1. SIGNIFICANCE

Over the last decade, the multidisciplinary team (MDT) management of diabetes has increasingly taken a prominent role in patient management in many medical centers in the US [2, 6]. A key to the success of the team-based approach is to manage and correlate *medical knowledge* from diverse domains and effectively share it among the clinicians in the same team [7, 8]. However, with the advent of biomedical technologies, current diabetes study becomes constrained by the lack of capabilities to handle and share large and rapidly growing information in a MDT [3]. There are two major technology challenges that prevent knowledge sharing: we lack 1) high-quality cross-domain diabetes knowledge maps (DKMs) to relate the knowledge bodies in different fields [4, 9], and 2) the ability to assess, adjust, filter, and share knowledge maps in a MDT. We will review the existing approaches and then explain why we must develop a new platform in the two aspects to accelerate cross-domain knowledge sharing. Once the new platform being built, DKMs can rapidly foster productive collaborations across multiple disciplines, facilitating the evaluation and management of diabetes and other co-morbidities.

1.1 Multidisciplinary Team Approach of Diabetes

Type 2 Diabetes Mellitus is a leading cause of death in North American [10, 11]. It is a complex, chronic illness requiring continuous medical care [1]. For diabetes patients, multiple medications are usually prescribed to manage hypertension, hyperglycemia, and other comorbidities [12]. Clinical team members must deal with the challenge of adjusting therapy plans from time to time to help patients overcome adverse drug effects, as well as changing medications in the face of renal failure and cardiovascular disease as the disease progresses [12]. Diabetes is typically treated by a MDT (including endocrinologists, nurses, surgeons, pharmacists, genetic consultants, nutritionists, and other healthcare professionals), who applies multifactorial risk-reduction strategies beyond glycemic control [13]. The MDT approach can effectively help patients cope with the vast array of complications that can arise from diabetes [2]. Knowledge map is an effective tool to help clinicians in a MDT extend boundaries of understanding on diabetes, improve team collaborations, and at the same time reduce information overload [4].

1.2 Knowledge Graph, Knowledge Map and Diabetes Knowledge Map

A knowledge map can be instantly built by searching against a knowledge graph [14]. A knowledge graph is a large knowledge base that helps search and understand the world [15-17]. In a knowledge graph, nodes are knowledge bodies, and they are connected with semantic information gathered from a wide variety of sources. Since 2016 the American Diabetes Association has collaborated with Google to incorporate diabetes information taken directly from the Association's Standards of Medical Care into Google's knowledge graph [11, 17]. This high-quality knowledge graph forms a solid foundation for providing frequently accessed information about diabetes treatment, empowering patients and clinicians to effectively manage diabetes. Specifically, in a medical knowledge graph, symptoms are connected to anatomy parts; drugs are related to diseases; diseases are coupled with genes; and so on. For example, both Insulin and Metformin are connected to diabetes since both medications are used to treat diabetes. The graph format enables the use of graph-based approaches to querying and working with the data.

A medical knowledge map is an emerging knowledge management and representation technique to leverage medical knowledge both within or across domains [18, 19]. Instead of maintaining an overwhelmingly long list of records, a medical knowledge map displays a graph that includes a reasonable amount of knowledge bodies (such as disease symptoms, drugs, diets, practices, and disease-related genes). The edges of a knowledge map represent the semantic relationships between any two closely related knowledge bodies. A knowledge map is usually built around a central knowledge body. The graph representation clearly describes relationships among clinical concepts, leading to better understanding of disease mechanisms and knowledge sharing. Note that although both knowledge graph and knowledge map are graphs, a knowledge graph has a huge amount of knowledge bodies that cover certain domains as complete as possible, whereas a knowledge map is a query result, small to medium scale, and only including knowledge bodies matched to the query.

Unlike the traditional single-core knowledge maps, a typical diabetes knowledge map (DKM) has multiple core knowledge bodies distributed in diverse domains. In a DKM, connections among core

knowledge bodies elucidate the complex semantic relationships in diabetes. Two core knowledge bodies may be connected via a path instead of a direct connection, and multiple core knowledge bodies may *share paths* connecting them. For example, remote concepts "insulin resistance" and "gain weight" can be connected via path "insulin resistance – Type 2 Diabetes Mellitus – weight loss – gain weight".

