تحسين ترتيب الربط باستخدام خوارزمية MCTS الموجَهة بمعلومات الاستعلام في قواعد البيانات

محمد حاج حسن* أ.د. يُسر السيد سليمان الأتاسي**

الملخص:

يُعدّ تحسين الاستعلامات مكوّناً أساسياً في نظم إدارة قواعد البيانات، إذ يرتبط ارتباطاً مباشراً بكفاءة تنفيذ الاستعلامات. ويُعدّ اختيار ترتيب عمليات الربط عاملاً حاسماً في أداء استعلامات قواعد البيانات، نظراً لتأثيره المباشر على زمن تنفيذ الاستعلامات. تكمن صعوبة هذه المهمة في اتساع فضاء البحث مع زيادة عدد جداول الاستعلام، مما يجعل مسألة اختيار ترتيب الربط من المسائل المعقدة حسابياً NP-hard.

في هذه الدراسة، نقدم نهجاً لتحسين أداء قواعد البيانات من خلال اختيار ترتيب عمليات الربط باستخدام تقنيات التعلم المعزّز. تعتمد طريقتنا المقترحة على خوارزمية Monte Carlo Tree Search (MCTS). ترتكز الفكرة الرئيسية على محاكاة العديد من ترتيبات الربط الممكنة في بنية شجرية واحدة وتطبيق MCTS الموجهة بمعلومات خاصة بالاستعلام (حجم الجدول) لاختيار ترتيب التنفيذ الذي يحقق أعلى أداء ممكن.

نقوم بإجراء مجموعة من التجارب لتقييم فعالية طريقتنا. تظهر النتائج أداء أكثر استقراراً وتحسيناً ملحوظاً عبر مختلف أحجام الاستعلامات والقدرة على التكيف مع تعقيد البيانات بشكل عام. على وجه التحديد تُظهر النتائج أن الطريقة المقترحة تحقق تسريع قدره 1.14x عند المقارنة مع Alphajoin1.0 على مقياس WRL في حين حققت تسريعاً قدره 1.3× مقارنة بـ Alphajoin1.0 على مقياس GMRL.

تساهم هذه الدراسة في المجال من خلال تقديم منهجية اختيار ترتيب الربط دون معرفة واسعة بمجال إدارة قواعد البيانات ويؤكد عملنا على إمكانيات دمج تعلم الآلة مع أنظمة إدارة قواعد البيانات لتحسين الأداء.

كلمات مفتاحية: عملية الربط، تحسين ترتيب الربط، تعلم الآلة، التعلم المعزز، خوارزمية MCTS.

*) طالب ماجستير: قسم هندسة البرمجيات ونظم المعلومات – كلية الهندسة المعلوماتية – جامعة حمص – حمص – سوريا

**) أستاذ في هندسة البرمجيات ونظم المعلومات: قسم هندسة البرمجيات ونظم المعلومات – كلية الهندسة المعلوماتية – جامعة حمص – حمص – سوريا

Join Order Optimization using query-information-driven MCTS algorithm in Databases

* M.H.Hasan ** Y.Atassi

Abstract:

Query optimization is a fundamental component of database management systems, as it is directly related to the efficiency of query execution. Selecting the join order is a critical factor in database query performance due to its direct impact on query execution time. The difficulty of this task lies in the rapid expansion of the search space as the number of tables in a

query increases, making join order selection a computationally complex NP-hard problem.

In this study, we propose an approach to improve database performance by selecting join orders using reinforcement-learning techniques. Our proposed method is based on the Monte Carlo Tree Search (MCTS) algorithm. The core idea is to simulate a large number of possible join orders within a single tree structure and apply MCTS guided by query-specific information (table size) to select the execution order that achieves the best possible performance.

We conduct a series of experiments to evaluate the effectiveness of our approach. The results demonstrate more stable performance and noticeable improvements across different query sizes, as well as an overall ability to adapt to data complexity. Specifically, the proposed method achieves a speedup of 1.14× compared to AlphaJoin1.0 under the WRL metric, while achieving a speedup of 1.3× compared to AlphaJoin1.0 under the GMRL metric.

This study contributes to the field by presenting a methodology for join order selection that does not require extensive domain knowledge in database management systems. Our work highlights the potential of integrating machine learning with database management systems to improve performance.

Keywords: join operation, join order optimization, machine learning, reinforcement learning, MCTS algorithm.

---

*) Master Student: Department of Software Engineering and Information Systems – Faculty of Informatics Engineering – Homs University – Homs – Syria

**) Professor of Software Engineering and Information Systems: Department of Software Engineering and Information Systems – Faculty of Informatics Engineering – Homs University – Homs – Syria

1. مقدمة:

تشكل قواعد البيانات حجر الأساس لأنظمة المعلومات والبرمجيات الحديثة. وتتجلى أهميتها ليس فقط في قدرتها على معالجة كميات ضخمة من البيانات، بل أيضاً في دورها المحوري في دعم عمليات اتخاذ القرار. توفّر أنظمة إدارة قواعد البيانات Database Management Systems (DBMS) مجموعة واسعة من الإعدادات الضرورية لضمان الأداء الأمثل لقواعد البيانات، مما يُسهِم في سرعة وفعالية استرجاع المعلومات ذات الصلة. ويُعتبر تحسين الاستعلامات أحد المكونات الجوهرية في هذه الأنظمة، ويشكّل تحسين ترتيب عمليات الربط (Join Ordering) –وهو مشكلة فرعية ضمن تحسين الاستعلامات –أحد أبرز التحديات البحثية في هذا المجال. وتتمثل هذه المهمة في إيجاد الترتيب الأمثل للعلاقات (الجداول) المشاركة في عملية الربط، بهدف تقليل الكلفة المقدّرة لعملية التنفيذ إلى أقل ما يمكن [2].

تعتمد الممارسات الحالية في تحسين أداء قواعد البيانات بصورة رئيسة على عمليات ضبط يدوية ومعقدة ومنهجيات وتوصيفات تجريبية، ينفذها مسؤولي قواعد البيانات (DBAs) ممن يمتلكون خبرة متقدمة ومعرفة متراكمة. ومع التوسع المستمر في أحجام قواعد البيانات وتزايد درجة تعقيدها، يغدو هذا النهج اليدوي أقل جدوى وأكثر إشكالية وغير قادر على تلبية متطلبات الأداء العالي لقواعد البيانات والتطبيقات المتنوعة، والمستخدمين المتنوعين.

يمكن توظيف تقنيات الذكاء الاصطناعي للتغلب على القيود المرتبطة بالأساليب التقليدية في تحسين أداء قواعد البيانات، وذلك عبر قدرتها على استكشاف فضاء تصميم أوسع مما هو ممكن للبشر، واستبدال النهج التجريبي في معالجة المشاكل المعقدة بآليات أكثر منهجية وفعالية [3]. قدم الباحثون في [9] نموذجاً يعتمد على تقنيات التعلّم المعزز العميق لتحسين ترتيب عمليات الربط في قواعد البيانات، وفي [2] فقد طُرح إطار عمل يصوغ مشكلة تحسين ترتيب عمليات الربط كمشكلة تعلم معزز مُلاحظة بالكامل وعلى الرغم من النتائج الجيدة التي تم تحقيقها في

الأعمال السابقة إلا أنها كانت تعتمد على نماذج كلفة تفتقر إلى الدقة مما يؤثر في كثير من الأحيان على جودة الخطة النهائية.

أظهرت الدراسات الحديثة في مجال التعلّم المعزّز قدرة ملحوظة على معالجة المشكلات التي تتطلب مستوى عالٍ من الحدس، حيث حققت نتائج مشجعة مدعومة بخبرة مكتسبة من الملاحظة والتجربة. وقد برز هذا بوضوح في تطبيقات ناجحة لتعلّم استراتيجيات لعب الألعاب المعقدة ذات المساحات البحثية الواسعة. وانطلاقاً من هذه الإنجازات، **يهدف هذا العمل إلى استكشاف إمكانية توظيف التعلّم المعزّز –وبالتحديد خوارزمية Monte Carlo Tree Search (MCTS)–في تحسين أداء أنظمة قواعد البيانات، وذلك من خلال تحسين اختيار عملية الربط بشكل أكثر كفاءة.**

تتجسّد مساهمات هذه البحث في:

- ✓ توجيه خوارزمية UCT بمعلومات بنيوية عن الاستعلام: إذ أتاح دمج المعرفة ببنية الاستعلام تعزيز فهم الخوارزمية لسياق الاستعلام المعالج، والحدّ من مستويات عدم اليقين الملازمة للخوارزمية.
- ✓ اعتماد سياسة موجهة بطبيعة الاستعلام بدلاً من السياسات العشوائية: لأن استكشاف فضاء البحث الضخم بشكل عشوائي غير فعال يؤدي إلى ترتيبات ربط سيئة وبالتالي الحصول على خطط استعلام كارثية، بينما يضمن التوجيه المستند إلى خصائص الاستعلام مساراً أكثر فاعلية.

2. أهداف البحث:

تهدف دراستنا إلى توظيف تقنيات تعلم الآلة، وبشكل خاص التعلم المعزز، في تحسين عملية اختيار ترتيب الربط داخل قواعد البيانات. نسعى من خلال ذلك إلى **الوصول إلى خطة ربط مثلى للاستعلام**، بما يضمن **تقليص زمن التنفيذ** والحصول على النتائج المطلوبة في أقصر وقت ممكن،

وهو ما ينعكس إيجاباً على أداء قاعدة البيانات ككل. ويركّز الأسلوب المقترح على تحقيق الأهداف التالية:

- ✓ تعزيز كفاءة وفعالية استعلامات الربط عبر تطبيق خوارزميات التعلم المعزز.
- ✓ ايجاد حلول فعالة من ناحية زمن تنفيذ الاستعلامات.
- ✓ الحدّ من مستويات عدم اليقين الملازمة لخوارزمية Upper Confidence Bounds applied to Trees.
- ✓ توجيه عملية البحث ضمن فضاء البحث عن خطط ربط مثلى للاستعلامات بمعلومات خاصة بالاستعلام.

3. الدراسة المرجعية:

يقترح المؤلفون في [9] ReJOIN بوصفه نموذجاً يعتمد على تقنيات التعلّم المعزز العميق لتحسين ترتيب عمليات الربط في قواعد البيانات. يعتمد على النهج التقليدي القائم على الكلفة في نظم إدارة قواعد البيانات الحديثة. يُصاغ ترتيب الربط كمسألة تعلّم معزز. يعامل كل استعلام كحلقة تتضمن حالات وإجراءات؛ تمثل الحالات الأشجار الفرعية لشجرة الربط مع شروط الربط والاختيار،و تُعرّف الإجراءات بدمج هذه الأشجار وصولاً إلى الحالة النهائية، أما المكافأة تعادل مقلوب الكلفة. يعتمد على خوارزمية تدرج السياسة باستخدام شبكة عصبية تولّد احتمالات الإجراءات الممكنة، وتُضبط أوزانها تدريجياً استناداً إلى المكافآت لتحسين الأداء. يستخدم النموذج متجهاً خطياً لتمثيل الأشجار الفرعية، ومصفوفة ثنائية لتمثيل شروط الربط، إضافة إلى متجه أخر لتمثيل شروط الاختيار. عند تقييمه باستخدام JOB أظهر تفوقاً على PostgreSQL وQuickpick، حيث خفّض متوسط الكلفة إلى 80% من كلفة PostgreSQL وأنتج خططاً أقل كلفة بنسبة 20%، مع أداء أفضل أو مماثل في زمن التنفيذ وزمن تحسين شبه ثابت خطي مقارنة بالزيادة غير الخطية في PostgreSQL. ومع ذلك، فإن أبرز القيود تكمن في اعتماده على نماذج كلفة قد تفتقر إلى الدقة، واقتصاره على جانب ترتيب الربط دون معالجة اختيار المشغلات أو الفهارس، فضلاً عن حاجته إلى فترة تدريب طويلة، إضافة إلى بساطة أسلوب الترميز الذي قد لا يستوعب الخصائص الهيكلية أو الدلالية للاستعلام.

