Efficient Approximation of Labeling Problems with Applications to Immune Repertoire Analysis

Yusuf Osmanlıoğlu*, Santiago Ontañón*, Uri Hershberg†, Ali Shokoufandeh*
{osmanlioglu, santi, ashokouf}@cs.drexel.edu , uri.hershberg@drexel.edu
*Dept. of Computer Science, Drexel University, Philadelphia,PA.
†Dept. of Biomedical Engineering, Science & Health Systems, Drexel University, Philadelphia,PA.

***Abstract*—Labeling problems are finding increasing applications to optimization problems. They usually get realized into linear or quadratic optimization problems, which are inefficient for large graphs. In this paper we propose an efficient primal-dual solution, ML_{PD} , for a family of labeling problems. We apply this algorithm to the analysis of immune repertoires, and compare it against our baseline approach based on refinement operators. We provide a comparative evaluation both in terms of accuracy and computational efficiency with respect to the baseline model, as well as to quadratic optimization.**

I. Introduction

Graph matching is a fundamental problem in computer science having real life applications in various domains of structural pattern recognition. Given two graphs G and H, determining an exact match among the nodes is known to be computationally intractable. Specifically, *subgraph isomorphism* which asks whether G contains an induced subgraph H' that is isomorphic to H, is a problem known to be NP-complete [1]. *Graph isomorphism*, which is a special case of the problem that is neither known to be polynomial time solvable nor NP-complete, is recently shown to be solvable in quasipolynomial time [2]. Hardness of finding and exact solution to the problem led the research to focus on *inexact matching* since early 80s [3].

We tackle inexact matching by utilizing *metric labeling*, a well known problem of combinatorial optimization [4]. Given an object graph and a label graph with pairwise relations among their nodes, the goal of metric labeling is to assign object nodes to label nodes by minimizing a quadratic cost function. Solving the quadratic optimization problem provides two outcomes. First is a mapping between individual nodes which has applications such as object tracking. Second is the value of the objective function which can be used as a similarity score between the two graphs for classification tasks. In this paper, we use the latter for a labeling task in the biomedical domain.

Solving the quadratic programming formulation of metric labeling becomes impractical as the size of graphs increase. Utilizing the linear programming formulation of the metric labeling [5], we propose a primal dual approximation algorithm, ML_{PD} , for the problem which is several orders of magnitude faster. We compare this approach against both a quadratic programming formulation, and an approach based on refinement operators. The refinement operator approach has shown to provide good results in the past for assessing the similarity between structures in representation formalisms such as typed feature structures [6] or Description Logics [7]. In this paper we provide an instantiation of this approach for labeled graphs, which serves as a baseline for ML_{PD} . We present an evaluation in the domain of immunology both in terms of labeling accuracy and running time.

The rest of the paper is organized as follows: §II gives an overview definitions. In §V, we define the classification task over B-cell mutation trees. §III and §IV provides the details of the approximation algorithm for the metric labeling and the refinement based approach. Next, we present the experiments in §VI and conclude the document in §VII.

II. Background

Inexact graph matching is a well studied problem in the theory community with several approaches including tabu search [8], error-correcting graph matching [9], graph edit distance based matching [10], and convex optimization formulations [11]. Another way to treat the problem is by approaching the inexact matching as a classification task. A fundamental set of problems in computer science deals with classifying a set of objects into clusters by minimizing a cost function. For problems such as *multiway cut problem* of Dahlhaus et al. [12] and 0-*extension problem* of Karzanov [13], the cost function can be as simple as counting the separation cost of the related objects. In other settings with the prior knowledge about cluster labels, the cost function also accounts for the assignment cost of an object to a specific cluster. The *quadratic assignment problem* [14] is an example of the latter with its various flavors. In its original form as devised by Koopmans and Beckmann, it can be stated in the context of finding optimal locations to facilities given two disjoint sets $\mathcal{L}$ and $\mathcal{P}$ of cardinality n. The main goal is to find a bijection between the two sets by assigning facilities to the locations with minimum assignment cost while minimizing the cost of transportation between facilities. *Metric labeling problem* of Kleinberg and Tardos [4] is a special case of the quadratic assignment where the two sets need not be of same size and several elements of the first set can be assigned to the same element of the second set. Specifically, given a set of objects P and a set of labels L with pairwise relationships defined among the elements of both sets, metric labeling assigns a label to each object by minimizing a cost function involving both

separation and assignment costs. Separation cost penalizes assigning loosely related labels to closely related objects while assignment cost penalizes labeling an object with an unrelated label. A natural quadratic programming formulation for the metric labeling problem is as follows:

