

UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

Implementación del algoritmo de Yannakakis con QDags utilizando prejoining

PROPUESTA DE TEMA DE MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL EN COMPUTACIÓN

Gabriel C. Díaz

MODALIDAD:
Memoria

PROFESOR GUÍA:
Gonzalo Navarro

SANTIAGO DE CHILE
2026

1. Introducción

Las bases de datos de grafos son bases de datos que emplean una estructura de grafo para consultas semánticas utilizando nodos y aristas para representar relaciones binarias entre entidades. En este modelo, una primitiva fundamental para realizar consultas es el BGP (Basic Graph Pattern, por sus siglas en inglés), que consiste en encontrar un subgrafo dado dentro de la base de datos. Si bien las BGPs se pueden expresar en el álgebra relacional, la mayoría de estas resultan en un número significativo de joins, lo que ha motivado la creación de nuevas técnicas que puedan manejarlos eficientemente.

Una de las principales líneas de investigación que abordan este problema son los algoritmos WCO (*Worst-Case Optimal*), cuyo tiempo de ejecución alcanza, salvo factores polilogarítmicos, la cota del peor caso para el tamaño del resultado de un join dadas las cardinalidades de sus relaciones de entrada [6]. Sin embargo, las implementaciones de estos algoritmos suelen requerir índices que consumen cantidades significativas de memoria, lo que dificulta su uso sobre bases de datos de gran tamaño. Esto ha motivado el desarrollo de representaciones compactas que permitan mantener tiempos competitivos de consulta utilizando poco espacio.

Otra estrategia consiste en aprovechar la estructura de las consultas mediante Generalized Hypertree Decompositions (GHDs) [8]. Intuitivamente, una GHD divide una consulta en un conjunto de subconsultas relacionadas entre sí en forma de árbol. Cada nodo de la descomposición puede resolverse mediante un multijoin, produciendo relaciones intermedias que conforman una consulta acíclica. Sobre esta última puede aplicarse el algoritmo de Yannakakis [13], que reduce las relaciones mediante semijoins antes de realizar los joins necesarios para obtener el resultado final. Este enfoque permite obtener cotas más refinadas para algunos tipos de consultas, pero introduce el costo de almacenar las relaciones intermedias generadas por la descomposición.

Los QDags [11] son representaciones compactas de relaciones basadas en quadtrees que permiten evaluar multijoins mediante un algoritmo WCO utilizando poco espacio. Trabajos posteriores han evaluado su extensión a consultas de mayor dimensionalidad y su combinación con GHDs [10]. Esta combinación resulta particularmente atractiva, dado que permite almacenar los resultados intermedios asociados a los nodos de una GHD de manera compacta. Sin embargo, la aplicación del algoritmo de Yannakakis sobre los QDags requiere el uso de semijoins, cuya implementación en QDags involucra el cálculo de proyecciones lo cual puede introducir un costo significativo.

Por otra parte, en [4] se introduce la técnica de *prejoining*, que consiste en resolver previamente joins de menor dimensionalidad y utilizar sus resultados para podar el espacio recorrido durante la evaluación de la consulta. En el caso de relaciones binarias, esta técnica mantiene precalculadas las proyecciones de cada relación sobre sus atributos individuales, de modo que estas pueden reutilizarse durante la evaluación sin ser calculadas en tiempo de consulta. Si bien el prejoining fue propuesto con un objetivo distinto, posteriormente surgió la idea de aprovechar esta propiedad para reproducir el efecto de los semijoins requeridos por el algoritmo de Yannakakis. Esta posibilidad no ha sido estudiada en los trabajos previos considerados y constituye el punto de partida de esta memoria. En particular, se busca determinar experimentalmente si trasladar parte del costo de las proyecciones a una etapa de preprocesamiento permite reducir el costo de la fase de reducción y bajo qué tipos de

consultas y descomposiciones esta estrategia resulta conveniente.