طرحت الدراسة [2] FOOP كإطار عمل يصوغ مشكلة تحسين ترتيب عمليات الربط كمشكلة تعلم معزز مُلاحظة بالكامل يتم تمثيلها وفق Markov Decision Process (MDP)، حيث تُعرِّف الحالات خطط الاستعلام الجزئية والأفعال عمليات الربط المحتملة بين الجداول ممثلة كمتجهات ميزات بينما المكافآت هي الكلفة السلبية لخطة الاستعلام النهائية. يتتبع FOOP جميع العلاقات والنتائج الوسيطة أثناء عملية التحسين و يدمج خوارزميات تعلم معزز متعددة: DQN, DDQN , PPO . استندت عملية التقييم إلى معيار JOB، فأظهرت DQN متوسط كلفة استعلام يقارب 0.2×10^{10} مع تباين كبير في النتائج أدى إلى خطط مرتفعة الكلفة لنصف الاستعلامات. بينما DDQN خفّضت المدى الربيعي والحد الأقصى للكلفة بنسبة 50% وتفوّقت PPO حيث قلّصت المدى الربيعي وأقصى كلفة خطة الاستعلام بأكثر من النصف مقارنةً بـ DDQN. أدى إعادة ترتيب مجموعات التدريب والاختبار وتطبيق التعلم الجماعي إلى تحسين أداء النماذج، إذ انخفضت التكاليف القصوى والمدى الربيعي والمتوسط. كما أظهرت النتائج أن DP تتفوق عادةً من حيث جودة الخطط، لكنها تعاني من تعقيد زمني $O(n!)$ في المقابل، تتميز PPO بتعقيد $O(n)$ وقد أظهر التحليل أن PPO يمكن أن تتفوق على DP في الاستعلامات التي تتراوح بين 3 و16 ربطاً. أما من حيث زمن التدريب، استغرقت DQN وDDQN حوالي 7 دقائق، بينما احتاجت PPO إلى 25 دقيقة. ومع التعلم الجماعي، ارتفع زمن التدريب ليصل إلى 45 دقيقة لـ DQN و125 دقيقة لـ PPO. وللمقارنة، يتطلب ReJoin نحو 3 ساعات. يواجه FOOP قيوداً تتعلق بالاعتماد على تقديرات غير دقيقة للكلفة، إضافةً إلى عبء حسابي مرتفع عند تطبيق التعلم الجماعي، مما يضعف قابليته للتوسع في بيئات قواعد البيانات الكبيرة.

اقترح الباحثون في [10] RTOS كمُحسِّن متعلّم يدمج التعلّم المعزز العميق مع بنية Tree-LSTM لمعالجة مسألة ترتيب الربط. تُصاغ كمسألة تعلّم معزز. تمثل الحالات غابات ربط وسيطة، والاجراءات شروط الربط الممكنة، والمكافآت تُستمد من الكلفة المقدّرة أو زمن الاستجابة الفعلي. يعتمد التدريب على التعلّم متعدد المهام يبدأ باستخدام قيم الكلفة للتعلّم الأولي ثم يُضبط

وفق زمن الاستجابة لتحسين النموذج. يستخدم شبكة DQN لتقريب قيم Q واختيار الاجراء الأمثل، مع الاحتفاظ بالخبرات السابقة في Memory Pool لدعم عملية التدريب. يقدّم تمثيلاً متعدد المستويات يشمل مصفوفة ثنائية للعلاقات، متجهات لترميز شروط الربط والاختيار، ومصفوفات لتضمينات الأعمدة لتمثيل الجداول. أما شجرة الربط فتعالج باستخدام Tree-LSTM عبر دمج نموذج N-ary لالتقاط ترتيب الأبناء وChild-Sum للأشجار غير المرتبة، مع بناء البيان ديناميكياً باستخدام البحث بالعمق أولاً لإنتاج تمثيلات تضمينية لكل شجرة ربط، ليُشكّل متجه الحالة النهائي دمجاً بين تمثيل الاستعلام والغابة. أظهر تقييم RTOS باستخدام PostgreSQL على معياري JOB وTPC-H أنه يحقق كلفة شبه مثالية مماثلة للبرمجة الديناميكية، متفوقاً على جميع الأساليب الأخرى وخاصة على مجموعة JOB بمتوسط كلفة نسبية (MRC) قدره 1.01. كما حقق أفضل أداء في زمن الاستجابة بقيم Geometric Mean Relevant Latency (GMRL) بلغت 0.67 على JOB و0.92 على TPC-H، أي تحسناً يصل إلى 37٪ مقارنة بالبرمجة الديناميكية، مع قدرة عالية على التكيّف مع تغييرات المخطط. يتطلب تدريب RTOS وقتاً طويلاً، إذ تستغرق مرحلة تدريب الكلفة نحو ساعة ونصف، يعقبها ضبط زمن الاستجابة لما يقارب عشر ساعات.فضلاً عن الحاجة إلى بيانات تدريب كبيرة وتغذية راجعة مكثفة، إضافة إلى كلفة زمنية عالية للتدريب المعتمد على زمن الاستجابة، كما أن زمن التخطيط أطول نسبياً بسبب حسابات Tree-LSTM.

تُقدِّم الدراسة GTDD [13] كنهج يجمع التعلّم المعزز العميق وتقنيات تعلّم التمثيل المتقدّم لتحسين اختيار ترتيب عمليات الربط.يعتمد على وكيل قائم على Dueling-DQN يختار إجراءات الربط تدريجياً حتى اكتمال الخطة، ثم يُنفَّذ الترتيب عبر نظام إدارة قواعد البيانات الذي يوفّر تغذية راجعة تُخزَّن وتُستخدم لتحديث الوكيل بشكل تكراري. يمثل الأعمدة كمتجه يصف مشاركته في عمليات الربط وشروط الاختيار المرتبطة به. أما لتمثيل الجداول يدمج جميع تمثيلات الأعمدة الخاصة بالجدول الواحد، مع تضمين العلاقات المستخرجة من المخطط. كما يمثل الاستعلام باستخدام الشبكات العصبية الرسومية حيث تعبر الجداول عقداً وتمثل الحواف شروط الربط. ولتمثيل الحالة يلتقط تطور خطة الربط حيث تُعالج الحالات الوسيطة باستخدام Tree-LSTM، ويُدمج التمثيل

النهائي عبر شبكة عصبية متعددة الطبقات قبل اتخاذ القرار. يعتمد الإطار على التعلّم المنهجي بتقسيم الاستعلامات وفق درجة التعقيد لضمان استقرار التقارب، بينما صُمِّمت دالة المكافأة كنسبة تقارن كلفة الخطة الأساسية بالخطة المتعلمة لتوحيد التغذية الراجعة عبر مختلف الاستعلامات. أظهرت التجارب أن GTDD يتفوّق باستمرار على أحدث المُحسِّنات المتعلَّمة مثل DQ، ReJOIN، RTOS، JOGGER وعلى المناهج الجينية. فقد حقق تقارباً أسرع وأدنى معدل خطأ متوسط علىJOB. كما حقق أفضل Geometric Mean Relevant Latency (GMRL) على JOB وTPC-H. على مستوى القوالب، تفوق GTDD على RTOS في 87% وعلى JOGGER في 78%. وبلغ GMRL قيمة 0.60381 على JOB، محققاً تحسناً مقارنةً بـ GTD (+20%)، RTOS (+23%)، JOGGER (+56%). يتطلب GTDD وقت تدريب أطول من الأساليب التقليدية بسبب العبء الحسابي لبنية Dueling-DQN، كما إن الكلفة المرتفعة للتدريب تحد من إمكانية تطبيقه عملياً في البيئات ذات المتطلبات الزمنية الصارمة. فضلاً عن أن دالة المكافأة صُمِّمت كنسبة بين كلفة الخطط الأساسية والمُتعلمة لتوفير تغذية راجعة موحّدة، إلا أن هذا التجريد قد يكون مبسطاً بشكل مفرط، مما قد يؤثر على جودة الخطة النهائية.

انجز الباحثون في [8] AlphaJoin وهو مُحسّن للاستعلامات مستوحى من AlphaGo، يعتمد على خوارزمية MCTS لتحسين ترتيب الربط. يتألف الإصدار الأول AlphaJoin 1.0 من شبكة OVN للتنبؤ بدرجة زمن التنفيذ لخطط الاستعلام الجزئية وخوارزمية MCTS لاستكشاف فضاء الترتيبات الربط باستخدام خوارزمية UCT لتحقيق التوازن بين الاستكشاف والاستغلال. أما الإصدار الثاني AlphaJoin 2.0 فأضاف شبكة ADN لتحديد ما إذا كان الأنسب استخدام AlphaJoin أو الاعتماد على PostgreSQL. يعتمد النظام على ترميزين متكاملين: ترميز SQL لتمثيل بنية الاستعلام عبر مصفوفة تجاور وترميز السمات المستخدمة في الشروط، وترميز خطة التنفيذ لتمثيل الخطط الجزئية أو الكاملة باستخدام أرقام ترتيبية تشير إلى أولوية عمليات الربط. أظهر تقييم AlphaJoin باستخدام معيار JOB تفوقاً واضحاً على PostgreSQL وNEO حتى مع احتساب زمن البحث. كما حقق AlphaJoin 2.0 انخفاضاً في زمن التنفيذ بنسبة 60.1% مقارنةً بـPostgreSQL، و38.7% مقارنةً بـNEO، و31.3% مقارنةً

بـAlphaJoin1.0،، تقليص نسبة الاستعلامات التي تفوّق فيها PostgreSQL من 48% في الإصدار الأول إلى 9% فقط في الإصدار الثاني.تُظهر نتائج AlphaJoin بعض القيود، أبرزها أن زيادة عامل البحث تُحسّن زمن التنفيذ لكنها ترفع زمن البحث بسبب كثافة محاكاة MCTS. كما أن شبكة OVN تحقق دقة محدودة (نحو 60.9% على JOB)، مما يجعل النظام يعتمد على المحاكاة للتعويض، إضافةً إلى ضعف أداء AlphaJoin 1.0 في الاستعلامات السريعة وعدم اليقين في خوارزمية UCT الذي قد يؤدي إلى ترتيبات غير مثالية.

يتضح من خلال مراجعة الدراسات المرجعية أن معظم الطرق السابقة كانت تعتمد في مناهجها على نماذج كلفة تقديرية تفتقر إلى الدقة مثل [2,9,10] وهو ما انعكس سلباً على جودة خطط الاستعلامات الناتجة. كما لجأت بعض الدراسات إلى أساليب محاكاة عشوائية وخوارزميات تتسم بدرجة من عدم اليقين مثل [8] الأمر الذي يؤدي إلى ترتيبات ربط دون المستوى الأمثل. وانطلاقاً من هذه الملاحظات، يهدف هذا العمل –على خلاف [2,9,10]– إلى تحسين اختيار ترتيب الربط بالاعتماد على أزمنة تنفيذ الاستعلامات بدلاً من النماذج التقديرية غير الدقيقة. كما نسعى إلى معالجة القيود المرتبطة في [8]، وذلك عبر تقليل مستويات عدم اليقين واستبدال المكوّنات العشوائية بأخرى موجّهة بمعلومات الاستعلام، بما يحدّ من احتمالية اختيار ترتيبات ربط ضعيفة ويعزز الوصول إلى حلول أكثر قرباً من المثالية.

