12, 16, 17])**.** *An ILP instance of size m with t variables can be solved in time $t^{\mathcal{O}(t)} \cdot m^{\mathcal{O}(1)}$.*

3 Reinterpretation Based on Eulerian Cycles

Our approach to CGE with k agents is as follows. Let G be a connected graph, let $v_{\mathsf{init}} \in V(G)$ and let $k \in \mathbb{N}$. Let $\{\mathsf{RC}_1, \ldots, \mathsf{RC}_k\}$ be a solution, let $1 \leq i \leq k$ and denote $\mathsf{RC}_i = (v_0 = v_{\mathsf{init}}, v_1, v_2, \ldots, v_\ell = v_{\mathsf{init}})$ for some $\ell \in \mathbb{N}$. If we define a multiset $\widehat{E}_{\mathsf{RC}_i} = \{\{v_j, v_{j+1}\} \mid 0 \leq j \leq \ell - 1\}$, then, clearly, $\mathsf{RC}_i = (v_0 = v_{\mathsf{init}}, v_1, v_2, \ldots, v_\ell = v_{\mathsf{init}})$ is an Eulerian cycle in $\mathsf{Graph}(\widehat{E}_{\mathsf{RC}_i})$. We call this graph the RC_i*-graph* (see Figure 1b):

▶ **Definition 3.1 (Robot Cycle-Graph).** *Let G be a graph, let $v_{\mathsf{init}} \in V(G)$ and let $\mathsf{RC} = (v_0 = v_{\mathsf{init}}, v_1, v_2, \ldots, v_\ell = v_{\mathsf{init}})$ be a robot cycle. The* RC-graph*, denoted by* $\mathsf{Graph}(\mathsf{RC})$*, is the multigraph $\mathsf{Graph}(\widehat{E}_{\mathsf{RC}})$, where $\widehat{E}_{\mathsf{RC}} = \{\{v_i, v_{i+1}\} \mid 0 \leq i \leq \ell - 1\}$ is a multiset.*

▶ **Observation 3.2.** *Let G be a graph, let $v_{\mathsf{init}} \in V(G)$ and let $\mathsf{RC} = (v_0 = v_{\mathsf{init}}, v_1, v_2, \ldots, v_\ell = v_{\mathsf{init}})$ be a robot cycle. Then RC is an Eulerian cycle in $\mathsf{Graph}(\mathsf{RC})$.*

On the opposite direction, let $\widehat{E}$ be a multiset with elements from $E(G)$, and assume that $v_{\mathsf{init}} \in V(\mathsf{Graph}(\widehat{E}))$. Let $\mathsf{RC} = (v_0, v_1, v_2, \ldots, v_\ell = v_0)$ be an Eulerian cycle in $\mathsf{Graph}(\widehat{E})$ and assume, without loss of generality, that $v_0 = v_\ell = v_{\mathsf{init}}$. It is easy to see that RC is a robot cycle in G:

▶ **Observation 3.3.** *Let G be a graph, let $v_{\mathsf{init}} \in V(G)$, let $\widehat{E}$ be a multiset with elements from $E(G)$ and assume that $v_{\mathsf{init}} \in V(\mathsf{Graph}(\widehat{E}))$. Let $\mathsf{RC} = (v_0 = v_{\mathsf{init}}, v_1, v_2, \ldots, v_\ell = v_{\mathsf{init}})$ be an Eulerian cycle in $\mathsf{Graph}(\widehat{E})$. Then, RC is a robot cycle in G.*

From Observations 3.2 and 3.3, we get that finding a solution is equal to find k multisets $\widehat{E}_1, \ldots, \widehat{E}_k$ such that: (i) for every $1 \leq i \leq k$, $v_{\mathsf{init}} \in V(\mathsf{Graph}(\widehat{E}_i))$ (ii) for every $1 \leq i \leq k$, there exists an Eulerian cycle in $\mathsf{Graph}(\widehat{E}_i)$ and (iii) $E(G) \subseteq \widehat{E}_1 \cup \ldots \cup \widehat{E}_k$, that is, each $e \in E$ appears at least once in at least one of $\widehat{E}_1, \ldots, \widehat{E}_k$.

Recall that, in a multigraph $\widehat{G}$, there exists an Eulerian cycle if and only if $\widehat{G}$ is connected and each $v \in V(\widehat{G})$ has even degree in $\widehat{G}$ [3]. Thus, we have the following lemma:

▶ **Lemma 3.4.** *Let G be a connected graph, let $v_{\mathsf{init}} \in V(G)$ and let $k, B \in \mathbb{N}$. Then, $(G, v_{\mathsf{init}}, k, B)$ is a yes-instance of* CGE *if and only if there exist k multisets $\widehat{E}_1, \ldots, \widehat{E}_k$ with elements from $E(G)$, such that the following conditions hold:*

1. *For every $1 \leq i \leq k$, $v_{\mathsf{init}} \in V(\mathsf{Graph}(\widehat{E}_i))$.*
2. *For every $1 \leq i \leq k$, $\mathsf{Graph}(\widehat{E}_i)$ is connected, and every vertex in $\mathsf{Graph}(\widehat{E}_i)$ has even degree.*
3. *$E(G) \subseteq \widehat{E}_1 \cup \ldots \cup \widehat{E}_k$.*
4. *$\max\{|\widehat{E}_1|, \ldots, |\widehat{E}_k|\} \leq B$.*

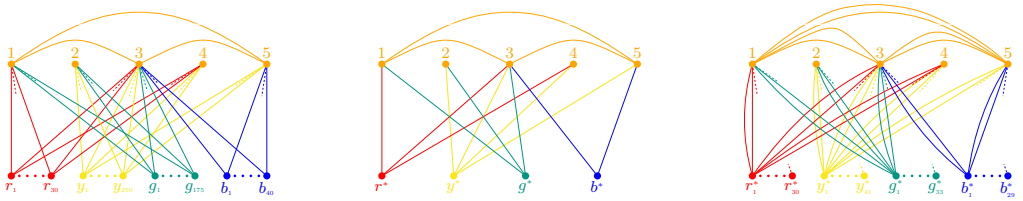
4 High-Level Overview

4.1 FPT Algorithm with Respect to Vertex Cover

Our algorithm is based on a reduction to the ILP problem. We aim to construct linear equations that verify the conditions in Lemma 3.4.

4.1.1 Encoding $\widehat{E}_i$ by a Valid Pair

First, we aim to satisfy the "local" conditions of Lemma 3.4 for each robot, that is, Conditions 1 and 2. Let us focus on the "harder" condition of the two, that is, Condition 2. We aim to encode any potential $\widehat{E}_i$ by smaller subsets whose union is $\widehat{E}_i$. In addition, we would like the



(a) G. **(b)** G^*. **(c)** $\overline{G}$.

Figure 2 An illustration of a graph G (in (a)), and its corresponding graphs G^* (in (b)) and $\overline{G}$ (in (c)). The vertex cover vertices and their edges are shown in orange. The 4 equivalence classes and their vertices are shown by red, yellow, green, and blue.

