Journal of Machine Learning Research 7 (2006) 2065-2086 Submitted 4/05; Revised 4/06; Published 10/06

Distance Patterns in Structural Similarity

Thomas Kämpke KAEMPKE@FAW-NEU-ULM.DE
Forschungsinstitut für Anwendungsorientierte Wissensverarbeitung/n FAW/n
Lise-Meitner-Str. 9
89081 Ulm, Germany

Editor: Peter Dayan

Abstract

Similarity of edge labeled graphs is considered in the sense of minimum squared distance between corresponding values. Vertex correspondences are established by isomorphisms if both graphs are of equal size and by subisomorphisms if one graph has fewer vertices than the other. Best fit isomorphisms and subisomorphisms amount to solutions of quadratic assignment problems and are computed exactly as well as approximately by minimum cost flow, linear assignment relaxations and related graph algorithms.

Keywords: assignment problem, best approximation, branch and bound, inexact graph matching, model data base

1. Introduction

Structural similarity is involved in a variety of pattern recognition problems when considered from an abstract perspective. The abstraction refers to measurements and observations whose specifics are ignored. One class of such problems is encountered in image processing, where a set of features or objects with topological interrelations is detected in several scenes. Whenever these are presumed to be similar according to position, proximity or else, the degree of similarity is of interest.

Structures are represented throughout by labeled graphs such as image graphs. In image graphs, vertices represent image edges, corners or regions of interest such as regions of constant intensity or homogenous texture. Graph edges represent relations such as neighborhoods or concept hierarchies. Edge labels represent distances, degrees of association or else.

Structural similarity is considered as similarity between two labeled graphs. Typical roles of the two graphs are that of a model graph from a model data base or a prototype data base and that of an instance graph representing an 'as is' structure which is encountered 'at run time'. The issue is then to determine the similarity between prototype and instance.

Similarity will be formulated as a best approximation problem. This involves minimization of squared distances which results in a quadratic assignment problem. The problem is approached by several algorithmic concepts including network algorithms with emphasis on linear assignment relaxations. Also, cost minimal flows of given strength will play a major role. The focus is on approximate algorithms for best graph approximation since the exact problem is NP-hard.

Besides image processing, structural similarity is encountered, for example, in document analysis and molecular graph search. However, the objective of this work is not to consider one particular real or potential application. Instead, common problem formulations and algorithms are presented.

The remainder of this work is organized as follows. Section 2 introduces the best approximation problem for edge-labeled graphs and reviews related work. Polynomial time approximation algorithms are stated in Section 3. Since the best graph approximation problem contains subgraph isomorphism as a special case, no exact algorithm can be expected to run in polynomial worst case time. The approximations are consequently based on linear assignment problems since the original problem is a quadratic assignment problem. Approximations have interesting side features such as being suited for grid computations. Section 4 contains two sketches of approaches for exact algorithms for the best graph approximation problem. One is based on flows, the other on branch and bound.

Unless otherwise stated, all graphs considered are undirected which means that edges have no preferred directions. Moreover, the graphs are simple which means that there is at most one edge between any two vertices and there are no loops so that no edge begins and ends in the same vertex. The edge labels themselves must allow to be subtractable from each other but are otherwise unconstrained. In particular, the edge labels themselves do not have to reflect any notion of similarity.

The 2-norm or Euclidean norm of any vector $z = (z_1, \ldots, z_n)$ of real numbers z_i is denoted by $||z|| = ||z||_2 = \sqrt{z_1^2 + \ldots + z_n^2}$. The number of elements of a finite set A is denoted by $|A|$.

2. Problem and Related Work

A distance pattern is understood to be an undirected graph with edge labels. The graph vertices denote objects or states and the edge labels denote distances, transition times etc. Though it may take quite some effort to generate these graphs in applications, this effort is ignored here and two such graphs are assumed to be given.

2.1 Problem Formulation

The best approximation of a labeled graph by another labeled graph is defined by a subisomorphism of the vertex set of the first graph to the vertex set of the second graph. Thereby edge labels of the first graph are approximated by corresponding edge labels of the second graph as minimum sum of squared differences.

Formally, two undirected graphs with edge labelings are given by $G_1 = (V_1, E_1)$ with $l_1 : E_1 \to \mathbb{R}$ and $G_2 = (V_2, E_2)$ with $l_2 : E_2 \to \mathbb{R}$. The first graph has $n = |V_1|$ vertices and the second graph has $m = |V_2|$ vertices with $n \leq m$. The best approximation of G_1 by G_2 is defined via an optimal approximating subisomorphism

$$\varphi^0 = argmin_{\varphi: V_1 \to V_2,\, \varphi \text{ invertible}} \sqrt{\sum_{\{v_i, v_j\} \in E_1} \Big(l_1(v_i, v_j) - l_2(\varphi(v_i), \varphi(v_j)) \Big)^2}.$$

The best approximation is given by the image of the first vertex set $\varphi^0(V_1)$ so that $\varphi^0(V_1) \subseteq V_2$. The minimum value is called the distance of the best approximation.

In order to be well defined, the problem entails a technical condition that, whenever two vertices of the first graph are joined by an edge, the images of the two vertices must be joined by an edge in the second graph. Without further restrictions to the subisomorphisms this implies that the second graph must be complete.

The objective of best approximation can be considered as Frobenius distance when both graphs are complete. The edge labelings then denote distance matrices D_1, D_2 with entries

$$D_1(v_i,v_j) = \begin{cases} l_1(v_i,v_j), & \text{for } v_i \neq v_j \\ 0, & \text{for } v_i = v_j \end{cases} \quad D_2(w_i,w_j) = \begin{cases} l_2(w_i,w_j), & \text{for } w_i \neq w_j \\ 0, & \text{for } w_i = w_j. \end{cases}$$

The best approximation objective can then be written as follows since counting over all edges amounts to counting twice over all vertex pairs. Pairs of identical vertices are negligible because they contribute value zero.

$$\begin{aligned} \sum_{\{v_i,v_j\}\in E_1} \Big(l_1(v_i,v_j) - l_2(\varphi(v_i),\varphi(v_j))\Big)^2 &= \frac{1}{2} \sum_{v_i,v_j\in V_1} \Big(D_1(v_i,v_j) - D_2(\varphi(v_i),\varphi(v_j))\Big)^2 \\ &= \frac{1}{2}\,||(D_1(\cdot,\cdot)) - (D_2(\varphi(\cdot),\varphi(\cdot)))||_F^2. \end{aligned}$$

The Frobenius norm of any matrix is the square root of the sum of all squared entries (Golub and van Loan, 1985).

Distance matrices allow to consider the best approximation problem also for graphs which are not complete. Whenever one of the two given graphs is not complete, all missing edges are inserted. The complete graph is then labeled by the shortest path distances according to the original edge labeling. Original edge labels are preserved when these satisfy the triangle inequality. Original edge labels may be overwritten when these do not satisfy the triangle inequality.

The celebrated graph isomorphism problem is contained in the best approximation problem as the following special case. Suppose H_1 and H_2 are two unlabeled graphs with same number of vertices and arbitrary edge sets. Each graph is extended to the complete graph on its vertex set and receives the edge labels

$$l_i(e) := \begin{cases} 1, & \text{if edge } e \text{ belongs to original graph } H_i \\ 0, & \text{if edge } e \text{ does not belong to original graph } H_i, \end{cases}$$

$i = 1,2$. These graphs are denoted G_1 and G_2 respectively. The original graphs are isomorphic if and only if the best approximation of G_1 by G_2 and the best approximation of G_2 by G_1 both have distance zero.