1.3 Constructing Diabetes Knowledge Maps (DKMs)

While the core knowledge bodies in a DKM can be precisely matched to a user's query using search algorithms [20], methods to concisely describe the semantic relationships between the core knowledge bodies have not been extensively studied [16].

Existing knowledge map construction tools include AIML, Agent/Group/Role, MOISE+, Agile Integration Modeling Language, Enterprise Ontology [21-23], each focusing on a specific aspect of knowledge management. These knowledge map construction and optimization methods are focused on single-core knowledge maps, which cannot be simply extended to explore the knowledge-based associations among multiple knowledge bodies. To the best of our knowledge, none of the existing knowledge map construction tools can be applied to build multi-core knowledge maps, which is essential for DKM construction. Therefore, new tools are needed to construct DKMs to bridge the otherwise isolated domain knowledge in the pursuit of solutions to transdisciplinary problems.

Moreover, a DKM should be *instantly constructed* upon users' search request. Considering a large number of diabetes-related knowledge bodies in a medical knowledge graph, the possible combinations of the knowledge bodies is enormous. It is time-consuming and inefficient to pre-build many DKMs and save them for the search. Rather, a DKM should be rapidly constructed when a query is delivered, and should be saved for knowledge sharing. However, in graph mining, the time complexity of finding the optimal paths among multiple (>2) nodes is NP-complete (see details in Section 3.2) [24]. Due to the high number of diabetes-related knowledge bodies, it is impractical to directly apply the existing graph mining algorithms for DKM construction.

2. INNOVATION

In this proposal, we focus on developing a new computational platform called **DKMiner** to efficiently construct DKMs to identify, manage and visualize the complex relationships among multiple core knowledge bodies in different domains, for advancing diabetes knowledge management and healthcare delivery. Compared to the existing approaches, our platform has three advantages. First, given a user's query, we provide an **ultra-fast algorithm to instantly search for the optimal solutions (Aim 1.1)**. Second, DKMiner is further enhanced to take into consideration the unique properties of medical knowledge graph (including node importance and noise backgrounds) to **significantly improve the model performance (Aim 1.2)**. Third, our platform includes a **graphical user interface** allowing clinicians to flexibly filter, adjust, assess and compare the automatically generated DKMs (Aim 2) to facilitate better the collaborative research and healthcare delivery, instead of providing a static knowledge map. The overall workflow of DKMiner is shown in Figure 1.

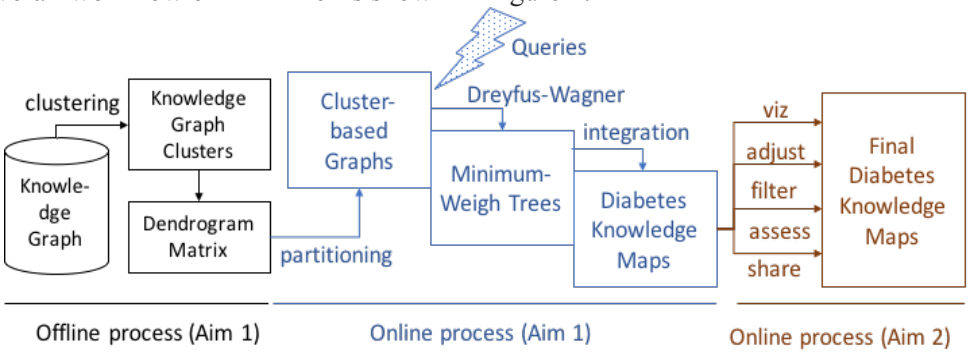

Figure 1. DKMiner consists three components. In the offline preprocessing component (in black), a medical knowledge graph is partitioned and indexed, ready for searching. In the online DKM construction component (in blue), a three-step approach is presented to quickly construct DKMs upon user's queries. Finally, in the DKM-based application, DKMs are visualized, adjusted, and shared among clinicians in a MDT (in brown).