$$\begin{aligned} \min \quad & \sum_{\substack{p\in \mathrm{P}\\ a\in \mathrm{L}}} c_{pa}\cdot x_{pa} + \sum_{\substack{p,q\in \mathrm{P}\\ a,b\in \mathrm{L}}} w_{pq}\cdot d_{ab}\cdot x_{pa}\cdot x_{qb} \\ \text{s.t.} \quad & \textstyle\sum_{a\in \mathrm{L}} x_{pa} = 1, && \forall p\in \mathrm{P} \\ & x_{pa}\in\{0,1\}, && p\in \mathrm{P}, a\in \mathrm{L} \end{aligned}$$

where, $c_{p,a}$ represents the cost of labeling an object $p \in \mathrm{P}$ with a label $a \in \mathrm{L}$, $w_{p,q}$ is the strenght of relation between objects $p, q \in \mathrm{P}$, and $d_{a,b}$ is a distance measure on the set L of labels. In order to eliminate the quadratic term in the separation cost, Kleinberg and Tardos used embedding of label graph into a special tree structure, called hierarchically well-separated trees (HST) [15], and measured the distances between labels over the tree albeit with the cost of $O(\log n)$ distortion. Although, there has been ample studies on solving classification problems using labeling methods, this work was the first study that provided a polynomial-time approximation algorithm with a nontrivial performance guarantee.

Although the linear programming (LP) formulation of Kleinberg and Tardos is very elegant in using the HST embedding for overcoming the quadratic term in the objective function, time it takes to embed the object graph into an HST and the large number of constraints in the resulting program limit the scalability of the method for large graphs. Chekuri et al.[5] introduced another LP formulation for the metric labeling which overcomes both problems while having the same distortion guarantee.

$$\begin{aligned} \min \quad & \sum_{\substack{p\in \mathrm{P}\\ a\in \mathrm{L}}} c_{pa}\cdot x_{pa} + \sum_{\substack{p,q\in \mathrm{P}\\ a,b\in \mathrm{L}}} w_{pq}\cdot d_{ab}\cdot x_{paqb} \\ \text{s.t.} \quad & \textstyle\sum_{a\in \mathrm{L}} x_{pa} = 1, && \forall p\in \mathrm{P} && \text{(1a)} \\ & \textstyle\sum_{b\in \mathrm{L}} x_{paqb} = x_{pa}, && \forall p\neq q\in \mathrm{P}, a\in \mathrm{L} && \text{(1b)} \\ & x_{paqb} = x_{qbpa}, && \forall p\neq q\in \mathrm{P}, a,b\in \mathrm{L} && \text{(1c)} \\ & x_{pa}, x_{paqb}\in\{0,1\}, && \forall p,q\in \mathrm{P}, a,b\in \mathrm{L} \end{aligned}$$

Since integer linear programs (ILP) are known to be NP-hard to solve, a feasible solution to (1) can be obtained by first relaxing the integrality condition of integer program and then solving the resulting LP. Fractional results of the LP are then rounded to discrete values with an auxiliary rounding procedure. Thus, it is necessary to find a suitable rounding algorithm with a bounded distortion guarantee.

Goemans and Williamson [16] introduced the primal-dual approximation scheme for finding approximate solutions to many combinatorial optimization problems that can be modeled with ILPs. Their method is a modified version of the classical primal-dual algorithm in that complementary slackness conditions are relaxed, i.e., only primal complementary slackness condition is imposed while the dual complementary slackness is relaxed. The algorithm achieves an approximation to the original problem and a feasible solution to its dual simultaneously. The main advantage of using modified primal-dual method over classical LP relaxation followed by rounding is twofold. First, it runs faster while guaranteeing a bounded distortion rate. Second, if the data changes during the course of algorithm running, the algorithm can recover by updating only the newly violated constraints without the need for solving the problem from scratch.

III. Approximation Algorithm for Matching

Taking shortest path metric as the distance measure between the nodes, problem of finding a similarity measure among graphs can be treated as an instance of metric labeling. Specifically, given two graphs P and L, distance function $d : \mathrm{L}\times\mathrm{L}\to\mathbb{R}$ denoting the shortest path between nodes of L, weight function $w : \mathrm{P}\times\mathrm{P}\to\mathbb{R}$ defined as the reciprocal of shortest path distance between pairs of nodes in P, and an assignment cost function $c : \mathrm{P}\times\mathrm{L}$ defining a measure of similarity among pairs of nodes of P and L, we can restate the problem as labeling nodes of P with nodes of L. Solving the metric labeling formulation in (1) with this setup provides a matching between individual nodes along with a similarity score between the two trees.