4. طرائق البحث:

سنبدأ بطرح مجموعة من المفاهيم الأساسية. نبدأ بتقديم تعريف شامل لعملية الربط في استعلامات SQL. يلي ذلك التطرق إلى مفهوم تحسين الاستعلامات، من ثم نستعرض خوارزمية MCTS بإيجاز وفي الختام، ننتقل إلى شرح الطريقة المقترحة التي تستند إلى مجموعة من الأفكار المستوحاة من العمل المقدم في [8].

1.4عملية الربط:

تُعد عملية الربط إحدى الركائز الجوهرية في استعلامات قواعد البيانات العلائقية، إذ تُستخدم لاسترجاع البيانات من علاقات متعددة بالاعتماد على السمات المشتركة فيما بينها. يمكن أن يُنفّذ الربط بصيغة ثنائية الاتجاه أو متعددة الاتجاهات؛ ففي الحالة الأولى يُطبّق الربط بين علاقتين فقط، بينما في الحالة الثانية يُستخدم لدمج أكثر من علاقتين. وعادةً ما يُنفّذ الربط بين n علاقة عبر سلسلة من $(n-1)$ عملية ربط ثنائية الاتجاه.

2.4 تحسين الاستعلامات:

يلعب تحسين الاستعلامات دوراً محورياً في أداء نظم إدارة قواعد البيانات الحديثة، لا سيما في ظل ظروف البيانات الضخمة. في الأنظمة الواقعية، يمكن أن تتسبب الاستعلامات غير الفعالة في تأخر كبير، واستهلاك مفرط للموارد، وتدهور الخدمة ومع ازدياد اعتماد المؤسسات على الاستعلامات المعقدة، يصبح تحسين الاستعلامات الفعال ضرورياً لضمان الاستجابة، وقابلية التوسع، وكفاءة الكلفة [5]. إن الهدف الأساسي من تحسين الاستعلامات هو الوصول إلى خطة تنفيذية ذات أقل كلفة ممكنة لا سيما من حيث زمن التنفيذ [6].

3.4 خوارزمية Monte Carlo Tree Search (MCTS) العامة:

هي خوارزمية لاتخاذ القرارات تتبع الإجراء العام للبحث في شجرة Monte Carlo [23]. تُوسّع الشجرة من خلال عمليات محاكاة أثناء البحث في الفضاء حتى اتخاذ قرار، وتُعيد النتيجة النهائية للقرار ("فوز" أو "خسارة") إلى عُقد الشجرة لتحديثها. بعد عدد كبير من عمليات المحاكاة، تُمثّل معلومات كل عُقدة نسبة عدد القرارات الصحيحة إلى إجمالي عدد عمليات المحاكاة عند هذه العُقدة، مما يُشير إلى قيمة هذه العُقدة [8]. وقد كان لها بالفعل تأثير عميق على مناهج الذكاء الاصطناعي في المجالات التي يمكن تمثيلها بأشجار من القرارات المتسلسلة، وخاصة الألعاب ومسائل التخطيط.

تعتمد MCTS على مفهومين أساسيين: أن القيمة الحقيقية لأي فعل يمكن تقريبها باستخدام المحاكاة العشوائية؛ وأن هذه القيم يمكن استخدامها بكفاءة لتعديل السياسة باتجاه استراتيجية

الأفضل أولاً. تقوم الخوارزمية تدريجياً ببناء شجرة جزئية، مسترشدة بنتائج الاستكشاف السابق لتلك الشجرة. تُستخدم الشجرة لتقدير قيم الحركات، وتصبح هذه التقديرات (خاصة تقديرات الحركات الواعدة) أكثر دقة مع تقدم بناء الشجرة. تتضمن الخوارزمية الأساسية بناء شجرة بحث بشكل تكراري حتى الوصول إلى حد معين من الموارد الحسابية المحددة مسبقاً –عادةً ما يكون قيداً زمنياً أو متعلقاً بالذاكرة أو بعدد التكرارات –وعند هذه النقطة، يتوقف البحث ويتم إرجاع أفضل إجراء. تمثل كل عقدة في شجرة البحث حالة من حالات المجال، وتمثل الروابط الموجهة إلى العقد الفرعية إجراءات تؤدي إلى الحالات اللاحقة [23].

يتم تطبيق أربع خطوات لكل تكرار بحث:

الاختيار (Selection)، التوسيع (Expansion)، المحاكاة (Roll-out)، والانتشار العكسي (Backup). إلا أنه بدلاً من تقييم العقد الطرفية في الشجرة عبر خطوة المحاكاة، يتم استخدام شبكة عصبية لتوفير سياسة وتقديرات لقيمة الحالة للعقدة الجديدة في الشجرة [7].

i. الاختيار: بدءاً من العقدة الجذر، يتم تطبيق سياسة اختيار بشكل متكرر للنزول عبر الشجرة حتى الوصول إلى العقدة القابلة للتوسيع الأكثر إلحاحاً. تكون العقدة قابلة للتوسيع إذا كانت تمثل حالة غير نهائية ولها عقد فرعية غير مُزارة (أي غير مُوسعة)
ii. التوسيع: تتم إضافة عقدة فرعية واحدة (أو أكثر) لتوسيع الشجرة، وفقاً للإجراءات المتاحة.
iii. المحاكاة: يتم تشغيل محاكاة من العقدة (العقد) الجديدة وفقاً للسياسة لإنتاج نتيجة.
iv. الانتشار العكسي: يتم نسخ نتيجة المحاكاة أي نشرها عكسياً عبر العقد المختارة لتحديث إحصاءاتها.

4.4 الطريقة المقترحة:

تتمثل الفكرة الرئيسية لمنهجيتنا المقترحة على تمثيل جميع ترتيبات الربط الممكنة ضمن بنية شجرية واحدة، بما يتيح تسهيل عملية البحث والاستكشاف. عقب ذلك، نطبق خوارزمية MCTS موجهة معلومات ذات صلة بالاستعلام وقاعدة البيانات (مثل حجم الجدول – من الجدير بالذكر بأننا أجرينا دراسة مطولة باستخدام معلومات أخرى غير حجم الجدول مثل انتقائية عملية الربط

وانتقائية الشروط لكن لم نتطرق لها هنا حرصاً على عدم إطالة المقالة)، بغرض تعلم وتوليد ترتيبات ربط عالية الكفاءة داخل مُحسّن الاستعلام. نوظف شبكة عصبية [8] للتنبؤ بزمن تنفيذ الاستعلام لخطة معينة، بحيث ندمج هذه الشبكة ضمن إطار عمل MCTS لتقييم خطط الاستعلام المُرشّحة.

1.4.4 قاعدة البيانات و عبء العمل:

قاعدة البيانات: تم اختيار قاعدة بيانات Internet Movie Database (IMDB). تتميز هذه القاعدة بغزارة بياناتها وتنوعها، حيث تحتوي على معلومات موسَّعة تتعلق بالأفلام، والممثلين، والمخرجين، وشركات الإنتاج، إضافةً إلى تفاصيل دقيقة حول تواريخ الإصدار وغيرها من المعلومات ذات الصلة [15].

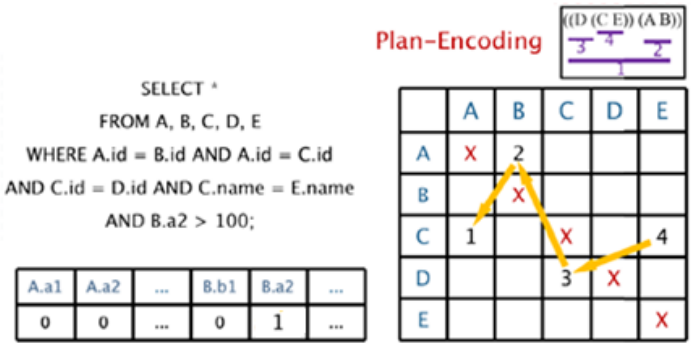
عبء العمل: يُعَدّ معيار ترتيب الربط (Join Order Benchmark – JOB) أحد المعايير المرجعية المهمة في تقييم أداء مُحسِّنات الاستعلامات. يتكوّن هذا المعيار من 113 استعلاماً تحليلياً صُمِّمت لاختبار كفاءة محسنات الاستعلامات على قاعدة بيانات IMDB المجمَّعة من مصادر واقعية [11]. تتراوح هذه الاستعلامات بين 3 و 16عملية ربط، بمتوسط يقارب 8 عمليات ربط لكل استعلام.

2.4.4 ترميز البيانات:

نقوم بتمثيل خطة التنفيذ، سواء كانت جزئية أو كاملة. ويُشترط أن يكون هناك تطابق صارم بين كل ترميز وبين ترتيب الربط المقابل له، بحيث تكون آلية الترميز قابلة للترميز وفك الترميز في آن واحد. يتكون ترميز الخطة أيضاً من عنصرين:

- ❖ مخطط الربط (Join Graph): يُمثَّل باستخدام مصفوفة تجاور تُظهر ترتيب عمليات الربط باستخدام أرقام مختلفة. كلما ارتفع الرقم في المصفوفة، دلّ ذلك على أولوية أعلى لعملية الربط بين الجدولين المعنيين.

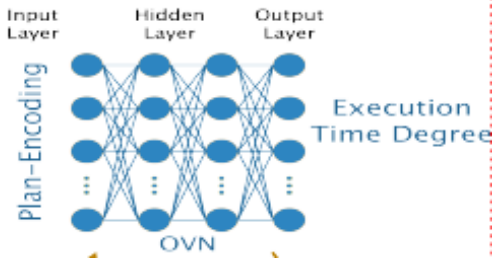
- ترميز الخصائص (Attributes Encoding): يمثل باستخدام الترميز الواحد الساخن One-Hot Encoding بحيث تأخذ الخاصية المشاركة في شرط الاستعلام القيمة 1، بينما تأخذ الخصائص غير المشاركة القيمة 0.



الشكل 4-1: طريقة الترميز Plan Encoding [8]

3.4.4 بناء نموذج الشبكة العصبية Order Value Network (OVN):

تُعد شبكة قيمة الترتيب (Order Value Network -OVN) نموذجاً عصبياً صُمِّم للتنبؤ بدرجة زمن التنفيذ المثلى لاستعلام جزئي ضمن خطة تنفيذية. كما هو موضح في الشكل 4-2، تتألف البنية المعمارية للشبكة من طبقة إدخال (I)، يليها ثلاث طبقات مخفية بأحجام (2048، 512، 128 وحدة) مفعّلة باستخدام دالة ReLU، ثم طبقة إخراج (O).



الشكل 4-2: البنية المعمارية للشبكة العصبية OVN [8]

يهدف هذا النموذج إلى تقدير ما إذا كانت خطط الاستعلام سريعة أو بطيئة، حيث تُغذّى الشبكة بترميز خطة الاستعلام، ويكون الإخراج على شكل تصنيف متعدد الفئات. في هذا السياق، تُستخدم أربع فئات محتملة (K = 4) لتمثيل درجات زمن التنفيذ (من 0 إلى 4)، بحيث تعكس الدرجة الأدنى زمن تنفيذ أقل. لتوليد توزيع احتمالي مناسب على درجات زمن التنفيذ، تعتمد الشبكة على دالة Softmax في طبقة الإخراج، كما يتم تطبيق تقنية Dropout Regularization للحد من مشكلة فرط التعلّم (Overfitting).

4.4.4 توليد بيانات مجموعة تدريب الشبكة العصبية OVN:

نجمع بيانات التدريب الخاصة بشبكة القيمة (OVN) من خلال تنفيذ عدد كبير من خطط الاستعلام على قاعدة بيانات IMDB باستخدام نظام PostgreSQL، وذلك بهدف توليد ترتيبات ربط متعددة، تنفيذها، وتسجيل كل من أزمنة التنفيذ وخطط الاستعلام الناتجة.