The most trivial case of the best approximation problem is given for the smaller graph being the smallest possible. This is a two vertex graph with one edge only. The single edge graph is best approximated by that edge from the larger graph whose label comes closest to the label of the single-edge graph. A non-trivial example of the best approximation problem is given in Figures 1 and 2.

Best graph approximation can be considered as search for a minimum weight clique of given size in a suitably defined graph, the association graph or correspondence graph. This relation is such that the given clique size equals the size of the smaller graph and that no larger cliques exist in the association graph. Mnemonically, best graph approximation can thus be remembered as search for a minimum weight clique of maximum size.

The association graph of two graphs is defined as their product. Each vertex of one graph is paired with each vertex of the other graph and each such pair is identified with a vertex of the association graph. Two vertices of the association graph are joined by an edge if the two vertices stem from four distinct vertices of the original graphs. The construction is illustrated in Figure 3. The

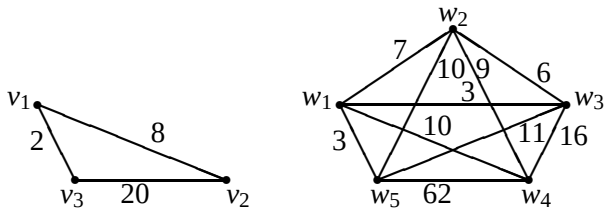

Figure 1: The left three vertex graph is best approximated by the triangle with edge labels $3, 10, 16$ in the larger graph. The subisomorphism is $\varphi(v_1) = w_1$, $\varphi(v_2) = w_4$ and $\varphi(v_3) = w_3$ with the two vertices w_2 and w_5 being unattained. The graph isomorphism is given explicitly in the next figure.

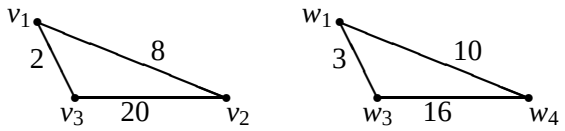

Figure 2: Original graph (left) and best approximating isomorphic substructure (right) for the situation of Figure 1. The squared distance between the two graphs is $(2-3)^2 + (8-10)^2 + (20-16)^2 = 21$.

cost of the edge between the vertices (v_i, w_j) and (v_k, w_l) is set equal to $(D_1(v_i, v_k) - D_2(w_j, w_l))^2$. The meaning of selecting any vertex of the form (v_i, w_j) is that the original vertex v_i is mapped to the original vertex w_j by a subisomorphism. These "associations" motivate the name association graph and the cost of a clique, which equals the cost sum over all edges between selected vertices, is the objective of graph approximation.

An alternative view of best graph approximation can be obtained from quadratic assignment problems such as the following

$$\begin{aligned} &\min_x x^T C x \\ &\text{such that } \sum_{i=1}^{n} x_{ij} \leq 1 \quad \forall\, j = 1, \ldots, m \\ &\qquad \sum_{j=1}^{m} x_{ij} = 1 \quad \forall\, i = 1, \ldots, n \\ &\qquad x_{ij} \in \{0, 1\} \quad \forall\, i, j. \end{aligned}$$

The binary variable x_{ij} attaining value one means that vertex v_i is assigned to vertex w_j and that variable attaining the value zero means that this assignment is not valid. The vector x has $n \cdot m$ coordinates and C is an $n \cdot m \times n \cdot m$ matrix denoting the cost incurred by pairwise assignments; the assignments $v_i \mapsto w_j$ and $v_k \mapsto w_l$ entail the cost $(D_1(v_i, v_k) - D_2(w_j, w_l))^2$. The cost matrix is

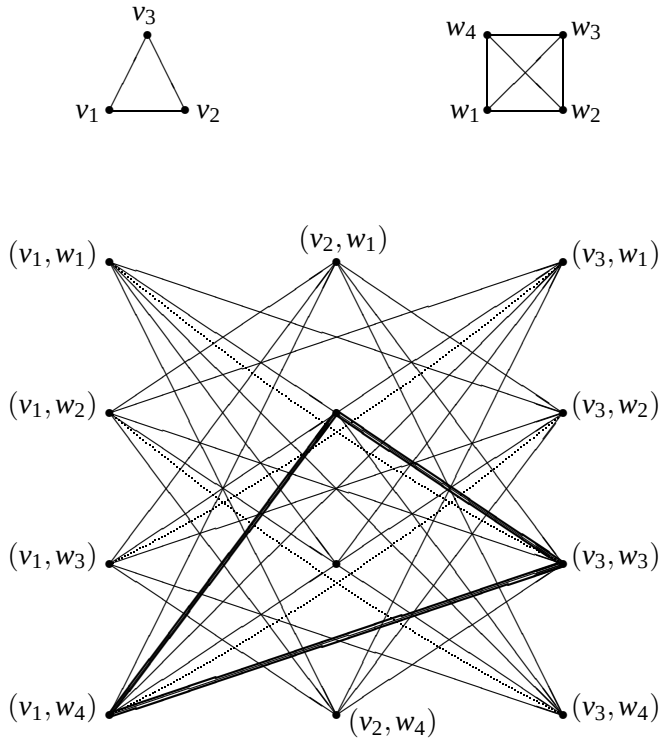

Figure 3: A "small" graph and a "large" graph (top) and their association graph (bottom). The indicated clique of the vertices (v_1,w_4), (v_2,w_2) and (v_3,w_3) denotes the subisomorphism $\varphi(v_1) = w_4$, $\varphi(v_2) = w_2$ and $\varphi(v_3) = w_3$.

computed as

$$C = \begin{pmatrix} (D_1(v_1,v_1) - D_2(\cdot,\cdot))^2 & \dots & (D_1(v_1,v_n) - D_2(\cdot,\cdot))^2 \\ \vdots & \vdots & \vdots \\ (D_1(v_n,v_1) - D_2(\cdot,\cdot))^2 & \dots & (D_1(v_n,v_n) - D_2(\cdot,\cdot))^2 \end{pmatrix}$$

where $D_2(\cdot,\cdot)$ is the $m \times m$ matrix of all distance values for the second graph and $(c-M)^2$, the square of a constant c minus a matrix M, is understood as a matrix of the size of M with each element denoting the squared distance from the constant so that $(c-M)^2 = ((c-m_{ab})^2)_{ab}$. The size of the cost matrix is unfortunately large as it already is a 15×15 matrix for the small graphs from Figure 1.

2.2 Related Work

The subisomorphism problem which is contained as special case of the best approximation problem must not be confused with that version of the SUBGRAPH ISOMORPHISM problem which is known to be NP-complete, see Garey and Johnson (1981, problem GT48). That problem does not admit edge labels and it considers the two operations of vertex removal and edge removal for the transition from the larger to the smaller graph. Even more, the LARGEST COMMON SUBGRAPH

problem also is NP-complete, see Garey and Johnson (1981, problem GT49). This problem allows edge removals only in order to find isomorphic subgraphs. Here, only vertex removals matter and edge removals are allowed only in so far as they are implied by vertex removals.

A widely used measure for so-called inexact graph matching is the edit distance between two unlabeled graphs. One graph is therefore modified by a minimum number of vertex insertions and deletions and by edge insertions and deletions. The concept in general as well as a particular focus on tree graphs is given in Wang et al. (1998). A simplification of the edit distance does not refer to the graphs themselves and isomorphism between them but to their degree histograms which are to be made equal by a minimum number of changes (Papadopoulos and Manolopoulos, 1999). The edit distance for subisomorphisms of labeled graphs is considered by Messmer and Bunke (1998a).