2. Situación Actual

En la literatura existen diversos motores e índices utilizados para la evaluación de consultas sobre grafos, entre ellos Apache Jena [2], Jena LTJ [9], BlazeGraph [12], Virtuoso [7], EmptyHeaded [1] y las estructuras basadas en Ring [5]. En particular, EmptyHeaded utiliza GHDs como parte de su estrategia de evaluación, mientras que Ring y sus variantes constituyen índices compactos para la resolución de BGPs. Los resultados experimentales reportados en [4] comparan estas alternativas con Qdag y PJ-Qdag, mostrando diferencias importantes en el espacio requerido por sus índices, como se resume en la Tabla 1.

Sistema	**Espacio**	**Tiempo de construcción**	**Sistema**	**Espacio**
Qdag	4.75	298	Jena	48.42
PJ-Qdag	6.17	441	Jena LTJ	96.83
RingP	4.90	3444	BlazeGraph	99.86
C-Ring	6.69	1398	Virtuoso	104.89
Ring	11.17	1337	EmptyHeaded	1292.28

Tabla 1: Espacio del índice, en bytes por arista, para todos los sistemas, y tiempo de construcción, en segundos, para los índices compactos. Datos obtenidos de [4].

Esto sugiere a los QDags como una alternativa viable para implementar algoritmos basados en GHD en poco espacio. Sin embargo, una de sus limitaciones principales es su implementación de los *semijoins*, pues requieren calcular proyecciones. Computar el resultado de esta operación utilizando QDags puede resultar en algoritmos cuadráticos, como se menciona en [11].

En [3] se explora la posibilidad de mitigar este sobrecosto seleccionando apropiadamente las descomposiciones mediante un modelo de costo formulado como una instancia de descomposición en árbol de peso mínimo. Sin embargo, algunas consultas continúan presentando tiempos de ejecución elevados. El análisis experimental muestra que, si bien este modelamiento permite reducir el costo asociado a los *multijoins*, los *semijoins* continúan constituyendo un cuello de botella. Se estudió también la posibilidad de incorporar su costo al modelo, sin obtener un modelo que permitiera predecir satisfactoriamente su comportamiento.

En [4] se estudian los prejoins como una heurística para acelerar la resolución de BGPs sobre QDags. La premisa de este trabajo es que esta técnica puede utilizarse para reproducir el efecto de la reducción por semijoins requerida por el algoritmo de Yannakakis. Esto daría lugar a una implementación alternativa de dicha fase, que podría reducir el principal cuello de botella identificado en [3] y facilitar el modelamiento de su costo.

3. Objetivos

Objetivo General

El objetivo de este trabajo es estudiar el uso de prejoins como mecanismo para implementar eficientemente la reducción por semijoins del algoritmo de Yannakakis sobre relaciones representadas por QDags. En particular, se busca diseñar e implementar una variante basada en prejoining, integrarla al algoritmo basado en GHDs desarrollado en trabajo previo ([3]) y caracterizar experimentalmente su comportamiento temporal y espacial en comparación a la implementación existente basada en semijoins. El trabajo buscará identificar en qué tipos de consultas y descomposiciones esta estrategia resulta beneficiosa, junto con determinar si es posible modelar de forma precisa su complejidad.

Objetivos Específicos

1. Diseñar una estrategia que permita reproducir las reducciones por semijoin requeridas por el algoritmo de Yannakakis utilizando operaciones de prejoining sobre Qdags.
2. Implementar la estrategia propuesta sobre la infraestructura de Qdags utilizada en trabajo previo e integrarla al evaluador de consultas basado en GHDs.
3. Verificar experimentalmente la correctitud de la nueva implementación, comparando sus resultados con los obtenidos por la implementación existente del algoritmo de Yannakakis y, cuando sea conveniente, con una implementación relacional de referencia.
4. Evaluar el comportamiento temporal y espacial de la estrategia propuesta sobre consultas de distintas características, desglosando el costo de las principales etapas de evaluación para determinar el impacto específico del prejoining.
5. Comparar la nueva estrategia con la implementación existente basada en semijoins, identificando las características de las consultas, de sus GHDs y de los datos que determinan cuándo cada estrategia resulta más conveniente.
6. Estudiar, a partir de los resultados anteriores, la posibilidad de incorporar el costo del prejoining a las técnicas de selección de GHDs desarrolladas en trabajo previo. En caso de que los resultados experimentales permitan identificar regularidades suficientemente claras, se diseñará y evaluará una extensión del modelo de costo o una heurística de selección apropiada.