3. APPROACH

A well-constructed DKM can be used to efficiently manage the diabetes-related knowledge in a multidisciplinary team, and effectively reduce information overload, leading to better team collaboration and knowledge sharing. However, constructing DKMs presents several computational challenges, as described in Section 1.3. The aim of the proposal is to develop DKMiner, a new platform to instantly construct multi-core knowledge maps by overcoming the computational challenges. Our main idea is to identify the backbone of the knowledge map by finding the minimum-weigh tree that connects all the core knowledge bodies, and then build a DKM by expanding the tree.

3.1 Intuition and Motivation

We propose a new platform DKMiner to address the computational challenges in DKM construction. We illustrate the intuition behind our main idea with an example. Consider a user's query "male patients with insulin resistance lost weight". With a search algorithm, we will first identify the core knowledge bodies "male", "insulin resistance", and "weight loss" that match the query. Second, we will look for the most appropriate paths connecting these three core knowledge bodies. We can solve the optimal path finding problem by constructing a minimum-weigh tree, in which leaf nodes are the three core knowledge bodies, and the overall cost of the paths to connect them is minimized. In graph theory, the minimum-weigh tree identification problem is called the Steiner trees in graphs problem, which can be solved using Dreyfus-Wagner algorithm or its variations [25]. The time complexity of the Dreyfus-Wagner algorithm is exponential to the number of the core knowledge bodies [24]. In this proposal, we will adopt the divide and conquer strategy and provide an **ultra-fast algorithm** to instantly identify the minimum-weigh tree.

3.2 DKM Construction (Aim 1.1)

Given a medical knowledge graph $KG(V,E,W)$, where V is a set of medical knowledge bodies, two medical knowledge bodies u and v are connected with an edge $e \in E$ if u and v are semantically related, and the non-negative edge weight $w \in W$ represents the correlation coefficient. Let core knowledge body set $S \subseteq V$ be a set of knowledge bodies that match a user's query; the aim is to construct the minimum-weigh tree T in KG that spans S. To efficiently identify S, we propose DKMiner with two phases, i.e. the offline phase and the online phase. In the offline phase, we preprocess the knowledge graph KG to identify the hierarchical clustering structure (Figure 1 in black). The offline phase is independent to user's queries. The online phase has three steps (Figure 1 in blue). First, based on a user's query, we will partition KG and generate a set of cluster-based graphs, saved as CG. Second, we will identify all the minimum-weigh trees in CG. Finally, we will construct a DKM by integrating all the minimum-weigh trees. In the following, we describe all the steps in detail and demonstrate the advantages of our algorithm using preliminary results.

3.2.1 Identifying Clustering Structure of Knowledge Graph. In the offline phase, we will study the structure of a given knowledge graph KG using a hierarchical clustering algorithm [26]. The clustering result is saved in a dendrogram, a tree diagram to illustrate the arrangement of the clusters, prepared for DKM construction. And then, we construct dendrogram matrix KM. KM is a square matrix, where each row/column is a knowledge body in KG, and the value for each cell $KM(m,n)$ is the lowest level in the dendrogram that m and n are separated into different two clusters. The idea is that with clustering, we can discard the clusters irrelevant to a user's query, significantly reducing the search space for the minimum-weigh tree construction and saving the online computation time.

3.2.2 Generating Cluster-based Graphs. In the online process, given a user's query, we will first identify the set of core knowledge bodies S in KG that match a user's query using a search algorithm [20]. Second, we will construct a small subset of KM that only includes the knowledge bodies in S, and save it as QM. Here $QM(m,n)$ indicates the lowest level in the cluster dendrogram at which core knowledge bodies m and n ($m \in S$ and $n \in S$) are separated. Third, we convert QM into a complete graph QG, where each node is a core knowledge body in S, and each edge weight is the corresponding value in QM. Fourth, we will separate QG into k subgraphs using a graph cut algorithm [27], where k is a system parameter ($k>2$). Finally, by applying the same cuts on the dendrogram generated in the offline phase, we will

compose a cluster-based graph. In a cluster-based graph, each node is a cluster, and two nodes are connected if the corresponding clusters have the common parent cluster in the dendrogram. An important property of a cluster-based graph is that it includes at most k clusters that have at least one core knowledge body. We will repeat the graph cut process on each subgraph of QG until all the subgraphs have less than k nodes, and save all the cluster-based graphs in CG. Note that all the processes in this step are fast because both QM and QG are generated based on a user's query, which usually is not long.