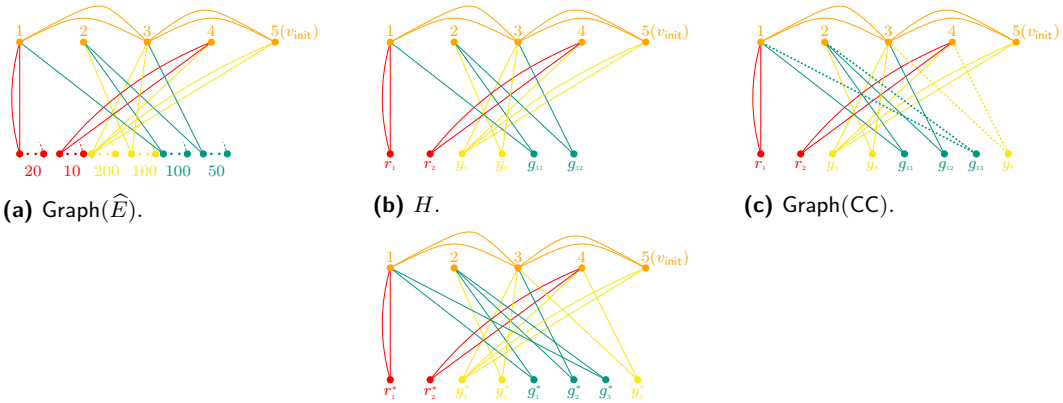
"reverse" direction as well: every collection of subsets that we will be able to unite must create some valid $\widehat{E}_i$. Note that we have two goals to achieve when uniting the subsets together: (i) derive a connected graph, where (ii) each vertex has even degree. In the light of this, the most natural encoding for the subsets are cycles, being the simplest graphs satisfying both aforementioned goals. Indeed, every cycle is connected, and a graph composed only of cycles is a graph where every vertex has even degree. Here, the difficulty is to maintain the connectivity of the composed graph. On the positive side, observe that every cycle in the input graph G has a non-empty intersection with any vertex cover VC of G. So, we deal with the connectivity requirement as follows. We seek for a graph $\overline{G}$ that is essentially (but not precisely) a subgraph of G that is (i) "small" enough, and (ii) for every valid $\widehat{E}_i$, there exists $\mathsf{CC} \subseteq E(\overline{G})$ such that $\mathsf{Graph}(\mathsf{CC})$ is a "submultigraph" of $\mathsf{Graph}(\widehat{E}_i)$, $\mathsf{Graph}(\mathsf{CC})$ is connected, and $V(\mathsf{Graph}(\mathsf{CC})) \cap \mathsf{VC} = V(\mathsf{Graph}(\widehat{E}_i)) \cap \mathsf{VC}$.

Equivalence Graph G^*. A first attempt to find such a graph is as follows. We define an equivalence relation on $V(G) \setminus \mathsf{VC}$ based on the sets of neighbors of the vertices in $V(G) \setminus \mathsf{VC}$ (see the 4 equivalence classes of the graph G in Figure 2a). We denote the set of equivalence classes induced by this equivalence relation by EQ. Then G^* is the graph defined as follows.

▶ **Definition 4.1 (Equivalence Graph G^*).** *Let G^* be the graph that: (i) contains VC, and the edges having both endpoints in VC, and (ii) where every equivalence class $u^* \in \mathsf{EQ}$ is represented by a single vertex adjacent to the neighbors of some $u \in u^*$ in G (which belong to VC). See Figure 2b.*

Unfortunately, this attempt fails, as we might need to use more than one vertex from the same $u^* \in \mathsf{EQ}$ in order to maintain the connectivity. E.g., see Figure 3b. If we delete r_2 and y_5, which are in the same equivalence class (in G) as r_1 and y_6, respectively, then the graph is no longer connected.

The Multigraph $\overline{G}$. So, consider the following second attempt. We use the aforementioned graph G^*, but instead of one vertex representing each $u^* \in \mathsf{EQ}$, we have $\mathsf{min}\{|u^*|, 2^{|\mathsf{N}_{G^*}(u^*)|}\}$ vertices. Observe that given a connected subgraph G' of G, and two vertices $u, u' \in u^*$ such that $\mathsf{N}_{G'}(u) = \mathsf{N}_{G'}(u')$, it holds that $G \setminus \{u'\}$ remains connected (e.g., see Figure 3a and 3b. The connectivity is still maintained even after deleting all but one vertex in the same equivalence class (in G) having same neighbourhood). Therefore, we have enough vertices for each $u^* \in \mathsf{EQ}$ in the graph, and its size is a function of $|\mathsf{VC}|$; so, we obtained the sought graph $\overline{G}$. Now, we would like to have an additional property for CC, which is that every vertex in $\mathsf{Graph}(\mathsf{CC})$ has even degree in it. To this end, we add to $\overline{G}$ more vertices for each



(a) Graph($\widehat{E}$). **(b)** H. **(c)** Graph(CC). **(d)** Graph(CC′).

Figure 3 The graphs shown here are with respect to the graph G shown in Figure 2a. An illustration of (a) a graph Graph($\widehat{E}$), (b) the graph H obtained by deleting all but one vertex from the same equivalence class in G and have the same neighbours in Graph($\widehat{E}$), (c) the graph Graph(CC) where CC is a skeleton of $\widehat{E}$ obtained from the graph in (b) by adding four more edges from Graph($\widehat{E}$) $\setminus H$, and (d) the graph Graph(CC′) where CC′ is the skeleton in $\overline{G}$ that is derived from the skeleton CC.

$u^* \in \mathsf{EQ}$. See Figure 3c. The vertex g_{13} having the same neighbours as g_{11} in H and being in the same equivalence class (in G) as g_{11} is added to make the degrees of 1 and 2 even. We have the following definition for $\overline{G}$.

▶ **Definition 4.2 (The Multigraph $\overline{G}$).** *Let $\overline{G}$ be the graph that: (i) contains* VC*, and the edges having both endpoints in* VC*, (ii) for every equivalence class $u^* \in \mathsf{EQ}$, there are exactly* $\min\{|u^*|, 2^{|\mathsf{N}_{G^*}(u^*)|} + |\mathsf{VC}|^2\}$ *vertices, adjacent to the neighbors of some $u \in u^*$ in G (which belong to* VC*), (iii) each edge in $\overline{G}$ appears exactly twice in $E(\overline{G})$ (for technical reasons). See Figure 2c.*

A Skeleton of $\widehat{E}_i$. We think of Graph(CC) as a "skeleton" of a potential $\widehat{E}_i$. By adding cycles with a vertex from $V(\mathsf{Graph}(\mathsf{CC})) \cap \mathsf{VC}$, we maintain the connectivity, and since every vertex in Graph(CC) has even degree, then by adding a cycle, this property is preserved as well. We have the following definition for a skeleton.

▶ **Definition 4.3 (A Skeleton CC).** *A* skeleton *of $\widehat{E}_i$ is* $\mathsf{CC} \subseteq \widehat{E}_i$ *such that: (i)* Graph(CC) *is a "submultigraph" of $\overline{G}$, (ii)* Graph(CC) *is connected, $v_{\text{init}} \in V(\mathsf{Graph}(\mathsf{CC}))$ and every vertex in* Graph(CC) *has even degree, and (iii) $V(\mathsf{Graph}(\mathsf{CC})) \cap \mathsf{VC} = V(\mathsf{Graph}(\widehat{E}_i)) \cap \mathsf{VC}$ (See Figure 3d).*

An $\widehat{E}_i$-Valid Pair. We prove that we might assume that the $\widehat{E}_i$'s are *nice multisets*, that is, a multiset where every element appears at most twice. We prove that every $\widehat{E}_i$ (assuming $\widehat{E}_i$ is nice) can be encoded by a skeleton CC (See Figure 3c.) and a multiset $\mathcal{C}$ of cycles (of length bounded by $2|\mathsf{VC}|$). We say that $(\mathsf{CC}, \mathcal{C})$ is an *$\widehat{E}$-valid pair*.

▶ **Definition 4.4 (A Valid Pair).** *A pair $(\mathsf{CC}, \mathcal{C})$, where* CC *is a skeleton of $\widehat{E}_i$ and $\mathcal{C}$ is a multiset of cycles in* Graph($\widehat{E}_i$)*, is an* $\widehat{E}_i$-valid pair *skeleton* CC *if:*

1. *The length of each cycle in $\mathcal{C}$ is bounded by $2|\mathsf{VC}|$.*
2. *At most $2|\mathsf{VC}|^2$ cycles in $\mathcal{C}$ have length other than 4.*
3. $\mathsf{CC} \cup \bigcup_{C \in \mathcal{C}} E(C) = \widehat{E}_i$ *(being two multisets).*

4.1.2 Robot and Cycle Types

Now, obviously, the number of different cycles in G (of length bounded by $2|\mathsf{VC}|$) is potentially huge. Fortunately, it is suffices to look at cycles in G^* in order to preserve Condition 2 of Lemma 3.4: assume that we have a connected $\mathsf{Graph}(\mathsf{CC})$ such that every vertex in $\mathsf{Graph}(\mathsf{CC})$ has even degree in it, and a multiset of cycles with a vertex from $V(\mathsf{Graph}(\mathsf{CC})) \cap \mathsf{VC}$ in G^*. By replacing each vertex that represents $u^* \in \mathsf{EQ}$ by any $u \in u^*$, the connectivity preserved, and the degree of each vertex is even.