Evaluación

La evaluación considerará de manera separada la correctitud y rendimiento de la propuesta. La correctitud se estudiará verificando que la evaluación mediante prejoins produzca los mismos resultados que el algoritmo de referencia para un conjunto de consultas de prueba. Además de casos provenientes de los benchmarks utilizados en trabajos anteriores, se podrán utilizar instancias pequeñas o generadas artificialmente que permitan comprobar de manera exhaustiva el resultado esperado.

El rendimiento se evaluará en términos de tiempo de ejecución y uso de espacio. Se comparará el algoritmo propuesto con la variante existente de Yannakakis sobre QDags junto a otros algoritmos de la literatura, utilizando el mismo conjunto de datos, consulta y entorno

experimental. El tiempo de ejecución será estudiado junto a un desglose de las distintas fases del algoritmo para identificar los costos afectados por el prejoining.

Dado el carácter exploratorio del trabajo, la evaluación no presupone que la estrategia propuesta deba superar a la implementación existente. Se considerará un resultado satisfactorio obtener una caracterización reproducible de su comportamiento que permita establecer en qué condiciones el uso de prejoins es beneficioso, neutro o perjudicial, y explicar las causas de dichas diferencias. Si se desarrolla un modelo de costo o heurística a partir de estos resultados, se evaluará adicionalmente su eficacia a la hora de generar planes de consulta.

4. Solución Propuesta

El punto de partida de este trabajo será la implementación de QDags y del algoritmo basado en GHDs desarrollada originalmente por Rivas [10] y posteriormente extendida en [3]. Esta implementación incorpora soporte para semijoins y utiliza el algoritmo de Yannakakis sobre las relaciones intermedias generadas por la descomposición. En dicho enfoque, cada nodo de la GHD corresponde a una subconsulta que puede resolverse mediante un multijoin sobre QDags. Una vez calculadas estas relaciones intermedias, el árbol resultante se procesa mediante el algoritmo de Yannakakis, cuya fase de reducción requiere ejecutar semijoins entre nodos adyacentes de la descomposición.

La implementación existente de estas operaciones requiere efectuar operaciones relacionadas con proyecciones en QDags. Estas operaciones introducen un costo considerable y constituyen una motivación para estudiar formas alternativas de realizar esta reducción [11]. Por otro lado, el trabajo desarrollado en [4] muestra que los prejoins pueden utilizar relaciones de menor dimensionalidad como filtros durante la evaluación de joins sobre Qdags, reduciendo considerablemente el espacio de búsqueda en determinadas consultas.

A partir de estas observaciones, se diseñará una variante de la fase de reducción de Yannakakis en la que el efecto de los semijoins se obtenga mediante prejoining. Posteriormente, la estrategia se implementará en C++ sobre el código utilizado en [3], manteniendo la implementación original como línea base para realizar comparaciones bajo un mismo entorno experimental.

La evaluación utilizará principalmente los conjuntos de datos y consultas empleados en trabajos anteriores consistentes en consultas sobre el grafo de Wikidata.

Finalmente, se estudiará la predictibilidad de los tiempos de ejecución según las descomposiciones escogidas. En caso de obtener buenos resultados, se explorará una extensión del modelo de costo o una heurística que pueda explotar las características de esta nueva estrategia para generar planes de consulta de mejor rendimiento.

5. Plan de Trabajo (Preliminar)

1. Estudiar formalmente la relación entre la reducción por semijoins utilizada por Yannakakis y las operaciones de prejoining disponibles sobre Qdags, y diseñar una variante que preserve la correctitud de la evaluación.