3.2.3 Identifying Minimum-Weigh Trees. Given a graph, the minimum-weigh tree identification problem is equivalent to the Steiner trees in graphs problem, which can be solved using the Dreyfus-Wagner algorithm [28]. Since its first version, the Dreyfus-Wagner algorithm has been further improved significantly [25]. Dreyfus-Wagner algorithm and its variations have effectively solved many practical problems in biomedical informatics, such as functional module identification in protein-protein interaction networks [29] and pathway discovery in metabolic networks [30]. In our case, the time complexity of the algorithm is $O\left(nm^{2m+(log2^m)(log2^n)}\right)$, where n is the total number of nodes in a knowledge graph and m is the total number of core knowledge bodies [25]. Clearly, it is time-consuming to directly apply Dreyfus-Wagner algorithm on a knowledge graph. By adopting the divide and conquer strategy in the previous subsection, we have decomposed the knowledge graph into a set of cluster-based graphs, in which the number of nodes including at least one core knowledge body is bounded by k (usually a small number), and have discarded all the clusters irrelevant to a query. Hence, all the minimum-weigh trees can be quickly identified.

3.2.4 DKM construction and visualization. Finally, we will connect all the minimum-weigh trees based on the clustering dendrogram, expand the integrated minimum-weigh tree to form a DKM, and visualize the DKM in a web browser using JavaScript and Cytoscape.js [31, 32].

3.2.5 Preliminary Results on DKM Construction. We have conducted a pilot project to test the performance of DKMiner using an integrated yeast co-function network upon querying a list of gene names. Although the diabetes knowledge graph and the yeast network are vastly different regarding scale, content, and complexity, they are mathematically equivalent. The yeast co-function network is an integration of the protein-protein interaction networks, gene co-expression networks, and genetic interaction networks [33]. It includes 4,987 yeast genes and 29,757 gene-to-gene relations, and the edge weights are defined by the evidence types that support the edges. We chose the yeast co-function network because it has been used to construct a widely-used ontology called Nexo ontology [33, 34]. With the yeast network, we can systematically evaluate the performance of DKMiner using Nexo ontology.

For performance evaluation, we prepared four sets of queries, i.e., easy query, medium query, hard query, and combined query (see Figure 2). Each set includes more than 100 queries. For the easy query, we randomly chose 10 genes that annotated to the same Nexo ontology term, simulating in-domain query. For the medium query, we randomly combine two easy queries, simulating queries across two domains. For the hard query, we randomly combine five easy queries, simulating queries across more than two domains. For the combined query, we combine all kinds of queries that have already generated.

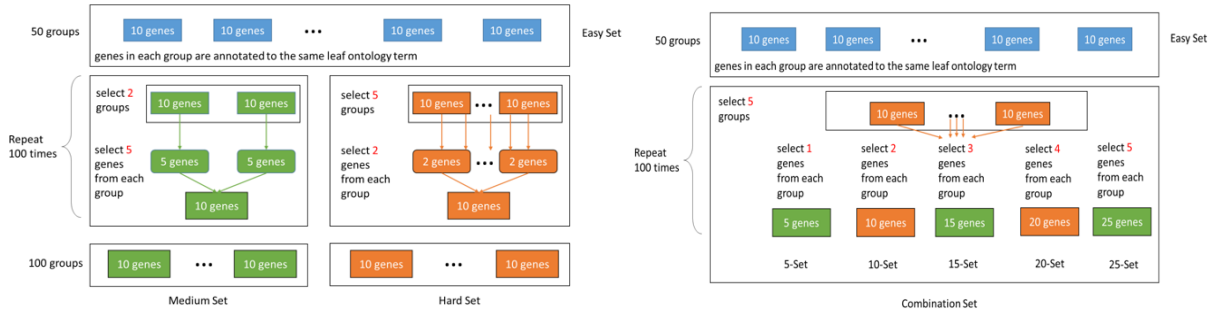

Figure 2. Queries at the different level of complexities were generated for DKMiner performance evaluation.

