تحسين طول الجدول النسبي للمهام المترابطة على وحدات معالجة بيانات غير متجانسة

طالبة الدكتوراه رويدة مهباني

كلية الهندسة الميكانيكية والكهربائية – جامعة البعث

بإشراف: د. م. بسيم عمران

الملخص

تُعتبر جدولة المهام المترابطة في الأنظمة غير المتجانسة أمراً بالغ الأهمية وذلك لتقليل زمن الإتمام الكلي حيث يُدخل عدم التجانس والترابط المزيد من التعقيد وقد ركزت دراسات عديدة على دراسة جدولة المهام في الأنظمة المتجانسة واقتصرت بعض الدراسات على الجدولة في الأنظمة غير المتجانسة.

تم في هذا البحث إنشاء خوارزمية جديدة وتم تصميم محاكي للخوارزمية المقترحة وتم اختبارها باستخدام مولد مخططات عشوائي فأعطت هذه الخ وارزمية نتائج أفضل من سابقاتها من حيث طول الجدول النسبي نسبة إلى محدد عدم التجانس.

الكلمات المفتاحية: زمن الإتمام الكلي–المسار الحرج–المهمة–الرتبة–السلف–الخلف.

Improving Schedule Length Ratio of Associated Task on heterogeneous data processing units

Abstract

Scheduling of associated tasks in heterogeneous systems is very important to reduce the total completion time as heterogeneity and coherence introduce more complexity. Several studies have focused on studying task scheduling in homogeneous systems and some studies have been limited to scheduling in heterogeneous systems.

In this paper, a new algorithm was created and a simulation of the proposed algorithm was designed and tested using a random schema generator. This algorithm showed better results than its predecessors in terms of schedule length ratio relative to heterogeneity factor.

Keywords: Makespan - Critical Path - Task - Rank - Predecessor - Successor.

1- مُقدّمة

في عصر البيانات الكبيرة لا يمكن أن تعتمد حوسبة البيانات الكثيفة على معالج واحد لتنجز. لكنها غالباً ما تعتمد على أنظمة الحوسبة غير المتجانسة، والتي تعرف بأنها ربط شبكة عالية السرعة من معالجات متعددة، تعتمد في عملها على الحوسبة المتوازية والموزعة.

تعتمد فعالية تنفيذ التطبيقات في الأنظمة غير المتجانسة على طرائق جدولة المهام، حيث تحسن طريقة الجدولة الفعالة بشكل خاص كفاءة النظام غير المتجانس. تهدف طرق الجدولة لإنقاص زمن الإتمام الكلي[20].

تحتاج خوارزميات الجدولة أن تسجل عمليات المعالجات وقرار انتهائها تحت معيار أسبقية المهام. تتضمن خوارزميات جدولة المهام الأساسية في الأنظمة غير المتجانسة[20]:

- خوارزمية زمن الانتهاء الأبكر غير المتجانس (Heterogeneous Earliest Finish Time: HEFT)
- المسار الحرج على المعالج (Critical Path On a Processor :CPOP)
- الخوارزمية المستندة إلى الانحراف المعياري (Standard Deviation-Based Algorithm for Task Scheduling :SDBATS)
- التنبؤ بأبكر زمن انتهاء (Predict Earliest Finish Time :PEFT)

على الرغم من أن تلك الخوارزميات استخدمت بشكل واسع في الأنظمة غير المتجانسة إلا أنها لاتزال تعاني من مساوئ عدة أبرزها:

أولاً: يتجاهل معظمها عدم تجانس مصادر الحوسبة المختلفة والاتصالات المختلفة بين مصادر الحوسبة.

ثانياً: ليس لديها سياسة فعالة للإدراج.

بهدف حل تلك المشكلات تم اقتراح خوارزمية جديدة، ومقارنتها مع الخوارزميات السابقة فأعطت هذه الخوارزمية نتائجاً أفضل.

2- مُشكلة البحث

يزداد زمن الإتمام الكلي Makespan كلما اختلف التجانس عند جدولة مهام مترابطة على وحدات معالجة بيانات غير متجانسة (أي يوجد زمن تنفيذ مختلف لنفس المهمة المراد جدولتها على أكثر من معالج، حيث يُلحق بكل مخطط يُراد جدولته جدول يُوضح ذلك).

3- الهدف من البحث

يهدف هذا البحث إلى تحسين جدولة المهام المترابطة على وحدات معالجة البيانات غير المتجانسة، بإنقاص طول الجدول النسبي كلما اختلف التجانس بشكل أفضل من الدراسات السابقة.

4- أهمية البحث

يعتبر الجدول الزمني في الحوسبة الموزعة الذي يتم فيه تعيين المهام للمعالجات أمراً بالغ الأهمية، وذلك لتقليل زمن الإتمام الكلي Makespan.

يعتبر تنفيذ المهام في الزمن الحقيقي أمراً مهماً للغاية مثل أنظمة التحكم في العمليات الصناعية، الشبكات الحاسوبية، الروبوتات، الحواسيب.... الخ.

5- آلية البحث

- تقديم دراسات مرجعية.
- دراسة الخوارزميات السابقة وتوضيح دراسة الباحث الحالية.
- استخدام برمجيات متخصصة لبناء محاكي للخوارزمية المُقترحة.
- الحصول على نتائج وتحليلها.
- مقارنة الخوارزمية المُقترحة مع أهم الخوارزميات المستخدمة سابقاً.

6- دراسات مرجعية Reference studies

تم مؤخراً اقتراح عدد من خوارزميات الجدولة في أنظمة الحوسبة غير المتجانسة. يمكن تقسيمها إلى نوعين رئيسين: جدولة ديناميكية وجدولة ستاتيكية.

يكون زمن التنفيذ وزمن الاتصال والعلاقات بين المهام في الجدولة الديناميكية غير معروفة، إذ يتم اتخاذ القرار خلال زمن التنفيذ، بينما في الجدولة الستاتيكية فإن جميع تلك المعلومات معروفة بشكل مسبق، إذ يتم اتخاذ القرار خلال زمن التنفيذ.

الجدولة الديناميكية مناسبة في الظروف التي تكون فيها محددات المهام مجهولة عند زمن المحاكاة[20].

من الخوارزميات الديناميكية:

- Batch Mode Mapping Heuristics [1]
- Dynamic Mapping Heuristics [2]
- Dynamic Scheduling Cycle Strategy [3]
- Dynamic Scheduling Method [4]

يتم تقسيم خوارزميات الجدولة الستاتيكية إلى صنفين رئيسين: خوارزميات الجدولة المعتمدة على البحث العشوائي وخوارزميات الجدولة المعتمدة على الحدس.

- يتضمن الصنف الأول جدولة مهام متعددة المعالجات باستخدام الخوارزميات الجينية (GA Task Scheduling Multiprocessor)[5] .
- نهج المعرفة الوراثية المعززة (Knowledge-Augmented Genetic Approach) [6]
- مشكلة فضاء الخوارزمية الجينية (Problem-Space Genetic Algorithm: PSGA) [7].

أعطت تلك الخوارزميات حلولاً متقاربة من خلال التكرار أكثر، والذي زاد التكلفة مقارنة مع الخوارزميات المعتمدة على الحدس.

تضم الخوارزميات المعتمدة على الحدس ثلاث فئات رئيسة: جدولة القائمة والعنقدة والازدواجية.

من خوارزميات الجدولة الستاتيكية المعتمدة على الحدس الأساسية نذكر الآتي:

- Heterogeneous Earliest Finish Time [8]
- Critical Path On a Processor [8]
- Standard Deviation-Based Algorithm for Task Scheduling [9]
- Predict Earliest Finish Time[10]
- Longest Dynamic Critical Path (LDCP) [11]
- Heterogeneous Critical Parent Trees (HCPT) [12]

- High-PerformanceTask Scheduling (HPS) [13]
- low complexity Performance Effective Task Scheduling (PETS) [14]
- Heterogeneous Earliest Finish with Duplicator (HEFD) [15]
- Selective Duplication Algorithm [16]

للجدولة بالعنقدة قيود كلما زاد عدم التجانس، بينما تسبب الجدولة التي تستند إلى تكرار المهمة تعقيداً زمنياً كبيراً، كما أن تكرار المهمة يستهلك طاقة معالج أكبر، وهذا لا يسبب فقط استهلاكاً أكثر في الطاقة بل استهلاكاً للموارد المشتركة من قبل المعالجات التي تستخدم لتنفيذ مهام أخرى.

تضمن خوارزمية جدولة القائمة جدولاً زمنياً أكثر كفاءة نسبياً مع تعقيد زمني تربيعي. تُعتبر خوارزمية الجدولة HEFT أكثر شيوعاً واستخداماً، إذ تنتج طول جدول زمني قصير [8].

