GEDLIB: A C++ Library for Graph Edit Distance Computation

Presented at GbRPR 2019, Tours, France, June 19–21, 2019

David B. Blumenthal, Johann Gamper, Sébastien Bougleux, and Luc Brun

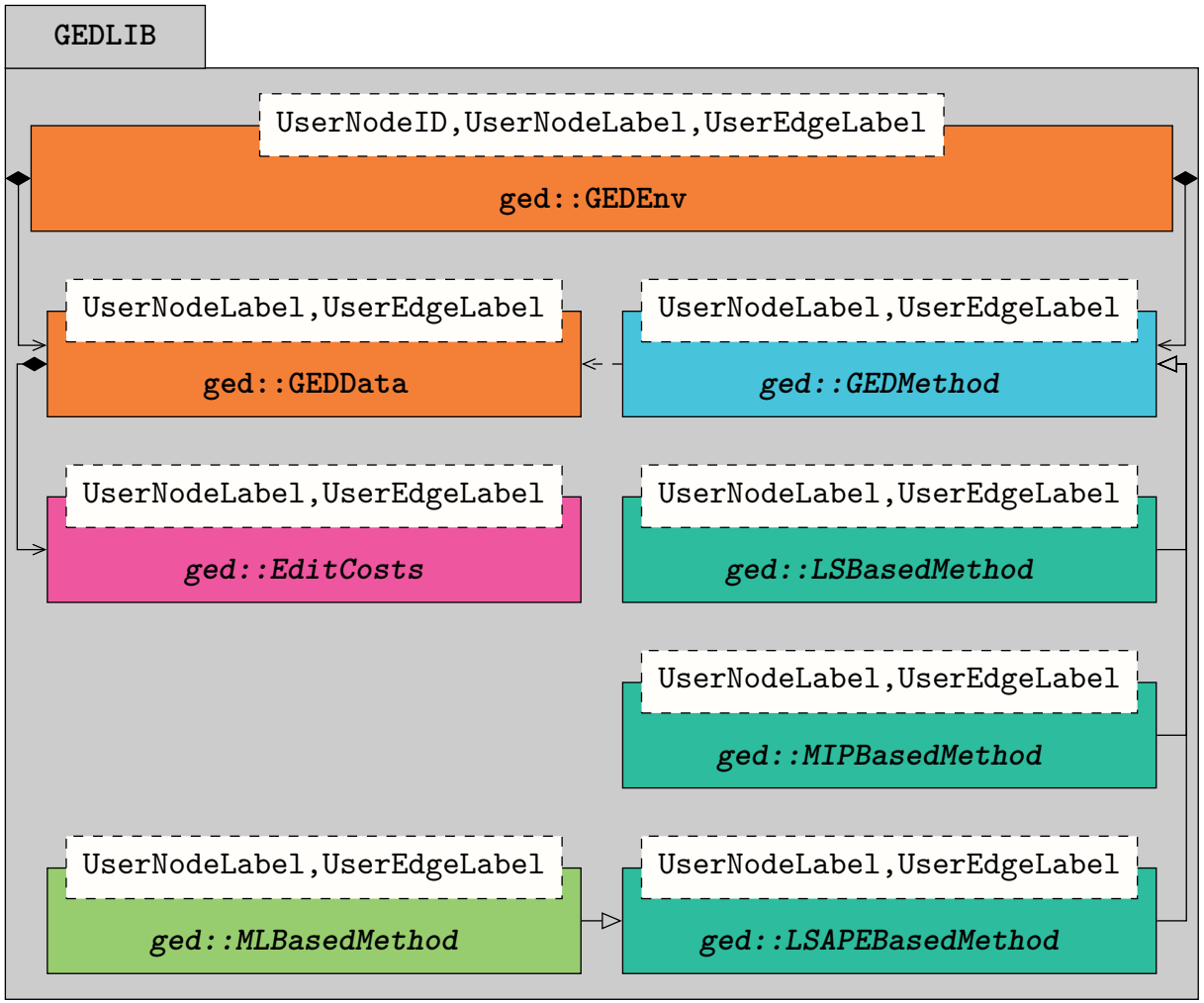
June 20, 2019

A Very Short Overview

Most Important Facts

- C++ library for computing GED between arbitrarily labeled, undirected graphs
- around 30 different algorithms are implemented in GEDLIB
- edit costs are available for IAM Graph Database [8], Graph Data Repository for Graph Edit Distance [1], and GREYC's Chemistry Database
- allows easy implementation of new algorithms and edit costs
- has been used for several publications, e. g.: D. B. Blumenthal, N. Boria, J. Gamper, S. Bougleux, and L. Brun, "Comparing heuristics for graph edit distance computation", *VLDB J.*, 2019, in press
- available on GitHub: `https://github.com/dbblumenthal/gedlib`