We compared DKMiner with two algorithms, i.e., the fastest Dreyfus-Wagner algorithm implementation running on the whole knowledge graph and the all-pairs shortest path algorithm [35], on all the four sets of queries. While the former one guarantees to generate the global minimum-weigh tree, it is NP-complete. The later one scales well, but the resulting knowledge map may include redundant paths. Figure 3 shows a sample output of DKMiner. To connect all the five genes in a query (in pink), DKMiner identifies a tree with six extra genes (in gray). With the same input, the all-pairs shortest path algorithm, however, returns all the shortest paths between any two pink nodes, introducing many foreign genes.

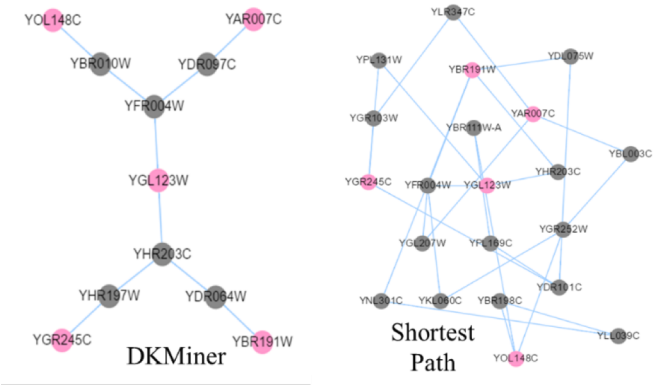

Figure 3. The outputs of DKMiner and the all-pairs shortest path algorithm using the same query (in pink). The output of DKMiner is more clear.

We first tested whether DKMiner scales well with the increase in the number of genes (core knowledge bodies). Figure 4 shows that to connect 20 genes, DKMiner finished in less than three seconds, even faster than the all-pairs shortest path algorithm, while the Dreyfus-Wagner algorithm needs roughly 300 years (estimated results). Second, since the Dreyfus-Wagner algorithm always returns the globally optimized results, we computed the difference between the results of DKMiner (noted as T_{DKMer}) and the Dreyfus-Wagner algorithm (noted as T_{DW}). Mathematically, for every node in T_{DKMer}, we look for its closest node in T_{DW} and compute the node-to-node distance. Similarly, we computed the difference between the results of shortest path algorithm (noted as T_{SP}) and T_{DW}. The averaged distances shown in Figure 5 indicate the results of DKMiner is significantly better that of the all-pairs shortest path algorithm on both the medium set and the hard set.

3.3 DKMiner Further Enhancement (Aim 1.2)

The preliminary results on the yeast network showed that DKMiner is significantly better than the Dreyfus-Wagner algorithm and the all-pairs shortest path algorithm. However, DKMiner needs to be further enhanced to accommodate the complex medical knowledge graph. First, we will adopt node ranking algorithms (such as topic-sensitive PageRank [36] and HubPPR [37]) to take node weight into consideration, so that knowledge bodies with rich information will have a high priority to be selected. Second, by fully aware the noise in medical knowledge graphs [38], we will develop a new model to improve path reliability [39], thus significantly improve the robustness of resulting DKM [40]. Third, to achieve a faster response, we will increase the platform scalability using Hadoop [41]. We will deploy the DKMiner platform for the diabetes teams at the University of Kentucky, allowing the domain experts to fully evaluate the platform performance. We will also improve the quality of medical knowledge graph. In practice, we will rebuild *KG* by incorporating multiple biomedical informatics resource such as Human Phenotype Ontology [42], Gene Ontology [43-45], Disease Ontology [46], human gene networks [47, 48], and the IDs in these resources will be matched using the Unified Medical Language System [49, 50].