محاسن ومساوئ كل من الخوارزميات السابقة:

الخوارزمية	المحاسن	المساوئ
HEFT	▶ تعتبر خوارزمية HEFT أكثر خوارزمية كلاسيكية للمقارنة. ▶ تستخدم هذه الخوارزمية القيمة المتوسطة لكلفة الحسابات والقيمة المتوسطة لكلفة الاتصالات كقيمة للرتبة لتحديد تسلسل الجدولة.	▶ تعتبر أقل معقولية في البيئات غير المتجانسة. إذا كانت كلفة الحساب لنفس المهمة على معالجات مختلفة كبيرة جداً، فإن هذه الخوارزمية لن تعطي نتائج جدولة مرضية. ▶ هذه الخوارزمية وُضعت لتحسن توازن الحمل حيث تحدد عدد محدد من المهام يمكن تعينها لكل معالج (L عدد المهام / عدد المعالجات) وتسند المهام لكل معالج إذ أن عدد المهام المسندة لكل معالج $\geq L$ ▶ لسوء الحظ يزيد هذا التعديل زمن الانتهاء الإجمالي مما يقلل من أداء هذه الخوارزمية.
CPOP	▶ تملك خوارزمية CPOPأداء جيد عندما تحقق مهام المسار الحرج شروط الجدولة المثلى.	▶ تملك خوارزمية CPOPأسوأ نتائج جدولة، لأنها تُجدول مهام المسار الحرج على نفس المعالج.

PEFT	▶ تأخذ خوارزمية PEFT بعين الاعتبار العقد الأبناء في حساب الأولويات للجدولة لذلك لديها بعض المحاسن عندما تكون التفرعية منخفضة. ▶ هذه الخوارزمية عبارة عن خليط من خوارزمية HEFT وCPOP حيث تحسب أولويات المهام بشكل مماثل لخوارزمية HEFT مع الأخذ بالاعتبار أصغر زمن تنفيذ وكلفة اتصال بين المهام.	▶ عندما تصبح التفرعية عالية فإن هذه المحاسن تقل خاصةً عندما يكون مخطط DAG على درجة عالية من الاختلاف في التجانس. ▶ عندما كلفة الحسابات مختلفة وكلفة الاتصالات للعقد الأبناء مرتفعة تفقد خوارزمية PEFT محاسنها، حتى تصبح في بعض الأحيان خوارزمية HEFT أفضل منها. ▶ تشكل ما يسمى جدول الكلفة الأمثل OCT لأولويات المهام وبالنسبة لطور تخصيص المعالجات للمهام يتم حساب زمن الانتهاء الأبكر الإجمالي OEFT الذي هو مجموع كلفة الحساب لكافة المهام على أطول مسار من عقد الدخول للخروج (المسار الحرج) كما في خوارزمية CPOP ويتم جدولة المهام على المعالج ذو أصغر OEFT.
SDBATS	▶ تُعتبر خوارزمية SDBATS أفضل من خوارزمية HEFT في بعض الأحيان.	▶ تفقد هذه الخوارزمية محاسنها عندما تكون كلفة الاتصالات كبيرة. ▶ تستخدم الانحراف المعياري لكلفة الحسابات لحساب قيمة الرتبة للأولوية بدلاً من القيمة المتوسطة لكلفة الحسابات. هذا سيسبب ظلم في جدولة المهام عندما تكون كلفة الاتصالات كبيرة جداً. ▶ تُشغل هذه الخوارزمية أيضاً مهمة الدخول على جميع المعالجات في بداية الجدولة. ستزيد هذه السياسة من طول الجدولة إن كان هنالك اختلاف ملحوظ في كلفة الحساب بين مختلف المعالجات.
HSIP	▶ خوارزمية HSIP أفضل من باقي الخوارزميات خاصة من حيث اختلافات التجانس الكبيرة. ▶ تضرب هذه الخوارزمية الانحراف المعياري بكلفة الحساب المتوسطة وتعتبرها كتكلفة للحساب، لذلك يمكن الحصول على أولوية أعلى للمهمة ذات فرق الحساب الأعلى وبالتالي فإن هذه الخوارزمية تعطي نتائج جدولة أفضل مقارنةً مع سابقاتها. ▶ تستخدم استراتيجية مضاعفة مهمة الإدخال مُحسنة، حيث تُقيم هذه الاستراتيجية ضرورة إنشاء نسخة على المعالجات الأخرى من عدمها وفق شروط. ▶ تستخدم استراتيجية الإدخال لفتحات الزمن المثالية ITS بالاعتماد على الأمثلة.	▶ لا تأخذ بالاعتبار مهام المسار الحرج التي تستغل معظم موارد الحاسوب حيث تتم جدولتها مثلها مثل أي مهمة.

الجدول () محاسن ومساوئ كل من الخوارزميات السابقة

7- مفاهيم

يتألف نموذج جدولة المهام من: التطبيق وبيئة حساب الهدف ومعايير الأداء. يمكن وصف التطبيق بمخطط بياني لا دوري مباشر

G=(V,E) (Directed Acyclic Graph: DAG)

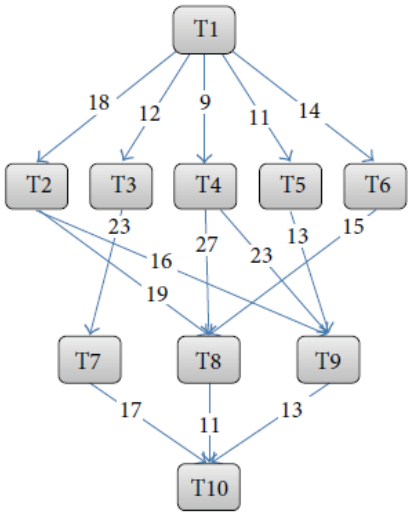
إذ أن: $T = \{t_1, t_2, \dots \dots . t_n\}$ مجموعة من العقد و $E = \{e_1, e_2, \dots . . e_n\}$ مجموعة من المسارات.

يبين الشكل (1) مثالاً عن مخطط DAG، إذ أن كل عقدة تمثل مهمة وكل $e(i,j) \in E$ تُمثل زمن الاتصال بين مهمتين تحت قيود الاعتمادية. تُسمى المهمة بدون أسلاف في المخطط مهمة الدخول بينما المهمة بدون أي خلف تسمى مهمة الخروج. يُلحق المخطط بمصفوفة

$$W = T_{\mathrm{i}} \times P_j$$

حيث أن: i=1: m :m عدد المهام، j=1:n :n عدد المعالجات في النظام. تمثل $W_{i.j}$ الزمن المخمن لإنجاز المهمة t_i على المعالج p_j.

بناءً على ذلك، لدينا في الشكل (1) عدد المهام m=10 وعدد المعالجات n=3 و زمن تنفيذ المهمة T_1 على المعالج $P_1 = 14$ وعلى المعالج $P_2 = 16$ وعلى المعالج $P_3 = 9$ وهكذا.. بالنسبة لبقية المهام، زمن الاتصال بين المهام مُوضح على المسار الواصل بينها.



Task	P_1	P_2	P_3
T1	14	16	9
T2	13	19	18
T3	11	13	19
T4	13	8	7
T5	12	13	10
T6	13	16	9
T7	7	15	11
T8	5	11	14
T9	18	12	20
T10	21	7	16

الشكل (1): مخطط بياني لا دوري مباشر (DAG).

يتّضح من الشكل (1) معنى اعتمادية المهام واختلاف التجانس، إذ أن المهمة T_2 مثلاً تعتمد على المهمة T_1 لتنفيذها، حيث تحتاج زمن اتصال بينها وبين المهمة T_1 مقداره 18 وحدة زمنية لنقل البيانات منها، وذلك يحقق مبدأ الاعتمادية وهكذا بالنسبة لأزمنة الاتصال بين بقية المهام. يتضح معنى اختلاف التجانس من الجدول المُرفق بالمخطط، إذ لدينا ثلاثة معالجات مختلفة في السرعة والأداء، فمثلاً تحتاج المهمة T_1 زمناً مقداره 14 وحدة زمنية لكي يتم تنفيذها على المعالج P_1 وزمناً مقداره 16 وحدة زمنية لكي يتم تنفيذها على معالج آخر مختلف بالسرعة والأداء P_2 و زمناً مقداره 9 وحدة زمنية لكي يتم تنفيذها على المعالج P_3 وذلك يحقق مبدأ عدم التجانس.

يتم حساب الزمن المتوسط لإنجاز المهمة t_i وفق العلاقة (1):

$$\overline{w_i} = \frac{\sum w_{i,j}}{n} \qquad (1)$$

تمثل $c_{i,j}$ في الشكل (1) زمن الاتصال بين المهمة t_i والمهمة t_j. عندما تنفذ المهمتان على المعالج نفسه فإن زمن الاتصال يصبح صفراً نظراً لتجاهل تكلفة الاتصالات داخل المعالج. يُوضع زمن الاتصال عادة على المسار الواصل بين المهام.

بعض التعاريف المهمة المستخدمة في الجدولة:

- **زمن الإتمام الكلي Makespan:** يمثل الزمن المُنقضي من لحظة بدء تنفيذ أول مهمة إلى لحظة تنفيذ آخر مهمة في المخطط يتم حسابه وفق العلاقة (2):

$$makespan = \max\{AFT(t_{exit})\} \qquad (2)$$

حيث يمثل $AFT(t_{exit})$ Actual Finish Time زمن الانتهاء الفعلي لمهمة الخروج.