2. Implementar un prototipo de la operación propuesta y construir pruebas de correctitud sobre casos pequeños y consultas controladas.
3. Integrar la nueva operación a la implementación existente del evaluador basado en GHDs, manteniendo disponible la implementación original de semijoins como línea base.
4. Instrumentar ambas implementaciones para obtener mediciones separadas de las principales fases de ejecución y de las estructuras intermedias relevantes.
5. Realizar una primera evaluación experimental utilizando consultas y datos provenientes de los experimentos de trabajos anteriores. Utilizar estos resultados para identificar casos representativos donde el nuevo método presente comportamientos distintos de la línea base.
6. Diseñar experimentos adicionales, incluyendo cuando sea necesario consultas sintéticas, para estudiar las causas de las diferencias observadas y determinar qué características de las consultas, datos o GHDs influyen en el rendimiento de ambas estrategias.
7. Analizar los resultados y caracterizar los escenarios en los que el uso de prejoining resulta beneficioso o perjudicial. Si la evidencia experimental permite identificar predictores adecuados del costo, estudiar una extensión del modelo de selección de GHDs o una heurística que incorpore la nueva estrategia.
8. Consolidar los experimentos finales, documentar las decisiones de diseño y sus limitaciones, y redactar el informe de memoria.

Referencias

[1] Aberger, Christopher R., Andrew Lamb, Susan Tu, Andres Nötzli, Kunle Olukotun y Christopher Ré: *EmptyHeaded: A Relational Engine for Graph Processing*. ACM Transactions on Database Systems, 42(4):20:1–20:44, 2017.

[2] Apache Software Foundation: *Apache Jena*. `https://jena.apache.org/`, 2026.

[3] Arroyuelo, Diego, Gabriel Carmona, Gonzalo Navarro y Matilde Rivas: *Speeding up Qdags with Generalized Hypertree Decompositions*. En *Proceedings of AMW'26*, 2026.

[4] Arroyuelo, Diego, José Cazorla y Gonzalo Navarro: *Boosting Graph Joins and Matrix Multiplications in Little Space*. En *Proceedings of GRADES-NDA'26*, 2026.

[5] Arroyuelo, Diego, Adrián Gómez-Brandón, Aidan Hogan, Gonzalo Navarro, Juan L. Reutter, Javiel Rojas-Ledesma y Adrián Soto: *The Ring: Worst-Case Optimal Joins in Graph Databases Using (Almost) No Extra Space*. ACM Transactions on Database Systems, 49(2):5:1–5:45, 2024.

[6] Atserias, Albert, Martin Grohe y Dániel Marx: *Size Bounds and Query Plans for Relational Joins*. En *Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2008)*, 2008.

[7] Erling, Orri: *Virtuoso, a Hybrid RDBMS/Graph Column Store*. IEEE Data Engineering Bulletin, 35(1):3–8, 2012.

[8] Gottlob, Georg, Nicola Leone y Francesco Scarcello: *Hypertree Decompositions and Tractable Queries*. Journal of Computer and System Sciences, 64(3):579–627, 2002.

[9] Hogan, Aidan, Cristian Riveros, Carlos Rojas y Adrián Soto: *A Worst-Case Optimal Join Algorithm for SPARQL*. En *Proceedings of the 18th International Semantic Web Conference (ISWC)*, páginas 258–275, 2019.

[10] Lagos, Matilde Rivas: *Enhancing Qdags for Higher-Dimensional Queries*. Tesis de Licenciatura, Universidad de Chile, 2026.

[11] Navarro, Gonzalo, Juan L. Reutter y Javiel Rojas-Ledesma: *Optimal Joins Using Compact Data Structures*. En *23rd International Conference on Database Theory (ICDT 2020)*, 2020.

[12] Thompson, Bryan B., Mike Personick y Martyn Cutcher: *The Bigdata RDF Graph Database*. En *Linked Data Management*, páginas 193–237. Chapman and Hall/CRC, 2014.

[13] Yannakakis, Mihalis: *Algorithms for Acyclic Database Schemes*. En *Proceedings of the 7th International Conference on Very Large Data Bases (VLDB)*, páginas 82–94. IEEE Computer Society, 1981.