Due to its efficiency over standard approach for solving metric labeling which consists of solving the LP and then rounding the fractional results, one would be interested in a primal-dual approximation algorithm for the metric labeling problem. Although Komadakis and Tziritas [17] proposed a primal dual algorithm for the problem, the LP formulation of the metric labeling that they used does not account for the constraints of type (1c). We propose another primal dual algorithm using the LP formulation (1). We first present the dual problem as follows:

$$\begin{aligned} \max \quad & \sum_{p\in \mathrm{P}} y_p \\ \text{s.t.} \quad & y_p - \sum_{q\in \mathrm{P}} y_{paq} \le c_{pa}, && \forall p\neq q\in \mathrm{P}, a\in \mathrm{L} && \text{(2a)} \\ & y_{paq} \pm y_{paqb} \le w_{pq}\cdot d_{ab}, && \forall p,q\in \mathrm{P}, a,b\in \mathrm{L} && \text{(2b)} \\ & y_p,\ y_{paq},\ y_{paqb} \text{ unrestricted}, && \forall p,q\in \mathrm{P}, a,b\in \mathrm{L} \end{aligned}$$

Note that variables of type y_{paqb} appears as a summation and a subtraction in (2b) type of constraints which accounts for the balancing constraints in (1c). Strictly following the primal dual method of G&W, which enforces dual feasibility throughout the algorithm, would require us to make assignments in tuples. Specifically, once an object is assigned a label, another object will be required to get assigned with the same label. Algorithm will thus assign two objects at each phase of its iteration, resulting in a poor overall assignment. This leads us to modify G&W primal dual method by relaxing the dual feasibility condition for the constraints of type (2a) which previously became tight.

An efficient primal-dual approximation algorithm for the problem, denoted ML_{PD} , is presented in Alg. 1. The algorithm starts with initializing assignment variables x_{pa}, set

Algorithm 1 ML_{PD} : A primal-dual approximation algorithm for the metric labeling problem

procedure ML-Primal-Dual(P, L)

1: $\forall p, q \in \mathrm{P}, a \in \mathrm{L} : x_{pa} \leftarrow 0, \mathcal{O} \leftarrow \mathrm{P}$
$c_{pa} \leftarrow similarity(p, a)$
$d_{ab} \leftarrow distance_{\mathrm{L}}(a, b)$
$w_{pq} \leftarrow 1/distance_{\mathrm{P}}(p, q)$
$\phi(p, a) \leftarrow c_{pa}$
2: **while** $\mathcal{O} \neq \emptyset$ **do**
3: Find $p \in \mathcal{O}$ that minimizes $\phi(p, a)$ for some $a \in \mathrm{L}$
4: $x_{pa} \leftarrow 1$
5: $\mathcal{O} \leftarrow \mathcal{O} \setminus \{p\}$
6: $\forall q \in \mathcal{O}, b \in \mathrm{L} \setminus \{a\} : \phi(q, b) = \phi(q, b) + w_{pq} \cdot d_{ab}$
7: **end while**
8: **return** $\mathcal{X} = \{x_{pa} : \forall p \in \mathrm{P}, a \in \mathrm{L}\}$

of unlabeled objects $\mathcal{O}$, and the cost, distance, and pairwise relation functions c, d and w. It further defines an adjusted assignment cost function ϕ where the value of $\phi(p, a)$ is initially set to be the assignment cost of p to a (line 1). At each iteration of the loop in lines $2-7$, the algorithm makes an assignment for the object-label pair (p, a) that minimizes the adjusted assignment cost function ϕ (lines $3-4$). Before proceeding to the next iteration, assigned object is removed from the set $\mathcal{O}$ (line 5) and ϕ function is updated for each of the unassigned objects by an amount of separation cost with respect to the recently assigned object (line 6). Algorithm iterates until no unassigned object node is left.

Proposition 1. *Given that* $|\mathrm{P}| = n$ *and* $|\mathrm{L}| = m$*, running time complexity of Alg. 1 is* $O(n^2m + m^2)$.