- $\boldsymbol{EST(t_i, p_j)}$: زمن البدء الأبكر للمهمة t_i على المعالج p_j.

$$EST(t_i, p_j) = max\left\{T_{Avl}(p_j), max_{t_m \in pred(t_i)}\{AFT(t_m) + c_{m,j}\}\right\} \quad (3)$$

حيث أن $T_{Avl}(p_j)$ الزمن الأبكر عندما يكون المعالج p_j جاهزاً. بينما $pred(v_i)$ مجموعة أسلاف المهمة t_i. يمثل القوس الكبير في التعبير السابق الزمن الذي تصل فيه جميع البيانات التي تم طلبها من قبل المهمة t_i إلى المعالج p_j. يكون زمن الاتصال $c_{m,i}$ صفراً عندما تنفذ المهمة السلف على المعالج نفسه p_j. لمهمة الدخول فإن:

$$EST(t_{entry}, p_j) = 0 \quad (4)$$

- $\boldsymbol{EFT(t_i, p_j)}$: زمن الانتهاء الأبكر للمهمة t_i على المعالج p_j.

$$EFT(t_i, p_j) = EST(t_i, p_j) + w_{i,j} \quad (5)$$

والتي تمثل زمن البدء الأبكر للمهمة t_i على المعالج p_j + كلفة حساب المهمة t_i على المعالج p_j. لمهمة الدخول فإن:

$$EFT(t_{entry}, p_j) = w_{t_{entry},j} \quad (6)$$

- **درجة عالية من وزن زمن الاتصال Out-degree communication cost weight (OCCW) للمهمة t_i**: المجموع الأكبر لزمن الاتصال المباشر بين المهمة t_i وخلفائها المباشرين وفق العلاقة التالية:

$$occw(t_i) = \sum_{t_{j \in succ(t_i)}} c_{ij} \quad (7)$$

$$occw(t_{exit}) = 0 \quad (8)$$

تؤثر الدرجة العالية من وزن زمن الاتصال أيضاً على أولويات المهمة. إذا لم تنفذ المهمة ذات عدد كبير من $occw$ فإن جميع خلفائها لن يكونو جاهزين.

تهدف الجدولة لتحديد إسناد المهام في مخطط DAG للمعالجات بحيث يقلل طول الجدول إلى أصغر حد.

8- مقياس الأداء[17]:

طول الجدولة النسبي SLR [18]:

يُعتبر makespan المعيار الذي يُستخدم عادةً لتقييم جدولة مخطط dag واحداً بينما المعيار الذي يُستخدم لمقارنة مخططات مختلفة البنية هو طول الجدولة النسبي SLR والذي يُحسب من العلاقة(11):

$$SLR = \frac{makespan(solution)}{\sum_{t_i \in CP_{MIN}} min_{p_j \in}(w(i,j))} \quad (11)$$

حيث المسار الحرج CP: Critical Path، المقام في التعبير السابق هو أصغر مجموع لزمن الحسابات للمهام في المسار الحرج. لا يوجد طول جدولة أصغر من المقام في التعبير السابق، لذلك فإن الخوارزمية ذات أصغر SLR هي الأفضل.

9- مولد مخططات عشوائي:

لتقييم أداء الخوارزمية المقترحة، تم استخدام مولد مخططات عشوائية معياري [19] إذ تم استدعاء المكتبة المتوفرة وتضمينها في برنامج Visual Studio باستخدام لغة C++ مع العلم أن الذي يحدد شكل المخطط المتولد المحددات الآتية:

- **n:** عدد العقد في المخطط (هذا يعني عدد المهام).
- **محدد الشكل fat:** يؤثر هذا المحدد على ارتفاع المخطط وعرضه. يتم إنشاء الارتفاع أو عدد المستويات حسب عدد المهام في المخطط. القيمة المنخفضة ستقود إلى مخطط صغير (مثلاً سلسلة) بتفرعية مهام أقل بينما القيمة المرتفعة ستؤدي لمخطط بتفرعية أعلى (مثلاً ارتباط شوكة).
- **محدد عدد المسارات density:** يُحدد هذا المحدد عدد المسارات بين مستويين في المخطط، إذ أن القيمة المنخفضة تؤدي إلى حواف أقل والمرتفعة تُنتج العديد من الحواف.
- **محدد التناسق regularity:** يُحدد هذا المحدد تناسق عدد المهام في كل مستوى. تنتج القيمة المنخفضة مخططاً بعدد مهام متباين في كل مستوى، بينما تؤدي القيمة المرتفعة لعدد مهام متماثل في كل مستوى.
- **درجة القفز jump:** يُشير هذا المحدد إلى أن الحافة يمكن أن تنتقل من المستوى L إلى المستوى L+ jump. القفزة 1 هي اتصال مباشر بين المستويات المتتابعة.
- **محدد نسبة الاتصال للحساب Communication to Computation Ratio (CCR):** نسبة مجموع أوزان المسارات إلى مجموع أوزان العقد في المخطط.
- **محدد عدم التجانس لسرعات المعالج (مجال النسبة المئوية لزمن الحساب على المعالجات) β:** تعني القيمة المرتفعة β تجانس أعلى وأزمنة مختلفة للحساب بين المعالجات، بينما تعني القيمة المنخفضة أن أزمنة الحساب للمهمة المُعطاة تقريباً متساوية بين المعالجات. يتم اختيار الكلفة المتوسطة للحساب للمهمة t_i في مخطط مُعطى $\overline{w_i}$ بشكل عشوائي من توزع مُوحد في النطاق $[0.2 \times \overline{w_{DAG}}]$ ،حيث $\overline{w_{DAG}}$ متوسط زمن الحساب للمخطط المُعطى والذي تم توليده بشكل عشوائي. يتم ضبط زمن الحساب لكل مهمة t_i على كل معالج p_j بشكل عشوائي من المجال في المعادلة (12):

$$\overline{w}_i \times (1 - \frac{\beta}{2}) \leq w_{i,j} \leq \overline{w}_i \times \left(1 + \frac{\beta}{2}\right) \quad (12)$$

- تم استخدام القيم الآتية للمحددات لتوليد المخططات العشوائية بشكل مشابه في الدراسات السابقة:

- n =10; 20; 30; 40
- CCR =0.1, 0.5, 1, 2, 5
- β =0.1, 0.2, 0.5, 1, 2
- jump =1
- regularity =0.2; 0.8
- fat =0.1, 0.4, 0.8
- density =0,2; 0,8
- Processors =4, 8, 16, 32

10- الخوارزمية المقترحة:

تم إجراء حسابات رياضية بشكل يدوي تجريبياً، لإيجاد الخوارزمية المقترحة، إذ تم في كل مرة المزج بين الخوارزميات، وإضافة ميزة من خوارزمية إلى خوارزمية أخرى ومقارنة الخوارزمية الهجينة الناتجة مع أفضل الخوارزميات المستخدمة سابقاً إلى أن تم الوصول إلى خوارزمية أعطت زمن الإتمام الكلي Makespan أقصر من الخوارزميات السابقة.

تُعتبر الخوارزمية المقترحة عبارة عن نظام هجين من خوارزميات سابقة عدة مع إضافة بعض التحسينات، إذ تم استخدام تابع الرتبة(والذي يُعرّف بأنه التابع الذي يحدد المهام ذات الألوية الأعلى في التنفيذ، حيث تختلف العلاقة الرياضية التي تحسبه من خوارزمية إلى أخرى، إذ تعتمد في خوارزمية HSIP على الانحراف المعياري ومتوسط زمن التنفيذ للمهمة وزمن اتصالها بالأخلاف ورتبتها وكلما كانت قيمة تابع الرتبة لمهمة أعلى كلما كانت أولويته أعلى) وكذلك مضاعفة مهمة الدخول والإدخال لفتحات الزمن المثالية من خوارزمية HSIP [20] وإضافة فكرة المسار الحرج من خوارزمية CPOP [21] والمضاعفة للمهام من خوارزمية HEFT [21] وPEFT [20] وSDBATS [20]. لذلك تم اصطلاح تسمية الخوارزمية المُقترحة AHPSHC اختصاراً لAdvanced HSIP PEFT SDBATS HEFT CPOP. أي خوارزميةHSIP PEFT SDBATS HEFT CPOP المتطورة.

تعتمد خوارزمية AHPSHC على تابع الرتبة التصاعدية الذي ينطلق في حساب أولوية المهام من مهمة النهاية إلى مهمة البداية، علماً أن هذا التابع ذاته المستخدم في خوارزمية

HSIP لحساب الرتبة، إذ يحقق أداء أفضل من الخوارزميات السابقة من حيث اعتماده على جداء متوسط كلفة حساب المهام في الانحراف المعياري وأكبر قيمة لمجموع زمن الاتصال مع الأخلاف ورتبتها.

$$rank_u(t_i) = max_{t_{j\in succ(t_i)}}\{\overline{w}_i \times \sigma_i + occw(t_i) + rank_u(t_j)\} \quad (13)$$

$$rank_u(t_{exit}) = \overline{w_{exit}} \times \sigma_{exit} \quad (14)$$

حيث:

$rank_u(t_i)$: الرتبة التصاعدية للمهمة t_i.