Thus, each robot is associated with a *robot type* RobTyp, which includes a skeleton CC of the multiset $\widehat{E}_i$ associated with the robot (along other information discussed later). In order to preserve Condition 1 of Lemma 3.4, we also demand that $v_{\mathsf{init}} \in V(\mathsf{Graph}(\mathsf{CC}))$. Generally, for each type we define, we will have a variable that stands for the number of elements of that type. We are now ready to present our first equation of the ILP reduction:

Equation 1: Robot Type for Each Robot. In this equation, we ensure that the total sum of robots of the different robot types is exactly k, that is, there is exactly one robot type for each robot:

1. $\sum_{\mathsf{RobTyp} \in \mathsf{RobTypS}} x_{\mathsf{RobTyp}} = k.$

In addition, the other "pieces" of the "puzzle", that is, the cycles, are also represented by types: Each cycle C of length at most $2|\mathsf{VC}|$ in G^* is represented by a *cycle type*, of the form $\mathsf{CycTyp} = (C, \mathsf{RobTyp})$ (along other information discussed later), where RobTyp is a robot type that is "able to connect to C", that is, $V(\mathsf{Graph}(\mathsf{CC})) \cap \mathsf{VC} \cap V(C) \neq \emptyset$ for $\mathsf{RobTyp} = \mathsf{CC}$. Similarly, we will have equations for our other types.

Satisfying the Budget Restriction. Now, we aim to satisfy the budget condition (Condition 2 of Lemma 3.4), that is, for every $i \in [k]$, $|\widehat{E}_i| \leq B$. Let $i \in [k]$ and let $(\mathsf{CC}, \mathcal{C})$ be an $\widehat{E}$-valid pair. So, $\widehat{E}_i = \mathsf{CC} \cup (\bigcup_{C \in \mathcal{C}} E(C))$ (being a union of two multisets). Now, we prove that "most" of the cycles in $\mathcal{C}$ are of length 4, that is, for every $2 \leq j \leq 2|\mathsf{VC}|, j \neq 4$, the number of cycles of length j in $\mathcal{C}$ is bounded by $2|\mathsf{VC}|^2$. Therefore, we add to the definition of a robot type also the number of cycles of length exactly j, encoded by a vector $\mathsf{NumOfCyc} = (N_2, N_3, N_5, N_6, \ldots, N_{2|\mathsf{VC}|})$. So, for now, a robot type is $\mathsf{RobTyp} = (\mathsf{CC}, \mathsf{NumOfCyc})$. Thus, in order to satisfy the budget condition, we verify that the budget used by all the robots of a robot type $\mathsf{RobTyp} = (\mathsf{CC}, \mathsf{NumOfCyc})$, is as expected together. First, we ensure that the number of cycles of each length $2 \leq j \leq 2|\mathsf{VC}|, j \neq 4$, is exactly as the robot type demands, times the number of robots associated with this type, that is, $N_j \cdot x_{\mathsf{RobTyp}}$. So, we have the following equation:

Equation 5: Assigning the Exact Number of Cycles of Length Other Than 4 to Each Robot Type. We have the following notation: $\mathsf{CycTypS}(\mathsf{RobTyp}, j)$ is the set of cycle types for cycles of length j assigned to a robot of robot type RobTyp.

5. For every robot type $\mathsf{RobTyp} = (\mathsf{CC}, \mathsf{NumOfCyc})$ and for every $2 \leq j \leq 2|\mathsf{VC}|$, $j \neq 4$, $\sum_{\mathsf{CycTyp} \in \mathsf{CycTypS}(\mathsf{RobTyp}, j)} x_{\mathsf{CycTyp}} = N_j \cdot x_{\mathsf{RobTyp}}$, where $\mathsf{NumOfCyc} = (N_2, N_3, N_5, N_6, \ldots, N_{2|\mathsf{VC}|})$.

Observe that once this equation is satisfied, we are able to arbitrary allocate N_j cycles of length j to each robot of type RobTyp. So, in order to verify the budget limitation, we only need to deal with the cycles of length 4. Now, notice that the budget left for a robot of type

$\mathsf{RobTyp} = (\mathsf{CC}, \mathsf{NumOfCyc})$ for the cycles of length 4 is $B - (|\mathsf{CC}| + \sum_{2 \leq j \leq 2|\mathsf{VC}|, j \neq 4} N_j \cdot j)$, where $\mathsf{NumOfCyc} = (N_2, N_3, N_5, N_6, \ldots, N_{2|\mathsf{VC}|})$. Now, the maximum number of cycles we can add to a single robot of type RobTyp is the largest number which is a multiple of 4, that is less or equal to $B - \mathsf{Bud}(\mathsf{RobTyp})$. So, for every robot type RobTyp let $\mathsf{CycBud}(\mathsf{RobTyp}) = \lfloor (B - (|\mathsf{CC}| + \sum_{2 \leq j \leq 2|\mathsf{VC}|, j \neq 4} N_j \cdot j)) \cdot \frac{1}{4} \rfloor \cdot 4$. Notice that $\mathsf{CycBud}(\mathsf{RobTyp})$ is the budget left for the cycles of length 4. Thus, we have the following equation:

Equation 6: Verifying the Budget Limitation. This equation is defined as follows.

6. For every $\mathsf{RobTyp} \in \mathsf{RobTypS}$,

$$\sum_{\mathsf{CycTyp} \in \mathsf{CycTypS}(\mathsf{RobTyp}, 4)} 4 \cdot x_{\mathsf{CycTyp}} \leq x_{\mathsf{RobTyp}} \cdot \mathsf{CycBud}(\mathsf{RobTyp}).$$

By now, we have that there exist $\widehat{E}_1, \ldots, \widehat{E}_k$ that satisfy Conditions 1, 2 and 4 of Lemma 3.4 if and only if Equations 1, 5 and 6 can be satisfied.

Covering Edges with Both Endpoints in VC. Now, we aim to satisfy Condition 3 of Lemma 3.4, that is, we need to verify that every edge is covered by at least one robot. First, we deal with edges with both endpoints in VC. Here, for every $\{u, v\}$ such that $u, v \in \mathsf{VC}$, we just need to verify that at least one cycle or one of the CC's contains $\{u, v\}$. This we can easily solve by the following equation:

Equation 4: Covering Each Edge With Both Endpoints in VC. We have the following notations: For every $\{u, v\} \in E$ such that $u, v \in \mathsf{VC}$, (i) let $\mathsf{CycTypS}(\{u, v\})$ be the set of cycle types $\mathsf{CycTyp} = (C, \mathsf{RobTyp})$ where C covers $\{u, v\}$, and (ii) let $\mathsf{RobTypS}(\{u, v\})$ be the set of robot types $\mathsf{RobTyp} = (\mathsf{CC}, \mathsf{NumOfCyc})$ where CC covers $\{u, v\}$. In this equation, we ensure that each $\{u, v\} \in E(G)$ with both endpoints in VC is covered at least once:

4. For every $\{u, v\} \in E$ such that $u, v \in \mathsf{VC}$,

$$\sum_{\mathsf{CycTyp} \in \mathsf{CycTypS}(\{u,v\})} x_{\mathsf{CycTyp}} + \sum_{\mathsf{RobTyp} \in \mathsf{RobTypS}(\{u,v\})} x_{\mathsf{RobTyp}} \geq 1.$$

Let $\mathsf{RobTypS}$ be the set of the robot types, and let $\mathsf{CycTypS}$ be the set of cycle types.