Proof. Initialization phase takes $O(n^2+m^2+nm)$ time. Each iteration of the loop resolves one violation of constraint type (1a) and once a violation gets fixed, it will never get violated again for the rest of the execution. Thus, the loop is executed n times, and lines $2, 4$, and 5 each take $O(n)$ time. Making an aggregate analysis for lines 3 and 6, we get $O(n^2)$ and $O(n^2m)$ time, respectively. Thus, overall running time of the algorithm is $O(n^2+m^2+nm+n^2m)$ which is asymptotically equal to $O(n^2m + m^2)$. □

Remark 1. *Alg. 1 is a greedy algorithm since the pair* (p, a) *which minimizes the adjusted assignment cost* $\phi(p, a)$ *is selected for assignment at each iteration.*

Remark 2. *In Alg. 1,* $\phi(p, a)$ *is not updated for an object node* p *once it gets assigned. At the end of the algorithm, summation* $\sum_{p,a} \phi(p, a)x_{pa}$ *gives the value of the objective function in (2) which is equal to the value of the objective function in (1).*

IV. Refinement-based Similarity Function

Recent work [6] has shown that refinement operators [18] can be used as a general framework to assess similarity between structured data representations such as typed feature structures or logical clauses. In this section, we present an instantiation for this approach to assess similarity between labeled graphs (the formalism used to represent the data used in our experiments), which will later be used as a baseline to compare against ML_{PD} in our experimental evaluation.

A. Refinement of Labeled Graphs

Definition 1 (Directed Labeled Graph, DLG)**.** *Given a finite set of labels* L*, a directed labeled graph* G *is defined as a tuple* $G = \langle V, E, l \rangle$*, where:*

- $V = \{v_1, ..., v_n\}$ *is a finite set of vertices.*
- $E = \{(v_{i_1}, v_{j_1}), ..., (v_{i_m}, v_{j_m})\}$ *is a finite set of edges.*
- $l : V \cup E \rightarrow L$*, is a function that assigns a label from* L *to each vertex and edge.*

For simplicity of notation, given a graph $G = \langle V, E, l \rangle$, and a vertex $v \in V$, we would also say that $v \in G$.

Given two graphs g_1 and g_2, we say that g_1 *subsumes* g_2 (noted $g_1 \sqsubseteq g_2$), if g_2 contains a subgraph that is equivalent to g_1. Formally subsumption is defined as follows:

Definition 2 (Subsumption)**.** *Given two DLGs,* $g_1 = \langle V_1, E_1, l_1 \rangle$ *and* $g_2 = \langle V_2, E_2, l_2 \rangle$*,* g_1 subsumes g_2 *(we write* $g_1 \sqsubseteq g_2$*) if there is a mapping* $m : V_1 \rightarrow V_2$ *such that:*

- $\forall (v, w) \in E_1 : (m(v), m(w)) \in E_2$*,*
- $\forall v \in V_1 : l_1(v) = l_2(m(v))$*, and*
- $\forall (v, w) \in E_1 : l_1((v, w)) = l_2((m(v), m(w)))$.

When a graph g_1 subsumes a graph g_2, we will say that g_1 is *more general* than g_2 (or, alternatively, that g_2 is *more specific* than g_1). This follows the intuition that if g_1 subsumes g_2, we can construct g_2 by *adding* vertexes and edges to g_1 (i.e., making it more specific).

Definition 3 (Anti-unification)**.** *Given two graphs* g_1*, and* g_2*, and a subsumption relation* $\sqsubseteq$ *(which can be any of the ones defined above),* g *is an* anti-unifier *of* g_1 *and* g_2 *(we write* $g = g_1 \sqcap g_2$*) if:* $g \sqsubseteq g_1$*,* $g \sqsubseteq g_2$*, and* $\nexists g' \sqsupset g : g' \sqsubseteq g_1 \ \wedge \ g' \sqsubseteq g_2$.

In other words, an anti-unifier of two graphs is the most specific graph that subsumes both of them. Let us call G to the infinite set of all the possible graphs we can create using a set of labels L. The subsumption relation $\sqsubseteq$ is a partial order over these graphs, and thus $(G, \sqsubseteq)$ is a quasi-ordered set. This allows us to define *refinement operators*, which intuitively are functions that let us *edit* graphs by either adding or removing elements from them to make them more specific (downward refinement operators) or more general (upward refinement operators). A downward refinement operator is defined as:

Definition 4 (Downward Refinement Operator)**.** *A* downward refinement operator *over a quasi-ordered set* $(G, \sqsubseteq)$ *is a function* $\rho : G \rightarrow 2^G$ *such that* $\forall g' \in \rho(g) : g \sqsubseteq g'$.

We create a refinement operator for labeled graphs called ρ_f. Given a DLG $g = \langle V, E, l \rangle$, ρ_f generates refinements by performing one of the following four operators: (1) if $E = \emptyset$, it refines it by adding one vertex labeled with a label $l \in L$, (2) picking a vertex $v \in V$, and adding a new vertex w to the graph and an edge form v to w, and giving both the new edge

and vertex labels from L, (3) the same thing but making the edge go from w to v, and (4) adding a new edge between two vertices in the graph, and giving it a label $l \in L$. Although a formal definition and proof is out of the scope of this paper, it can be shown that ρ_f is *locally finite* (given a graph, it generates a finite number of refinements) and *proper* (given a graph, none of the refinements it generates are equivalent to the original graph).