Architecture



Using GEDLIB and Implementing New Edit Costs

User Interface

- `ged::GEDEnv<UserNodeID,UserNodeLabel,UserEdgeLabel>`: use this class to load graphs, set edit costs, run methods, obtain results, etc.
- more details in demo later in this presentation

Abstract Class for Implementing New Edit Costs

- `ged::EditCosts<UserNodeLabel,UserEdgeLabel>`: generic interface
- implement node and edge edit cost functions for label types `UserNodeLabel` and `UserEdgeLabel`

Implementing new Algorithms

- `ged::GEDMethod<UserNodeLabel,UserEdgeLabel>`: generic interface
- `ged::LSBasedMethod<UserNodeLabel,UserEdgeLabel>`: interface for methods based on local search
- `ged::MIPBasedMethod<UserNodeLabel,UserEdgeLabel>`: interface for methods based on (mixed integer) linear programming
- `ged::LSAPEBasedMethod<UserNodeLabel,UserEdgeLabel>`: interface for methods based on transformations to the linear sum assignment problem with error-correction (LSAPE)
- `ged::MLBasedMethod<UserNodeLabel,UserEdgeLabel>`: interface for LSAPE based methods that employ machine learning to construct their LSAPE instances

A GEDLIB Implementation of Median Graph Computation for Letter (H) Graphs

Median Graph Computation

Problem Definition

- given: finite collection of graphs $\mathcal{G}$ from domain $\mathbb{G}$
- task: find $G \in \mathbb{G}$ that minimizes $\sum_{H \in \mathcal{G}} \mathrm{GED}(G, H)$

Employed Approach

- paper: N. Boria, S. Bougleux, B. Gaüzère, and L. Brun, "Generalized median graph via iterative alternate minimizations", in *GbRPR*, 2019, pp. 99–109
- algorithm: compute set median $\tilde{G} \in \mathcal{G}$ with (close to) optimal node maps $\pi^H \in \Pi(\tilde{G}, H)$ for all $H \in \mathcal{G}$; then iterate the following steps until convergence:
  1. optimize median $\tilde{G}$, keeping node maps π^H fixed
  2. optimize node maps π^H, keeping median $\tilde{G}$ fixed

The Letter (H) Dataset

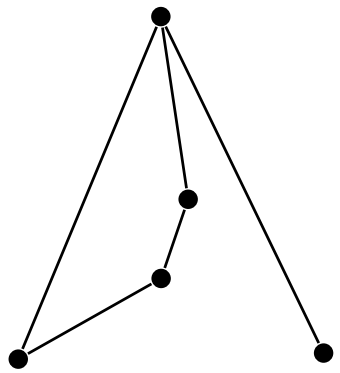


Figure: Set median of graphs that represent the letter "A".

- contained in IAM Graph Database [8]
- graphs represent highly distorted drawings of capital letters with only straight lines
- very small graphs, contain up to 9 nodes only
- nodes labeled with Euclidian coordinates, edges unlabeled
- edit costs: constant node and edge insertion and deletion costs, Euclidean node substitution costs

Including GEDLIB

- at the beginning of `<GEDLIB_ROOT>/median/src/median_letter.cpp`:

```
/* Use GEDLIB as shared library for GXL graphs. */
#define GXL_GEDLIB_SHARED

/* Include the main header file. */
#include "../../src/env/ged_env.hpp"
```

- `GXL_GEDLIB_SHARED`: define this macro if you want to link your application against a pre-compiled template instantiation of GEDLIB for graphs given as GXL files
- if `GXL_GEDLIB_SHARED` is undefined, GEDLIB is used as a header-only library

Setting Up the Environment

- inside `int main(int argc, char* argv[])`:

```
/* Set up the environment. */
ged::GEDEnv<ged::GXLNodeID,ged::GXLLabel,ged::GXLLabel> env;
```

- `ged`: global namespace of GEDLIB
- `ged::GXLNodeID` a. k. a. `std::string`: generic type for node IDs of graphs given in GXL format
- `ged::GXLLabel` a. k. a. `std::map<std::string,std::string>`: generic type for node and edge labels of graphs given in GXL format

Node and Edge Labels for Letter (H) Graphs

- nodes of Letter (H) graphs in GXL files:

```
<node id="_0">
  <attr name="x"><float>0.687437</float></attr>
  <attr name="y"><float>0.271509</float></attr>
</node>
```