Subisomorphism with vertices and edges both carrying labels is considered by Hlaoui and Wang (2002). Conflicts of tentatively assigning several vertices of the smaller graph to one vertex of the larger graph are resolved in a hierarchical manner. The method is reported to be well suited for small graphs. Best graph approximation in terms of matching problems is considered by Gold and Rangarajan (1996). There, the problem is extended to a sequence of continuous surrogate problems. This leads to an iterative, matrix-based solution scheme which is controlled by a continuous parameter that intends to drive continuous relaxations to a discrete vertex assignment. While leaving slight uncertainties about the control of this parameter and while using a linear instead of a quadratic distance between edge labels, the method makes, like the approaches given below, use of the assignment problem. A quite different, probabilistic approach which makes use of potential functions is given in Caetano et al. (2005).

Best graph approximation can be considered as ”dual” to joint edge and label construction. This construction was developed for trees by Desper and Vingron (2002). Only distance information is required in order to build one tree with edge weights so that the given distances are approximated by path lengths between leaves of the tree. The construction is formulated as a least square approximation problem so that solution methods have a strong algebraic component.

Matching techniques for graphs whose vertices denote positions in space have been developed from the analogy of physical elasticity, compare Wiskott et al. (1997) and Wiskott and Malsburg (2002). In addition, the elasticity idea supports the generation of the model graph from examples. These physical methods are complemented by probabilistic methods for unlabeled graph subisomorphisms by Bengoetxea (2002).

Motivated by graphs that describe the structure of SQL data bases, an interactive fixpoint algorithm for similarity computing has been proposed by Melnik et al. (2002). The method is based on the assumption that adjacent vertices are more similar than non-adjacent vertices. The quality of the matching result is measured by the number of human adjustment steps that are eventually needed. For a similar data base purpose, case-based reasoning, similarity of graphs with character-string labels has been considered (Champin and Solnon, 2003). Similarity is measured by weighted counts of identical labels with vertex correspondence being generated by a greedy algorithm. This algorithm maximizes the similarity score in each iteration.

Best graph approximation relates given lists of numbers by vertex-edge incidences. When these are dropped, that is, when numbers are given as mere list entries of one list and when the numbers are viewed as Euclidean distances on the real line, a complete line graph may be searched for such that the given numbers form a coherent distance labeling. This is the NP-complete PARTIAL DIGEST problem from genomic mapping, see Skiena and Sundaram (1994). Whenever the best graph approximation problem refers to graphs with Euclidean distances on the line and whenever

the smaller graph is known to be contained in the larger graph, methods for PARTIAL DIGEST can help to identify the actual subgraph isomorphism.

Subgraph isomorphism for unlabeled graphs has been studied in Messmer and Bunke (1998b) motivated by symbol recognition problems. A polynomial time algorithm is given which requires preprocessing and exponential space in the worst case. Graph similarity has even been investigated for machine learning (Pope and Lowe, 1996). A quite sketchy outline of graph search methods over molecular graphs is presented in Shasha et al. (2002), while the perspective of distance methods for molecular similarity and superstructure retrieval is given in Kämpke (2004).

3. Approximate Algorithms

The best approximation problem obviously is finite and, thus, can, in principle, be solved by enumeration over all $\binom{m}{n}n!$ selections for admissible functions φ. Approximate algorithms of different types as well as a strategy for exact algorithms are given in the sequel. The present approximations mainly focus on linear assignment problems since the original problem is a quadratic assignment problem.

To illustrate the intuitive aim of best graph approximation, the squared approximation distance is rewritten for the special case of equally sized graphs. The best isomorphism then is a solution of the maximization problem

$$\max_{\varphi:V_1\to V_2,\ \varphi \text{ invertible}} \sum_{\{v_i,v_j\}\in E_1} D_1(v_i,v_j)\cdot D_2(\varphi(v_i),\varphi(v_j)).$$

The sum can be considered as an inner product over edges. When all edges of the first graph are sorted increasingly then the maximization is obtained by an isomorphism that maintains monotonicity "as far as possible". This is motivated by the well known inner product maximization $\sum_{i=1}^{N} a_i b_{\pi(i)}$ problem over all permutations π. The solution is a permutation with $b_{\pi(1)} \leq \ldots \leq b_{\pi(N)}$ whenever all coordinates of the first vector are sorted as $a_1 \leq \ldots \leq a_N$, compare with Hardy et al. (1948). A monotonicity preserving isomorphism does generally not exist for the edge labels.

3.1 Distance Lists

A heuristic procedure for best subgraph isomorphism can be devised on local, that is, vertex-oriented decisions. These are based on distance lists. The distance list of a vertex contains the labels of all edges that are incident with the vertex. Any distance list can also be considered as a vector. The distance list of a vertex v is denoted by $distlist(v)$. A distance list in a complete graph with edge labels $D(\cdot,\cdot)$ is given by $distlist(v) = (D(v,v_i))_{v_i\in V-\{v\}}$. For the ease of comparability, all vertex lists are sorted increasingly. Distance lists generalize vertex degrees since they count the "ones" when unlabeled graphs receive the binary edge labeling as given above for the embedding of the graph isomorphism problem.

The distance lists for the three-vertex graph from Figure 1 are given by $distlist(v_1) = (2,8)$, $distlist(v_2) = (8,20)$ and $distlist(v_3) = (2,20)$. The five-vertex graph of the same figure has the distance lists $distlist(w_1) = (3,3,7,10)$, $distlist(w_2) = (6,7,7,9)$, $distlist(w_3) = (3,6,8,16)$, $distlist(w_4) = (9,10,16,62)$ and $distlist(w_5) = (3,7,8,62)$.

Viewing distance lists as vectors allows to consider Euclidean distances between distance lists. This only requires standard notions for vector distances as long as distance lists have the same number of coordinates. The lists being sorted makes these differences meaningful. Whenever two

distance lists have different numbers of coordinates, a proper selection is made from the larger distance list. In the foregoing example, selections of two out of the four coordinates from the distance lists of the five-vertex graph are made.

The approximation of a distance list by a selection from a larger distance list can be formulated as a weighted or cost minimal assignment problem. Therefore, two distance lists $distlist(v) = (x_1,\ldots,x_{n-1})$ and $distlist(w) = (y_1,\ldots,y_{m-1})$, $n \leq m$, are endowed with a complete bipartite graph. Each coordinate receives one vertex and each coordinate of one distance list is connected by an edge to each coordinate of the other distance list. No two coordinates of the same list are connected. The edge connecting coordinates x_i and y_j is labeled by the squared difference of the list entries $(x_i - y_j)^2$, compare with Figure 4.

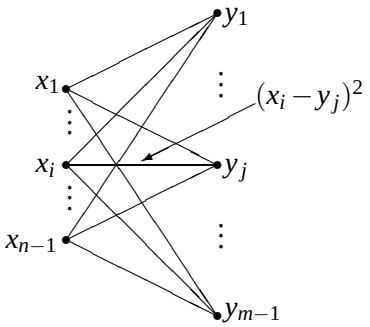

Figure 4: Complete bipartite graph for two distance lists. The vertices correspond to the coordinates of the distance lists rather than to the vertices of the original graphs. Only one of the $(n-1)\cdot(m-1)$ edge labels is sketched.

Any approximation of the first distance list by the second distance list amounts to a selection of $n-1$ distinct coordinates or indices from the second list. The approximation objective can be formulated as a linear function of the edge labels

$$\min_{1\leq j_1<\ldots<j_{n-1}\leq m-1} \sum_{i=1}^{n-1} (x_i - y_{j_i})^2.$$

Alternatively, the best approximation problem for distance lists can be formulated as cost minimal perfect matching problem and as a cost minimal integral flow problem with flow value set to level $n-1$, compare with Section 4.