Definition 5 (Refinement Path)**.** *A finite sequence of graphs* $[g_1, ..., g_n]$ *is a* refinement path $g_1 \xrightarrow{\rho} g_n$ *between* g_1 *and* g_n *when for each* $1 \leq i < n$, $g_{i+1} \in \rho(g_i)$.

We will write $|g_1 \xrightarrow{\rho} g_2|$ to denote the length (# of refinements) of the shortest path between g_1 and g_2.

B. Similarity Function

In our previous work [6], we introduced the concept of refinement-based similarity measures in the context of feature-terms [19] (a representation formalism used in structured machine learning and in natural language processing), and later extend this idea to other formalisms such as Description Logics [7], and partial-order plans [20]. Here, we extend these ideas further to directed labeled graphs (DLGs) by using the refinement operator introduced above.

The intuition of the refinement-based similarity function presented here is that the amount of information contained in a graph can be measured by the number of times a refinement operator needs to be applied in order to generate such graph starting from the empty graph ($g_\top$).

Definition 6 (Anti-unification-based Similarity)**.** *Given two graphs* g_1*, and* g_2*, a refinement operator* ρ *and a subsumption relation* $\sqsubseteq$*, the* anti-unification-based similarity *is defined as:*

$$S_\lambda(g_1, g_2) = \frac{a}{a + b + c}$$

where $a = |g_\top \xrightarrow{\rho} (g_1 \sqcap g_2)|$, $b = |(g_1 \sqcap g_2) \xrightarrow{\rho} g_1|$, *and* $c = |(g_1 \sqcap g_2) \xrightarrow{\rho} g_1|$.

Intuitively, this measures the amount of information shared between g_1 and g_2 (size of their anti-unifier), and normalizes it by the total amount of information. The reader is referred to our previous work [6] for a more in-depth description and analysis of the anti-unification-based similarity. Moreover, the distance between two graphs can be computed as $1 - S_\lambda(g_1, g_2)$, since S_λ is normalized to the interval $[0, 1]$.

V. Description of the Problem and the Data

In order to evaluate the proposed techniques, in this paper we focus on an important biomedical problem, namely the analysis of immune repertoire diversification and selection. The adaptive immune response depends on the differentiation and affinity maturation of B cells. Each of these cells encodes for a specific B cell receptor. During an immune response, B cells proliferate, mutate, and die, such that the surviving B cells have receptors of improved affinity to the disease that triggers the immune response. Mutant members can be associated to their unmutated source and a lineage can be constructed from their common mutation patterns. Recent high throughput sequencing techniques enable the acquisition of sequences which span different tissues and B cell subset types. We can use the methods described in this paper to test if these different functional and phenomenological differences in B cell type relate to the patterns of mutation and selection described by the shape of the lineage.

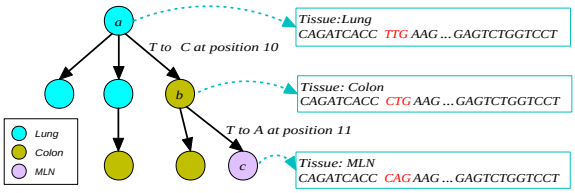

Fig. 1. Structure of a sample mutation tree: each node in the tree denotes an immune cell along with the germline of the cell. Each edge represents a mutation of a nucleotide at a certain position in the germline. Each of the cells are color coded according to the tissue they were encountered in.

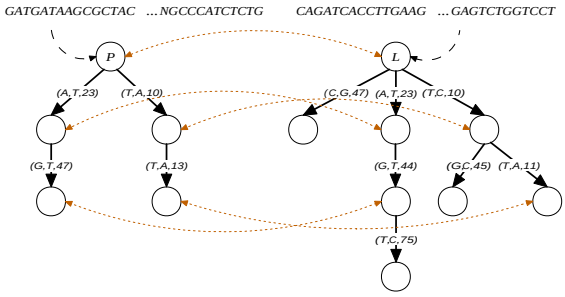

Fig. 2. Matching two mutation trees: nodes of the tree P is labeled with the nodes of the tree L. Note that tissue labels are removed before matching.