نبدأ بتهيئة مجموعة البيانات، ثم قراءة ملفات الاستعلام المتاحة. ولكل استعلام، نولد مجموعة ترتيبات الربط الممكنة عبر استكشاف تركيبات الربط المتوافقة مع شروط الربط المحددة ضمن الاستعلام بشكل تكراري.

ننفّذ كل ترتيب ربط صالح على PostgreSQL مع تفعيل إضافة pg_hint_plan [19]، التي تتيح فرض ترتيبات ربط محددة على النظام بدلاً من الاعتماد على الترتيب الافتراضي الذي يقترحه المُحسِّن الأصلي، وذلك باستخدام ما يُعرف بتلميحات hints.

ولضمان قياس أزمنة تنفيذ الاستعلامات في ظروف متسقة وخالية من تأثيرات التخزين المؤقت، نقوم بإعادة تشغيل PostgreSQL ومسح ذاكرة التخزين المؤقت للنظام قبل تنفيذ الاستعلامات، مما يزيل أي تحيز محتمل في الأداء ناجم عن البيانات المخزنة مسبقاً.

بعد ذلك، نسجِّل نتائج التنفيذ التي نستخرجها باستخدام الأمر EXPLAIN ANALYZE في PostgreSQL، بما يشمل زمن التنفيذ، خطة الاستعلام، وترتيب الربط المستخدم.

تتكرر هذه العملية بصورة مستمرة، بما يضمن كافة الاستعلامات وتغطية فضاء واسع من ترتيبات الربط، وبالتالي توفير بيانات تدريبية غنية ومتنوعة.

ملاحظة: نقوم بفرض قيود على عدد ترتيبات الربط في الاستعلامات التي تتضمن عدداً كبيراً من الجداول (خمسة جداول فأكثر)، وذلك بهدف تحقيق توازن بين تنوع البيانات والجدوى الحسابية.

تنبع هذه الحاجة من أننا لم نفرض أي قيود على بنية شجرة الاستعلام الناتجة، سواء كانت ذات عمق يساري أو يميني أو ذات بنية كثيفة، ونتيجة لذلك يصبح فضاء البدائل الممكنة غير قابل للإدارة بسرعة كبيرة عند أخذ جميع أشكال الأشجار الكثيفة بعين الاعتبار؛ فعلى سبيل المثال، ينتج عن استعلام يتكون من خمسة جداول فقط 1680 ترتيباً محتملاً.

5.4.4 المعالجة المسبقة لمجموعة بيانات التدريب المولَّدة:

من أجل تدريب الشبكة العصبية OVN للتنبؤ بزمن تنفيذ الاستعلامات على مجموعة البيانات المولَّدة، كان من الضروري إجراء معالجة مسبقة تتضمن الخطوات التالية:

- استبعاد بعض الاستعلامات ذات الحجم الكبير: هناك مجموعة من الاستعلامات ضمن نماذج مختلفة من JOB تحتوي على عدد ضخم من الجداول (12 جدولاً فأكثر). لم تتمكن الإضافة pg_hint_plan من فرض ترتيبات ربط بديلة على نظام PostgreSQL، إذ بقي النظام يعتمد على الترتيب الذي يوصي به المحسّن الداخلي. ونتيجة لذلك، لم يكن بالإمكان الحصول على زمن التنفيذ لهذه الفئة من الاستعلامات، مما استدعى استبعادها من مجموعة البيانات. وتشمل هذه النماذج: 24، 26، 27، 28، 29، 30، 33.

- تحديد عتبة زمنية لتنفيذ الخطط: بعض الخطط ضمن مجموعة البيانات المولَّدة استغرقت زمناً طويلاً جداً للتنفيذ، وصل في بعض الحالات إلى أكثر من ساعة. ولتفادي الاستهلاك المفرط للوقت والموارد الحاسوبية، كان من الضروري وضع عتبة زمنية قصوى. ولتحديد هذه القيمة، أجرينا سلسلة من التجارب باستخدام عتبات زمنية مختلفة: 5 دقائق، 10 دقائق، 15 دقيقة، 20 دقيقة، 30 دقيقة، 60 دقيقة. وقد أظهرت النتائج أن قيمة 5 دقائق هي الأكثر كفاءة.

- ترميز الخطط المولَّدة: من أجل تدريب الشبكة العصبية OVN على التنبؤ بزمن تنفيذ الاستعلامات، كان من الضروري تحويل مجموعة الخطط الناتجة إلى تمثيل قابل للمعالجة من قبل النموذج. ولتحقيق ذلك، قمنا بترميز جميع الخطط وفقاً لترميز الخطة المشروح سابقاً.

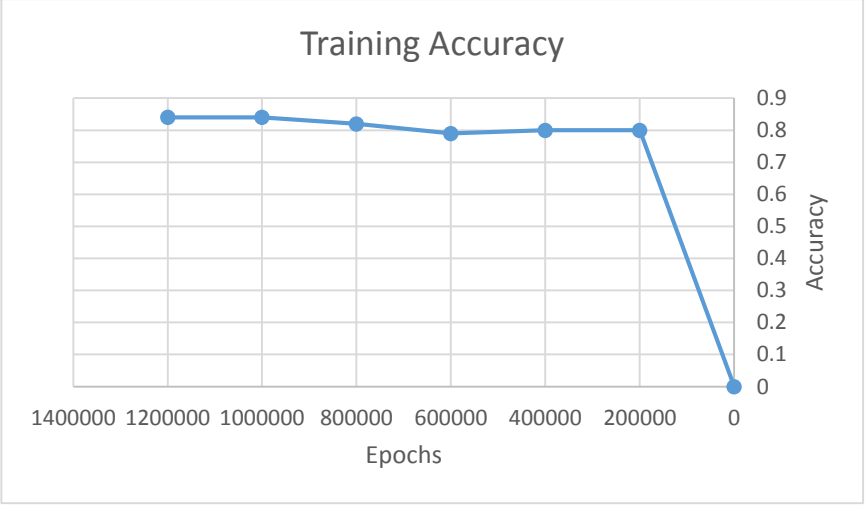
- إضافة ترميز الخصائص: يتم الدمج بين ترميز الخطة وشعاع ترميز الخصائص لنشكل ترميز SQL.

- إضافة الفئات: نقوم بترتيب مجموعة البيانات تصاعدياً اعتماداً على أزمنة التنفيذ ومن ثم نقسم مجموعة البيانات المرتبة إلى k مجموعة بحيث كل مجموعة لها نفس الفئة (تستخدم لتدريب الشبكة باستخدام التعلم الخاضع للإشراف) والتي تمثل درجة سرعة التنفيذ.

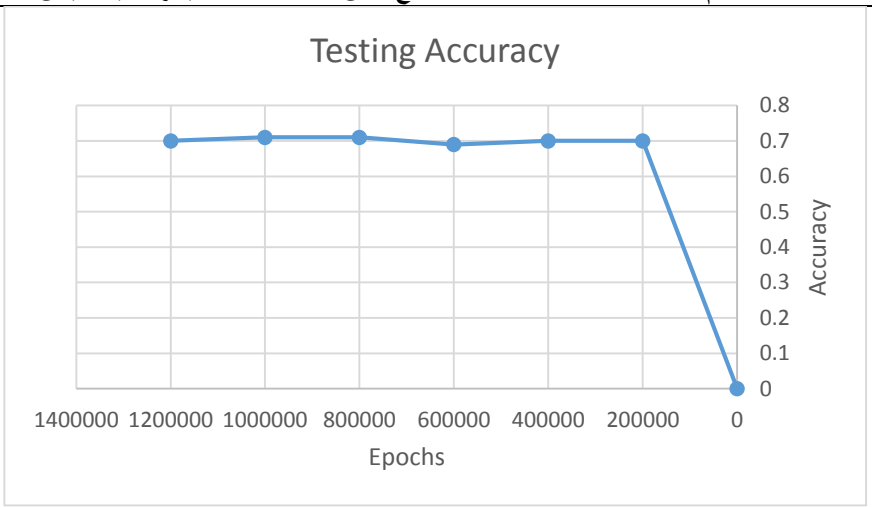
وفي نهاية المطاف نقسم مجموعة البيانات إلى 70% بيانات تدريب و30% بيانات اختبار.

6.4.4 تدريب الشبكة العصبية OVN:

يجري تدريب نموذج الشبكة العصبية OVN بالاعتماد على مجموعة بيانات التدريب، وذلك بهدف تقليل قيمة دالة الخسارة Cross Entropy [16] بين خطط الاستعلام التاريخية وزمن التنفيذ المتوقع لها. وتُعد دالة Cross Entropy المعيار القياسي في مشاكل التصنيف متعدد الفئات، إذ يعد مقياساً لمدى التقارب التوزيعات الاحتمالية بين القيم المتوقعة والقيم الفعلية[8]. استغرقت عملية التدريب ما يقارب خمس ساعات، وأسفرت النتائج عن تحقيق دقة بلغت 84% على مجموعة بيانات التدريب، مقابل 70% على مجموعة بيانات الاختبار. يوضح الشكل 4-3 والشكل 4-4 المنحنى البياني لمستوى الدقة المحقق على كل من مجموعة التدريب ومجموعة الاختبار على الترتيب.

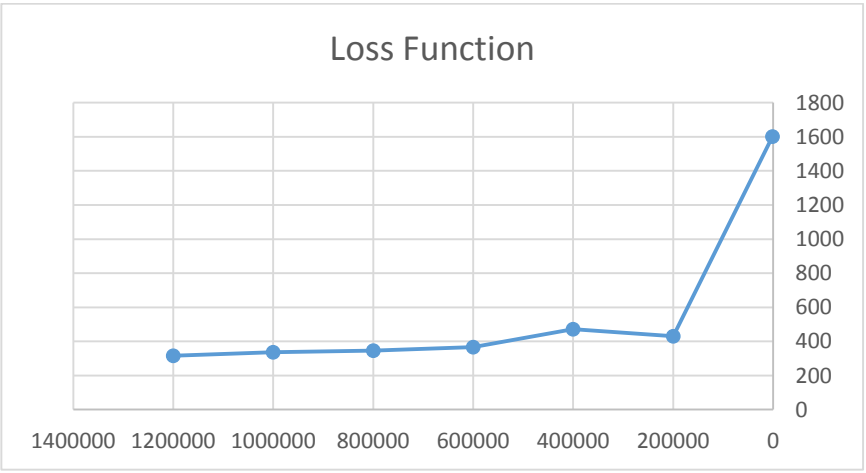


الشكل 4-3: المنحني البياني لدقة تدريب الشبكة OVN على مجموعة بيانات التدريب



الشكل 4-4: المنحني البياني لدقة تدريب الشبكة OVN على مجموعة بيانات الاختبار

يوضح الشكل 4-5 المنحني البياني لدالة الخسارة أثناء عملية التدريب:



الشكل 4-5: المنحني البياني لدالة الخسارة أثناء عملية تدريب الشبكة

7.4.4 خوارزمية Monte Carlo Tree Search (MCTS) المعدلة:

قمنا بإعادة تنفيذ خوارزمية Monte Carlo Tree Search (MCTS) كما وردت في [8]، مع إدخال مجموعة من التحسينات البنيوية والمنهجية على مراحلها المختلفة، وذلك على النحو الآتي:

- ✓ تطوير نسخة معدّلة من خوارزمية UCT: قمنا بتعديل خوارزمية Upper Confidence Bounds applied to Trees (UCT) لتعتمد على معلومات موجّهة بالاستعلام (حجم الجدول)، إضافةً إلى مجموعة من القواعد الاستدلالية التي تُستخدم في مرحلة اختيار العقدة التالية، بما يعزز جودة عملية التوسّع داخل الشجرة.
- ✓ استبدال سياسة المحاكاة العشوائية: قمنا باستخدام سياسة محاكاة موجهة بالمعلومات ذات الصلة بالاستعلام بدلاً من السياسة العشوائية.