  ⇝ each node label `ged::GXLLabel node_label` has two keys: `"x"` and `"y"`

- edges of Letter (H) graphs in GXL files:

```
<edge from="_0" to="_1"/>
```

  ⇝ each "edge label" `ged::GXLLabel edge_label` is empty

Select Edit Costs

- use predefined edit costs for Letter (H) graphs with default constants:

```
env.set_edit_costs(ged::Options::EditCosts::LETTER);
```

- or use predefined edit costs for Letter (H) graphs with non-default constants:

```
env.set_edit_costs(ged::Options::EditCosts::LETTER,
  {0.7,1.5,0.25});
```

- or implement your own edit cost class `CustomEditCosts<ged::GXLLabel,ged::GXLLabel>` as derived class of `ged::EditCosts<ged::GXLLabel,ged::GXLLabel>` and use it like so:

```
CustomEditCosts<ged::GXLLabel,ged::GXLLabel> edit_costs;
env.set_edit_costs(&edit_costs);
```

Load the Graphs and Initialize the Environment

```
/* Path to XML file specifying which graphs should be loaded. */
std::string collection("../collections/Letter_A.xml");

/* Path to directory containing the graphs. */
std::string dir("../../data/datasets/Letter/HIGH/");

/* Load the graphs, obtain vector of graph IDs.*/
std::vector<ged::GEDGraph::GraphID> graph_ids(env.load_gxl_graphs(
  dir, collection, ged::Options::GXLNodeEdgeType::LABELED, ged::
  Options::GXLNodeEdgeType::UNLABELED));

/* Allocate space for median graph. */
ged::GEDGraph::GraphID median_id{env.add_graph("median", "A")};

/* Initialize the environment */
env.init(ged::Options::InitType::EAGER_WITHOUT_SHUFFLED_COPIES);
```

- to avoid expensive re-initialization, always allocate space for all graphs you want to add later on

Select the GED Method

```
std::string ipfp_options("--threads 6 --initial-solutions 5 --
  initialization-method RANDOM");
env.set_method(ged::Options::GEDMethod::IPFP, ipfp_options);
```

- `ged::Options::GEDMethod`: contains macros to select all methods that are implemented in GEDLIB
- many methods accept options `"[--<option> <arg>] [...]"`
- for instance, we here set up IPFP [3, 5–7] to run in 6 threads from 5 different randomly constructed initial solutions

Get Modifiable Representations of All Graphs

```
std::vector<ged::ExchangeGraph<ged::GXLNodeID,ged::GXLLabel,ged::
  GXLLabel>> graphs;
for (auto graph_id : graph_ids) {
  graphs.emplace_back(env.get_graph(graph_id));
}
```

- `ged::ExchangeGraph<UserNodeID,UserNodeLabel,UserEdgeLabel>`: structure to inspect and/or modify the graphs after loading them into the environment
- you don't have to load the exchange graphs if you only want to run GED queries

Compute Set Median (1)

```
ged::GEDGraph::GraphID set_median_id{0};
/* Sums of distances for all encountered medians. */
std::vector<double> sums_dists({std::numeric_limits<double>::
  infinity()});
for (auto g_id : graph_ids) {
  double sum_dists{0};
  for (auto h_id : graph_ids) {
    /* Run the selected method (IPFP in our case). */
    env.run_method(g_id, h_id);
    /* Add obtained distance to sum of distances. */
    sum_dists += env.get_upper_bound(g_id, h_id);
  }
  /* Update ID and sum of distances of set median. */
  if (sum_dists < sums_dists.at(0)) {
    sums_dists[0] = sum_dists;
    set_median_id = g_id;
  }
}
```

Compute Set Median (2)

```
/* Get the node maps for the set median computed by IPFP. */
std::vector<ged::NodeMap> node_maps;
for (auto h_id : graph_ids) {
  node_maps.emplace_back(env.get_node_map(set_median_id, h_id));
}

/* Get the ExchangeGraph representation of the set median. */
ged::ExchangeGraph<ged::GXLNodeID,ged::GXLLabel,ged::GXLLabel>
  median(graphs.at(set_median_id));
```

- `ged::NodeMap`: class that represents node maps a. k. a. error-correcting matchings

Main Loop (1)