An example of one such lineage tree is shown in Figure 1. Each edge in the tree represents a set of mutations, and each node contains the set of cells that share such mutation. Each node in the tree is thus labeled by which tissues they were found on and the *copy number*, i.e., the number of cells found sharing the corresponding mutations. Each edge is labeled with the specific mutations (only the number of such mutations is used for the experiments reported in this paper, although we plan to incorporate more information in our future work). Specifically, we analyzed 5000 B cell clone lineages of size 3 to 50 nodes, found across up to 7 tissues types (bone marrow, colon, lung, etc.) selected at random from the a wider set of immune repertoires of 9 individuals.

The proposed graph matching methods can be applied to finding similarities between lineage trees due to trees being special graph structures (see Fig. 2). Adapting the ML_{PD} to the lineage tree dataset requires defining similarity and distance functions since the assignments made by the algorithm at line 3 rely on the distance measure d_{ab} and the similarity function c_{pa}. We take the distance of an edge between two neighboring nodes as the number of mutated nucleotides. Shortest path distance is then used as the distance measure between any two nodes in the tree. For calculating

the similarity between nodes, we defined a two-dimensional feature vector for each node which consists of the copy number and the depth of a node in the tree. Similarity score is then calculated as the l_1 distance between the two feature vectors.

VI. Experiments

This section presents an empirical evaluation of the proposed graph similarity approaches. For each lineage tree, we generated a *ground truth* label which we call the *tissue flow* (TF) label. The TF label specifies how the cells described by the lineage tree flow between tissues as they mutate. For example, if all the cells in the root of a lineage tree were found in the lung, and the cells in the leaves of the tree where found in the spleen, the TF label for this lineage tree would be "lung to spleen". If the tree were to have additional leaves containing cells found in colon, then the label would be "lung to spleen and colon". The TF is represented by a small tree where each node is labeled with a tissue or set of tissues (since some cells might be found in a mixture of tissues).

So, the prediction task is: given a lineage tree (like the one in Figure 2), predict the TF label in the absence of tissue labels. Basically, this corresponds to asking the question: are the mutation patterns (captured by the shape of the lineage tree) correlated in any way with the tissues in which the different cells in the lineages are found.

When generating TF labels for the 5000 trees in our dataset, this results in 1190 different TF labels. Moreover, some of those labels are very similar. For example, one tree might be labeled with a TF label "lung to spleen and colon" and the other with "lung to spleen". So, in our experiments, we score each prediction by the similarity (using S_λ) of the predicted TF with the ground truth. Moreover, since some TF labels are very uncommon, we also performed experiments by removing all of those trees from the dataset for which there was less than a minimum threshold T of 10, 50, 100 or 400 trees sharing the same label.

In order to evaluate our similarity approaches, we employed then a k-nearest neighbor (kNN) algorithm using a *leave-one-out* procedure. Moreover, all the tissue annotations were removed from the trees for these experiments, in order to ensure we are predicting TF and tissue distribution from the tree structure alone. When using kNN for predicting TF, the most common label amongst the kNNs is used as the final prediction.

Results of the experiment with ML_{PD} and S_λ algorithms are shown in Table I. We compared the results against two baselines: a *random* predictor, which picks one of the labels at random, and the *most-common*, which predict the most common label in the training set. As can be seen in Table I, results improve with higher values of k, meaning that there is a significant amount of either uncertainty or noise in the dataset, which higher values of k can compensate for. As a matter of fact, we observed that results keep improving up to values of $k = 31$, after which they plateau, and then drop quickly to those of the *most-common* predictor, as expected. We only report results for $k \in \{1, 11, 31\}$ since it suffices for

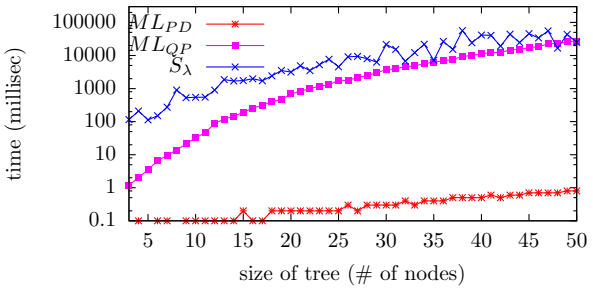

Fig. 3. Performance comparison of ML_{PD} ,ML_{QP} , and S_λ. Algorithms are run with pairs of same size trees and results are log scaled.

comparison purposes. Moreover, we also observe that in terms of classification accuracy, ML_{PD} and S_λ outperform both of the baseline methods where S_λ performs slightly better. We also compare ML_{PD} to the algorithm that solves metric labeling using quadratic programming solver, denoted ML_{QP} . Due to high computational complexity of the ML_{QP} , we ran an experiment on a subsampled dataset of 1000 trees. ML_{QP} achieves accuracy rates of 0.156 and 0.204 for 1 and 31 kNN while ML_{PD} achieving 0.191 and 0.211, respectively. One would expect ML_{QP} to outperform ML_{PD} since the the latter is an approximation of the former. We conjecture that ML_{QP} performs relatively poor due to feature vectors carrying limited information, which causes assignment cost of unrelated nodes to be small. Heuristic that ML_{PD} pursues compensates this inaccuracy of features for obtaining better results. We expect ML_{QP} to perform better with more descriptive feature vectors, which we leave as a future work.