The best approximation of a distance list $distlist(v)$ by the distance list $distlist(w)$ is denoted as the projection $pr(distlist(w),\ distlist(v))$. The squared approximation error equals $DL(v,w) = ||pr(distlist(w),\ distlist(v)) - distlist(v)||^2$. Samples of distance lists, their best approximations and approximation errors are given in the following table whose data refer to Figure 1.

	$distlist(v_i)$	$distlist(w_j)$	$pr(distlist(w_j), distlist(v_i))$	$DL(v_i, w_j)$
$i=1$	$(2,8)$	$(3,3,7,10)$	$(3,7)$	$(2-3)^2+(8-7)^2=2$
	$(2,8)$	$(6,7,9,10)$	$(6,7)$ or $(6,9)$	$(2-6)^2+(8-7)^2=17$
	$(2,8)$	$(3,6,11,16)$	$(3,6)$	$(2-3)^2+(8-6)^2=5$
	$(2,8)$	$(9,10,16,62)$	$(9,10)$	$(2-9)^2+(8-10)^2=53$
	$(2,8)$	$(3,10,11,62)$	$(3,10)$	$(2-3)^2+(8-10)^2=5$
$i=2$	$(8,20)$	$(3,3,7,10)$	$(7,10)$	$(8-7)^2+(20-10)^2=101$
	$(8,20)$	$(6,7,9,10)$	$(7,10)$ or $(9,10)$	$(8-7)^2+(20-10)^2=101$
	$(8,20)$	$(3,6,11,16)$	$(6,16)$	$(8-6)^2+(20-16)^2=20$
	$(8,20)$	$(9,10,16,62)$	$(9,16)$	$(8-9)^2+(20-16)^2=17$
	$(8,20)$	$(3,10,11,62)$	$(10,11)$	$(8-10)^2+(20-11)^2=85$
$i=3$	$(2,20)$	$(3,3,7,10)$	$(3,10)$	$(2-3)^2+(20-10)^2=101$
	$(2,20)$	$(6,7,9,10)$	$(6,10)$	$(2-6)^2+(20-10)^2=116$
	$(2,20)$	$(3,6,10,16)$	$(3,16)$	$(2-3)^2+(20-16)^2=17$
	$(2,20)$	$(9,10,16,62)$	$(9,16)$	$(2-9)^2+(20-16)^2=65$
	$(2,20)$	$(3,10,11,62)$	$(3,10)$	$(2-3)^2+(20-10)^2=101$

3.2 Best Approximations by Distance Lists

The idea of cost minimal assignments can be carried over from distance lists of single vertices to the whole graph. This results in an efficient heuristic algorithm for best graph approximation. Again, the approximation problem is formulated as cost minimal assignment problem over a complete bipartite graph. One set of vertices corresponds to the smaller graph and the other to the larger graph. The edges are labeled by the errors of best distance list approximations. The general situation is sketched in Figure 5.

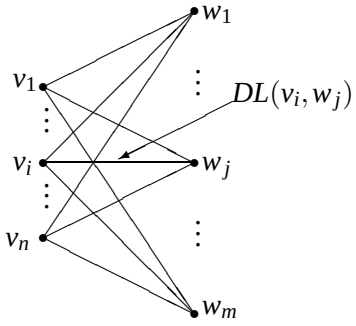

Figure 5: Complete bipartite graph for a heuristic solution of best graph approximation. The edge labels refer to best approximations of distance lists.

A greedy heuristic for best graph approximation can now be based on iterative decisions according to minimum distance list errors.

ApproxDistList

1. Input complete labeled graphs G_1, G_2 with $|V_1| \leq |V_2|$.
   Initialization. Computation of distance list errors $DL(v,w)$ for all $v \in V_1$, $w \in V_2$. $A = V$, $B = W$.
2. While $A \neq \emptyset$ do
   (a) Computation of $W(v) = argmin_{w \in B} DL(v,w)$ and $level(v,B) = DL(v,W(v))$ and for all $v \in A$.
   (b) Selection of $v_0 = argmin_{v \in A} level(v,B)$.
   (c) $\varphi(v_0) = w(v_0)$ with $w(v_0) \in W(v_0)$.
   (d) $A = A - \{v_0\}$.
   (e) $B = B - \{\varphi(v_0)\}$.
3. Output subisomorphism $\varphi(\cdot)$ on V_1.

The level computations in step 2(a) determine the best fit decision that can be made without taking back prior decisions. The sets $W(v)$ indicate all vertices from the second graph that attain the minimum. Ties for selections in step 2(b) as well as for selections from the set $W(v_0)$ in step 2(c) are broken arbitrarily. While the sets A and B of unassigned vertices decrease along the iterations of the algorithm, the distance lists to consider become fewer but not smaller. Thus, the edge labels $DL(v,w)$ do not have to be updated along the iterations.

The greedy procedure applied to the graphs of Figure 1 can be traced with the data from the table of Section 3.1. The resulting subisomorphism is given by selecting the minima for each vertex from the smaller graph. This is exactly the solution indicated by Figure 2.

The foregoing algorithm need not solve the cost minimum assignment problem exactly. An optimal solution of this problem (which still need not lead to the best graph approximation since the assignment problem is only an approximative encoding for best graph approximation problem) can be found by any weighted assignment algorithm for bipartite graphs. These algorithms are typically based on transformations to cost minimum flow problems. Therefore, all vertices of the smaller graph are connected to an extra source vertex and all vertices of the larger graph are connected to an extra sink vertex. All the extra edges receive a unit capacity on the flow. All edges become oriented edges or arcs as indicated in the flow network in Figure 6.

Among the cost minimal flows of strength n there is one with all integer values. This flow amounts to a cost minimal assignment of all vertices from the smaller graph. The out of kilter algorithm allows to compute the desired cost minimum flow, see Ahuja et al. (1993).

A different solution for cost minimal assignment problems can be obtained from straightforward transformations to cost minimal perfect matching problems by introducing additional vertices for the smaller graph. Both sets of the vertex partition then have the same size. All dummy vertices for the smaller graph are connected to all vertices from the larger graph by edges with zero cost. A matching is a set of edges such that any two edges do not have a common vertex. A matching is perfect if each vertex of the graph is covered by an edge.

A polynomial time algorithm for cost minimal perfect matchings can be based on augmenting paths that are constructed by shortest paths. Implementations thereof are available in the LEDA system, see Mehlhorn and Näher (2000, Chapter 7). More recent scaling algorithms are given in Ahuja et al. (1993).

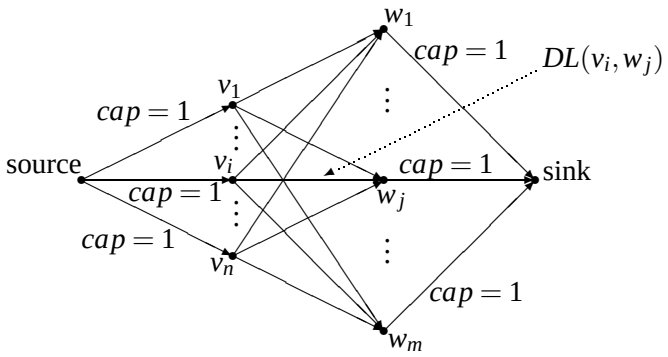

Figure 6: Flow network for a distance label heuristic. The edges that are incident either to the source or the sink carry capacities but no cost coefficients while the edges with cost coefficients do not carry capacities.

3.3 Direct Methods

Direct methods for graph approximation will use the original edge labels and the unaltered best approximation errors for all fitting assessments.