بداية سنقدم خوارزمية UCT ثم نسلط الضوء على النسخة المعدلة منها يلي ذلك مناقشة تابع الجائزة وفي الختام، نستعرض خوارزمية MCTS بمراحلها الأربعة.

1.7.4.4 خوارزمية Upper Confidence Bounds applied to Trees (UCT):

تُعد خوارزمية حدود الثقة العليا المطبقة على الأشجار (UCT) إحدى خوارزميات البحث في أشجار الألعاب، وتهدف إلى معالجة مسألة اختيار العقدة المثلى ضمن فضاء البحث. وتعتمد هذه الخوارزمية على مبدأ الاستكشاف مقابل الاستغلال، بحيث توازن بين الاستفادة من المعرفة المتراكمة حول العقد التي تم اختبارها سابقاً، وبين استكشاف عقد جديدة لم تُجرَّب بعد، بما يحدّ من احتمالية الوقوع في الحلول المثلى المحلية. صيغة حساب UCT هي:

$$\mathbf{UCT}(v_i, v) = \frac{Q(v_i)}{N(v_i)} + C\sqrt{\frac{\log N(v)}{N(v_i)}}$$

حيث أن:

- تُمثل v_i العقدة الحالية التي يتم دراستها
- v العقدة الأبوية للعقدة v_i.
- $Q(v_i)$ عدد مرات اكتساب ميزة في العقدة الحالية.
- $N(v_i)$ $(N(v)$ إجمالي عدد مرات الوصول إلى العقد الحالية (العقد الأبوية).
- C مُعامل لضبط حساسية الاستكشاف.

يُشير الجزء الأول من الصيغة إلى الاستغلال الذي يُحدد احتمالية "الفوز" بعد اختيار هذه العقدة. أما الجزء الثاني من الصيغة فيُشير إلى الاستكشاف، حيث تزداد احتمالية استكشاف العقد الأخرى غير الممسوحة (بدلاً من v_i) إذا كانت قيمة $N(v_i)$ كبيرة. أن المسار من كل عقدة فرعية إلى العقدة الجذر في الشجرة يُمثل ترتيب ربط كامل [8].

بالاستفادة من [14] التي قامت بتطبيق خوارزمية PUCT (Predictor + UCT) في مرحلة الاختيار من MCTS كالتالي:

$$a = \arg\max_{a} Q(s,a) + c\ P(s,a)\ \frac{\sqrt{N(s)}}{1+N(s,a)}$$

حيث أن:

- $P(s, a)$ يرمز إلى الاحتمال المسبق لاختيار الإجراء a في الحالة s.
- $Q(s,a)$ هي متوسط قيمة الإجراء (الاستغلال).
- c هو مُعامل فائق يُعدِّل التوازن بين الاستكشاف والاستغلال.

- N(s) هو إجمالي عدد الزيارات للحالة s الحالية.
- N(s,a) هو عدد الزيارات لاتخاذ الإجراء a من الحالة s.

قمنا بتعديل خوارزمية UCT لتقوم بتقدير جودة الحالة (العقدة) من خلال تحليل بنية الربط الحالية وتطبيق قاعدة استدلالية تُفضل الربط بين الجداول الأصغر حجماً (وهو مبدأ شائع في تحسين الاستعلامات). غالباً ما يُقلل ربط الجداول الأصغر حجماً أولاً من أحجام النتائج الوسيطة مما يُساعد هذا في توجيه البحث نحو خطط استعلام أكثر كفاءة ويُحسّن الأداء.

صيغة حساب UCT المعدلة:

$$UCT = Exploitation + Exploration + Heuristic\_value$$

حيث أن:

Exploitation تشير إلى جزء الاستغلال من صيغة UCT الاساسية

Exploration تشير إلى جزء الاكتشاف من صيغة UCT الاساسية

Heuristic_value قيمة استدلالية تعتمد على معلومات الاستعلامات (مثل حجم الجدول).

2.7.4.4 تابع الجائزة:

تمت إعادة بناء تابع المكافأة وفقاً لما ورد في المرجع [8]. في سياق تحسين الاستعلام، يتمثل الهدف الرئيسي في توليد خطة استعلام تحقق أدنى زمن تنفيذ ممكن؛ إذ إن انخفاض زمن التنفيذ يعكس زيادة احتمالية تحقيق الأداء الأمثل. وقد صُمم تابع المكافأة على النحو التالي:

$$R_j = (K - k_j) / K$$

حيث تمثل R_j قيمة المكافأة الخاصة بترتيب الربط j الذي يتم تقييمه باستخدام خوارزمية MCTS. تشير k إلى القيمة المرجعية لزمن التنفيذ المحددة مسبقاً ضمن شبكة OVN، بينما يرمز K_j (يتراوح بين 0 و k) إلى زمن التنفيذ المتوقع لخطة الربط j. تصل المكافأة إلى قيمتها العظمى، أي 1، عندما يُتنبأ بأن زمن التنفيذ للخطة يساوي الحد الأدنى الممكن، وهو 0.

3.7.4.4 خوارزمية MCTS لمشكلة تحسين ترتيب عمليات الربط:

تم تنفيذ خوارزمية MCTS وفقاً للمنهجية الواردة في المرجع [8]، مع إدخال مجموعة من التعديلات في مرحلتي الاختيار والمحاكاة. تُعد خوارزمية MCTS إحدى خوارزميات اتخاذ القرار التي تعتمد على توسيع الشجرة عبر عمليات محاكاة متكررة ضمن فضاء البحث حتى الوصول إلى قرار نهائي. يتم بعد ذلك إرجاع نتيجة القرار (فوز أو خسارة) إلى العقد الممثلة في الشجرة بغرض تحديثها. ومع تزايد عدد عمليات المحاكاة، تصبح قيمة كل عقدة معبّرة عن نسبة القرارات الصحيحة إلى إجمالي عدد المحاكاة المنفذة عند تلك العقدة، وهو ما يُحدد تقدير قيمتها. ولتقليص مساحة البحث الممتدة من العقدة الجذر إلى العقد الطرفية، تم تعريف عدد عمليات المحاكاة المطلوبة لكل عقدة على النحو التالي:

$$S_n = N_C * F_s$$

حيث يشير N_C إلى عدد العقد الفرعية المنبثقة من العقدة الحالية، بينما يمثل F_s عامل البحث الذي يتحكم في إجمالي زمن المحاكاة. وكلما ارتفعت قيمة F_s، ازداد زمن البحث الذي تستغرقه الخوارزمية. أما إجمالي عدد عمليات المحاكاة لكل استعلام SQL فيُعطى بالعلاقة:

$$\sum_1^J S_n$$

حيث يرمز J إلى عدد عمليات الربط في الاستعلام. تستمر هذه العملية حتى يتم استكشاف جميع ترتيبات الربط الممكنة أو بلوغ الحد الأقصى المحدد مسبقاً لعدد عمليات المحاكاة.

يمكن تقسيم كل عملية محاكاة في خوارزمية MCTS إلى أربع مراحل أساسية [8]:

- الاختيار: نطبيق خوارزمية UCT المعدلة لاختيار ترتيب الربط بدءاً من العقدة الجذرية وصولاً إلى العقدة الطرفية. عند مواجهة عقدة فرعية طرفية أثناء التنقل، تنتقل الخوارزمية إلى مرحلة التوسيع.
- التوسيع: إضافة عقدة فرعية جديدة (تمثل ترتيب ربط جديد) إلى الشجرة عند العقدة التي تم الوصول إليها بشكل أمثل خلال عملية الاختيار [8].
- المحاكاة: بدلاً من إجراء محاكاة عشوائية كاملة حتى نهاية خطة الاستعلام، نستخدم التوجيه الاستدلالي لتسريع عمليات المحاكاة وتحسين كفاءتها. بدلاً من اختيار إجراء عشوائي، نختار أفضل إجراء استدلالي حيث تُعطى الأولوية لعمليات الربط بين الجداول الأصغر لأن عمليات الربط الأصغر عادةً ما تُقلل وقت التشغيل. تجعل هذه السياسة التي قمنا بتنفيذها خوارزمية MCTS للبحث عن الجداول أكثر كفاءة في تحسين الاستعلام من حيث:
  - ✓ عمليات المحاكاة العشوائية البحتة تُهدر الوقت في استكشاف ترتيبات الربط غير المناسبة.
  - ✓ عبر إيلاء الأولوية لعمليات الربط ذات الحجم الأصغر، فإننا نوجَّه عملية المحاكاة نحو خطط أكثر كفاءة وقابلية للتنفيذ.
- الانتشار العكسي: إعادة تمرير نتائج المحاكاة من العقدة الجديدة إلى العقدة الجذرية. خلال هذه العملية، يُزاد العدد الإجمالي لعمليات المحاكاة المسجلة في كل عقدة بمقدار واحد. وإذا أسفرت المحاكاة عن زمن تنفيذ أقل، تُمنح العقدة مكافأة إضافية [8].

8.4.4 أهم الأدوات و المكتبات المستخدمة في البحث:

- ✓ PostgreSQL هو نظام قواعد بيانات علائقية كائنية قوي ومفتوح المصدر، يستخدم لغة SQL ، بالإضافة إلى العديد من الميزات التي تُخزّن وتُوسّع أحجام البيانات الأكثر تعقيدًا بأمان يتميز بموثوقيته، وسلامة بياناته، ومجموعة ميزاته القوية، وقابليته للتوسيع. يعمل PostgreSQL على جميع أنظمة التشغيل الرئيسية، وهو متوافق مع معايير ACID [17].

- ✓ PostgreSQL JDBC 42.2.6: برنامج تشغيل JDBC مفتوح المصدر مكتوب بلغة جافا، ويتصل باستخدام بروتوكول الشبكة الأصلي native network protocol لـ PostgreSQL. يُمكّن برنامج تشغيل JDBC لـ PostgreSQL برامج جافا من الاتصال بقاعدة بيانات PostgreSQL باستخدام شفرة جافا قياسية ومستقلة عن قواعد البيانات. لهذا السبب، فإن برنامج التشغيل مستقل عن النظام الأساسي يمكن استخدامه على أي نظام [18].

- ✓ pg_hint_plan : تتيح الإضافة pg_hint_plan تعديل خطط تنفيذ PostgreSQL باستخدام تلميحات hints في تعليقات SQL. تقرأ عبارات التلميح في تعليق ذي شكل خاص مُعطى لعبارة SQL. يمكن تحديد التلميح بإضافة البادئة "/*+" في بدايته ونهاية العبارة "*/". تتكون عبارات التلميح من أسماء التلميحات والمعلمات محاطة بأقواس ومفصولة بمسافات بيضاء. يمكن استخدام أسطر جديدة في عبارات التلميح لتحسين سهولة القراءة [19].

- ✓ PyTorch: إطار عمل مفتوح المصدر للتعلم الآلي يستخدم على نطاق واسع لبناء وتدريب نماذج التعلم العميق [20].