$succ(t_i)$: خلف المهمة t_i.

$\overline{w}_i$: متوسط كلفة حساب المهمة t_i.

σ_i: الانحراف المعياري للمهمة t_i.

$occw(t_i)$: زمن الاتصال مع الأخلاف.

$rank_u(t_j)$: الرتبة التصاعدية للخلف.

$rank_u(t_{exit})$: الرتبة التصاعدية لمهمة الخروج.

$\overline{w_{exit}}$: متوسط كلفة حساب مهمة الخروج.

σ_{exit}: الانحراف المعياري لمهمة الخروج.

تتميز خوارزمية AHPSHC بميزتين إضافيتين:

1- نسخ السلف الأكبر لمهام المسار الحرج فقط والذي يقلل بشكل كبير من زمن الإتمام الكلي ويخفف من أعباء الحسابات على المعالج، نظراً لأنه عندما تتم جدولة المهمة وسلفها على المعالج ذاته فإن كلفة الاتصال تكون صفراً، كما أن مهام المسار الحرج تتطلب بالذات كلفة بالحسابات واستغلال موارد الحاسوب أكثر بكثير من غيرها من المهام في المخطط.

2- نسخ السلف الثاني أو البقية لمهمة الخروج فقط إن حقق زمن إتمام كلي أقصر أو نسخ السلف الأول والثاني معاً إن حقق كلاهما زمن إتمام كلي أقصر.

مصطلحات:

تم اصطلاح التسميات الآتية:

- كل مهمة من المسار الحرج بالمهمة ذات الأهمية الشديدة VIT.

- السلف الأكبر Maximum parent MP T_k لمهمة ما T_i بين مجموعة أسلافها إذا كان المجموع: $\mathrm{EFT}\ (T_k\ ,P_j) + \mathrm{c}(\ T_k,\ T_i)$ أكبر ما يمكن بين الأسلاف.
- زمن جاهزية البيانات Data Ready Time (DRT) على الوقت المثالي الذي تنتظره المهمة T_i لكي تبدأ التنفيذ على معالج ما.

ملاحظة:

- إذا كانت المهمة الحالية المراد جدولتها VIT، فإنه يتم إنشاء نسخة من سلفها MP على المعالج P_j إذا كان $\mathrm{DRT} > W_{T_{MP},p_j}$، وإلا يتم جدولة المهة على المعالج ذو أصغر EFT.
- في حال كان EFT للمهمة VIT المراد جدولتها على أحد المعالجات أصغر من EFT في حال النسخ وهذا نادراً ،فإننا لا نقوم بإنشاء النسخة في هذه الحالة.
- إذا تساوى EFT لمهمة ما على معالجين، فإننا نختار أي منهما للجدولة.

استراتيجية مضاعفة مهمة الإدخال:

أهمية هذه الاستراتيجية:

بما أن بقية المعالجات ستكون خاملة في حال تمت جدولة مهمة البدء(الدخول) على أحدها، وكون أن العُقد الخلف تعتمد على هذه المهمة في نقل البيانات لها أي أن الخلف عليه أن ينتظر زمن انتهاء تنفيذ مهمة الدخول وزمن الاتصال بها، وعلى اعتبار أن زمن الاتصال سيكون صفر في حال تمت جدولة المهمة السلف والخلف على نفس المعالج، نحسب عندها زمن الانتهاء الأبكر على كل معالج في حالة نسخ مهمة الدخول من عدمها ونختار الجدولة والنسخ على المعالج الذي يعطي زمن الانتهاء الأصغر، وهذا يقلل بالنتيجة من زمن الإتمام الكلي Makespan.

تعطي هذه الاستراتيجية طول جدول أقل لكنها محدودة بالحمل الناتج عن استخدام المعالج. ولكن بالنسبة لمهمة الدخول عندما تعمل على أحد المعالجات، فإن جميع المعالجات الأخرى ستكون خاملة في الوقت ذاته، لذلك لا حاجة لأخذ مشكلة الحمل الزائد للمعالج بالحسبان. كما أن المهام الأخرى ليس عليها أن تنتظر عندما تشغل المعالجات الأخرى نسخة. وطالما أن هذه السياسة مطبقة فقط على مهمة الدخول، فإن ذلك لن يسبب حملاً زائداً على المعالج. تستخدم هذه الخوارزمية هذه السياسة لتجنب التحميل الزائد للمعالج وتحسين فعالية الجدولة الكلية. لجعل زمن نقل البيانات للعقد الخلف أسرع، فإن هذه الاستراتيجية تُقيم ضرورة إنشاء

نسخة على المعالجات الأخرى من عدمها. لن تؤثر هذه الاستراتيجية على نتائج جدولة المهام الأخرى، لأنه لن يتم إنشاء النسخة إن لم تحسن نتائج الجدولة وفق الشرطين الآتيين:

- اختر المعالج p_j ذو أصغر زمن EFT لمهمة الدخول.
- حدد فيما إن كانت مهمة الدخول تحتاج إلى نسخة على معالج آخر p_i. إذا تحققت العلاقة الآتية قم بإنشاء نسخة، وإلا لا تقم بأي شيء.

$$w_{v_{entry},i} < w_{v_{entry},j} + c_{v_{entry},v_n}$$

حيث v_n الخلف المباشر لمهمة الدخول.

ينتهي التحقق من الشرط السابق في إحدى الحالتين الآتيتين:

- أسندت مهام لجميع المعالجات، أي تم اختبار شرط الحكم بإنشاء نسخة من مهمة الدخول على كل معالج.
- تمت جدولة جميع العقد الخلف v_n لمهمة الدخول، أي أنهم لا يحتاجون أن تنقل مهمة الدخول لهم البيانات.

استراتيجية الإدخال لفتحات الزمن المثالية ITS بالاعتماد على الأمثلة:

اعتمدت هذه الاستراتيجية من قبل خوارزميات جدولة عدة، لكن لا يوجد وصف رياضي دقيق لهذه الآلية. عندما تقابل ITS شرط الإدخال، فإن الخوارزميات السابقة تختار أول ITS بدلاً من الأسرع. هذا يسبب مشكلة جدولة غير معقولة.

تم حل هذه المشكلة وفق الآتي:

- بعد الانتهاء من تخصيص مهمة حدث فتحات ITS لجميع المعالجات.
- ابحث عند تخصيص مهمة v_i لمعالج عن فتحات خاملة تحقق الشرط $w_{i,j} \leq$ ITS.
- طبق الشرط السابق على جميع ITS وحدد فيما إن كانت المهمة المسندة إلى ITS قد نفذت وفيما إن كان EFT أقل أو يساوي الحد النهائي ITS.
- إذا هناك العديد من فتحات ITS المحققة للشرط في البند الثاني اختر ITS ذو أصغر EFT.

منهجية PUSUDO لخوارزمية AHPSHC:

Input: DAG,set of tasks V,set of processors P

Output: Schedule result,Makespan

(1) Starting from the exit node, compute $rank_u$ for all tasks by using " improved task priority strategy ".
(2) Sort tasks in schedualing list by decreasing order of $rank_u$ value.
(3) Compute the Critical Path using $CP_x = Max\{\sum_1^n w(t_i)_{max} + \sum \overline{c_{i,succ(t_i)}}\}$.
(4) Select the first task $v_i$from the list for schedualing.
(5) If the task is the entry task
(6) Use " entry task duplication selection policy "
(7) Elseif ti is VIT
(8) If DRT (t_i, pj)>w (MP,pj)
(9) Duplicate MP on pj without violate the dependency constraints
(10) Update EFT of ti on pj
(11) End if
(12) If satisfy the condition of ITS insertion-based optimizing policy
(13) Use " ITS insertion-based optimizing policy "
(14) Else
(15) For each processor in the processor set do
(16) Compute EFT(t_i , pj) value
(17) End for
(18) Assign t_i to the pj that minimizes EFT
(19) For latest task if second MP or any other MP gives shorter scheduale length duplicate it
(20) elseif first and second MP give shorter scheduale length duplicate them
(21) End if

الدخل: المخطط والجدول المُرفق معه.

الخرج: مخطط جانت (زمن الإتمام الكلي).

1-البدء من مهمة الخروج وحساب رتبتها ورتبة المهام السابقة وفق سياسة أولوية المهام المحسنة حسب العلاقة 9 والعلاقة 10.

2-فرز المهام في قائمة الجدولة بترتيب تنازلي لقيمة الرتبة.

3-حساب المسار الحرج باستخدام العلاقة: $CP_x = Max\{\sum_1^n w(t_i)_{max} + \sum \overline{c_{i,succ(t_i)}}\}$

4-اختر المهمة الأولى من قائمة الجدولة.

5-إذا كانت المهمة مهمة دخول

6-استخدم سياسة اختيار تكرار مهمة الدخول.

7-وإلا إذا كانت المهمة شديدة الأهمية

8-وكانت الفتحة الخاملة أكبر من زمن تنفيذ السلف الأكبر

9-انسخ السلف الأكبر دون انتهاك قيود الاعتمادية.

