Budgeted Interactive Property Graph Repair with GNN

Anonymous Author(s)

CCS Concepts

• **Do Not Use This Code → Generate the Correct Terms for Your Paper**; *Generate the Correct Terms for Your Paper*; Generate the Correct Terms for Your Paper; Generate the Correct Terms for Your Paper.

Keywords

Do, Not, Use, This, Code, Put, the, Correct, Terms, for, Your, Paper

ACM Reference Format:
Anonymous Author(s). 2018. Budgeted Interactive Property Graph Repair with GNN. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation email (Conference acronym 'XX).* ACM, New York, NY, USA, 4 pages. https://doi.org/XXXXXXX.XXXXXXX

1 NP-Hardness proof

Definition 1.1 (Budget-Constrained GRDG Covering Problem). Given a GRDG graph $\mathcal{G} = (V, F)$ with node difficulties D_v and a set of users $U = \{u_1, \ldots, u_N\}$, each user u_i having a skill value K_i, cognitive capacity CC_i, and cost c_i, and a total budget $B \in \mathbb{R}_{\geq 0}$, find a subset of users $U^* \subseteq U$ and an assignment of disjoint subgraphs $\{P_i \subseteq V \mid u_i \in U^*\}$ such that:

$$U^* = \arg\max_{U' \subseteq U} \sum_{u_i \in U'} \sum_{v \in P_i} EQ(D_v, K_i) \tag{1}$$

subject to:

$$\text{Each } P_i \text{ is connected,} \tag{2}$$

$$\sum_{v \in P_i} D_v \leq CC_i \quad \forall u_i \in U', \tag{3}$$

$$P_i \cap P_j = \emptyset \quad \forall u_i \neq u_j, \tag{4}$$

$$\bigcup_{u_i \in U'} P_i = V, \tag{5}$$

$$\sum_{u_i \in U'} c_i \leq B. \tag{6}$$

where $EQ(D_v, K_i)$ is a component-wise monotonic function defined in Section ??, Equation (??), estimating the expected quality of a user u_i with skill level K_i on a violation with difficulty D_v.

The above formulation captures the key aspects of the violation assignment task. Constraint (2) ensures that each user is assigned a connected subgraph of violations, preserving local context and facilitating coherent resolution. Constraint (3) guarantees that the total difficulty of assigned violations does not exceed the user's cognitive capacity, reflecting individual workload limits. Constraint (4) enforces disjointness in the assignment, ensuring each violation is handled by at most one user. Constraint (5) ensures that the assignment covers all necessary violations. Finally, constraint (6) is the budget constraint.

Theorem 1.2. *The Budget-Constrained GRDG Covering Problem (bud-GCP) is NP-hard.*

Proof. We reduce the classical Multiple Knapsack problem (MKP) to bud-GCP. An instance of MKP consists of items $I = \{1, \ldots, n\}$, where item i has weight $w_i \geq 0$ and profit $p_i \geq 0$, and knapsacks $M = \{1, \ldots, m\}$, where knapsack j has capacity $C_j \geq 0$. In the decision version of MKP, given a threshold P, the question is whether there exists a packing of each item into at most one knapsack respecting the capacity constraints such that the total profit of the packing is at least P.

Given an instance $\mathcal{I}$ of MKP, we create an instance $\mathcal{J}$ of bud-GCP as follows: create a node v_i for each item $i \in I$ and set $V = \{v_1, \ldots, v_n\}$. Construct a complete graph $G = (V, F)$.

For each node v_i assign the difficulty $D_{v_i} := w_i$. Create a user u_j for each knapsack $j \in M$ and set its cognitive capacity to $CC_j := C_j$. We also create a dummy user u_0 with very large capacity, say $CC_0 := \sum_{i \in I} w_i$. Set $K_j = 1$ for $j \geq 1$ and $K_0 = \varepsilon \in (0, 1)$. Set the user costs $c_i := 1$, $i \geq 0$ and set the budget to $B := m + 1$.

Define the expected-quality function as

$$EQ(D_{v_i}, K_j) = \begin{cases} p_i, & \text{if } K_j = 1, \\ 0, & \text{if } K_j < 1. \end{cases}$$

It is easy to verify that this definition is component-wise monotone and that this reduction is polynomial time in n and m.