3.3.1 SEQUENTIAL ASSIGNMENTS

A subisomorphism can be constructed by sequentially assigning vertices from the smaller graph to the larger graph so that the sum of squared label distances over all new edge pairs is minimal. The first assignment may stem from two best matching edges. No assignment is ever revised by the following procedure.

SeqAssign

1. Input complete labeled graphs G_1, G_2 with $|V_1| \leq |V_2|$.
   Initialization. Computation of $(e_0, f_0) = argmin_{e \in E_1, f \in E_2} (D_1(e) - D_2(f))^2$.
   Selection of one vertex v_1 of the two vertices incident with e_0.
   Selection of one vertex w_1 of the two vertices incident with f_0.
   $\varphi(v_1) = w_1$.
   Labeling all other vertices from V_1 by $v_2, \ldots, v_n$.
   $B = V_2 - \{\varphi(v_1)\}$.

2. For $i = 2, \ldots, n$ do

   (a) Computation of $w_0 = argmin_{w \in B} \sum_{j=1}^{i-1} (D_1(v_j, v_i) - D_2(\varphi(v_j), w)^2$.

   (b) $\varphi(v_i) = w_0$.

   (c) $B = B - \{\varphi(v_i)\}$.

3. Output subisomorphism $\varphi(\cdot)$ on V_1.

3.3.2 IMPROVEMENTS

Whenever a subisomorphism is not optimal or not known to be optimal, improvements can be aimed at by swapping two vertex assignments or by swapping an assigned with an unassigned vertex. Swaps of both types can easily be evaluated.

For notational ease the vertices are numbered such that $\varphi(v_k) = w_k$, $k = 1,\ldots,n$, for some given subisomorphism $\varphi : V_1 \to V_2$. First, two vertices v_i, v_j, $1 \leq i \neq j \leq n$ are considered for swapping their assignments while all other assignments are preserved.

$$\varphi'(v_k) = \begin{cases} w_j & \text{if } k = i \\ w_i & \text{if } k = j \\ w_k & \text{if } k \neq i, j. \end{cases}$$

The new subisomorphism leads to a smaller approximation error if and only if

$$\sum_{k=1,k\neq i,j}^{n} \Big(l_1(v_i,v_k) - l_1(v_j,v_k)\Big) \cdot \Big(l_2(w_j,w_k) - l_2(w_i,w_k)\Big) > 0.$$

This condition being true is denoted by $\mathsf{Imp(i,j)} = \mathsf{true}$. Second, one vertex v_i, $1 \leq i \leq n$ is considered for changing its assignment to an unassigned vertex $w_0 \in V_2 - \varphi(V_1)$, that is, the present subisomorphism is compared to the new subisomorphism

$$\varphi'(v_k) = \begin{cases} w_0 & \text{if } k = i \\ w_k & \text{if } k \neq i. \end{cases}$$

The new subisomorphism leads to a smaller approximation error if and only if

$$\sum_{k=1,k\neq i}^{n} l_2^2(w_i,w_k) - l_2^2(w_0,w_k) > 2 \sum_{k=1,k\neq i}^{n} l_1(v_i,v_k) \cdot \Big(l_2(w_i,w_k) - l_2(w_0,w_k)\Big).$$

This condition being true is denoted by $\mathsf{Imp(i,}w_0\mathsf{)} = \mathsf{true}$. The improvement conditions results in the following procedure.

Imp

1. Input complete labeled graphs G_1, G_2.
   Subisomorphism $\varphi : V_1 \to V_2$ with $\varphi(v_k) = w_k$, $k = 1,\ldots,n$.
2. While $\mathsf{Imp(i,j)} = \mathsf{true}$ or $\mathsf{Imp(i,}w_0\mathsf{)} = \mathsf{true}$ for some w_0 do
   (a) If $\mathsf{Imp(i,j)} = \mathsf{true}$ then $\varphi(v_i) = w_j$ and $\varphi(v_j) = w_i$
   else $\varphi(v_i) = w_0$
   (b) Vertex relabeling such that $\varphi(v_k) = w_k$, $k = 1,\ldots,n$.
3. Output swap-improved subisomorphism $\varphi : V_1 \to V_2$.

Swaps for triples, quadruples etc. can be considered instead of pairwise swaps. Though the complexity of evaluating such swaps increases only little, the number of swap candidates increases by one order of m for each size increase of the swap candidates.

3.4 Relaxation Method

The previous methods can all be considered as primally feasible which means that all assignments actually are subisomorphisms though not necessarily optimal. The subisomorphism constraint may tentatively be relaxed in analogy to so-called dual optimization techniques. Starting from some promising structure, a sequence of changes will be made that eventually attain feasibility and that tend to incur as little additional cost as possible per step.

3.4.1 INITIAL STRUCTURE

A promising initial structure is constructible by enumerating the first vertex set. Each vertex from that set is paired with all vertices from the second vertex set which amounts to considering their common vertices in the association graph. The label sum of all edges which emanate from each of these common vertices is minimized under the choice of the second vertex. This can be expressed as the following nested minimization:

$$\mu(i) = argmin_{j=1,\ldots,m} \sum_{k=1,k\neq i}^{n} \min_{l=1,\ldots,m} \Big(D_1(v_i,v_k) - D_2(w_j,w_l) \Big)^2 .$$

This minimization implies a function from the first vertex set into the second vertex set by $v_i \mapsto w_{\mu(i)}$. The inner minimizations are independent of each other and, thus, may lead to overassignments which means that the same index l is attained as minimum for different outer indices. The analogue is true for the outer minimization. The presence of overassignments implies that the overall function is not a subisomorphism. However, the independence of the minimizations makes them easy to compute and the resulting cost value provides a lower bound for the cost of best graph approximation. The edges for summation in one minimization of the initial relaxation are sketched in Figure 7. It is worth noticing that the minimization for the initial structure may lead to even

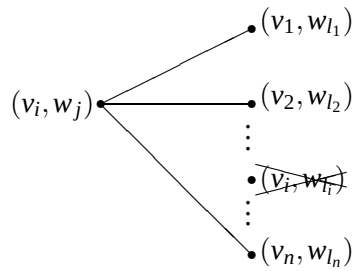

Figure 7: Vertices and edges of the association graph that are considered while computing the value $\mu(i)$ for the initial structure of the relaxation method.

smaller objective values than the distance lists.

3.4.2 IMPROVEMENT

After initialization, the relaxation method proceeds by iteratively selecting an overassigned vertex and redirecting or backtracking at least one assignment to a yet unassigned vertex. Several vertex assignments may be altered in each iteration. Each iteration is organized by computing a wave front of node potentials that emanates from an overassigned vertex.

The edges for the wave front computations are directed. All edges of the current structure are oriented as backward edges from the second vertex set to the first vertex set and all other edges are oriented as forward edges from the first to the second vertex set, see Figure 8. Not all edge labels are initially known in quite a contrast to the ordinary dual method for assignment problems. Actually, node potentials will be computed according to edge transitions rather than by explicitly given edge labels.

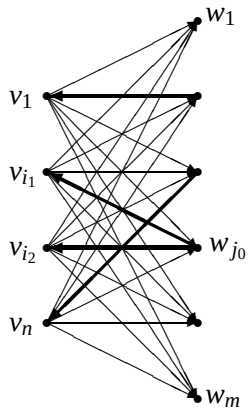

Figure 8: Forward edges (thin) and four backward edges (bold) for the wave front computations to resolve the double assignment of vertex w_{j_0}. The wave front begins in that vertex and ends in a suitable vertex from the second set from which no edge leads back into the first set.