Covering Edges with an Endpoint in $V(G) \setminus \mathsf{VC}$. Now, we aim to cover the edges from $E(G)$ with (exactly) one endpoint in $V(G) \setminus \mathsf{VC}$. Here, we need to work harder. Let x_z, for every $z \in \mathsf{RobTypS} \cup \mathsf{CycTypS}$, be values that satisfy Equations 1 and 4–6. As for now, we will arbitrary allocate cycles to robots according to their types. Then, we will replace every $u^* \in V(\mathsf{Graph}(\mathsf{CC}_i))$ and $u^* \in V(C)$, for every cycle C allocated to the i-th robot, by an arbitrary $u \in u^*$. Then, we will define $\widehat{E}_i$ as the union of edge set of the cycles and CC_i we obtained. We saw that due to Equations 1, 5 and 6, Conditions 1, 2 and 4 of Lemma 3.4 are satisfied. In addition, due to Equations 4, we ensure that each $\{u, v\} \in E(G)$ with both endpoints in VC is covered. The change we need to do in order to cover edges with an endpoint in $V(G) \setminus \mathsf{VC}$ is to make a smarter choices for the replacements of u^* vertices.

4.1.3 Vertex Type

Allocation of Multisets with Elements from $\mathsf{N}_{G^*}(u^*)$. Observe that each $u_j^* \in V(\mathsf{Graph}(\mathsf{CC}_i))$ that is replaced by some $u \in u^*$, covers the multiset of edges $\{\{u, v\} \mid v \in \widehat{\mathsf{N}}_{\mathsf{Graph}(\mathsf{CC}_i)}(u_j^*)\}$. In addition, every $u^* \in V(C)$ that is replaced by $u \in u^*$, covers the multiset of edges $\{\{u, v\}, \{u, v'\}\}$, where v and v' are the vertices right before and right after u in

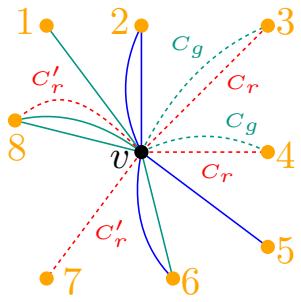


Figure 4 An illustration of the parts of a solution around an independent set vertex v. The three colors represent the parts of the multisets corresponding to three robots. The solid edges belong to the skeleton of the specific robot. The dashed edges belong to a cycle, labelled in the figure, of the multiset of the cycles corresponding to the specific robot. The vertex type of v derived from the solution shown in the figure is $(v^*, \{\{1,6,8,8\},\{3,4\},\{7,8\},\{2,2,5,6\}\})$, where $v \in v^* \in \mathsf{EQ}$.

C, respectively. Now, in order to cover every edge with an endpoint in $V(G) \setminus \mathsf{VC}$, we need to cover the set $\{\{u,v\} \mid v \in \mathsf{N}_{G^*}(u^*)\}$ for every $u \in u^* \in \mathsf{EQ}$. Therefore, we would like to ensure that the union of multisets of neighbors "allocated" for each u, when we replace some u^* by u, contains $\{\{u,v\} \mid v \in \mathsf{N}_{G^*}(u^*)\}$.

The Set NeiSubsets of Multisets Needed to Allocate to a Vertex. Now, the reverse direction holds as well: let $\widehat{E}_1, \ldots, \widehat{E}_k$ be multisets satisfying the conditions of Lemma 3.4, for every $i \in [k]$, let $(\mathsf{CC}_i, \mathcal{C}_i)$ be an $\widehat{E}_i$-valid pair, and let $u \in u^* \in \mathsf{EQ}$. Consider the following multisets (*): (i) for every $i \in [k]$ such that $u \in V(\mathsf{Graph}(\mathsf{CC}_i))$, the multiset $\widehat{\mathsf{N}}_{\mathsf{Graph}(\mathsf{CC}_i)}(u)$; (ii) for every $i \in [k]$ and $C \in \mathcal{C}_i$ and every appearance of u in C, the multiset $\{v, v'\}$, where v and v' are the vertices in C right before and right after the appearance of u. By Condition 3 of Lemma 3.4, every edge appears in at least one among $\widehat{E}_1, \ldots, \widehat{E}_k$. So, as for every $i \in [k]$, $\widehat{E}_i = \mathsf{CC}_i \bigcup_{C \in \mathcal{C}_i} E(C)$, the union of the multisets in (*) obviously contains $\mathsf{N}_{G^*}(u^*)$, e.g. see Figure 4. We would like to store the information of these potential multisets that ensures we covered $\mathsf{N}_{G^*}(u^*)$. The issue is that there might be a lot of multisets, as u might appear in many $\widehat{E}_i$'s. Clearly, it is sufficient to store one copy of each such multiset, as we only care that the union of the multisets contains $\mathsf{N}_{G^*}(u^*)$. Now, as we assume that $\widehat{E}_1, \ldots, \widehat{E}_k$ are nice multisets, each element in every multiset we derived appears at most twice in that multiset. In addition, since every edge in $E(\overline{G})$ appears at most twice, for each skeleton $\mathsf{CC} \subseteq E(\overline{G})$, each edge appears at most twice in $E(\mathsf{Graph}(\mathsf{CC}))$. So, for each $u_j^* \in V(\mathsf{Graph}(\mathsf{CC}))$ we replace by some $u \in u^*$, in the multiset of neighbors that are covered, every element appears at most twice. Moreover, since the degree is even, we have that the number of element in each multiset is even.

For a set A we define the multiset $A \times 2 = \{a, a \mid a \in A\}$. That is, each element in A appears exactly twice in $A \times 2$. Thus, we have the following definition for a vertex type.

▶ **Definition 4.5 (Vertex Type).** *Let G be a connected graph and let VC be a vertex cover of G. Let $u^* \in \mathsf{EQ}$ and let $\mathsf{NeiSubsets} \subseteq 2^{\mathsf{N}_{G^*}(u^*) \times 2}$. Then, $\mathsf{VerTyp} = (u^*, \mathsf{NeiSubsets})$ is a* vertex type *if for every $\mathsf{NeiSub} \in \mathsf{NeiSubsets}$, $|\mathsf{NeiSub}|$ is even, and $\mathsf{N}_{G^*}(u^*) \subseteq \bigcup \mathsf{NeiSubsets}$.*

Now, given $\widehat{E}_1, \ldots, \widehat{E}_k$ satisfying the conditions of Lemma 3.4, for every $i \in [k]$, an $\widehat{E}_i$-valid pair $(\mathsf{CC}_i, \mathcal{C}_i)$, and $u \in u^* \in \mathsf{EQ}$, we derive the vertex type of u as follows. We take the set $\mathsf{NeiSubsets}$ of multisets as described in (*). Clearly, $(u^*, \mathsf{NeiSubsets})$ is a vertex type.

For the reverse direction, we will use vertex type in order to cover the edges incident to each $u \in u^* \in \mathsf{EQ}$. Let $\mathsf{VerTypS}$ be the set of vertex types. We have a variable x_z for every $z \in \mathsf{VerTypS}$. First, each $u \in u^* \in \mathsf{EQ}$ is associated with exactly one vertex type $\mathsf{VerTyp} = (u^*, \mathsf{NeiSubsets})$, for some $\mathsf{NeiSubsets}$. To achieve this, we first ensure that for every $u^* \in \mathsf{EQ}$, the total sum of x_z for $z \in \mathsf{VerTypS}_{u^*}$, is exactly $|u^*|$, where $\mathsf{VerTypS}_{u^*} = \{(u^*, \mathsf{NeiSubsets}) \in \mathsf{VerTypS}\}$.

Equation 2: Vertex Type for Each Vertex. This equation is defined as follows.

2. For every $u^* \in \mathsf{EQ}$, $\sum_{\mathsf{VerTyp} \in \mathsf{VerTypS}_{u^*}} x_{\mathsf{VerTyp}} = |u^*|$

Given values for the variables that satisfy the equation, we arbitrary determine a vertex type $(u^*, \mathsf{NeiSubsets})$ for each $u \in u^*$, such that there are exactly x_{VerTyp} vertices of type VerTyp.