9.4.4 مقارنة بين الطريقة المقترحة و Alphajoin1.0:

يسلط الجدول 4-1 الضوء على أوجه التشابه والاختلاف بين منهجيتنا المقترحة والعمل المنجز في [8]:

الجدول 4-1: أوجه التشابه والاختلاف بين الطريقة المقترحة و Alphajoin1.0 [8]

المقارنة	أوجه التشابه	أوجه الاختلاف
طريقة الترميز	كلا الطريقتين تستخدمان الترميز plan encoding	لا يوجد
تقييم زمن خطط الاستعلامات	كلا الطريقتين تستخدمان OVN	لا يوجد
MCTS من أجل JOS	كلا الطريقتين تستخدمان MCTS لتوليد ترتيبات ربط للاستعلامات	[8] تستخدم خوارزمية UCT لاختيار العقد وتعتمد على اختيارات عشوائية في مرحلة المحاكاة بينما طريقتنا المقترحة تستخدم نسخة معدلة من خوارزمية UCT في مرحلة اختيار العقد وتطبق سياسات مقادة بمعلومات ذات صلة بالاستعلام لتنفيذ مرحلة المحاكاة

5. التجارب والنتائج:

في إطار التجارب المنفّذة، تم الاعتماد على بيئة Apache NetBeans IDE 14 لتوليد مجموعة البيانات المستخدمة في تدريب النماذج. كما استُخدم نظام إدارة قواعد البيانات PostgreSQL 16، وتم تحقيق الاتصال بين PostgreSQL 16 و Apache NetBeans IDE 14 عبر المشغّل PostgreSQL JDBC 42.2.6 [18]، وهو مشغّل JDBC مفتوح المصدر مكتوب بلغة جافا، يتيح للبرامج المطوّرة بلغة جافا الاتصال بقاعدة بيانات PostgreSQL باستخدام شيفرة قياسية ومستقلة عن نوع قاعدة البيانات.أما فيما يتعلّق ببناء النماذج، فقد تم استخدام PyTorch

[21]، في حين تم تطبيق خوارزمية التعلّم المعزّز MCTS بالاعتماد على التنفيذ المتاح في [8]، وذلك باستخدام الإصدار 3.10.12 من لغة البرمجة Python.

أجريت التجارب باستخدام جهاز حاسوب محمول يعمل بنظام تشغيل Linux Mint 21.3 Cinnamon والجدول 5-1 يظهر مواصفات الجهاز.

الجدول 5-1: مواصفات الجهاز الذي أجريت عليه التجارب

الجهاز	المواصفات
نظام التشغيل	Linux Mint 21.3 Cinnamon
إصدار Cinnamon	6.0.4
المعالج	Intel© Core™ i5-5200U CPU @ 2.20GHz × 2
ذاكرة الوصول العشوائي	12 GB
التخزين	SSD 256 GB HDD 1TB
بطاقة الرسوميات	Intel Corporation HD Graphics 5500

1.5 إعداد التجارب:

مجموعة البيانات وأعباء العمل: تم الاعتماد على مجموعة بيانات مفتوحة المصدر وشائعة الاستخدام على نطاق واسع، وهي IMDB، إلى جانب عبء العمل المقابل لها وهو JOB.
إعدادات قاعدة البيانات وبيئة التنفيذ: أجرينا التجارب على نظام إدارة قواعد البيانات PostgreSQL 16، حيث استخدمنا الإضافة pg_hint_plan [19] لتمكين التلميحات (hints) لمُحسِّن الاستعلامات، بهدف تنفيذ خطط استعلام بديلة عن تلك التي يولّدها PostgreSQL

بشكل افتراضي. كما قمنا بتحديد عتبة زمنية قدرها 5 دقائق للاستعلامات التي تستغرق وقتاً طويلاً للتنفيذ و أخيراً قمنا بتقسيم مجموعة استعلامات حمل العمل لدينا إلى ثلاثة مجموعات كالتالي:

- ✓ استعلامات صغيرة: يبلغ عددها 23 استعلام يتراوح عدد جداولها بين 4 و5 جدول.
- ✓ استعلامات متوسطة: يبلغ عددها 39 استعلام يتراوح عدد جداولها بين 6 و7 و8 جدول.
- ✓ استعلامات ضخمة: يبلغ عددها 31 استعلام يتراوح عدد جداولها بين 9 و10 و11 جدول

المقاييس: نُطبّق عدة مقاييس لتقييم الأداء وهي Workload Relative Latency (WRL) و Geometric mean Relative Latency (GMRL) وإجمالي زمن التنفيذ (يشمل على أزمنة توليد الخطط أيضاً) وعدد الخطط ذات أقل زمن تنفيذ.

يمثل زمن الاستجابة النسبي لحمل العمل WRL إجمالي زمن التنفيذ على كامل حمل العمل. في حمل العمل Q، لتكن S_{bi} وS_i الخُطّتين المولّدتين بواسطة مُحسِّن نظام إدارة قواعد البيانات وطريقتنا المقترحة للاستعلام q_i على التوالي، يُحسب WRL كما يلي:

$$\mathbf{WRL}(Q) = \frac{\sum_{q_i \in Q} T_{\mathrm{exe}}(S_i)}{\sum_{q_i \in Q} T_{\mathrm{exe}}(S_{bi})}$$

تمثل القيمة WRL/1 تسريع زمن استجابة حمل العمل بينما يمثل متوسط زمن الاستجابة النسبي الهندسي (GMRL) المتوسط الهندسي لزمن الاستجابة النسبي ضمن حمل العمل، ويُحسب على النحو التالي:

$$\mathbf{GMRL}(Q) = \sqrt[|Q|]{\prod_{q_i \in Q} \frac{T_{\mathrm{exe}}(S_i)}{T_{\mathrm{exe}}(S_{bi})}}$$

تشير الصيغ السابقة إلى أن القيم الدنيا لكل من WRL وGMRL تعكس أداءً أفضل في التحسين. يقيم كل من WRL وGMRL الأداء من منظور مختلف: حيث يبيّن WRL زمن التنفيذ الكلي، مع تركيز أكبر على الاستعلامات البطيئة، بينما يركّز GMRL على تسريع زمن

الاستجابة النسبي على مستوى كل استعلام، موفراً تقييماً عادلاً لكل من الاستعلامات السريعة والبطيئة، إلا أنه غير قادر على قياس التسريع الكلي على مستوى حمل العمل [12].

المقارنة: نقارن نموذجنا المقترح بالطرق التقليدية وأحدث الطرق المتقدمة على النحو التالي:

- ✓ PostgreSQL: نستخدم مُحسِّن الاستعلامات الخاص بـ PostgreSQL (الإصدار 16) بالإعدادات الافتراضية.
- ✓ PG-NoMerge يحسّن أداء PostgreSQL من خلال إلغاء تفعيل المعامل merge join على كامل حمل العمل [12].
- ✓ alphajoin1.0: منهج يستخدم MCTS لتعلّم وتوليد ترتيب عملية الربط بكفاءة عالية داخل مُحسِّن الاستعلامات [8].

طريقة التقييم: نكرر كل تجربة 10 مرات ثم نعرض متوسط أداء المقاييس على مجموعة النتائج التي حصلنا عليها.

2.5 التجربة الأولى:

في هذه التجربة، قمنا بإعادة تحقيق خوارزمية MCTS كما وردت في [8] مع إدخال مجموعة من التعديلات على النحو التالي:

- ✓ تم تعديل خوارزمية UCT في مرحلة اختيار العقدة لتأخذ بعين الاعتبار معلومات عن الاستعلام، حيث أصبح توجيه اختيارات الخوارزمية يتم من خلال إضافة وزن يعتمد على حجم الجداول الموجودة في الاستعلام المعالج، بدلاً من الاعتماد فقط على عدد زيارات العقدة.
- ✓ تم استبدال السياسات العشوائية في مرحلة المحاكاة بسياسات موجهة بمعلومات عن الاستعلام، بحيث أصبحت اختيارات السياسة تُدار وفق حجم الجداول بدلاً من الاعتماد على الاختيارات العشوائية.

بعد الانتهاء من تنفيذ هذه التجربة، قمنا بمقارنة النتائج التي حصلنا عليها مع كل من:

PostgreSQL , PG-NoMerge , Alphajoin1.0

وذلك باستخدام مجموعة المقاييس التالية:

- ✓ عدد الخطط الناتجة من المُحسِّن التي حققت أقل زمن تنفيذ مقارنةً ببقية المُحسّنات

- ✓ WRL و GMRL
- ✓ إجمالي زمن التنفيذ (بما في ذلك أزمنة توليد الخطط)

وأظهرت النتائج ما يلي:

أولاً عدد الخطط المولدة ذات أقل زمن تنفيذ:

الجدول5-2 : متوسط عدد الخطط المولدة ذات أقل زمن تنفيذ عبر عشرة تكرارات عبر مجموعات الاستعلامات

Query Group \ Optimizer	Alphajoin1.0 Vs PostgreSQL	Alphajoin Table Size Vs PostgreSQL	Alphajoin1.0 Vs PG-No Merge	Alphajoin Table Size Vs PG-No Merge
Small Queries	50	52.61	53.04	57.83
Medium Queries	52.31	61.79	62.82	74.1
Large Queries	25.16	38.71	29.03	44.19

يوضح الجدول 5-2 متوسط عدد الخطط المولدة ذات أقل زمن تنفيذ عبر عشرة تكرارات كنسبة مئوية على مجموعات الاستعلامات الصغيرة والمتوسطة والكبيرة وكانت النتائج كالتالي:
نلاحظ أن طريقتنا المقترحة Alphajoin Table Size تحقق أداءً أفضل بشكل عام مقارنةً بـ AlphaJoin 1.0 في جميع السيناريوهات، سواء عند المقارنة مع PostgreSQL أو مع PG-NoMerge.

بالنسبة إلى الاستعلامات الصغيرة، تسجل الطريقة المقترحة تحسناً بنسبة 2.6% مقارنةً بـ AlphaJoin 1.0 على PostgreSQL وبنسبة 4.78% على PG-NoMerge. أما في الاستعلامات المتوسطة، يكون التحسّن أكثر وضوحاً إذ بلغت نسبة التحسين 9.49% مقارنة مع Alphajoin1.0 على PostgreSQL وبمقدار 11.28% على PG-No Merge وبالنسبة إلى الاستعلامات الكبيرة فتستمر الطريقة المقترحة في تحقيق أفضلية واضحة فبلغت نسبة التحسين 13.54% مقارنة مع Alphajoin1.0 على PostgreSQL و15.16% على PG-No Merge. بشكل عام، تؤكد النتائج أن الطريقة المقترحة تؤدي إلى أداء أكثر استقراراً وتحسيناً ملحوظاً عبر مختلف أحجام الاستعلامات، مع أفضلية واضحة عند المقارنة بخطوط الأساس التقليدية (PostgreSQL و PG-NoMerge).

الجدول 5-3: متوسط عدد الخطط ذات أقل زمن تنفيذ عبر عشرة تكرارات

المقارنة	المتوسط
Alphajoin1.0 Vs PostgreSQL	39.7
Alphajoin Table Size Vs PostgreSQL	48.2
Alphajoin Table Size Vs Alphajoin1.0	57.8
PG-No Merge Vs PostgreSQL	7.7
Alphajoin1.0 Vs PG-No Merge	45.7
Alphajoin Table Size Vs PG-No Merge	55.9

يوضح الجدول 5-3 متوسط عدد الخطط الناتجة ذات أقل زمن تنفيذ عبر التكرارات العشرة ونلاحظ التالي:

الطريقة المقترحة هي الأفضل من حيث عدد الخطط المثلى (ذات أقل زمن تنفيذ)، حيث تتفوق على جميع المحسنات الأخرى، خصوصاً عند مقارنتها بـ Alphajoin1.0 و PG-No Merge.

كما أن الطريقة المقترحة تتميز بقدرتها على توليد عدد أكبر من الخطط ذات زمن تنفيذ منخفض، مما يدل على كفاءة عالية في تحسين الأداء بينما Alphajoin1.0 يقدم أداء أفضل من PG-

No Merge لكنه لا يرقى إلى مستوى الطريقة المقترحة أو PostgreSQL. بشكل عام تُظهر النتائج أن الطريقة المقترحة تتفوق بوضوح في اختيار الخطط المثلى، مما يجعلها خياراً واعداً لتحسين أداء أنظمة قواعد البيانات.