10-حدث قيمة زمن الانتهاء الأبكر للمهمة t_i على المعالج P_j

11–أنهي حلقة if.

12–في حال تحقّق شرط سياسة التحسين القائمة على الإدراج لفتحات الزمن الخاملة

13–استخدم هذه السياسة

14–وإلا

15–من أجل كل معالج من مجموعة المعالجات

16–احسب زمن الانتهاء الأبكر للمهمة t_i على المعالج P_j

17–أنهي حلقة for

18–أسند المهمة t_i على المعالج ذو زمن الانتهاء الأبكر الأصغر.

19–من أجل آخر مهمة إذا كان السلف الأكبر الثاني أو أي سلف آخر يعطي طول جدولة أقصر انسخه.

20–وإلا إن كان السلف الأكبر الأول والثاني سوياً يعطيان طول جدولة أقصر انسخهما

21–أنهي if.

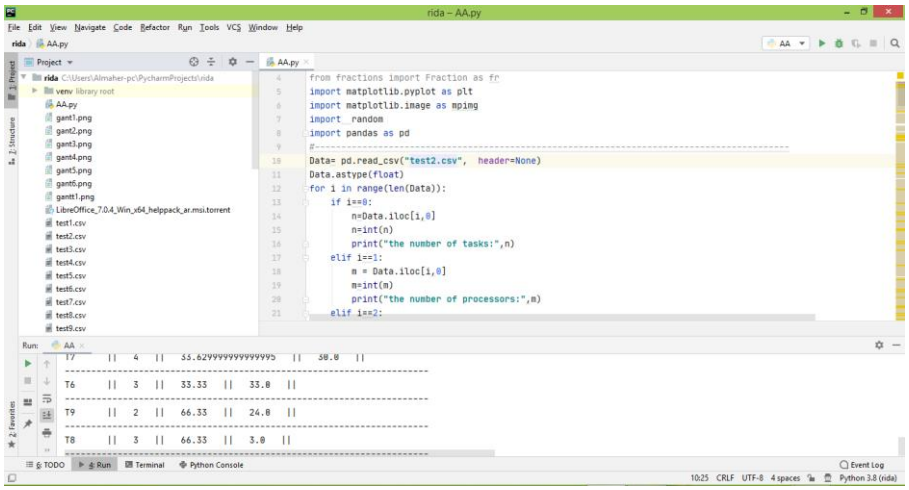
منهجية PUSUDO لخوارزمية HSIP:

Input: DAG, set of tasks V, set of Processors P
Output: Schedule result, makespan
(1) Starting from the exit node, compute $rank_u$ for all tasks by using "**Improved Task Priority Strategy**".
(2) Sort the tasks in scheduling list by decreasing order of $rank_u$ value.
(3) **While** there are unscheduled tasks in the list **do**
(4) Select the first task v_i from the list for scheduling
(5) **If** the task is the entry task
(6) Use "**Entry Task Duplication Selection Policy**"
(7) **Else** (task v_i is not the entry task)
(8) **if** satisfy the condition of ITS insertion-based optimizing policy
(9) Use "**ITS Insertion-based Optimizing Policy**"
(10) **else**
(11) **for** each processor p_j in the processor set ($p_j \in P$) **do**
(12) Compute the earliest finish time (EFT) by (5)
(13) **end**
(14) Assign task v_i to the processor p_j that minimize EFT of task v_i
(15) **End if**
(16) **End if**
(17) Update list
(18) **End while**

لا تأخذ خوارزمية HSIP بالاعتبار مهام المسار الحرج التي تستغل معظم موارد الحاسوب حيث تتم جدولتها مثلها مثل أي مهمة وهذا يزيد من زمن الإتمام الكلي Makespan، حيث تم تحسين ذلك في خوارزمية AHPSHC .

11-بناء محاكي لخوارزمية AHPSHC:

تم بناء محاكي الخوارزمية المقترحة باستخدام لغة البرمجة Python وبيئة العمل PyCharm Community Edition 2020.1 x64 كما هو مُوضح في الشكل (2).



الشكل (2) مُحاكي خوارزمية AHPSHC

بعد أن يتم توليد مخطط DAG كما في الشكل (7) تدخل بياناته على شكل ملف اكسل إلى المحاكي الذي تم بناؤه ليعطينا النتائج.

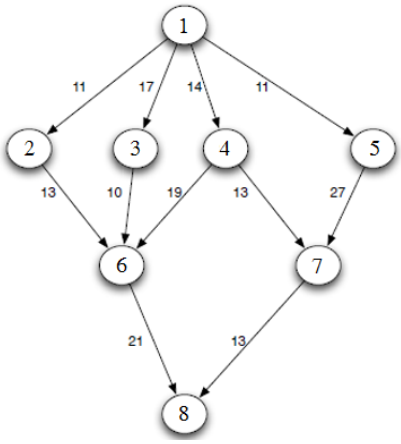
لنطبق الخوارزمية المقترحة على مخطط DAG التالي[22]:

لدينا في الشكل (3) 8 مهام مترابطة يُراد جدولتها على نظام مكون من 3 معالجات مختلفة في السرعة والأداء، إذ أن لكل مهمة زمن تنفيذ مختلف على كل معالج أي أن النظام غير متجانس.

تعتمد المهام T_2 و T_3 و T_4 و T_5 على المهمة T_1 لتنفيذها، إذ تحتاج المهمة T_2 مثلاً زمن اتصال مقداره 11 وحدة زمنية لتنقل المهمة T_1 البيانات لها وهكذا بالنسبة لبقية المهام. ذلك يحقق مفهوم الاعتمادية.

المسار الحرج هو أطول طريق يتم حسابه من مهمة الدخول إلى مهمة الخروج.

Task	p_1	p_2	p_3	$\overline{p_i}$
1	11	13	9	11.0
2	10	15	11	12.0
3	09	12	14	11.6
4	12	16	10	12.3
5	15	11	19	15.0
6	13	09	05	8.60
7	11	15	13	12.3
8	11	15	10	12.0



الشكل (3) مخطط بياني لا دوري مباشر (DAG)

النتائج حسابياً:

1- طور حساب أولوية المهام:

$$rankT_8 = 12 * 2.16 = 25.92$$
$$rankT_7 = Max(12.3 * 1.632 + 13 + 25.92) = 58.993$$
$$rankT_6 = Max(8.6 * 3.265 + 21 + 25.92) = 74.999$$
$$rankT_5 = Max(15 * 3.265 + 27 + 58.993) = 134.968$$

للمهمة T_4 خلفين وهما T_6 و T_7 بالتالي نختار الرتبة الأعلى وفق التالي:

$$T_6:\ rankT_4 = Max(12.3 * 2.494 + 13 + 19 + 74.999) = 137.675$$
$$T_7:\ rankT_4 = Max(12.3 * 2.494 + 13 + 19 + 58.993) = 121.669$$
$$rankT_3 = Max(11.6 * 2.054 + 10 + 74.999) = 108.825$$
$$rankT_2 = Max(12 * 2.16 + 13 + 74.999) = 113.919$$

للمهمة T_1 أربعة أخلاف وهي T_5 ، T_3،T_4 ، T_2 بالتالي نختار الرتبة الأعلى وفق التالي:

$$T_2: rankT_1 = Max(11 * 1.63 + 53 + 113.919) = 184.849$$
$$T_3: rankT_1 = Max(17.93 + 53 + 108.825) = 179.755$$
$$T_4: rankT_1 = Max(17.93 + 53 + 124.675) = 195.605$$
$$T_5: rankT_1 = Max(17.93 + 53 + 134.968) = 205.898$$

بالتالي تصبح أولوية المهام بالترتيب التنازلي:

2-حساب المسار الحرج:

$$CP_{12} = 13 + 11 + 15 = 39$$
$$CP_{13} = 13 + 17 + 14 = 44$$
$$CP_{14} = 13 + 14 + 16 = 43$$
$$CP_{15} = 13 + 11 + 19 = 43$$
$$CP_{126} = 39 + 13 + 13 = 65$$
$$CP_{136} = 44 + 10 + 13 = 67$$
$$CP_{146} = 43 + 19 + 13 = 75$$
$$CP_{147} = 43 + 13 + 15 = 71$$
$$CP_{157} = 43 + 27 + 15 = 85$$
$$CP_{1268} = 65 + 21 + 15 = 101$$
$$CP_{1368} = 67 + 21 + 15 = 103$$
$$CP_{1468} = 75 + 21 + 15 = 111$$
$$CP_{1478} = 71 + 13 + 15 = 99$$
$$CP_{1578} = 85 + 13 + 15 = 113$$

بالتالي فإن المسار الحرج $T_1 \rightarrow T_5 \rightarrow T_7 \rightarrow T_8$

3- طور إسناد المهام للمعالجات:

T_1: مهمة الدخول نجدولها على المعالج ذو أصغر زمن تنفيذ وهو P_3

المهمة ذات الأولوية الأدنى هي T_4:

$$EST_{4,1} = Max(0, \max(9 + 14)) = 23$$
$$EFT_{4,1} = 23 + 12 = 35$$

$$EST_{4,2} = Max(0, \max(9 + 14)) = 23$$
$$EFT_{4,2} = 23 + 16 = 39$$
$$EST_{4,3} = Max(9, \max(9 + 0)) = 9$$
$$EFT_{4,3} = 9 + 10 = 19$$

شرط إنشاء نسخة من مهمة الدخول على المعالج P_1 :

$11 < 9 + 14 = 23$ محقق

شرط إنشاء نسخة من مهمة الدخول على المعالج P_2 :

$13 < 9 + 14 = 23$ محقق

نجد أنه في حال إنشاء نسخة من مهمة الدخول على كل من المعالجين P_1 و P_2 :

$$EST_{4,1} = Max(11, \max(11 + 0)) = 11$$
$$EFT_{4,1} = 11 + 12 = 23$$
$$EST_{4,2} = Max(13, \max(13 + 0)) = 13$$
$$EFT_{4,2} = 13 + 16 = 23$$

نُلاحظ أن زمن الانتهاء الأبكر للمهة T_4 على المعالجات أصغر منه في حال عدم إنشاء النسخ بالتالي نُجدول المهمة T_4 على المعالج P_3 .