Considering the computational cost, S_λ required 9322 minutes (over 155 hours) to generate the 5000×5000 distance matrix used in our experiments, whereas the metric labeling approach required 54 minutes, making the latter method over 150 magnitudes of order faster then the former for the entire task. Although the tree sizes vary between 3 to 50, majority of the trees are of small sizes, making the average tree size to be 6.88. To obtain a detailed timing analysis, we performed another experiment where we run all three algorithms to compare trees of same size from 3 to 50 nodes. Fig. 3 shows the timing measurements in milliseconds where the results are presented in log scale. While the running time of the ML_{PD} is less than 2 ms for all trees, it increases exponentially for S_λ and ML_{QP} reaching up to 60 seconds for the largest trees in the dataset.

Moreover, in order to get more insight into how well the different labels are separated by our similarity measures, we studied the average *intra* and *inter* label distance. Given a label l, for each instance i with label l, we compute the distance to the nearest other instance with label l. The *intra* label distance of l is the average of all such distances. The *inter* label distance of a label l is the same, but finding the nearest instance with *different* label. Ideally, the *inter/intra* label distance ratio should be higher than 1, and the higher the better. The top of Table II shows the *inter/intra* ratio for

TABLE I

TISSUE FLOW PREDICTION ACCURACY (S_λ SIMILARITY BETWEEN PREDICTED AND GROUND TRUTH) FOR DIFFERENT THRESHOLDS T (MINIMUM NUMBER OF INSTANCES WITH A GIVEN LABEL IN THE DATASET), RESULTING IN DIFFERENT DATASET SIZES ($|D|$) AND NUMBER OF LABELS ($|L|$).

T	$\|D\|$	$\|L\|$	random	most-common	S_λ			ML_{PD}		
					1-kNN	11-kNN	31-kNN	1-kNN	11-kNN	31-kNN
-	5000	1190	0.093	0.164	0.188	0.221	0.230	0.180	0.208	0.221
10	3393	48	0.124	0.209	0.235	0.273	0.279	0.232	0.270	0.284
50	2764	12	0.128	0.234	0.249	0.294	0.309	0.250	0.298	0.319
100	2292	6	0.161	0.272	0.288	0.334	0.351	0.290	0.356	0.372
400	1183	2	0.495	0.528	0.603	0.690	0.690	0.592	0.638	0.639

TABLE II

inter/intra LABEL DISTANCE RATIO FOR DIFFERENT DATA SUBSETS AS COMPUTED BY THE S_λ AND THE ML_{PD} DISTANCES (HIGHER IS BETTER).

	S_λ	ML_{PD}
$\geq$10 with same label	1.916	1.549
$\geq$50 with same label	1.627	1.403
$\geq$100 with same label	1.314	1.272
"PBL" vs rest	1.666	2.245
"Lung" vs rest	1.500	1.458
"BM to MLN" vs rest	1.086	1.047
"MLN to mixed MLN, Jejunum" vs rest	1.112	1.105

subsets of all of those trees for which there was less than a minimum of 10, 50 or 100 trees sharing the same label, showing that in average labels tend to be well separated: a ratio of 1.916 means that, in average, the nearest neighbor with a different label is 1.916 times further than the nearest neighbor with the same label. Additionally, we also show (in the bottom three rows rows of Table II) the two labels that have the best ratio, and the two that have the lowest ratio. This shows that there are some labels that are very well separated by both of our similarity measures, and some that are not, which indicates that the structure of the lineage trees might not contain enough information to distinguish these labels.

VII. CONCLUSION AND FUTURE WORK

In this paper we proposed an efficient primal-dual approximation algorithm, ML_{PD} , for a family of labeling problems. We demonstrated the efficiency of our method on the analysis of immune repertoires, and compared it against baseline approaches based on refinement operators and quadratic programming formulation. One of the major factors effecting the accuracy of matching is the descriptive nature of the node features. Adding further features pertaining to the genome sequence is among our future plans such as the dissimilarity between the genetic sequence of the two nodes at the level of nucleotides, regions of genome, and aminoacids.