ثانياً WRL وGMRL:

الجدول5-4 : متوسط قيم المقياس WRL على مستوى عشرة تكرارات

المقارنة	المتوسط
WRL (.Alphajoin Table Size, PostgreSQL)	0.9
WRL (.Alphajoin Table Size, PG- No Merge)	0.79
WRL (.Alphajoin Table Size, Alphajoin1.0)	0.87

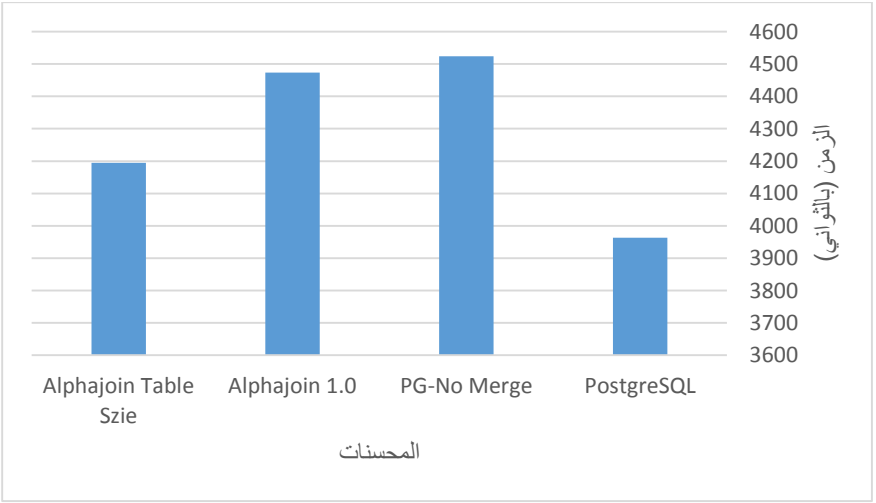
يوضح الجدول 5-4 متوسط قيم المقياس WRL على فترة 10 تكرارات كنسبة مئوية ونلاحظ أن الطريقة المقترحة Alphajoin Table Size تحقق تسريعاً قدره 1.11x مقارنة مع محسن PostgreSQL و1.26x مقارنة مع PG-No Merge و1.14x مقارنة مع alphajoin1.0. بشكل عام تؤكد النتائج السابقة أن الطريقة المقترحة هي الأفضل من حيث تخفيض زمن الاستجابة النسبي.

الجدول 5-5: متوسط قيم المقياس GMRL على مستوى عشرة تكرارات

المقارنة	المتوسط
GMRL (.Alphajoin Table Size, PostgreSQL)	1.15
GMRL (.Alphajoin Table Size, PG- No Merge)	0.96
GMRL (.Alphajoin Table Size, Alphajoin1.0)	0.78

يوضح الجدول 5-5 متوسط قيم المقياس GMRL على فترة 10 تكرارات كنسبة مئوية ونلاحظ أن الطريقة المقترحة Alphajoin Table Size تحقق تسريعاً قدره 0.9x مقارنة مع محسن PostgreSQL (أداء مقارب لـ PostgreSQL) و1.04x مقارنة مع PG-No Merge و1.3x مقارنة مع alphajoin1.0. ويعزى سبب التراجع الطفيف عند المقارنة مع PostgreSQL أن المتوسط النسبي الهندسي حساس لكافة الاستعلامات السريعة والبطيئة ومن الجدول 5-3 فإن PostgreSQL يتفوق بنسبة 44.8 بالمتوسط بعدد الخطط المثلى الناتجة على الطريقة المقترحة (تبقى الطريقة المقترحة أفضل بنسبة 48.2).

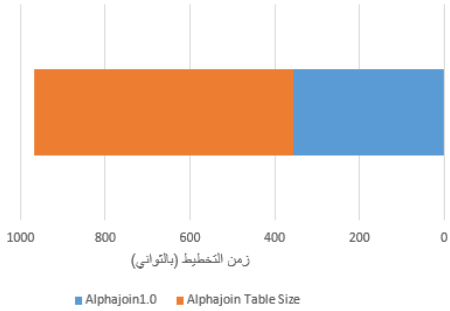
ثالثاً إجمالي زمن التنفيذ:



الشكل 5-1: متوسط إجمالي أزمنة تنفيذ المحسنات عبر عشرة تكرارات (أزمنة التخطيط مضمنة)

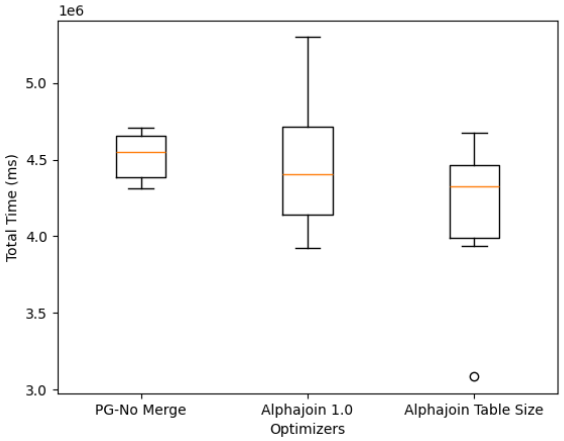
يوضح الشكل 5-1 متوسط إجمالي أزمنة تنفيذ المحسنات بما يشمل أزمنة التخطيط أيضاً عبر عشرة تكرارات ونلاحظ التالي: PostgreSQL يتفوق من حيث سرعة التنفيذ، بينما تقدم الطريقة المقترحة أداء جيد، أما Alphajoin 1.0 سجل متوسط زمن تنفيذ أبطأ من الطريقة المقترحة،

مما يبرز أثر التحسينات الجديدة واخيراً يعاني PG-No Merge من أعلى زمن تنفيذ، ما يعكس ضعفاً واضحاً في الأداء.



الشكل 5-2: متوسط أزمنة التخطيط عبر عشرة تكرارات

يوضح الشكل 5-2 متوسط أزمنة التخطيط عبر عشرة تكرارات ونلاحظ ارتفاع متوسط زمن تخطيط الطريقة المقترحة ويعزى ذلك بسبب أنها تجري حسابات أكثر في عملية اختيار العقد وتنفيذ السياسة الموجهة بمعلومات الاستعلام وبالتالي تجبر عملية البحث على تقييم قرارات (عقد) بشكل أكثر تنظيماً بدلاً من الاعتماد على عمليات المحاكاة العشوائية السريعة ومنه صحيح أننا ندفع كلفة حسابية إضافية لكننا نحصل على توجيه أذكى وبالتالي خطط أسرع.



الشكل 5-3: تقييم استقرار أداء المحسنات

يوضح الشكل 5-3 مخطط Box Plot يقارن إجمالي زمن التنفيذ عبر توزيع القيم وليس فقط المتوسط ونلاحظ التالي: يسجل Alphajoin 1.0 أعلى زمن تنفيذ وأكبر تشتت، مما يدل على عدم استقرار الأداء ووجود استعلامات بطيئة جداً بينما يقدم PG-No Merge أداءً أفضل من Alphajoin 1.0 بينما Alphajoin Table Size هي الأكثر استقراراً والأقل زمناً مما يعكس كفاءة عالية في اختيار الخطط وتنفيذها. الوسيط المنخفض في Alphajoin Table Size يعني أن معظم الاستعلامات تنفذ بسرعة. بشكل عام تؤكد النتائج أن الطريقة المقترحة تتفوق من حيث استقرار الأداء، مقارنة بالمحسنات الأخرى، مما يجعلها خياراً مثالياً في بيئات قواعد البيانات التي تتطلب استجابة سريعة ومتسقة.

3.5 التجربة الثانية:

تتيح دراسات الإزالة Ablation Study فهماً دقيقاً لدور كل مكوّن داخل النظام ومدى مساهمته في الأداء الكلي. ومن خلال إزالة عنصر واحد في كل مرة أو تعطيل وظيفة محددة، يمكن قياس التأثير المباشر لهذا المكوّن على جودة النتائج أو كفاءة التنفيذ مما يعزز موثوقية النموذج عبر سيناريوهات مختلفة. وفي هذا السياق، تهدف التجارب التالية إلى تحليل مكونات الطريقة المقترحة. قمنا في هذه التجربة بدراسة تأثير ما يلي:

- خوارزمية UCT الموجهة بمعلومات خاصة بالاستعلام.
- السياسة الموجهة بمعلومات الاستعلام.

تم إجراء التجربة لكل مكون من المكونات السابقة لخمسة تكرارات مع تحديد قيمة العتبة الزمنية للاستعلامات التي لها زمن تنفيذ طويل بمقدار 3 دقائق.

بعد الانتهاء من تنفيذ التجارب، قمنا بمقارنة النتائج التي حصلنا عليها مع كل من:

PostgreSQL , Alphajoin 1.0

وذلك باستخدام المقاييس التالية:

- عدد الخطط الناتجة من المُحسِّن التي حققت أقل زمن تنفيذ مقارنةً ببقية المُحسّنات.
- إجمالي زمن التنفيذ (بما في ذلك أزمنة توليد الخطط).

دراسة تأثير خوارزمية UCT الموجهة بمعلومات خاصة بالاستعلام:

ملاحظة: سوف نستخدم الاسم Alphajoin UCT-D للإشارة إلى هذه التجربة في النتائج.

في هذه التجربة قمنا بإعادة تحقيق MCTS كما ورد في [8] لكن مع استبدال خوارزمية UCT بالخوارزمية الموجهة بمعلومات الاستعلام. وأظهرت النتائج ما يلي:

أولاً عدد الخطط المولدة ذات أقل زمن تنفيذ:

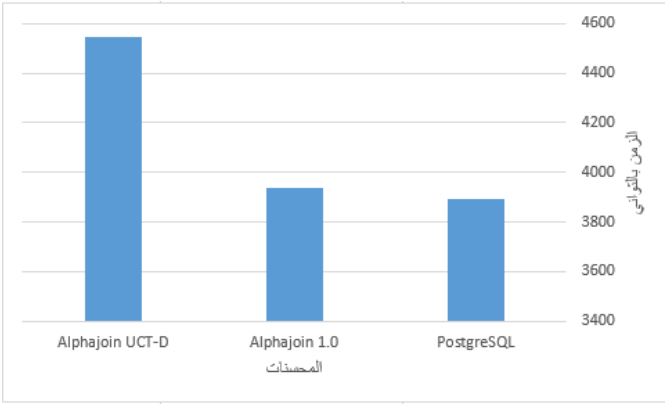
الجدول 5-6: متوسط عدد الخطط ذات أقل زمن تنفيذ عبر خمسة تكرارات

Query Group \ Optimizer	Alphajoin1.0 Vs PostgreSQL	Alphajoin UCT-D Vs PostgreSQL
Small Queries	49	50
Medium Queries	53	51
Large Queries	25	25

يشير الجدول 5-6 إلى متوسط عدد خطط الاستعلامات التي حققت أقل زمن تنفيذ عبر خمسة تكرارات، ومنه نلاحظ التالي: تشير النتائج إلى أن Alphajoin UCT-D قريبة جداً من النسخة الأساسية. بالنسبة للاستعلامات الصغيرة (فضاء البحث لا يزال قابلاً للإدارة)، نلاحظ تحسناً

طفيفاً، يعزى ذلك إلى أن معلومات الاستعلام تساعد عملية البحث على التركيز مبكراً على ترتيبات الربط الواعدة. أما بالنسبة للاستعلامات المتوسطة، نلاحظ انخفاضاً طفيفاً والسبب هو أنه مع ازدياد عدد الجداول، يتسع نطاق الخطة بسرعة كبيرة، وقد تصبح التوجيهات أقل دقة، ففي نطاق بحث أكبر، تبدأ Alphajoin UCT-D في تفويت بعض المناطق المثلى من فضاء البحث، في حين قد تستكشف Alphajoin1.0 ذات العشوائية الأكبر أحياناً فروعاً أكثر تنوعاً وتصل بالصدفة إلى خطط أكثر مثالية.