المهمة ذات الأولوية الأدنى في المستوى التالي هي T_5 وهي VIT بالتالي نقوم بالحسابات اللازمة وفق الآتي:

$$EST_{5,1} = Max(0, \max(9 + 11)) = 20$$
$$EFT_{5,1} = 20 + 15 = 35$$

يتم حساب slot وفق الآتي:

زمن جاهزية المعالج: t_{ready} ; $slot = EST - t_{ready}$

بالتالي: slot =20

$$EST_{5,2} = Max(0, \max(9 + 11)) = 20$$
$$EFT_{5,2} = 20 + 11 = 31$$

slot =20

$$EST_{5,3} = Max(9, \max(9 + 0)) = 9$$
$$EFT_{5,3} = 9 + 19 = 28$$

في حال إنشاء النسخة:

- باختيار المعالج P_1 :

نتحقق أولاً من الشرط: $W_{T_{1,1}} = 11 < 20$ محقق بالتالي نستطيع إنشاء نسخة

$$EST_{5,1} = Max(11, \max(11 + 0)) = 11$$
$$EFT_{5,1} = 11 + 15 = 26$$

- باختيار المعالج P_2 :

نتحقق أولاً من الشرط: $W_{T_{1,2}} = 13 < 20$ محقق بالتالي نستطيع إنشاء نسخة

$$EST_{5,2} = Max(13, \max(13 + 0)) = 13$$
$$EFT_{5,2} = 13 + 11 = 24$$

بالتالي سيتم جدولة المهمة T_5 على المعالج P_2 .

المهمة ذات الأولوية الأدنى هي T_2:

نتبع سياسة إنشاء نسخة من مهمة الدخول المتبعة في خوارزمية HSIP على جميع المعالجات الأخرى إذا تحقق $\mathrm{DRT} > W_{T_{entry},p_j}$

$$EST_{2,1} = Max(9, \max(9 + 11)) = 20$$
$$EFT_{2,1} = 20 + 10 = 30$$

slot =20

$$EST_{2,2} = Max(24, \max(11 + 0)) = 24$$
$$EFT_{2,2} = 24 + 15 = 39$$
$$EST_{2,3} = Max(19, \max(9 + 0)) = 19$$
$$EFT_{2,3} = 19 + 11 = 30$$

في حال إنشاء النسخة

- باختيار المعالج P_1 :

نتحقق أولاً من الشرط: $W_{T_{1,1}} = 11 < 20$ محقق بالتالي نستطيع إنشاء نسخة

$$EST_{2,1} = Max(11, \max(11 + 0)) = 11$$
$$EFT_{2,1} = 11 + 10 = 21$$

أما المعالجين الآخرين فقمنا بإنشاء النسخة سابقاً وحساب EFT.

بالتالي سيتم جدولة المهمة T_2 على المعالج P_1 .

المهمة ذات الأولوية الأدنى هي T_3:

$$EST_{3,1} = Max(21, \max(11 + 0)) = 21$$

$$EFT_{3,1} = 21 + 9 = 30$$
$$EST_{3,2} = Max(24, \max(13 + 0)) = 24$$
$$EFT_{3,2} = 24 + 12 = 36$$
$$EST_{3,3} = Max(19, \max(9 + 0)) = 19$$
$$EFT_{3,3} = 19 + 14 = 33$$
$$EST_{3,1} = Max(21, \max(11 + 0)) = 21$$
$$EFT_{3,1} = 21 + 9 = 30$$
$$EST_{3,2} = Max(24, \max(13 + 0)) = 24$$
$$EFT_{3,2} = 24 + 12 = 36$$
$$EST_{3,3} = Max(19, \max(9 + 0)) = 19$$
$$EFT_{3,3} = 19 + 14 = 33$$

بالتالي سيتم جدولة المهمة T_3 على المعالج P_1 .

المهمة ذات الأولوية الأدنى هي T_6:

$$EST_{6,1} = Max(30, \max(21 + 0{,}30 + 0{,}19 + 19)) = 38$$
$$EFT_{6,1} = 38 + 13 = 51$$
$$EST_{6,2} = Max(24, \max(21 + 13{,}30 + 10{,}19 + 19)) = 40$$
$$EFT_{6,2} = 40 + 9 = 49$$
$$EST_{6,3} = Max(19, \max(34{,}40{,}19 + 0)) = 40$$
$$EFT_{6,3} = 40 + 5 = 45$$

بالتالي سيتم جدولة المهمة T_6 على المعالج P_3 .

المهمة ذات الأولوية الأدنى هي T_7 وهي VIT:

$$EST_{7,1} = Max(30, \max(19 + 13{,}24 + 27)) = 51$$
$$EFT_{7,1} = 51 + 11 = 62$$

Slot=19

$$EST_{7,2} = Max(24, \max(32{,}24 + 0)) = 32$$
$$EFT_{7,2} = 32 + 15 = 47$$

Slot=8

$$EST_{7,3} = Max(45, \max(19 + 0{,}49)) = 49$$
$$EFT_{7,3} = 49 + 13 = 62$$

Slot=4

في حال إنشاء النسخة:

- باختيار المعالج P_1 :

نتحقق أولاً من الشرط: $W_{T_{5,1}} = 15 < 19$ محقق بالتالي نستطيع إنشاء نسخة

$$EST_{7,1} = Max(45, \max(32,45)) = 45$$
$$EFT_{7,1} = 45 + 11 = 56$$

المهمة T_5 مجدولة على المعالج P_2 بالتالي نختبر إمكانية إنشاء نسخة على المعالج P_3

- باختيار المعالج P_3 :

نتحقق أولاً من الشرط: $W_{T_{5,3}} = 19 > 4$ بالتالي لانستطيع إنشاء نسخة على هذ المعالج.

نلاحظ أن EFT على المعالج P_2 بدون إنشاء النسخة أصغر منه في حال إنشاء النسخة على المعالج P_1 بالتالي نُجدول المهمة T_7 على المعالج P_2.

المهمة ذات الأولوية الأدنى هي T_8 وهي VIT:

$$EST_{8,1} = Max(30, \max(47 + 13,45 + 21)) = 66$$
$$EFT_{8,1} = 66 + 11 = 77$$

Slot=36

$$EST_{8,2} = Max(47, \max(47,66)) = 66$$
$$EFT_{8,2} = 66 + 15 = 81$$

Slot=19

$$EST_{8,3} = Max(45, \max(60,45)) = 60$$
$$EFT_{8,3} = 60 + 10 = 70$$

Slot=15

MP: T_6

في حال إنشاء النسخة:

- باختيار المعالج P_1 :

نتحقق أولاً من الشرط: $W_{T_{6,1}} = 13 < 36$ محقق بالتالي نستطيع إنشاء نسخة

$$EST_{8,1} = Max(43, \max(43 + 0,47 + 13)) = 60$$

$$EFT_{8,1} = 60 + 11 = 71$$

- باختيار المعالج P_2 :

نتحقق أولاً من الشرط: $W_{T_{6,2}} = 9 < 19$ محقق بالتالي نستطيع إنشاء نسخة

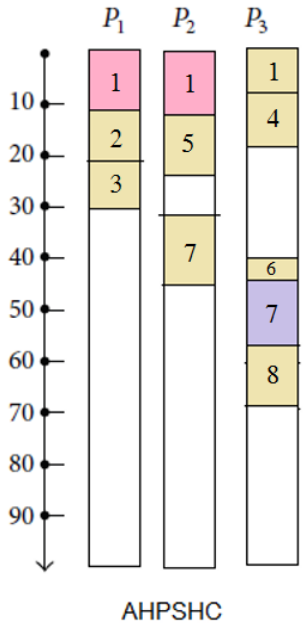
$$EST_{8,2} = Max(56, \max(56 + 0{,}47 + 0)) = 56$$
$$EFT_{8,2} = 56 + 15 = 71$$

نلاحظ أن EFT في حال النسخ أكبر منه بدون النسخة وبالتالي تقترح هذه الخوارزمية فقط بالنسبة لمهمة الخروج إن لم يحقق السلف ذو MP أصغر EFT لطول الجدولة مقارنةً به قبل النسخ فإننا ننشئ نسخة من السلف الآخر على المعالج الذي تمت جدولة المهمة ذات MP عليه وهو في مثالنا P_3 .