ACKNOWLEDGMENT

This work is supported in part by support of the National Science Foundation under Grant Number IIS-1551338.

REFERENCES

[1] S. A. Cook, "The complexity of theorem-proving procedures," in *Proceedings of the third annual ACM symposium on Theory of computing*. ACM, 1971, pp. 151–158.

[2] L. Babai, "Graph isomorphism in quasipolynomial time," *arXiv preprint arXiv:1512.03547*, 2015.

[3] L. G. Shapiro and R. M. Haralick, "Structural descriptions and inexact matching," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, no. 5, pp. 504–519, 1981.

[4] J. Kleinberg and E. Tardos, "Approximation algorithms for classification problems with pairwise relationships: Metric labeling and markov random fields," *J. ACM*, vol. 49, no. 5, pp. 616–639, Sep. 2002.

[5] C. Chekuri, S. Khanna, J. Naor, and L. Zosin, "A linear programming formulation and approximation algorithms for the metric labeling problem," *SIAM Journal on Discrete Mathematics*, vol. 18, no. 3, pp. 608–625, 2004.

[6] S. Ontañón and E. Plaza, "Similarity measures over refinement graphs," *Machine Learning Journal*, vol. 87, pp. 57–92, 2012.

[7] A. A. Sanchez-Ruiz, S. Ontañón, P. A. Gonzalez-Calero, and E. Plaza, "Refinement-based similarity measure over dl conjunctive queries," in *Proceedings of ICCBR 2013*, 2013, pp. 270–284.

[8] M. L. Williams, R. C. Wilson, and E. R. Hancock, "Deterministic search for relational graph matching," *Pattern Recognition*, vol. 32, no. 7, pp. 1255–1271, 1999.

[9] H. Bunke, "Error correcting graph matching: On the influence of the underlying cost function," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 21, no. 9, pp. 917–922, 1999.

[10] S. Berretti, A. Del Bimbo, and P. Pala, "A graph edit distance based on node merging," in *Image and Video Retrieval*. Springer, 2004, pp. 464–472.

[11] H. Almohamad and S. O. Duffuaa, "A linear programming approach for the weighted graph matching problem," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 15, no. 5, pp. 522–525, 1993.

[12] E. Dahlhaus, D. S. Johnson, C. H. Papadimitriou, P. D. Seymour, and M. Yannakakis, "The complexity of multiway cuts (extended abstract)," in *Proceedings of the* 24^{th} *Annual ACM Symposium on Theory of Computing*, ser. STOC '92. New York, NY, USA: ACM, 1992, pp. 241–251. [Online]. Available: http://doi.acm.org/10.1145/129712.129736

[13] A. V. Karzanov, "Minimum 0-extensions of graph metrics," *Eur. J. Comb.*, vol. 19, no. 1, pp. 71–101, Jan. 1998. [Online]. Available: http://dx.doi.org/10.1006/eujc.1997.0154

[14] T. C. Koopmans and M. Beckmann, "Assignment problems and the location of economic activities," *Econometrica: journal of the Econometric Society*, pp. 53–76, 1957.

[15] Y. Bartal, "Probabilistic approximation of metric spaces and its algorithmic applications," in *Proceedings of the* 37^{th} *Annual Symposium on Foundations of Computer Science*, ser. FOCS '96. Washington, DC, USA: IEEE Computer Society, 1996, pp. 184–. [Online]. Available: http://dl.acm.org/citation.cfm?id=874062.875536

[16] M. X. Goemans and D. P. Williamson, "The primal-dual method for approximation algorithms and its application to network design problems," in *Approximation Algorithms for NP-hard Problems*, D. S. Hochbaum, Ed. Boston, MA, USA: PWS Publishing Co., 1997, pp. 144–191.

[17] N. Komodakis and G. Tziritas, "Approximate labeling via graph cuts based on linear programming," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 29, no. 8, pp. 1436–1453, 2007.

[18] P. R. van der Laag and S.-H. Nienhuys-Cheng, "Completeness and properness of refinement operators in inductive logic programming," *The Journal of Logic Programming*, vol. 34, no. 3, pp. 201–225, 1998.

[19] B. Carpenter, *The Logic of Typed Feature Structures*, ser. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1992, vol. 32.

[20] A. A. Sanchez-Ruiz and S. Ontañón, "Least common subsumer trees for plan retrieval," in *Proceedings of ICCBR 2014*, 2014, pp. 405–419.