The relaxation method assigns tentative and permanent labels to graph nodes in analogy to the Dijkstra algorithm. The labels are potentials which equal the cost of functions from the first vertex set or a subset thereof into the second vertex set. Such functions need not be one-to-one. Formally, the potential of a function φ is computable as $cost(\varphi) = \sum_{v \in dom(\varphi)} \sum_{v' \in dom(\varphi) - \{v\}} (D_1(v,v') - D_2(\varphi(v), \varphi(v')))^2$, where $dom(\varphi)$ denotes the domain of function φ. Whenever a function is altered by deleting an assignment like $v_4 \mapsto w_3$ it is denoted by $\varphi - (v_4 \mapsto w_3)$, when then the assignment $v_4 \mapsto w_8$ is inserted, the function is denoted by $\varphi - (v_4 \mapsto w_3) + (v_4 \mapsto w_8)$ etc. Backward edges amount to deleting assignments and forward edges amount to inserting assignments.

NoPo

1. Input Structure φ, overassigned vertex $w_0 \in V_2$.
   Initialization $L = V_1 \cup V_2 - \{w_0\}$, $m(l) = \infty \; \forall l \in L$, and $m(w_0) = cost(\varphi)$.
2. While $(V_2 - \varphi(V_1) \subseteq L)$ do:
   (a) Selection of $u = argmin_{l \in L} m(l)$.
   (b) $L = L - \{u\}$.
   (c) $\forall z \in L \cap S(u)$ do:
      i. $\varphi_z = \varphi_u + (u \mapsto z)$ if $z \in V_2$ and $u \in V_1$
         $\varphi_z = \varphi_u - (u \mapsto z)$ if $z \in V_1$ and $u \in V_2$.

ii. Computation of $cost(\varphi_z)$.
iii. If $cost(\varphi_z) < m(z)$ then $m(z) = cost(\varphi_z)$.

3. Termination. Output φ_z for that $z \in V_2 - \varphi(V_1)$ which received its permanent label most recently.

The set $S(u)$ denotes the set of all immediate successors of vertex u which is the set of all vertices to which an edge points from vertex u. The list L contains all vertices that are tentatively labeled. The graph vertices which are not contained in the list are permanently labeled. The algorithm terminates as soon as the first yet unassigned vertex from the second set receives a permanent label.

It may occur during wave front propagation that an already assigned vertex from the second set is permanently labeled. This means that an assignment of the input function φ is revised. The node potential algorithm terminates with exactly one additional vertex assignment from the second graph. The algorithm is applied repeatedly until all overassignments are eliminated.

The improvement algorithm **Imp** from Section 3.3.2 can be obtained from the mode potential algorithm **NoPo** by starting at a vertex that is attained exactly once (with all vertices of the second set being attained at most once). The search for cost reductions proceeds along wave fronts of length two or four and by allowing the wave front to return to its origin.

3.5 Grid Computing Methods

The recently celebrated framework of grid computing is based on the idea of a system which coordinates distributed resources using standard, open, general purpose protocols and interfaces to deliver nontrivial qualities of services (Foster and Kesselman, 2004). Though the distribution of a computational problem into subproblems is not an inherent feature of grid computing in general, it is here considered as exactly that. The breakdown of a computational problem into subproblems that amend to partial computations without any communication between them is here called grid distribution. Grids with several thousand computing nodes have already become feasible.

The lack of any communication between computations means that neither intermediate results nor data are shared. Whenever common data are required, they are physically copied and stored separately before computations begin in order to avoid any access collision. The avoidance of intermediate result communication is a trivial concept. But this makes distributed computations feasible from a practical perspective. Multiple execution of certain operations is the price to be paid. The computational subproblems are generated, distributed and possibly queued by a master. The master also collects the individual computing results and aggregates them to the final computing result.

Best graph approximation lends to grid computing in a straightforward manner. To this end, the strategy of pivoting is here proposed for grid distribution. The idea is that of selecting a vertex from the larger graph as pivot element. This means that best graph approximation is tentatively reduced to only those subisomorphisms which attain that vertex and the optimal of such pivot-subisomorphisms is computed exactly or approximately. Eventually, all vertices of the second graph are chosen as pivot elements and the pivoting-subisomorphism with smallest objective value is reported as the best one. Heuristics which make use of assignment problems are particularly suited for pivoting.

Any subisomorphism which attains the pivot element $w_{j_0} \in V_2$ from some vertex $v_{i_0} \in V_1$ is denoted by $\varphi_{i_0 \mapsto j_0}$. A candidate for the best subisomorphism of this type will be computed and the

best φ_{j_0} of these over all vertices from the first graph is selected by independent computations. Then, the best φ_0 of these over all vertices from the second graph is centrally computed. Schematically this is denoted as

$$\underbrace{\varphi_{i_0 \mapsto j_0} \overset{\min_{i_0 \in \{1,\ldots,n\}}}{\longrightarrow} \varphi_{j_0}}_{grid} \quad pass\ result \longrightarrow \quad \varphi_{j_0} \underbrace{\overset{\min_{j_0 \in \{1,\ldots,m\}}}{\longrightarrow} \varphi_0}_{central}.$$

The weighted assignment problems that are solved for each of the subisomorphisms $\varphi_{i_0 \mapsto j_0}$ is sketched in Figure 9.

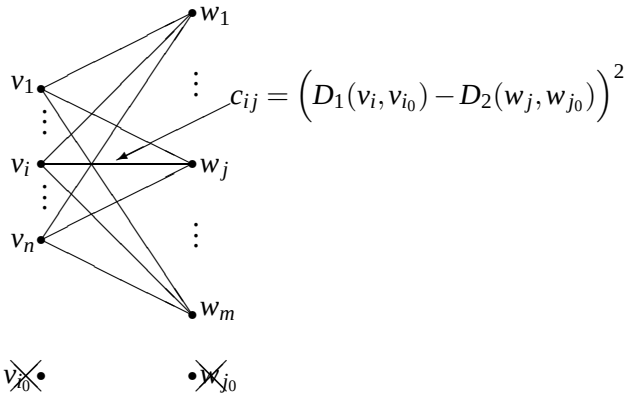

Figure 9: Complete bipartite graph for pivot vertex and preselected vertex from the first graph.

The resulting algorithm that has to be executed at the grid nodes is as follows.

GridPivot

1. Input complete labeled graphs G_1, G_2 with $n \leq m$ and $j_0 \in \{1, \ldots, m\}$.
2. Computations
   (a) Computation of $\varphi_{i_0 \mapsto j_0}$ as minimal weighted assignment for all $i_0 \in \{1, \ldots, n\}$.
   (b) Selection of φ_{j_0} with minimum objective of $\varphi_{i_0 \mapsto j_0}$ over all $i_0 \in \{1, \ldots, n\}$.
3. Output subisomorphism φ_{j_0} on V_1.

The minimization in step 2(b) adheres to the original objective of best graph approximation and no longer to the objective functions of the assignment problems from step 2(a). The computations in step 2(a) can be coupled by using the optimal assignment for one problem—for one value of i_0—as an initial assignment for the next problem—for the next value of i_0. This is feasible for primal methods as well as for dual methods such as the relaxation method, see above.

The grid distribution of the complete problem into subproblems and the selection φ_0 of the best of their results is conceptually obvious.

4. Towards Exact Methods

Since the focus is on practical algorithms, the descriptions of exact algorithmic solutions of best graph approximations is kept to an informal level. First, flows are extended and second, a branch and bound method is sketched.

4.1 Flows

Best graph approximation can be formulated as a cost minimal flow problem in loose analogy to the distance list heuristic. However, the graph is more complicated and additional constraints are necessary. These include submodular edge capacities and integrality conditions for the flow through some edges.