- باختيار المعالج P_3 :

نتحقق أولاً من الشرط: $W_{T_{7,3}} = 13 < 15$ محقق بالتالي نستطيع إنشاء نسخة

$$EST_{8,3} = Max(58, \max(45 + 0{,}58 + 0)) = 58$$
$$EFT_{8,3} = 58 + 10 = 68$$



الشكل (4) نتائج خوارزمية AHPSHC حسابياً

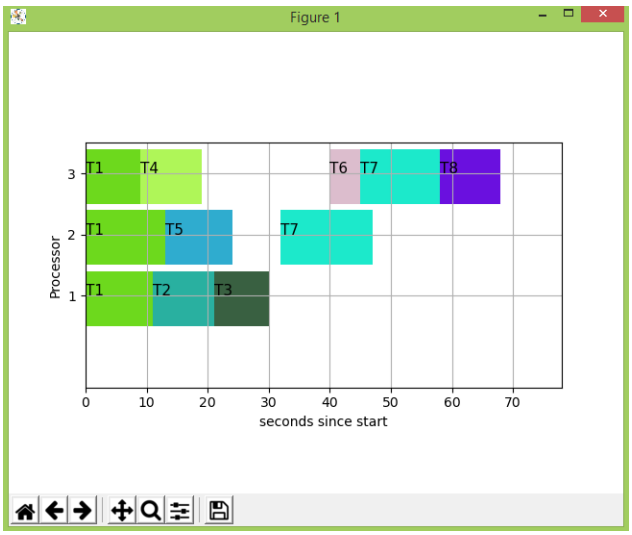
نُلاحظ أن Makespan=68.

النتائج باستخدام المحاكي:

```
:--Table of rank --tabrank
||  T1    ||  206.41766444215938
-------------------------------------------------------------------------
||  T5    ||  135.4547396617494
-------------------------------------------------------------------------
||  T4    ||  133.980582565453
-------------------------------------------------------------------------
||  T2    ||  113.98602272047596
-------------------------------------------------------------------------
||  T3    ||  108.9019301461593
-------------------------------------------------------------------------
||  T6    ||  75.06305992684452
-------------------------------------------------------------------------
||  T7    ||  59.46494480608584
-------------------------------------------------------------------------
||  T8    ||  25.92296279363144
-------------------------------------------------------------------------
Table of cp --tapc:--[]

<==  T1   ==>  T2   ==>  T6   ==>  T8   ==>  101.0
-------------------------------------------------------------------------
--
<==  T1   ==>  T3   ==>  T6   ==>  T8   ==>  103.0
-------------------------------------------------------------------------
--
<==  T1   ==>  T4   ==>  T6   ==>  T8   ==>  111.0
-------------------------------------------------------------------------
--
<==  T1   ==>  T4   ==>  T7   ==>  T8   ==>  99.0
-------------------------------------------------------------------------
--
<==  T1   ==>  T5   ==>  T7   ==>  T8   ==>  113.0
```

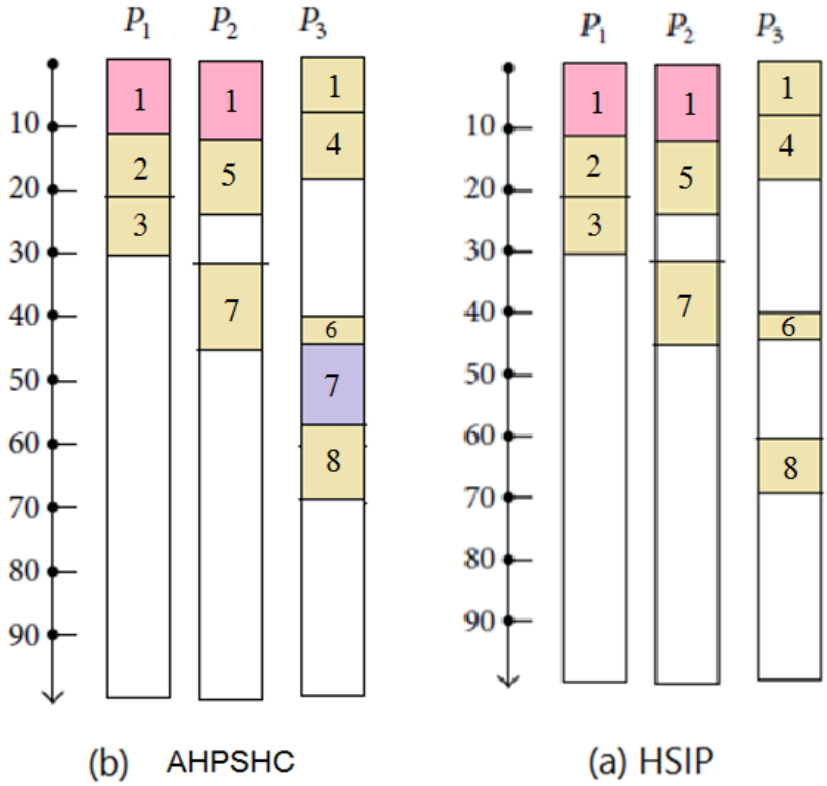
```
--------------------------------------------------------------------------
--
<==  T1    ==>  T5    ==>  T7    ==>  T8    ==>  113.0
--------------------------------------------------------------------------
--
Critical Path:
T1    ==>  T5    ==>  T7    ==>  T8    ==>  113.0
```



الشكل (5) نتائج خوارزمية AHPSHC باستخدام المحاكي

وهذا مطابق للنتائج الحسابية.

المقارنة مع خوارزمية HSIP:



	AHPSHC	HSIP
Makespan	68	**70**

الشكل (6) مقارنة بين خوارزمية HSIP وخوارزمية AHPSHC

نُلاحظ أنه تم تكرار مهمة الدخول حسب استراتيجية مضاعفة مهمة الإدخال المُوضحة سابقاً حيث تم تطبيقها بالحسابات الرياضية، وتم الشرح بشكل مفصل بالحسابات كيفية حساب تابع الرتبة للمهام السابقة، وكذلك شرح مفهوم الاعتمادية وعدم التجانس لهذا المثال على الشكل (3).

نلاحظ حسب الشكل (6) أن المهمة T_8 لديها سلفين T_6 و T_7 تعتمد عليهما في تنفيذها حيث تحتاج المهمة T_8 حسب الشكل (3) زمن اتصال مقداره 13 وحدة زمنية لتنقل لها المهمة T_7 البيانات فعلى المهمة T_8 أن تنتظر 13 وحدة من انتهاء تنفيذ المهمة T_7 لتبدأ بالتنفيذ، أما بالنسبة لزمن الاتصال بين المهمة T_6 و المهمة T_8 فهو يُهمل في هذه الحالة كونه تتم جدولة المهمة وسلفها على المعالج ذاته وهو في الشكل (6) المعالج 3.

إن زمن الإتمام الكلي Makespan=70 في خوارزمية HSIP وهو الزمن المُستغرق من مهمة الدخول لمهمة الخروج وباستخدام خوارزميتنا المقترحة أصبح Makespan=68.

12-الاستنتاجات والتوصيات:

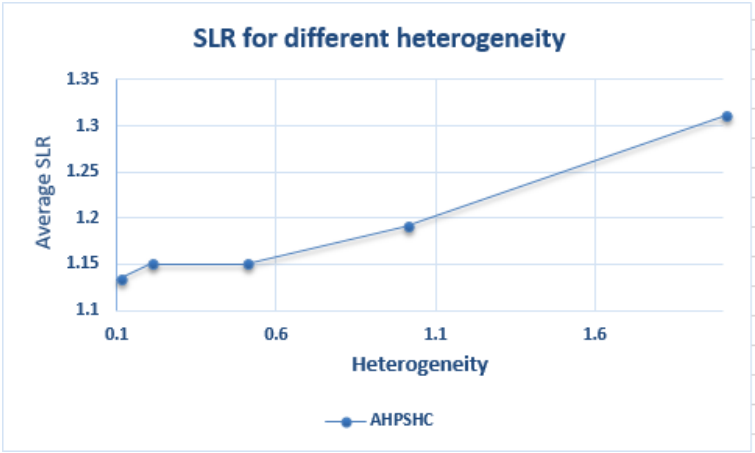
تم توليد عدة مخططات من أجل كل قيمة ل β حيث يوضح الشكل (7) مثالاً عن أحد المخططات المولدة، فكانت النتائج كما هو مُبين في الشكل (8) ،إذ يُوضح العلاقة ما بين قيم

β =0.1, 0.2, 0.5, 1, 2 و طول الجدولة النسبي المقابل وبمقارنة هذا المنحني مع المنحنيات التي تم الحصول عليها في الدراسات السابقة نجد وفق الشكل (9) أن خوارزمية AHPSHC أبدت أداء أفضل في الأوساط مختلفة التجانس حيث أعطت طول جدولة نسبي أقل.