ثانياً إجمالي زمن التنفيذ:



الشكل 5-4: متوسط إجمالي أزمنة تنفيذ المحسنات عبر خمسة تكرارات (أزمنة التخطيط مضمنة)

يوضح الشكل 5-4 متوسط أزمنة تنفيذ المحسنات عبر خمسة تكرارات ونلاحظ أن الطريقة المقترحة تسجل أعلى متوسط زمن تنفيذ ويعزى ذلك إلى الاعتماد على العشوائية في عمليات المحاكاة مما يؤدي إلى التأثير على اختيارات الطريقة المقترحة وبالتالي توليد خطط سيئة.

دراسة تأثير السياسة الموجهة بمعلومات الاستعلام:

ملاحظة: سوف نستخدم الاسم Alphajoin P-D للإشارة إلى هذه التجربة في النتائج.

في هذه التجربة قمنا بإعادة تحقيق MCTS كما ورد في [8] لكن مع استبدال السياسة العشوائية بأخرى موجهة بمعلومات الاستعلام. وأظهرت النتائج ما يلي:

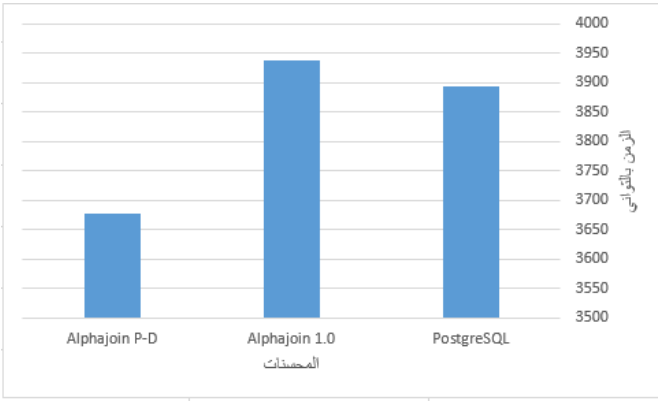
أولاً عدد الخطط المولدة ذات أقل زمن تنفيذ:

الجدول 5-7: متوسط عدد الخطط ذات أقل زمن تنفيذ عبر خمسة تكرارات

Query Group \ Optimizer	Alphajoin1.0 Vs PostgreSQL	Alphajoin P-D Vs PostgreSQL
Small Queries	49	53
Medium Queries	53	56
Large Queries	27	32

يشير الجدول 5-7 إلى متوسط عدد خطط الاستعلامات التي حققت أقل زمن تنفيذ عبر خمسة تكرارات، وذلك في سياقات مقارنة متعددة بين المحسنات المختلفة وتشير النتائج إلى تفوق Alphajoin P-D على Alphajoin1.0 في جميع المقارنات، مما يثبت أن السياسة الموجهة بمعلومات الاستعلام أفضل من السياسة العشوائية كما أنها توجه عملية البحث نحو أجزاء واعدة من فضاء البحث. ومنه نجد أن الطريقة المقترحة قادرة على التكيف مع التوزيع المتفاوت لحجم البيانات وتعقيد الربط.

ثانياً إجمالي زمن التنفيذ:



الشكل 5-5: متوسط إجمالي أزمنة تنفيذ المحسنات عبر خمسة تكرارات (أزمنة التخطيط مضمنة)

يوضح الشكل 5-5 متوسط أزمنة تنفيذ المحسنات عبر خمسة تكرارات ونلاحظ أن الطريقة المقترحة تسجل أقل متوسط زمن تنفيذ بين المحسنات محل المقارنة مما يعكس مدى تفوق السياسة الموجهة بمعلومات الاستعلام على توليد خطط استعلامات فعالة ذات أقل زمن تنفيذ. بشكل عام نستنتج أن الطريقة المقترحة أسرع، وتحقق أداءً أفضل من كل البدائل عبر جميع المقارنات.

6. الاستنتاجات والتوصيات:

تُبرز هذه الدراسة، استناداً إلى مراجعة منهجية وشاملة للأدبيات ذات الصلة، التوجّه المتنامي نحو توظيف تقنيات التعلّم الآلي في تحسين كفاءة وأداء أنظمة قواعد البيانات. وقد برزت تقنيات التعلم المعزز بوصفه أحد الأساليب الواعدة في تحسين ترتيب عمليات الربط، في ظل تزايد الاهتمام بتطبيق هذه التقنيات لمعالجة التحديات المتصاعدة التي تواجهها أنظمة إدارة قواعد البيانات الحديثة.

اعتمدت هذه الدراسة على توظيف خوارزمية MCTS بالتكامل مع معلومات خاصة بالاستعلام، بما يمثّل توجهاً منهجياً نحو تحسين ترتيب عمليات الربط وتعزيز دقّة اتخاذ القرار في توليد خطط تنفيذ فعّالة زمنياً. أظهرت النتائج التجريبية المحققة ضمن إطار العمل المقترح تحسّناً ملحوظاً في جودة خطط الاستعلامات المولّدة، الأمر الذي انعكس مباشرة على تقليص أزمنة التنفيذ. وعلى نحو أكثر تحديداً، حققت المنهجية المقترحة انخفاضاً ملموساً في زمن تنفيذ الاستعلامات، إلى جانب تسريع ملحوظ وتحسين واضح في أداء عبء العمل وزيادة في عدد الخطط الأمثلية الناتجة، مما ينعكس إيجاباً على الاداء العام.

تتجسّد مساهمات هذه الدراسة في محورين رئيسين:

- ✓ توجيه خوارزمية UCT بمعلومات بنيوية عن الاستعلام: إذ أتاح دمج المعرفة ببنية الاستعلام تعزيز فهم الخوارزمية لسياق الاستعلام المعالج، والحدّ من مستويات عدم اليقين الملازمة للخوارزمية.
- ✓ اعتماد سياسة موجهة بطبيعة الاستعلام بدلاً من السياسات العشوائية: لأن استكشاف فضاء البحث الضخم بشكل عشوائي غير فعال يؤدي إلى ترتيبات ربط سيئة وبالتالي الحصول على خطط استعلام كارثية، بينما يضمن التوجيه المستند إلى خصائص الاستعلام مساراً أكثر فاعلية.

7. الأعمال المستقبلية:

استناداً إلى نتائج الدراسة الحالية وما تم تحقيقه من تقدم، يمكن توسيع نطاق العمل البحثي مستقبلاً عبر عدد من المسارات الواعدة، من أبرزها:

- ✓ دراسة إمكانية دمج طريقتنا المقترحة في مُحسِّن قواعد البيانات التقليدي، مما يُتيح لتقنيات التحسين التقليدية الاستفادة من التعلّم الآلي.
- ✓ تسريع عملية التعلم من خلال دمج المعرفة المكتسبة من الخبراء في مرحلة التدريب نظراً لما تتطلبه بعض النماذج من موارد حسابية كبيرة وزمن تدريب طويل عند بنائها من الصفر.
- ✓ دراسة إمكانية مكاملة المزيد من المعلومات الخاصة بالاستعلام بخوارزمية MCTS بما يسهم في رفع كفاءة الخوارزمية وتحسين قدرتها على اتخاذ القرارات المثلى مما ينعكس إيجاباً على أزمنة تنفيذ الاستعلامات.

تلك المقترحات يمكن أن تسهم في تحسين أداء خوارزميات تحسين ترتيب عمليات الربط وزيادة كفاءتها في التعامل مع أعباء عمل متنوعة وقواعد بيانات مختلفة.

8. المراجع:

[1] Vidhya, V., Jeyaram, G. and Ishwarya, K.R, 2016- Database Management Systems. Alpha Science International.

[2] Heitz J, Stockinger K. Join query optimization with deep reinforcement learning algorithms. arXiv preprint arXiv:1911.11689. 2019 Nov 26.

[3] Li G, Zhou X, Cao L. AI meets database: AI4DB and DB4AI. InProceedings of the 2021 international conference on management of data 2021 Jun 9 (pp. 2859-2866).

[4] Sutton R., Barto, A., 2014- Reinforcement Learning: An Introduction. IEEE Transactions on Neural Networks, Vol. 9, No. 5, 1054.

[5] Hettiarachchi N, Yapa P. Advancements in SQL Query Optimization: A Review of Join Order and Index Selection. In2025 5th International Conference on Machine Learning and Intelligent Systems Engineering (MLISE) 2025 Jun 13 (pp. 31-39). IEEE.

[6] Elmasri R, Navathe SB. Fundamentals of database systems seventh edition.

[7] Mazyavkina N, Sviridov S, Ivanov S, Burnaev E. Reinforcement learning for combinatorial optimization: A survey. Computers & Operations Research. 2021 Oct 1;134:105400.

[8] Zhang J, Zhou K, Schelter S. AlphaJoin: Join Order Selection à la AlphaGo. PhD@ VLDB. 2020 Aug 31;2652.

[9] Marcus R, Papaemmanouil O. Deep reinforcement learning for join order enumeration. InProceedings of the First International Workshop on Exploiting Artificial Intelligence Techniques for Data Management 2018 Jun 10 (pp. 1-4).

[10] Yu X, Li G, Chai C, Tang N. Reinforcement learning with tree-lstm for join order selection. In2020 IEEE 36th international conference on data engineering (ICDE) 2020 Apr 20 (pp. 1297-1308). IEEE.

[11] Xu X, Zhao Z, Zhang T, Kang R, Sun L, Chen J. Coool: A learning-to-rank approach for sql hint recommendations. arXiv preprint arXiv:2304.04407. 2023 Apr 10.

[12] Chen T, Gao J, Chen H, Tu Y. Loger: A learned optimizer towards generating efficient and robust query execution plans. Proceedings of the VLDB Endowment. 2023 Mar 1;16(7):1777-89.

[13] Liu C, Kamali A, Kantere V, Zuzarte C, Corvinelli V. A Novel Framework Using Deep Reinforcement Learning for Join Order Selection. arXiv preprint arXiv:2412.10253. 2024 Dec 13.

[14] Kemmerling M, Lütticke D, Schmitt RH. Beyond games: a systematic review of neural Monte Carlo tree search applications. arXiv preprint arXiv:2303.08060. 2023 Mar 14.

[15] Leis V, Radke B, Gubichev A, Mirchev A, Boncz P, Kemper A, Neumann T. Query optimization through the looking glass, and what we found running the join order benchmark. The VLDB Journal. 2018 Oct;27(5):643-68.

[16] Goodfellow I, Bengio Y, Courville A, Bengio Y. Deep learning. Cambridge: MIT press; 2016 Nov 18.

[17] https://www.postgresql.org/about/. Last Visited 29/1/2026

[18] https://jdbc.postgresql.org/. Last Visited 29/1/2026

[19] https://pg-hint-plan.readthedocs.io/en/latest/description.html. Last Visited 29/1/2026

[20] https://pytorch.org/. Last Visited 29/1/2026

[21] https://pypi.org/project/torch/. Last Visited 29/1/2026

[22] Marcus R, Papaemmanouil O. Towards a hands-free query optimizer through deep learning. arXiv preprint arXiv:1809.10212. 2018 Sep 26.

[23] Browne CB, Powley E, Whitehouse D, Lucas SM, Cowling PI, Rohlfshagen P, Tavener S, Perez D, Samothrakis S, Colton S. A survey of monte carlo tree search methods. IEEE Transactions on Computational Intelligence and AI in games. 2012 Feb 3;4(1):1-43.