We claim that $\mathcal{I}$ is a YES-instance of MKP, i.e., it admits a valid packing of items into knapsacks with total profit at least P subject to budget B iff $\mathcal{J}$ admits a valid assignment of users to violations that covers the entire graph, whose total quality is at least P and the total cost is at most B.

Let $\pi : I \rightharpoonup M$ be a partial function denoting the packing of a subset of items I into the m knapsacks for the MKP instance $\mathcal{I}$. We can construct a corresponding solution to the bud-GCP instance $\mathcal{J}$: assign violation node v_i to user u_j whenever $\pi(i) = j$; for each unassigned item i, assign the violation v_i to the dummy user u_0, i.e., assign a violation v_i to the dummy user u_0 whenever $\pi(i)$ is undefined. Let P_j denote the set of violations assigned to user u_j. Since G is complete, each assigned part P_j is connected, so (2) holds. The cognitive-capacity constraints (3) hold since $\sum_{v \in P_j} D_v = \sum_{i \in P_j} w_i \leq C_j = CC_j$, $j \in \{1, \ldots, m\}$. Disjointness (4) follows from the fact that each item is placed in at most one knapsack and thus each violation is assigned to at most one user. By construction, the assignment covers all violations, satisfying (5). The budget constraint (6) holds by our choice of costs and B. By construction, the bud-GCP objective value equals the MKP profit P.

Conversely, consider a feasible solution for the instance $\mathcal{J}$ of bud-GCP with quality Q and cost $\leq B$. We obtain a feasible MKP item packing as follows: for each user u_j, $j \geq 1$, place item i into knapsack j iff $v_i \in P_j$, i.e., $\pi(i) = j, \forall v_i \in P_j$. Constraints (3) and (4) guarantee that the knapsack capacities are respected and that no item appears in more than one knapsack. Violations assigned to the dummy user, if any, correspond to items left unassigned in MKP. Removing the unassigned nodes doesn't reduce the profit of the item packing since the dummy user contributes 0 quality (and thus 0 profit). Thus, the items assigned to non-dummy users form a feasible MKP packing whose total profit is Q. The theorem follows. □

2 Time Complexity

Let $|V|$ be the number of nodes of the GRDG, $|F|$ the number of edges, $|U|$ the number of users, $C = |C_i|$ the size of candidates subgraphs per user and T the number of iterations of the algorithm.

Complexity of BuildCandidatesSet. Let Δ be the maximum degree of the GRDG. For user u_i, denote by CC_i the capacity and by D_v the node difficulties. Let $X_i := \min\left\{|V|, \left\lfloor \frac{CC_i}{\min_{v \in V} D_v} \right\rfloor\right\}$ be an upper bound on the number of nodes a candidate subgraphs for u_i can contain. Assume we start from S_i seeds of violations per user and run t_i iterations of 1-exchange local search (add/drop/swap one node) per seed, keeping at most $C_i \leq S_i$ candidate subraphs. The greedy growth step performs at most X_i insertions, checking at most $O(X_i\Delta)$ neighboring nodes to select the one with maximal gain (cost $O(\log n)$), giving $T_{\text{greedy}}(i) = O\big(X_i\,\Delta\,\log n\big)$. One step of local search scans a 1-exchange neighborhood of size $O(X_i\Delta)$ with cached deltas; over t_i iterations this is $T(i) = O\big(t_i\,X_i\,\Delta\big)$. Hence the per-seed cost is $T_{\text{seed}}(i) = O\big(X_i\,\Delta\,(\log n + t_i)\big)$. With S_i seeds, the complexity for finding the candidate subgraphs for a user is $T_{\text{user}}(i) = S_i \cdot T_{\text{seed}}(i) = O\Big(S_i\,X_i\,\Delta\,(\log n + t_i)\Big)$,

Complexity of SubModularAssignment. For each user i, let C_i be the set of candidate subgraphs produced by BuildCandidatesSet. The total candidates for all the users are $M = \sum_{i=1}^{N} |C_i|$. and the total nodes accross the candidates are $Y = \sum_{i=1}^{N} \sum_{P \in C_i} |P|$. We can leverage an inverted index from each node v for the list of candidate items (i, P) that contain v (of size $O(M)$). Let $c_{\min} = \min_i c_i$ be the minimum cost among the users, and let X_{sel} be the number of selected candidates; clearly $X_{\text{sel}} \leq \min\big\{M, |U|, \lfloor B/c_{\min} \rfloor\big\}$.