The main problem of transforming the best graph approximation into a flow problem is that subisomorphisms refer to vertices while the costs refer to edge pairs. The cost issue is therefore dealt by introducing a biquadratic number of network vertices that represent all possible edge pairings induced by the vertex assignments. Each pair of distinct vertices from the smaller graph may correspond to each pair of distinct vertices from the larger graph. The incurred cost is an edge label with the edge connecting a network vertex (v_i, v_j, w_k, w_l) with the sink in the flow network.

A subisomorphism in the network is specified by all considering each vertex v_i of the smaller graph and all its possible assignments by introducing the network vertices $(v_i, w_1), \ldots, (v_i, w_m)$. These are connected by edges. Since each vertex of the second graph is attained at most once by any subisomorphism, the edges into the network vertices $(v_1, w_j), \ldots, (v_n, w_j)$ have one common capacity constraint. The joint flow into all these vertices is bounded by one. The situation is depicted by Figure 10.

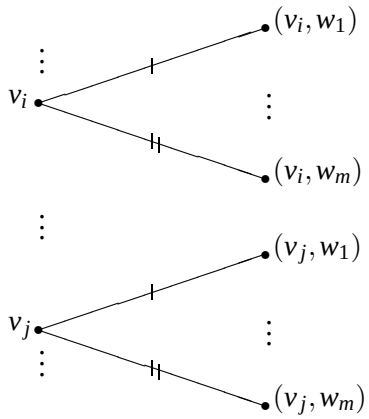

Figure 10: The m vertices $(v_i, w_1), \ldots, (v_i, w_m)$ together allow an inflow of strength one only for each of the vertices v_i. Edges with common capacity constraints are indicated with identical number of ticks.

Common edge capacities are known as submodular flow constraints, see Fujishige (1991). The flow from the source vertex to each of the graph vertices is bounded by one and the flow out of each vertex (v_i, w_j) is bounded by $\frac{1}{n-1}$. The reason for this bound is that each vertex of the smaller graph

is incident with all $n-1$ other vertices and thus $n-1$ edges of the smaller graph are incident with each vertex. The complete construction is illustrated in Figure 11 for the problem from Figure 1.

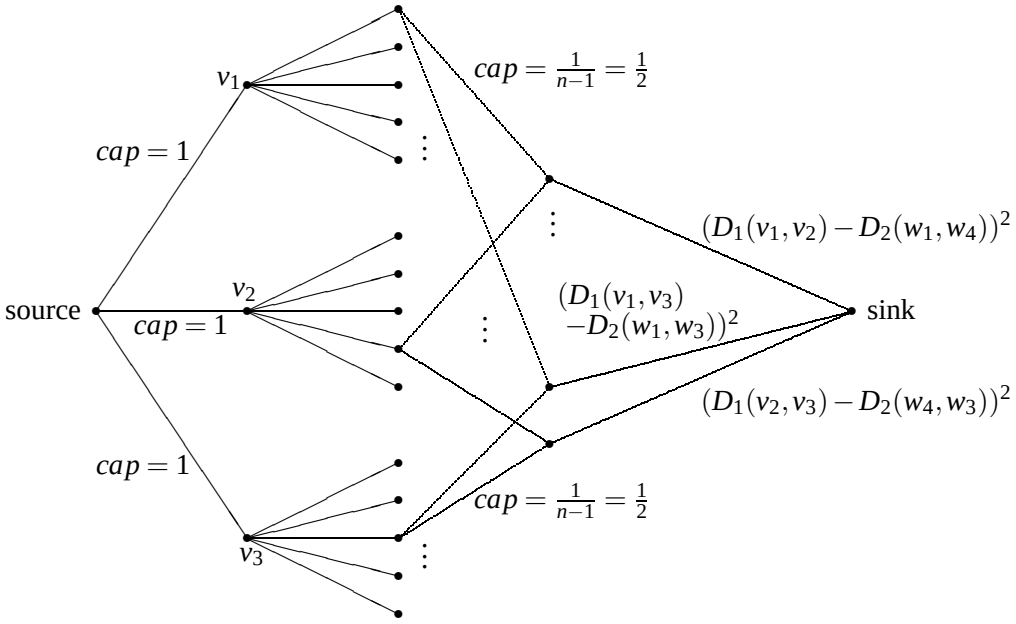

Figure 11: Part of graph for an exact solution of the best approximation problem. The submodular flow constraints as well as arc orientations are not indicated. Arcs are directed in the general direction "from left to right". The arcs for which cost labels are specified refer to the solution for the problem from Figure 1.

A best graph approximation amounts to a cost minimal flow of strength n from source to sink such that the flows along the edges between all v_i and (v_i, w_j) are integer. The other constraints then imply that the flows are binary over these edges.

4.2 Branch and Bound

The complicated structure of the foregoing flow problem motivates to organize an exact best graph approximation by branch and bound. Subisomorphisms will be built up sequentially by either pruning or refining a partial subisomorphism. A partial subisomorphism is a subisomorphism defined over a subset of the vertex set of the smaller graph. The domain of a partial subisomorphism is denoted by $A(V_1)$ and the partial subisomorphism itself is denoted by $\varphi|_{A(V_1)}$. The special case of the partial subisomorphism being defined over the complete vertex set of the smaller graph is denoted by $\varphi = \varphi|_{V_1}$.

A lower bound for the approximation distance of a partial subisomorphism can be obtained by independently minimizing distances between unassigned and assigned vertices. Formally, the lower bound is given by

$$\sum_{\{v_i,v_j\}\in E_1} \Big(D_1(v_i,v_j) - D_2(\varphi(v_i),\varphi(v_j))\Big)^2$$

$$\begin{aligned} &\geq \sum_{\{v_i,v_j\}\in E_1, v_i,v_j\in A(V_1)} \Big(D_1(v_i,v_j) - D_2(\varphi(v_i),\varphi(v_j))\Big)^2 + \sum_{v_0\in A(V_1)} c(v_0)^2 \\ &=: Val(\varphi|_{A(V_1)}), \end{aligned}$$

where $c(v_0) = \min_{v\in V_1 - A(V_1), w\in V_2 - \varphi(A(V_1))} |D_1(v_0,v) - D_2(\varphi(v_0),w)|$ for all $v_0 \in A(V_1)$.

Improved lower bounds can be constructed by cost minimal assignments in analogy to the heuristic distance list constructions of Section 3. Independent minimization over unassigned vertices is replaced by joint minimization. The vertex set of the bipartite graph for the assignment problem consist of both sets of unassigned vertices which are $V_1 - A(V_1)$ and $V_2 - \varphi(A(V_1))$. The cost values of the edges are given by squared errors of distance list approximations $DL(v,w)$, see Figure 12.

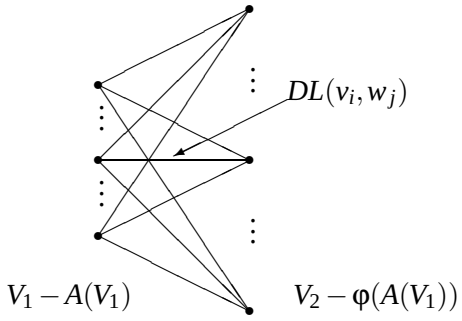

Figure 12: Assignment problem for improved lower bound of partial subisomorphism $\varphi|_{A(V_1)}$.