Allocation Functions of Multisets to Vertex Types. Now, let $u \in u^* \in \mathsf{EQ}$ of a vertex type $(u^*, \mathsf{NeiSubsets})$. We aim that when we do the replacements of u^*'s by vertices from u^*, each u gets an allocation of at least one of any of the multisets in $\mathsf{NeiSubsets}$. This ensures that we covered all of the edges adjacent to u. Instead of doing this for each $u \in u^* \in \mathsf{EQ}$, we will ensure that each $\mathsf{NeiSub} \in \mathsf{NeiSubsets}$ is allocated for vertices of type $(u^*, \mathsf{NeiSubsets})$ at least x_{VerTyp} times. To this end, we add more information for the robot types. For a robot type with a skeleton CC, recall that we replace each $u_j^* \in V(\mathsf{Graph(CC)})$ by some $u \in u^*$. The robot type also determines what is the vertex type of u that replaces u_j^*. In particular, we add to the robot type an allocation for each of $\{(u_j^*, \widehat{\mathsf{N}}_{\mathsf{Graph(CC)}}(u_j^*)) \mid u_j^* \in V(\mathsf{Graph(CC)})\}$, that is, a function $\mathsf{Alloc}_{\mathsf{Graph(CC)}}$ from this set into $\mathsf{VerTypS}$ (e.g., a robot of a robot type associated with the skeleton illustrated by Figure 3d, needs to allocate the pair $(r_1^*, \{1,1\})$, along with the other pairs shown in the figure). Observe that u_j^* is the vertex being replaced, and $\widehat{\mathsf{N}}_{\mathsf{Graph(CC)}}(u_j^*)$ is the multiset of neighbors that are covered. So, we demand that each $(u_j^*, \widehat{\mathsf{N}}_{\mathsf{Graph(CC)}}(u_j^*))$ is allocated to a vertex type $(u^*, \mathsf{NeiSubsets})$ that "wants" to get $\widehat{\mathsf{N}}_{\mathsf{Graph(CC)}}(u_j^*)$, that is, $\widehat{\mathsf{N}}_{\mathsf{Graph(CC)}}(u_j^*) \in \mathsf{NeiSubsets}$ (e.g, a robot of a robot type associated with the skeleton illustrated by Figure 3d, might allocate $(r_1^*, \{1,1\})$ to a vertex type $(r^*, \{\{1,1\}, \{3,4\}\})$). Now, we are ready to define a robot type as follows.

▶ **Definition 4.6 (Robot Type).** *A* robot type *is* $\mathsf{RobTyp} = (\mathsf{CC}, \mathsf{Alloc}_{\mathsf{Graph(CC)}}, \mathsf{NumOfCyc})$ *such that:*

1. $\mathsf{CC} \subseteq E(\overline{G})$.
2. $\mathsf{Graph(CC)}$ *is connected, every vertex in* $\mathsf{Graph(CC)}$ *has even degree and* $v_{\mathsf{init}} \in V(\mathsf{Graph(CC)})$.
3. $\mathsf{Alloc}_{\mathsf{Graph(CC)}}$ *is an allocation of* $\{(u_j^*, \widehat{\mathsf{N}}_{\mathsf{Graph(CC)}}(u_j^*)) \mid u_j^* \in V(\mathsf{Graph(CC)})\}$ *to vertex types.*
4. $\mathsf{NumOfCyc} = (N_2, N_3, N_5, N_6, \ldots, N_{2|\mathsf{VC}|})$, *where* $0 \leq N_i \leq 2|\mathsf{VC}|^2$ *for every* $2 \leq i \leq 2|\mathsf{VC}|$, $i \neq 4$.

Similarly, we add to a cycle type with a cycle C in G^* an allocation of the multiset $\{\{v, v'\} \mid u^* \in V(C)$, v and v' are the vertices appears right before and right after $u^*\}$ to vertex types (given by a function $\mathsf{PaAlloc}_C$). Now, we are ready to define a cycle type as follows.

▶ **Definition 4.7 (Cycle Type).** *Let* $C \in \mathsf{Cyc}_{G^*}$*, let* $\mathsf{PaAlloc}_C$ *be an allocation of* $\{\{v, v'\} \mid u^* \in V(C)$*,* v *and* v' *are the vertices appears right before and right after* $u^*\}$ *to vertex types, and let* $\mathsf{RobTyp} = (\mathsf{CC}, \mathsf{Alloc}_{\mathsf{Graph}(\mathsf{CC})}, \mathsf{NumOfCyc})$ *be a robot type. Then,* $\mathsf{CycTyp} = (C, \mathsf{PaAlloc}_C, \mathsf{RobTyp})$ *is a* cycle type *if* $V(\mathsf{Graph}(\mathsf{CC})) \cap V(C) \cap \mathsf{VC} \neq \emptyset$.

We have the following notations.

For every $\mathsf{VerTyp} = (u^*, \mathsf{NeiSubsets}) \in \mathsf{VerTypS}$, every $\mathsf{NeiSub} = \{v, v'\} \in \mathsf{NeiSubsets}$ and $1 \leq j \leq 2|\mathsf{VC}|$, $\mathsf{CycTypS}(\mathsf{VerTyp}, \mathsf{NeiSub}, j)$ is the set of cycle types that assign NeiSub to VerTyp exactly j times. For every $\mathsf{VerTyp} = (u^*, \mathsf{NeiSubsets}) \in \mathsf{VerTypS}$, every $\mathsf{NeiSub} \in \mathsf{NeiSubsets}$ and $1 \leq j \leq 2^{|\mathsf{VC}|} + |\mathsf{VC}|^2$, $\mathsf{RobTypS}(\mathsf{VerTyp}, \mathsf{NeiSub}, j)$ is the set of robot types that assign NeiSub to VerTyp exactly j times. Finally, we have the following equation:

Equation 3: Assigning Enough Subsets for Each Vertex Type. The equation is defined as follows.

3. For every $\mathsf{VerTyp} = (u^*, \mathsf{NeiSubsets}) \in \mathsf{VerTypS}$, and every $\mathsf{NeiSub} \in \mathsf{NeiSubsets}$,
$$\sum_{j=1}^{2|\mathsf{VC}|} \sum_{\mathsf{CycTyp} \in \mathsf{CycTypS}(\mathsf{VerTyp}, \mathsf{NeiSub}, j)} j \cdot x_{\mathsf{CycTyp}} + \sum_{j=1}^{2^{|\mathsf{VC}|}+|\mathsf{VC}|^2} \sum_{\mathsf{RobTyp} \in \mathsf{RobTypS}(\mathsf{VerTyp}, \mathsf{NeiSub}, j)} j \cdot x_{\mathsf{RobTyp}} \geq x_{\mathsf{VerTyp}}.$$

4.1.4 The Correctness of The Reduction

We denote the ILP instance associated with Equations 1–6 by $\mathsf{Reduction}(G, v_{\mathsf{init}}, k, B)$. Now, we give a proof sketch for the correctness of the reduction:

▶ **Lemma 4.8.** *Let* G *be a connected graph, let* $v_{\mathsf{init}} \in V(G)$ *and let* $k, B \in \mathbb{N}$*. Then,* $(G, v_{\mathsf{init}}, k, B)$ *is a yes-instance of* CGE*, if and only if* $\mathsf{Reduction}(G, v_{\mathsf{init}}, k, B)$ *is a yes-instance of the* Integer Linear Programming.

Proof. Let x_z, for every $z \in \mathsf{VerTypS} \cup \mathsf{RobTypS} \cup \mathsf{CycTypS}$, be values satisfying Equations 1–6. For every vertex type $\mathsf{VerTyp} = (u^*, \mathsf{NeiSubsets})$ and each $\mathsf{NeiSub} \in \mathsf{NeiSubsets}$, let $\mathsf{Alloc}(\mathsf{VerTyp}, \mathsf{NeiSub})$ be the set of every allocation of NeiSub to VerTyp by cycles or robots. We arbitrary allocate each element in $\mathsf{Alloc}(\mathsf{VerTyp}, \mathsf{NeiSub})$ to a vertex in u^*, such that every vertex $u \in u^*$ of type VerTyp gets at least one allocation. Due to Equation 3, we ensure we can do that. Then, we replace every $u^* \in V(\mathsf{Graph}(\mathsf{CC}_i))$ and every $u^* \in V(C)$ (for every $C \in \mathcal{C}_i$) by the $u \in u^*$ derived by the allocation. This ensures we covered every edge adjacent to a vertex in $V(G) \setminus \mathsf{VC}$. As seen in this overview, the other conditions of Lemma 3.4 hold.