Computing initial gains $\sum_{v \in P} EQ(D_v, K_i)$ for all (i, P) and building the inverted index costs $O(S)$.

At each iteration of the greedy loop, we extract the current best feasible item in $O(\log M)$ time. Upon choosing $(i^\star, P^\star)$ we (i) remove all other patches of user $i^\star$ (at most $|C_{i^\star}|$ items overall across the run), and (ii) invalidate all candidates that overlap $P^\star$. Using the inverted index, step (ii) takes $O\big(\sum_{v \in P^\star} \Gamma_v\big)$ time, where Γ_v is the number of candidate items containing node v. Because subgraphs are enforced disjoint, each node is invalidated at most once; hence across the whole run $\sum_{\text{selected } P^\star} \sum_{v \in P^\star} \Gamma_v = \sum_v \Gamma_v = Y$. The total number of operations on the lists of candidates is $O(M + X_{\text{sel}})$, so heap work is $O\big((M + X_{\text{sel}}) \log M\big)$.

Summing up, $T_{\text{assignment}} = O\big(Y + (M + X_{\text{sel}}) \log M\big)$. Under uniform parameters ($|C_i| = L$ candidates per user, each of size at most X), we have $M = |U|L$ and $X \leq |U|LX$, giving

$$T_{\text{assignment}} = O\big(|U|LX + (|U|L + X_{\text{sel}}) \log(|U|L)\big).$$

Complexity of the Final Outer Loop. Let T be the maximum number of Lagrangian iterations. The remaining outer-loop work per iteration is lightweight: (i) computing the current primal value Z is $O\big(\sum_{i \in U'} |P_i|\big) \leq O(|V|)$, (ii) the dual quantity UB_t is $O(1)$ once Z and $\sum_{i \in U'} c_i$ are known, (iii) the subgradient update and projection of λ are $O(1)$, and (iv) updating the incumbent $(U^\star, \{P_i^\star\})$ is $O(1)$.

Hence the worst-case total time (without early stopping) is $T_{\text{total}} = T \cdot \Big(|U| \cdot T_{\text{user}} + T_{\text{assignmnet}} + O(|V|)\Big)$. Therefore, T_{total} is

$$O\Big(T \cdot \big(|U|\,S\,X\,\Delta\,(\log|V| + t) + |U|LX + (|U|L + X_{\text{sel}}) \log(|U|L) + |V|\big)\Big).$$

In practice the loop halts at some $T' \leq T$ when either the dual–primal gap $UB - Z^\star$ drops below tolerance or the price λ stabilizes; then T above can be replaced by T'.

3 Balancing the human involvement

In this experiment, we analyzed the influence of GNN involvement and the available budget on the final repair quality.

We vary θ from 0 (the GNN performs all the repairs) to 1 (only humans repair the graph). The budget depends on the dataset, and it varies from 0 (no human can be involved), to the amount necessary to involve all the humans in the repair of all the violations.

Figure 1 illustrates the results. Each graph shows the F1 score as a function of the budget, with different lines corresponding to the employed θ values (e.g., `LARA-0.1` corresponds to $\theta = 0.1$). The reported F1 score is computed as the average between the scores obtained by the GNN and the users. The dashed lines represent the values of θ such that the GNN repairs all the violations.

We can observe a common behavior across the different dataset where the charts are divided in four regions. First, there is the *unfeasible* area, where there are missing points. It corresponds to settings where the budget and the GNN involvement are not enough to assign all the violations. This occurs rarely and only at higher level of θ as more humans need to be involved (e.g., the first two or three steps for `LARA-0.8`). Second, there is the *feasible* region, where given the available budget, all the violations are assigned but the more expert (and costly) users cannot be involved. Then, there is the *inflection point*, that correspond to the amount of budget needed to make the assignment resulting the highest quality (i.e. we can afford to involve high skilled users). Finally, there is the *plateau* area, corresponding to cases where augmenting the budget does not increase the quality of the repairs.