The improved lower bound is

$$\begin{aligned} &\sum_{\{v_i,v_j\}\in E_1} \Big(D_1(v_i,v_j) - D_2(\varphi(v_i),\varphi(v_j))\Big)^2 \\ &\geq \sum_{\{v_i,v_j\}\in E_1, v_i,v_j\in A(V_1)} \Big(D_1(v_i,v_j) - D_2(\varphi(v_i),\varphi(v_j))\Big)^2 + val(\varphi|_{A(V_1)}) \\ &=: Val^*(\varphi|_{A(V_1)}), \end{aligned}$$

where $val(\varphi|_{A(V_1)})$ is the minimum cost value of the lower bounding assignment problem from Figure 12. The bounding strategy of a branch and bound algorithm for best graph approximation can now be readily specified as follows.

Bound
Case $A(V_1) \neq V_1$.
If $Val^*(\varphi|_{A(V_1)}) \leq M$ then refine $\varphi|_{A(V_1)}$
else ignore $\varphi|_{A(V_1)}$. (prune or bound).
Case $A(V_1) = V_1$.
If $Val^*(\varphi) = M$ then $L_{opt} = L_{opt} \cup \{\varphi\}$.
If $Val^*(\varphi) < M$ then $L_{opt} = \{\varphi\}$ and $M = Val^*(\varphi)$.

The value M is the approximation distance of the best subisomorphism found so far and L_{opt} is a list of all these best subisomorphisms. This results in the following branch and bound approach.

The algorithm operates on a list U of unexplored partial subisomorphisms until this list becomes empty. The initial setting of this list consists of partial subisomorphisms that make exactly one assignment.

B+B

1. Input graphs G_1, G_2 with distances D_1, D_2.
   Initialization. M cost of arbitrary subisomorphism. $U = \{\varphi|_{v_1}(v_1) = w_1, \ldots, \varphi|_{v_1}(v_1) = w_m\}$.
   $L_{opt} = \emptyset$.
2. While $U \neq \emptyset$ do
   (a) Selection $\varphi|_{A(V_1)} \in U$.
   (b) $U = U - \{\varphi|_{A(V_1)}\}$.
   (c) Computation of $Val^*(\varphi|_{A(V_1)})$.
   (d) (Bound)
   If $A(V_1) = V_1$ then
   If $Val^*(\varphi) = M$ then $L_{opt} = L_{opt} \cup \{\varphi\}$.
   If $Val^*(\varphi) < M$ then $L_{opt} = \{\varphi\}$ and $M = Val^*(\varphi)$.
   (e) (Branch)
   If $Val^*(\varphi|_{A(V_1)}) < M$ then $U = U \cup \bigcup_{w_0 \in V_2 - \varphi(A(V_1))} \{\varphi|_{A(V_1)\cup\{v_0\}}$ with $\varphi|_{A(V_1)\cup\{v_0\}}(v_0) = w_0\}$ for one $v_0 \in A(V_1)$.
3. Output list of optimal subisomorphisms L_{opt}.

The branch and bound procedure is informal in so far as the selection step 2(a) and the branching step 2(e) leave many ways of specialization. Removal is specified implicitly here which means that a partial subisomorphism selected in step 2(a) is removed anyway in step 2(b). Refinements of the partial subisomorphism are possibly added to U in the branching step. Whenever a partial subisomorphism is not pruned by the bounding step, the next iteration of step 2 may or may not select a refinement of this partial subisomorphism to continue with.

5. Conclusion

Best graph approximation has been formulated for labeled graphs in analogy to isomorphism for unlabeled graphs. Polynomial time approximation algorithms in terms of linear assignment problems have been given and exact algorithms have been outlined. All algorithms are independent from application domains.

Whenever an instance graph is to be matched to several instead of one model graph, this can obviously be done sequentially. The best match is given by the minimum over all approximation distances and a ranking of the matching results is given by increasingly sorted approximation distances.

References

Ravindra, K. Ahuja, Thomas L. Magnanti and James Orlin. *Network Flows*. Prentice Hall, Englewood Cliffs, 1993.

Endika Bengoetxea. *Inexact graph matching using estimation of distribution algorithms*. Ph.D. dissertation, University of the Basque Country, San Sebastian, 2002.

Tiberio S. Caetano, Terry Caelli, Dale Schuurmanns and Dante A.C Barone. Graphical models and point pattern matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI*, forthcoming.

Pierre-Antoine Champin and Christine Solnon. Measuring the similarity of labeled graphs. In *Proceedings of the Fifth International Conference on Case-Based Reasoning*, pages 80-95, Springer, LNCS 2689, Berlin, 2003.

Richard Desper and Martin Vingron. Tree fitting: topological reconstruction from ordinary least-squares edge length estimates. *Journal of Classification*, 19:87-112, 2002.

Ian Foster and Carl Kesselman, editors. *The Grid: Blueprint to a new Computing Infrastructure*. 2nd ed., Elsevier, Amsterdam, 2004.

Satoru Fujishige. *Submodular Functions and Optimization*. North Holland, Amsterdam, 1991.

Michael R. Garey and David S. Johnson. *Computers and Intractability*. Freeman, San Francisco, 1981.

Steven Gold and Anand Rangarajan. A graduated assignment algorithm for graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 18:377-388, 1996.

Gene H. Golub and Charles F. van Loan. *Matrix Computations*. 4th printing, John Hopkins University Press, Baltimore, 1985.

Godefrey H. Hardy, John E. Littlewood and George Polya. *Inequalities*. Cambridge University Press, Cambridge, 1948.

Adel Hlaoui and Shengrui Wang. A new algorithm for inexact graph matching. In *Proceedings of the International Conference on Pattern Recognition ICPR'02*, pages 180-183, Quebec, 2002.

Thomas Kämpke. Scalable distance similarity of chemical structures. *Combinatorial Chemistry and High Throughput Screening*. 7:11-21, 2004.

Sergey Melnik, Hector Garcia-Molina and Erhard Rahm. Similarity flooding: a versatile graph matching algorithm and its application to schema matching. In *Proceedings of the 16th International Conference on Data Engineering ICDE*, 12 pages, San Diego, 2002.

Kurt Mehlhorn and Stephan Näher. *LEDA A Platform for Combinatorial and Geometric Computing*. Cambridge University Press, Cambridge, 2000.

Bruno T. Messmer and Horst Bunke. A new algorithm for error-tolerant subgraph isomorphism detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI*, 20:493-504, 1998.

Bruno T. Messmer and Horst Bunke. A decision tree approach to graph and subgraph isomorphism detection. *Pattern Recognition* 32:1979-1998, 1999.

Apostolos N. Papadopoulos and Yannis Manolopoulos. Structure-based similarity search with graph histograms. In *Proceedings of the 10th International Workshop on Database and Expert System Applications DEXA*, pages 174-178, 1999.

Arthur R. Pope and David G. Lowe. Learning appearance models for object recognition. In *Proceedings of International Workshop on Object Representation for Computer Vision*, pages 201-219, Springer, Berlin, 1996.

Dennis Shasha, Jason T.L. Wang and Rosalba Giugno. Algorithmics and applications of tree and graph searching. In *Proceedings of the Symposium on Principles of Database Systems*, pages 39-52, 2002.

Steven S. Skiena and Gopalakrishnan Sundaram. A partial digest approach to restriction site mapping. *Bulletin of Mathematical Biology*, 56:275-294, 1994.

Jason T.L. Wang, Bruce A. Shapiro, Dennis Shasha, Kaizhong Zhang and Kathleen M. Currey. An algorithm for finding the largest approximately common substructures of two trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20:889-895, 1998.

Laurenz Wiskott, Jean-Marc Fellous, Norbert Krüger, Christoph Malsburg. Face recognition by elastic bunch graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI*, 19:775-779, 1997.

Laurenz Wiskott and Christoph Malsburg. Labeled bunch graphs for image analysis. US patent no. 6,356,659, 2002.