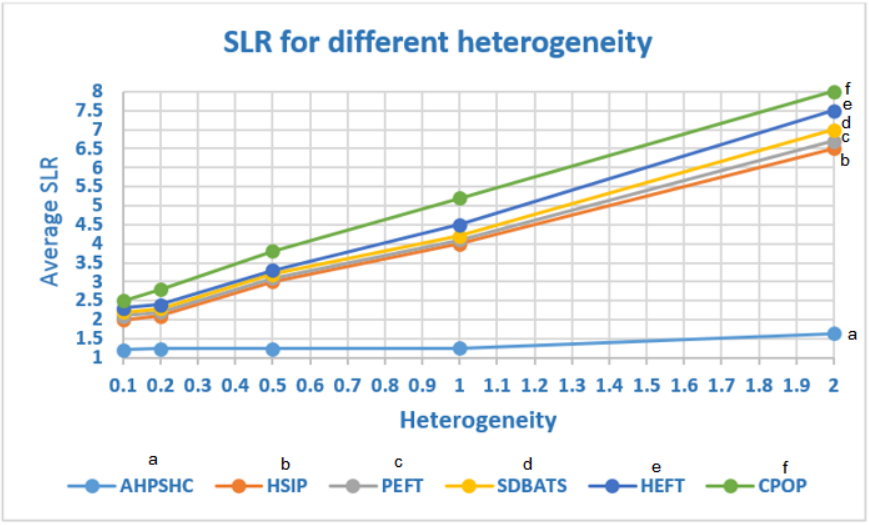
```
digraph G {
NODE 0 1 ROOT 0.0 0.0
  1 [size="1153751040", alpha="0.26"]
2,3 COMPUTATION 1153751040 0.26
  1 -> 4 [size ="75497472"]
  1 -> 5 [size ="75497472"]
  4 [size="218411008", alpha="0.03"]
6 COMPUTATION 218411008 0.03
  4 -> 8 [size ="8388608"]
  5 [size="2247688192", alpha="0.27"]
```

```
7 COMPUTATION 2247688192 0.27
  5 -> 8 [size ="134217728"]
  8 [size="212246528", alpha="0.14"]
9 COMPUTATION 212246528 0.14
  8 -> 10 [size ="8388608"]
  10 [size="537665536", alpha="0.19"]
11,12 COMPUTATION 537665536 0.19
  10 -> 13 [size ="33554432"]
  10 -> 14 [size ="33554432"]
  13 [size="4337934336", alpha="0.39"]
15 COMPUTATION 4337934336 0.39
  13 -> 18 [size ="75497472"]
  14 [size="4057049088", alpha="0.00"]
16,17 COMPUTATION 4057049088 0.00
  14 -> 18 [size ="75497472"]
  14 -> 19 [size ="75497472"]
  18 [size="327264256", alpha="0.41"]
20 COMPUTATION 327264256 0.41
  18 -> 22 [size ="8388608"]
  19 [size="3153723392", alpha="0.32"]
21 COMPUTATION 3153723392 0.32
  19 -> 22 [size ="134217728"]
  22 [size="4751972352", alpha="0.46"]
23 COMPUTATION 4751972352 0.46
NODE 22 23 COMPUTATION 4751972352 0.46
}
```

الشكل (7) مثال عن أحد المخططات المولدة باستخدام مولد المخططات العشوائي



الشكل (8) طول الجدول النسبي في خوارزمية AHPSHC في الأوساط مختلفة التجانس



الشكل (9) مقارنة بين طول الجدولة النسبي في الأوساط مختلفة التجانس مع الخوارزميات السابقة

نقترح أن تتم دراسة تأثير محدد أخر على هذه الخوارزمية ومعرفة السلوك الذي ستسلكه.

المراجع:

[1] Maheswaran. M, Ali. S, Siegel .H. J, Hensgen.D, Freund.R.F, 1999 -Dynamic mapping of a class of independent tasks onto heterogeneous computing systems , Journal of Parallel &Distributed Computing, vol. 59, no. 2, pp. 107–131.

[2] Kim. J-K, Shivle. S, Siegel et al. H. J, 2007- Dynamically mapping tasks with priorities and multiple deadlines in a heterogeneous environment, Journal of Parallel and Distributed Computing,vol. 67, no. 2, pp. 154–169.

[3] Sun.W, Zhang. Y, and Inoguchi. Y, 2007- Dynamic task flow scheduling for heterogeneous distributed computing: algorithm and strategy ,IEICE Transactions on Information and Systems,vol. E90-D, no. 4, pp. 736–744.

[4] Barbosa.J.G and Moreira.B, 2011 -Dynamic scheduling of a batch of parallel task jobs on heterogeneous clusters ,Parallel Computing, vol. 37, no. 8, pp. 428–438.

[5] Hou.E. S. H, Ansari.N, and Ren.H, 1994- A genetic algorithm for multiprocessor scheduling ,IEEE Transactions on Parallel and Distributed Systems, vol. 5, no. 2, pp. 113–120.

[6] Correa.R.C, Ferreira.A, and Rebreyend.P,1996- Integrating list heuristics into genetic algorithms for multiprocessor scheduling ,in Proceedings of the 8th IEEE Symposium on Parallel and Distributed Processing, pp. 462–469, New Orleans, La, USA.

[7] Dhodhi.M.K, Ahmad.I, Yatama.A, and Ahmad.I,2002- An integrated technique for taskmatching and scheduling onto distributed heterogeneous computing systems ,Journal of Parallel and Distributed Computing, vol. 62, no. 9, pp. 1338–1361.

[8] Topcuoglu.H, Hariri.S, and Wu.M.Y,2002- Performance-effective and low-complexity task scheduling for heterogeneous computing,IEEE Transactions on Parallel and Distributed Systems, vol.13, no. 3, pp. 260–274.

[9] Munir.E.U, Mohsin.S, Hussain.A, Nisar.M.W, and Ali.S,2013- SDBATS: a novel algorithm for task scheduling in heterogeneous computing systems ,in Proceedings of the IEEE 27th International Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW ’13), pp. 43–53, IEEE,Cambridge, Mass, USA.

[10] Arabnejad.H and Barbosa.J.G, 2014- List scheduling algorithm

for heterogeneous systems by an optimistic cost table ,IEEETransactions on Parallel and Distributed Systems, vol. 25, no. 3,pp. 682–694.
[11] Daoud.M.I and Kharma.N, 2008- A high performance algorithm for static task scheduling in heterogeneous distributed computing systems, Journal of Parallel and Distributed Computing, vol. 68, no. 4, pp. 399–409.
[12] Hagras.T and Janecek.J, 2003- A simple scheduling heuristic for heterogeneous computing environments ,in Proceedings of the Second International Conference on Parallel and Distributed Computing (ISPDC ’03), pp. 104–110, IEEE.
[13] Ilavarasan.E, Thambidurai.P, and Mahilmannan.R, 2005- High performance task scheduling algorithmfor heterogeneous computing system ,in Distributed and Parallel Computing, pp. 193–203, Springer, Berlin, Germany.
[14] Ilavarasan.E and Thambidurai.P, 2007- Low complexity performance effective task scheduling algorithm for heterogeneous computing environments, Journal of Computer Science, vol. 3,no. 2, pp. 94–103, 2007.
[15] Tang.X, Li.K, Liao.G, and Li.R, 2010- List scheduling with duplication for heterogeneous computing systems ,Journal of Parallel and Distributed Computing, vol. 70, no. 4, pp. 323–329.
[16] Bansal.S, Kumar.P, and Singh.K, 2003- An improved duplication strategy for scheduling precedence constrained graphs in multiprocessor systems ,IEEE Transactions on Parallel and Distributed Systems, vol. 14, no. 6, pp. 533–544.
[17] Kwok.Y.K and Ahmad.I, 1998- Benchmarking the task graph scheduling algorithms ,in Proceedings of the 1st Merged International and Symposium on Parallel and Distributed Processing, and IEEE Parallel Processing Symposium (IPPS/SPDP ’98), pp. 531–537.
[18] Arabnejad.H and Barbosa.J.G, 2014- List scheduling algorithm for heterogeneous systems by an optimistic cost table ,IEEE Transactions on Parallel and Distributed Systems, vol. 25, no. 3, pp. 682–694.
[19]https://github.com/frs69wq/daggen. . Accessed in: 14/11/2020.
[20]GuanWang,1,2 YuxinWang,3 Hui Liu,1 and He Guo1,2016- HSIP: A Novel Task Scheduling Algorithm for Heterogeneous Computing,

Hindawi Publishing Corporation,Scientific Programming, Article ID 3676149, 11 pages.
[21] Topcuoglu.H, Hariri.S and Wu.M, ”Task scheduling algorithms for heterogeneous processors”, 8th Proceedings Heterogeneous Computing Workshop (HCW), pp. 3-14, 1999.
[22] Soto.C, Santiago.A, Fraire.H, Dorronsoro.B, 2018- " Optimal Scheduling for Precedence-Constrained Applications on Heterogeneous Machines", MOL2NET, International Conference Series on Multidisciplinary Sciences, vol. 4, pp. 1–5.