For instance, let us consider the case of `LARA-0.8` applied to the `FAERS` dataset (Figure 1(a)). When the budget is set to 0 or 80, the assignment algorithm is unable to allocate a sufficient number of human participants to repair all the violations; this situation corresponds to the *unfeasible* region. When the budget increases to values between 240 and 720, the graph can be repaired; however, not all highly skilled users are involved in the process. As a consequence, the F1 score, remains around 0.7, which characterizes the *feasible* region. The *inflection point* is reached at a budget of 960, where the involvement of additional skilled users results in a improvement of the repair quality, leading to an F1 score of 0.92. Beyond this point, further increases in the budget do not produce significant gains.

It's interesting to observe that for all the datasets, the plateau is reached remaining under the 70% of the total maximum budget. In

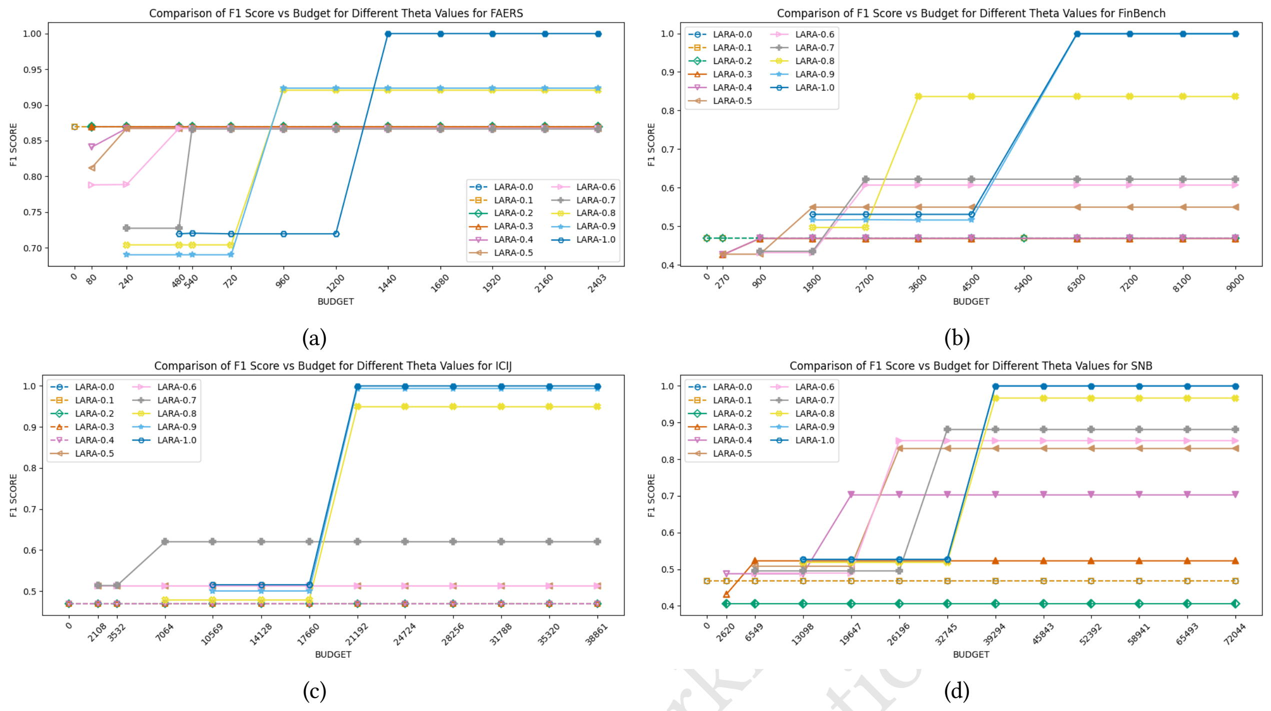

Figure 1: F1-Score of our approach using different θ across different datasets at different budget values.

fact, for F1 scores > 0.8 the the plateau is reached within three or four steps before the max budget. On `ICIJ`, LARA exhibits sharper inflection points: with $\theta < 0.6$ fewer than two users are involved, but higher θ drastically increases user demand (2 at `0.6` vs. 263 at `0.7`), leading to a sharp quality rise for $\theta \geq 0.8$ (Table ??).