For the reverse direction, let $\widehat{E}_1, \ldots, \widehat{E}_k$ be multisets satisfying the conditions of Lemma 3.4. For every $1 \leq i \leq k$ let $(\mathsf{CC}_i, \mathcal{C}_i)$ be an $\widehat{E}_i$-valid pair. Then, we first derive the vertex type of each $u \in V(G) \setminus \mathsf{VC}$, according to its equivalence class in EQ, and the set of multisets derived from $((\mathsf{CC}_i, \mathcal{C}_i))_{1 \leq i \leq k}$ (e.g. see Figure 4). Then, we derive the robot type $\mathsf{RobTyp} = (\mathsf{CC}, \mathsf{Alloc}_{\mathsf{Graph}(\mathsf{CC})}, \mathsf{NumOfCyc})$ for each $i \in [k]$: (i) the skeleton CC is determined by CC_i (e.g., see Figure 3d)), (ii) $\mathsf{NumOfCyc}$ is determined by the number of cycle of each length in $\mathcal{C}_i$ and (iii) the allocation of the multisets of $\mathsf{Graph}(\mathsf{CC}_i)$ is determined by the vertex types of $u \in V(\mathsf{Graph}(\mathsf{CC}_i)) \cap (V(G) \setminus \mathsf{VC})$ we have already computed. Then, for every $i \in [k]$ and every $C' \in \mathcal{C}_i$, we determine the cycle type $\mathsf{CycTyp} = (C, \mathsf{PaAlloc}_C, \mathsf{RobTyp})$ of C': (i) C is determined by C' (we replace each $u \in u^* \in \mathsf{EQ}$ in C' by u^*), (ii) RobTyp is the robot type of i we have already computed, and (iii) $\mathsf{PaAlloc}_C$ is determined by the

vertex types of $u \in V(C') \cap (V(G) \setminus \mathsf{VC})$ we have already computed. Then, for every $z \in \mathsf{VerTypS} \cup \mathsf{RobTypS} \cup \mathsf{CycTypS}$, we define x_z to be the number of elements of type z. As seen in this overview, the values of the variables satisfy Equations 1–6. ◀

Observe that the number of variables is bounded by a function of $|\mathsf{VC}|$, so we will get an FPT runtime with respect to vc. Thus, we conclude the correction of Theorem 1.1.

4.2 Approximation Algorithm with Additive Error of $\mathcal{O}(\mathsf{vc})$

Our algorithm is based on a greedy approach. Recall that, our new goal (from Lemma 3.4) is to find k multisets $\widehat{E}_1, \ldots, \widehat{E}_k$ such that for every $1 \leq i \leq k$, $v_{\mathsf{init}} \in V(\mathsf{Graph}(\widehat{E}_i))$, $\mathsf{Graph}(\widehat{E}_i)$ is connected and each $u \in V(\mathsf{Graph}(\widehat{E}_i))$ has even degree in $\mathsf{Graph}(\widehat{E}_i)$. Now, assume that we have a vertex cover VC of G such that $G[\mathsf{VC}]$ is connected and $v_{\mathsf{init}} \in \mathsf{VC}$, and let $I = V \setminus \mathsf{VC}$. We first make the degree of every vertex in I even in G, by duplicating an arbitrary edge for vertices having odd degree. Observe that, after these operations, G may be a multigraph.

We initialize $\widehat{E}_1, \ldots, \widehat{E}_k$ with k empty sets. We partition the set of edges of G with one endpoint in I in the following manner. We choose the next multiset from $\widehat{E}_1, \ldots, \widehat{E}_k$ in a round-robin fashion and put a pair of edges, not considered so far, incident to some vertex $v \in I$, in the multiset. This ensures that the degree of every vertex in I is even in each multiset. Let $\widehat{E}'_1, \ldots, \widehat{E}'_k$ be multisets satisfying the conditions of Lemma 3.4. Then, due to Condition 2 of Lemma 3.4, the degree of every vertex is even in every multiset $\widehat{E}'_i$. Thus, the total number of edges (with repetition) incident to any vertex in $\mathsf{Graph}(\widehat{E}'_1 \cup \ldots \cup \widehat{E}'_k)$ is even. Therefore, there must be at least one additional repetition for at least one edge of every vertex with odd degree in G. So, adding an additional edge to each vertex with odd degree is "must" and it does not "exceed" the optimal budget. Then, we partition the edges with both endpoints in VC, in a balanced fashion, as follows. We choose an edge, not considered so far, and add it to a multiset with minimum size.

Observe that, after this step, we have that: i) every edge of the input graph belongs to at least one of the multisets $\widehat{E}_i$, ii) the degree of each vertex of I in each multiset is even, and iii) we have not exceeded the optimal budget. We still need to ensure that i) $\mathsf{Graph}(\widehat{E}_i)$ is connected, for every $i \in [k]$, ii) the degree of each vertex of VC in each multiset is even, and iii) $v_{\mathsf{init}} \in V(\mathsf{Graph}(\widehat{E}_i))$ for every $i \in [k]$. Next, we add a spanning tree of $G[\mathsf{VC}]$ to each of the $\widehat{E}_i$, in order to make $\mathsf{Graph}(\widehat{E}_i)$ connected and to ensure that $v_{\mathsf{init}} \in V(\mathsf{Graph}(\widehat{E}_i))$. Lastly, we add at most $|\mathsf{VC}|$ edges, with both endpoints in VC, to every $\widehat{E}_i$ in order to make the degree of each $u \in \mathsf{VC}$ even in each of the multiset. Observe that the multisets $\widehat{E}_1, \ldots, \widehat{E}_k$ satisfy the conditions of Lemma 3.4. Moreover, we added at most $\mathcal{O}(|\mathsf{VC}|)$ additional edges to each $\widehat{E}_i$, comparing to an optimal solution.

References

1 Igor Averbakh and Oded Berman. A heuristic with worst-case analysis for minimax routing of two travelling salesmen on a tree. *Discret. Appl. Math.*, 68(1-2):17–32, 1996. doi:10.1016/0166-218X(95)00054-U.

2 Igor Averbakh and Oded Berman. (p - 1)/(p + 1)-approximate algorithms for p-traveling salesmen problems on a tree with minmax objective. *Discret. Appl. Math.*, 75(3):201–216, 1997. doi:10.1016/S0166-218X(97)89161-5.

3 Béla Bollobás. *Modern graph theory*, volume 184. Springer Science & Business Media, 1998.

4 Peter Brass, Flavio Cabrera-Mora, Andrea Gasparri, and Jizhong Xiao. Multirobot tree and graph exploration. *IEEE Trans. Robotics*, 27(4):707–717, 2011. doi:10.1109/TRO.2011.2121170.

5 Romain Cosson, Laurent Massoulié, and Laurent Viennot. Breadth-first depth-next: Optimal collaborative exploration of trees with low diameter. *CoRR*, abs/2301.13307, 2023. doi:10.48550/arXiv.2301.13307.

6 Shantanu Das, Dariusz Dereniowski, and Christina Karousatou. Collaborative exploration of trees by energy-constrained mobile robots. *Theory Comput. Syst.*, 62(5):1223–1240, 2018. doi:10.1007/s00224-017-9816-3.

7 Dariusz Dereniowski, Yann Disser, Adrian Kosowski, Dominik Pajak, and Przemyslaw Uznanski. Fast collaborative graph exploration. *Inf. Comput.*, 243:37–49, 2015. doi:10.1016/j.ic.2014.12.005.

8 Yann Disser, Frank Mousset, Andreas Noever, Nemanja Skoric, and Angelika Steger. A general lower bound for collaborative tree exploration. *Theor. Comput. Sci.*, 811:70–78, 2020. doi:10.1016/j.tcs.2018.03.006.