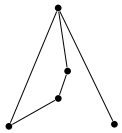
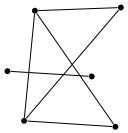
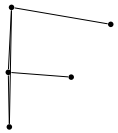
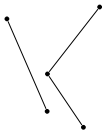
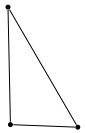
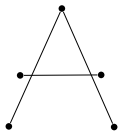
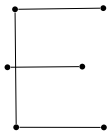
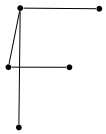
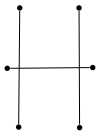
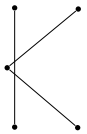
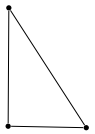
```
bool median_was_modified{true};
bool node_maps_were_modified{true};
while (median_was_modified or node_maps_were_modified) {
  /* Update the median graph. */
  median_was_modified = update_median_graph(median, graphs,
    node_maps, graph_ids);
  /* Load modified median into the environment. */
  env.load_exchange_graph(median, median_id);
  env.init(ged::Options::InitType::EAGER_WITHOUT_SHUFFLED_COPIES);
  /* Compute costs of old node maps w.r.t. updated median. */
  for (auto h_id : graph_ids) {
    env.compute_induced_cost(median_id, h_id, node_maps.at(h_id));
  }
  /* ... (continues on next slide) */
```

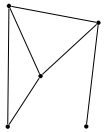
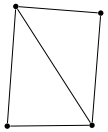
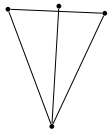
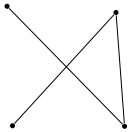
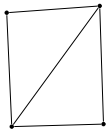
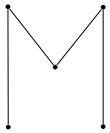
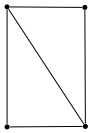
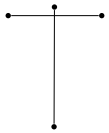
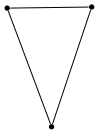
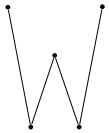
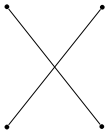
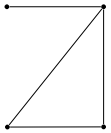
- `update_median_graph(median, graphs, node_maps, graph_ids)`: helper function to update the `median` given as `ged::ExchangeGraph<ged::GXLNodeID,ged::GXLLabel,ged::GXLLabel>`

Main Loop (2)

```
  /* ... (continuation from previous slide) */
  /* Update the node maps. */
  node_maps_were_modified = false;
  for (auto h_id : graph_ids) {
    env.run_method(median_id, graph_id);
    double new_dist{env.get_upper_bound(median_id, h_id)};
    if (new_dist < node_maps.at(h_id).induced_cost() - 0.0001) {
      node_maps[h_id] = env.get_node_map(median_id, h_id);
      node_maps_were_modified = true;
    }
  }
  /* Compute sum of distances for current median and node maps. */
  sums_dists.emplace_back(0);
  for (auto graph_id : graph_ids) {
    sums_dists.back() += node_maps.at(graph_id).induced_cost();
  }
}
```

Median Graphs for Letter (H) Dataset

letter	A	E	F	H	I	K	L
set median							
generalized median							

letter	M	N	T	V	W	X	Y	Z
set median								
generalized median								

Questions?

References

[1] Z. Abu-Aisheh, R. Raveaux, and J.-Y. Ramel, "A graph database repository and performance evaluation metrics for graph edit distance", in *GbRPR*, 2015, pp. 138–147.

[2] D. B. Blumenthal, N. Boria, J. Gamper, S. Bougleux, and L. Brun, "Comparing heuristics for graph edit distance computation", *VLDB J.*, 2019, in press.

[3] D. B. Blumenthal, É. Daller, S. Bougleux, L. Brun, and J. Gamper, "Quasimetric graph edit distance as a compact quadratic assignment problem", in *ICPR*, pp. 934–939.

[4] N. Boria, S. Bougleux, B. Gaüzère, and L. Brun, "Generalized median graph via iterative alternate minimizations", in *GbRPR*, 2019, pp. 99–109.

[5] S. Bougleux, L. Brun, V. Carletti, P. Foggia, B. Gaüzère, and M. Vento, "Graph edit distance as a quadratic assignment problem", *Pattern Recognit. Lett.*, vol. 87, pp. 38–46, 2017.

[6] S. Bougleux, B. Gaüzère, and L. Brun, "Graph edit distance as a quadratic program", in *ICPR*, 2016, pp. 1701–1706.

[7] É. Daller, S. Bougleux, B. Gaüzère, and L. Brun, "Approximate graph edit distance by several local searches in parallel", in *ICPRAM*, 2018, pp. 149–158.

[8] K. Riesen and H. Bunke, "IAM graph database repository for graph based pattern recognition and machine learning", in *S+SSPR*, 2008, pp. 287–297.