Notice that some values of θ result in a flat line. This typically occurs for lower values (e.g., `LARA-0.2` on `FinBench` in Figure 1(b)), where the violations predominantly repaired with GNN-Repairs. Because the GNN already fixes the majority, assigning the remaining violations to experts has little impact on overall quality.

In general, for all the datasets, $\theta \in \{0.8, 0.9\}$ represent the best trade-off between automated repairs and users intervention, suggesting that selecting a θ close to the average difficulty value results in a good quality/cost balance.

Finally, for a fixed budget, different θ values yield solutions with varying quality and certified approximation guarantees. To combine them, run LARA concurrently across multiple θ values, apply each instance's stopping rule, and select the one that returns the certified solution. This parallel strategy does not increase the algorithm's asymptotic complexity.

4 Constraints

4.1 LDBC Finbench

An account cannot transfer money if it is blocked

$$\phi_1 := (Transfer(X,Y), X.isBLocked = True \longrightarrow \bot) \tag{7}$$

A guarantor needs to have more than 5M in the bank account

$$\phi_2 := (Guarantee(X,Y) \wedge Own(Y,Z), \tag{8}$$
$$Z.balance < 500000000 \wedge id(X) \neq id(Y) \longrightarrow \bot) \tag{9}$$

A person cannot be guarantor of themself

$$\phi_3 := (Guarantee(X,Y), id(X) = id(Y) \longrightarrow \bot) \tag{10}$$

Interest Rates for company need to be greater than 0.2

$$\phi_4 := (Apply(X,Y), Y.interestRate < 0.2 \longrightarrow \bot) \tag{11}$$

4.2 LDBC SNB

A person cannot know themself

$$\phi_1 := (Knows(X,Y), id(X) = id(Y) \longrightarrow \bot) \tag{12}$$

Two people that know each other need to live in the same city

$$\phi_2 := (Knows(X,Y), id(X) <> id(Y) \tag{13}$$
$$\wedge X.LocationCityId \neq Y.LocationCityId \longrightarrow \bot) \tag{14}$$

4.3 FAERS (Real)

A drug cannot be primary suspect and secondary suspect of the same case

$$\phi_1 := (IS\_PRIMARY\_SUSPECT(X,Y) \tag{15}$$
$$\wedge IS\_SECONDARY\_SUSPECT(Y,Z), \tag{16}$$
$$id(X) = id(Y) \longrightarrow \bot) \tag{17}$$

A drug cannot be primary suspect of itself

$$\phi_1 := (IS\_PRIMARY\_SUSPECT(X, Y) \tag{18}$$
$$id(X) = id(Y) \rightarrow \perp) \tag{19}$$

A case cannot fall under two different age groups

$$\phi_3 := (FALLS\_UNDER(X, Y) \wedge FALLS\_UNDER(X, Z), \tag{20}$$
$$id(Z) \neq id(Y) \rightarrow \perp) \tag{21}$$

A drug cannot be prescribed to a child

$$\phi_3 := (PRESCRIBED(X, Y) \wedge RECEIVED(Y, Z) \tag{22}$$
$$\wedge\ FALLS\_UNDER(Z, W) \tag{23}$$
$$W.ageGroup = \text{"}Child\text{"} \rightarrow \perp) \tag{24}$$

4.4 ICIJ (Real)

An entity cannot be the same as itself

$$\phi_1 := (same\_as(X, Y), id(X) = id(Y) \rightarrow \perp) \tag{25}$$

The registered address of an entity must be the same as the one of the address

$$\phi_2 := (registered\_address(X, Y), \tag{26}$$
$$X.country\_codes \neq Y.country\_codes \rightarrow \perp) \tag{27}$$

References

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009