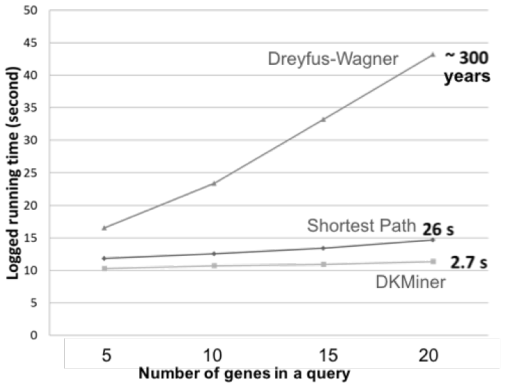

Figure 4. Running time with queries in different size. DKMiner is significantly faster than the Dreyfus-Wagner algorithm running on the whole knowledge graph.

3.4 Developing Human-Computer Interaction Tools (Aim 2). While Aim 1 focuses on DKM construction, Aim 2 shifts its focuses onto the development of DKM-based

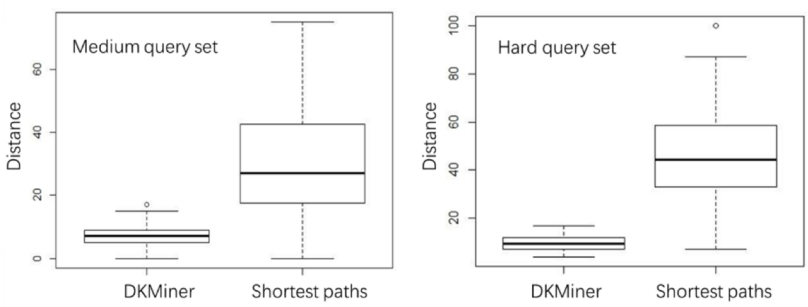

Figure 5. Compute the distances between the results of DKMiner (or the shortest path algorithm) and the gold standard. It shows DKMiner is significantly more accurate than the shortest path approach.

applications (Figure 1 in brown). Specifically, we will develop and deploy a human-computer interaction tool to reduce the cognitive differences in different disciplines, extend boundaries of understanding on diabetes, and improve team collaborations. Using this tool, clinicians in a MDT can 1) collaboratively adjust the automatically generated DKMs by reassigning core knowledge bodies, 2) view and evaluate the DKMs generated by other team members, 3) save and share DKMs, 4) select multiple DKMs and then merge or compare them, 5) filter and tailor an existing DKM. Technically, we will develop an active learning model to analyze users' inputs and to optimize DKMs automatically [51].

3.5 Dissemination

By guiding a clinical team through the knowledge flow, our platform will allow clinicians to effectively and conveniently evaluate, adjust and master cross-domain multi-core knowledge maps. We will periodically update our models and the resultant knowledge maps as domain knowledge grows, and further improvements will be incorporated in methodology. The software platform and the DKMs will be made publicly available as free and open source resources.

3.6 Limitations

The performance of DKMiner heavily relies on knowledge graph clustering. If the clustering results are not optimized, the quality of the final DKMs may drop significantly. Meanwhile, knowledge graphs are usually very large and keep evolving. Periodically regenerating high-quality graph clusters is a difficult task. Our alternative approach is, instead of simply applying a hierarchical clustering method, we will adopt the meta-clustering strategy [52], and incorporate multiple large graph clustering methods, such as link-based clustering [53], to repeatedly partition a knowledge graph to generate the cluster dendrogram.

3.7 Timeline

The following table outlines the timeline of the project tasks. The first year of the project will focus on developing the DKM construction algorithm (Aim 1.1), DKM enhancement (Aim 1.2), performance evolution method, and perform algorithm evaluation. In the second year, we will continue to develop DKM enhancement techniques (Aim 1.2), develop human-computer interface tools (Aim 2), and apply clinician-oriented evaluations and dissemination of end products.

Tasks	Year 1		Year 2	
	1/2	2/2	1/2	2/2
Knowledge graph clustering and indexing	■			
Cluster-based graph construction	■			
Minimum-weigh tree identification	■	■		
DKM construction	■	■		
DKM enhancement (noise reduction)		■		
DKM enhancement (node importance)		■	■	
DKM interface design			■	■
DKM Evaluation	■	■	■	■