9 Miroslaw Dynia, Miroslaw Korzeniowski, and Christian Schindelhauer. Power-aware collective tree exploration. In Werner Grass, Bernhard Sick, and Klaus Waldschmidt, editors, *Architecture of Computing Systems - ARCS 2006, 19th International Conference, Frankfurt/Main, Germany, March 13-16, 2006, Proceedings*, volume 3894 of *Lecture Notes in Computer Science*, pages 341–351. Springer, 2006. doi:10.1007/11682127_24.

10 Miroslaw Dynia, Jaroslaw Kutylowski, Friedhelm Meyer auf der Heide, and Christian Schindelhauer. Smart robot teams exploring sparse trees. In Rastislav Kralovic and Pawel Urzyczyn, editors, *Mathematical Foundations of Computer Science 2006, 31st International Symposium, MFCS 2006, Stará Lesná, Slovakia, August 28-September 1, 2006, Proceedings*, volume 4162 of *Lecture Notes in Computer Science*, pages 327–338. Springer, 2006. doi:10.1007/11821069_29.

11 Pierre Fraigniaud, Leszek Gasieniec, Dariusz R. Kowalski, and Andrzej Pelc. Collective tree exploration. *Networks*, 48(3):166–177, 2006. doi:10.1002/net.20127.

12 András Frank and Éva Tardos. An application of simultaneous diophantine approximation in combinatorial optimization. *Combinatorica*, 7(1):49–65, 1987. doi:10.1007/BF02579200.

13 Siddharth Gupta, Guy Sa'ar, and Meirav Zehavi. Collective graph exploration parameterized by vertex cover, 2023. arXiv:2310.05480.

14 Yuya Higashikawa, Naoki Katoh, Stefan Langerman, and Shin-ichi Tanigawa. Online graph exploration algorithms for cycles and trees by multiple searchers. *J. Comb. Optim.*, 28(2):480–495, 2014. doi:10.1007/s10878-012-9571-y.

15 Klaus Jansen, Stefan Kratsch, Dániel Marx, and Ildikó Schlotter. Bin packing with fixed number of bins revisited. *J. Comput. Syst. Sci.*, 79(1):39–49, 2013. doi:10.1016/j.jcss.2012.04.004.

16 Hendrik W. Lenstra Jr. Integer programming with a fixed number of variables. *Math. Oper. Res.*, 8(4):538–548, 1983. doi:10.1287/moor.8.4.538.

17 Ravi Kannan. Minkowski's convex body theorem and integer programming. *Math. Oper. Res.*, 12(3):415–440, 1987. doi:10.1287/moor.12.3.415.

18 Hiroshi Nagamochi and Kohei Okada. A faster 2-approximation algorithm for the minmax p-traveling salesmen problem on a tree. *Discret. Appl. Math.*, 140(1-3):103–114, 2004. doi:10.1016/j.dam.2003.06.001.

19 Christian Ortolf and Christian Schindelhauer. A recursive approach to multi-robot exploration of trees. In Magnús M. Halldórsson, editor, *Structural Information and Communication Complexity - 21st International Colloquium, SIROCCO 2014, Takayama, Japan, July 23-25, 2014. Proceedings*, volume 8576 of *Lecture Notes in Computer Science*, pages 343–354. Springer, 2014. doi:10.1007/978-3-319-09620-9_26.

A W[1]-Hardness for CGE

In this section, we aim to prove the following theorem:

▶ **Theorem 1.3.** CGE *is W[1]-hard with respect to k even on trees whose treedepth is bounded by* 3.

We prove Theorem 1.3 by showing a reduction from Exact Bin Packing (see Definition 2.6).

First, we show that unary Exact Bin Packing is W[1]-hard with respect to k. It is known that unary Bin Packing is W[1]-hard with respect to k [15]. So, we give a reduction from Bin Packing to Exact Bin Packing in order to prove the following lemma:

▶ **Lemma A.1.** *Unary* Exact Bin Packing *is W[1]-hard with respect to* k.

Proof. Let (I, s, B, k) be an instance of Bin Packing problem. Let $t = B \cdot k - \sum_{i \in I} s(i)$ and let $s' : I \cup \{i_1, \ldots, i_t\} \to \mathbb{N}$ be a function defined as follows. For every $i \in I$, $s'(i) = s(i)$, and for every $i_\ell \in \{i_1, \ldots, i_t\}$, $s'(i_\ell) = 1$. Observe that $(I \cup \{i_1, \ldots, i_t\}, s', B, k)$ is an instance of Exact Bin Packing. We show that (I, s, B, k) is a yes-instance of Bin Packing if and only if $(I \cup \{i_1, \ldots, i_t\}, s', B, k)$ is a yes-instance of Exact Bin Packing.

Assume that (I, s, B, k) is a yes-instance of Bin Packing. Let $I_1, \ldots, I_k$ be a partition of I into disjoint sets such that for every $1 \leq j \leq k$, $\sum_{i \in I_j} s(i) \leq B$. For every $1 \leq j \leq k$, let $t_j = B - \sum_{i \in I_j} s(i)$. Let $I'_1, \ldots, I'_k$ be a partition of $\{i_1, \ldots, i_t\}$ into k disjoint sets such that for every $1 \leq j \leq k$, $|I'_j| = t_j$. Observe that there exists such a partition since $\sum_{1 \leq j \leq k} t_j = t$. Clearly, $I_1 \cup I'_1, \ldots, I_k \cup I'_k$ is a partition of $I \cup \{i_1, \ldots, i_t\}$ into disjoint sets such that for every $1 \leq j \leq k$, $\sum_{i \in I_j \cup I'_j} s(i) = B$. Therefore, $(I \cup \{i_1, \ldots, i_t\}, s', B, k)$ is a yes-instance of Exact Bin Packing.

Now, assume that $(I \cup \{i_1, \ldots, i_t\}, s', B, k)$ is a yes-instance of Exact Bin Packing. Let $I_1, \ldots, I_k$ be a partition of $I \cup \{i_1, \ldots, i_t\}$ into disjoint sets such that for every $1 \leq j \leq k$, $\sum_{i \in I_j} s(i) = B$. Observe that $I_1 \setminus \{i_1, \ldots, i_t\}, \ldots, I_k \setminus \{i_1, \ldots, i_t\}$ is a partition of I into disjoint sets such that for every $1 \leq j \leq k$, $\sum_{i \in I_j} s(i) \leq B$. Therefore, (I, s, B, k) is a yes-instance of Bin Packing.

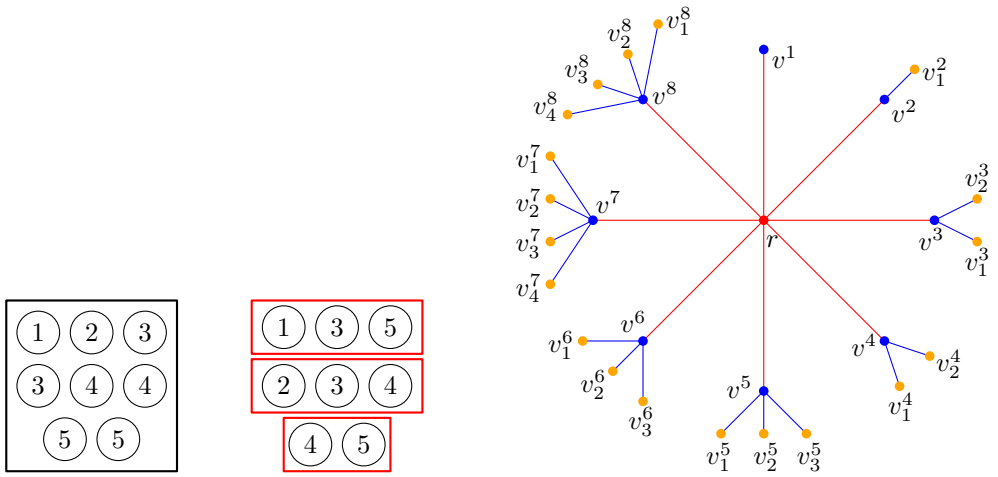
Clearly, the reduction works in polynomial time when the input is in unary. Thus, since unary Bin Packing is W[1]-hard with respect to k [15], unary Exact Bin Packing is W[1]-hard with respect to k. ◀

A.1 Reduction From Exact Bin Packing to CGE

Given an instance (I, s, B, k) of Exact Bin Packing problem, denote by $\mathsf{BinToRob}(I, s, B, k)$ the instance of CGE defined as follows. First, we construct the graph T as follows. For each $i \in I$ we create a star with $s(i) - 1$ leaves. We connect each such star with an edge to a vertex r. Formally, $V(T) = \{v^i, v^i_1 \ldots, v^i_{s(i)-1} \mid i \in I\} \cup \{r\}$ and $E(T) = \{\{v^i, v^i_j\} \mid i \in I, 1 \leq j \leq s(i) - 1\} \cup \{\{r, v_i\} \mid i \in I\}$. Now, we define $\mathsf{BinToRob}(I, s, B, k) = (T, r, k, 2B)$. See Figure 5 for an example. Next, we prove the correctness of the reduction:

▶ **Lemma A.2.** *Let* (I, s, B, k) *be an instance of* Exact Bin Packing. *Then,* (I, s, B, k) *is a yes-instance if and only if* $\mathsf{BinToRob}(I, s, B, k)$ *is a yes-instance of* CGE.

Proof. First, assume that (I, s, B, k) is a yes-instance. Let $I_1, \ldots, I_k$ be a partition of I into disjoint sets such that for every $1 \leq j \leq k$, $\sum_{i \in I_j} s(i) = B$. We prove that $\mathsf{BinToRob}(I, s, B, k) = (T, r, k, 2B)$ is a yes-instance of CGE, by showing that there exist k multisets $\widehat{E}_1, \ldots, \widehat{E}_k$ such that the conditions of Lemma 3.4 are satisfied. For every $1 \leq j \leq k$, let $\widehat{E}_j = \{\{v^i, v^i_t\}, \{v^i, v^i_t\} \mid \ i \in I_j, 1 \leq t \leq s(i) - 1\} \cup \{\{v^i, r\}, \{v^i, r\}\}$. Clearly, $r \in V(\mathsf{Graph}(\widehat{E}_j))$, $\mathsf{Graph}(\widehat{E}_j)$ is connected, and every vertex in $\mathsf{Graph}(\widehat{E}_j)$ has even degree. Therefore, Conditions 1 and 2 are satisfied. In addition, since $I = I_1 \cup \ldots \cup I_k$, we have that $E \subseteq \widehat{E}_1 \cup \ldots \cup \widehat{E}_k$, so Condition 3 is satisfied. Now, for every $1 \leq j \leq k$,



(a) **(b)**

Figure 5 An illustration of a Exact Bin Packing instance, a solution (in sub-figure (a)) and the equivalent instance of CGE constructed by the BinToRob function (in sub-figure (b)).

$|\widehat{E}_j| = |\{\{v^i, v^i_t\}, \{v^i, v^i_t\} \mid i \in I_j, 1 \leq t \leq s(i)-1\} \cup \{\{v^i, r\}, \{v^i, r\}\}| = \sum_{i \in I_j} 2(s(i)-1) + \sum_{i \in I_k} 2 = 2\sum_{i \in I_k} s(i) = 2B$. Thus, Condition 4 is satisfied. Therefore, all the conditions of Lemma 3.4 are satisfied, so $\mathsf{BinToRob}(I, s, B, k)$ is a yes-instance of CGE.

Now, we prove the reverse direction. Assume that $\mathsf{BinToRob}(I, s, B, k) = (T, r, k, 2B)$ is a yes-instance of CGE. From Lemma 3.4, there exist k multisets $\widehat{E}_1, \ldots, \widehat{E}_k$ such that the conditions of Lemma 3.4 hold. Let $1 \leq j \leq k$. We first show that every $\{u, v\} \in \widehat{E}_j$ appears at least twice in $\widehat{E}_j$. Let $\{u, v\} \in \widehat{E}_j$. We the following two cases:

Case 1: $\{u, v\} = \{v^i, v^i_t\}$ for some $i \in I$ and $1 \leq t \leq s(i) - 1$. From Condition 2 of Lemma 3.4, v^i_t has even degree in $\mathsf{Graph}(\widehat{E}_j)$. Since $\{v^i, v^i_t\}$ is the only edge having v^i_t as an endpoint in T, $\{v^i, v^i_t\}$ appears an even number of times in $\widehat{E}_j$, and so it appears at least twice in $\widehat{E}_j$.

Case 2: $\{u, v\} = \{v^i, r\}$ for some $i \in I$. From Condition 2 of Lemma 3.4, v^i has even degree in $\mathsf{Graph}(\widehat{E}_j)$. From Case 1, each $\{v^i, v^i_t\} \in \widehat{E}_j$ appears an even number of times in $\widehat{E}_j$. Therefore, since r is the only neighbor of v^i other than v^i_t, $1 \leq t \leq s(i) - 1$, $\{v^i, r\}$ appears an even number of times, which is greater or equal to 2, in $\widehat{E}_j$.

Now, observe that $|E(T)| = \sum_{i \in I} s(i) = B \cdot k$, and from Condition 4 of Lemma 3.4, $\sum_{1 \leq j \leq k} |\widehat{E}_j| \leq 2B \cdot k$. In addition, from Condition 3 of Lemma 3.4, (1): $E(T) \subseteq \widehat{E}_1 \cup \ldots \cup \widehat{E}_k$. So, since we have already proved that for every $1 \leq j \leq k$, each $\{u, v\} \in \widehat{E}_j$ appears at least twice in $\widehat{E}_j$, we get that for every $1 \leq j < j' \leq k$, $\widehat{E}_j \cap \widehat{E}_{j'} = \emptyset$, and $\sum_{1 \leq \ell \leq k} |\widehat{E}_\ell| = 2B \cdot k$; in turn, for every $1 \leq j \leq k$, $|\widehat{E}_j| = 2B$, and each $\{u, v\} \in \widehat{E}_j$ appears exactly twice in $\widehat{E}_j$. Moreover, from Conditions 1 and 2 of Lemma 3.4, for every $1 \leq j \leq k$, $r \in V(\mathsf{Graph}(\widehat{E}_j))$ and $\mathsf{Graph}(\widehat{E}_j)$ is connected. Therefore, for every $1 \leq j \leq k$ and $i \in I$, if $v^i \in V(\mathsf{Graph}(\widehat{E}_j))$ then $\{\{v^i, v^i_t\} \mid 1 \leq t \leq s(i) - 1\} \cup \{r, v^i\} \subseteq \widehat{E}_j$. Thus, for every $1 \leq j < j' \leq k$, (2): $V(\mathsf{Graph}(\widehat{E}_j)) \cap V(\mathsf{Graph}(\widehat{E}_{j'})) = \{r\}$.

Now, for every $1 \leq j \leq k$, let $I_j = \{i \in I \mid v^i \in V(\mathsf{Graph}(\widehat{E}_j))\}$. By (1) and (2), $I_1, \ldots, I_k$ is a partition of I into disjoint sets. We show, that for every $1 \leq j \leq k$, $\sum_{i \in I_j} s(i) = B$. Let $1 \leq j \leq k$. Then, $\sum_{i \in I_j} s(i) = \sum_{i \in I_j} |\{\{v_i, v_{i_t}\} \mid 1 \leq t \leq s(i) - 1\} \cup \{r, v_i\}| = \frac{1}{2}|\widehat{E}'_{j'}| = \frac{1}{2} \cdot 2 \cdot B = B$. Therefore $I_1, \ldots, I_k$ is a solution for (I, s, B, k), so (I, s, B, k) is a yes-instance of Exact Bin Packing problem. This ends the proof. ◀

Clearly, the reduction works in polynomial time when the input is in unary. In addition, observe that the treedepth of the tree, obtained by the reduction, is bounded by 3. Now, recall that, by Lemma A.1, unary Exact Bin Packing is W[1]-hard with respect to k. Thus, we conclude from Lemma A.2 the correctness of Theorem 1.3.