Journal of Engineering Sciences and Information Technology
Volume (5), Issue (3) : 30 Sep 2021
P: 1 - 16

ISSN: 2522-3321

مجلة العلوم الهندسية وتكنولوجيا المعلومات
المجلد (5)، العدد (3) : 30 سبتمبر 2021م
ص: 1 - 16

Improved Associated Task Scheduling in Different Heterogeneous systems

Bassim Ali Oumran

Rowayda AAdel Muhbani

Faculty of Mechanical and Electrical Engineering || Al Baath University || Syria

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 and Communication to Computation Ratio factor.

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

تحسين جدولة المهام المترابطة في الأنظمة مختلفة التجانس

بسيم علي عمران

رويده عادل مهباني

كلية الهندسة الميكانيكية والكهربائية || جامعة البعث || سوريا

المستخلص: 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 الخوارزمية نتائج أفضل من سابقاتها من حيث طول الجدول النسبي نسبة إلى محدد عدم التجانس ومحدد نسبة كلفة الاتصالات للحسابات.

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

1- مُقدّمة.

في عصر البيانات الكبيرة لا يمكن أن تعتمد حوسبة البيانات الكثيفة على معالج واحد لتنجز. إنها غالباً ما تعتمد على أنظمة الحوسبة غير المتجانسة التي 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 طرق جدولة المهام. حيث تحسن طريقة الجدولة الفعالة بشكل خاص كفاءة النظام غير المتجانس. تهدف طرق الجدولة لإنقاص زمن الإتمام الكلي.

تحتاج خوارزميات الجدولة أن تسجل عمليات المعالجات وقرار انتهائها تحت معيار أسبقية المهام. تتضمن خوارزميات جدولة المهام الأساسية في الأنظمة غير المتجانسة 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)، والتنبؤ بأبكر زمن انتهاء Predict Earliest Finish Time (PEFT).

على الرغم أن تلك الخوارزميات استخدمت بشكل واسع في الأنظمة غير المتجانسة إلا أنها لاتزال تعاني من عدة مساوئ أبرزها. أولاً: يتجاهل معظمها عدم تجانس مصادر الحوسبة المختلفة والاتصالات المختلفة بين مصادر الحوسبة. ثانياً: ليس لديها سياسة فعالة للإدراج.

بهدف حل تلك المشاكل تم اقتراح خوارزمية جديدة وتمت مقارنتها مع الخوارزميات السابقة وأبدت هذه الخوارزمية نتائج أفضل.

2- مُشكلة البحث

اقتصرت بضع دراسات حديثة على جدولة المهام على وحدات معالجة بيانات غير متماثلة، حيث يُدخل عدم التجانس في النظم المتوازية درجة إضافية من التعقيد لمشكلة الجدولة إذ تُعتبر الجدولة ليست بالأمر السهل، خاصة أن النوى أو وحدات المعالجة غير متماثلة، كونه يوجد زمن تنفيذ مختلف لنفس المهمة على وحدات معالجة مختلفة نتيجة الاختلاف في السرعة والأداء والتكوين، كما يؤثر اختلاف نسبة كلفة الاتصالات للحسابات على الجدولة أيضاً، مما يزيد التعقيد في تقليل زمن الإتمام الكلي (Makespan).

3- الهدف من البحث

يهدف هذا البحث إلى تحسين جدولة المهام المترابطة على وحدات معالجة البيانات غير المتجانسة من حيث طول الجدول النسبي نسبة إلى محدد عدم التجانس ومحدد نسبة كلفة الاتصالات للحسابات.

4- أهمية البحث

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

يعتبر تنفيذ المهام في الزمن الحقيقي أمراً بالغ الأهمية.

5- مواد البحث وطرائقه

- تقديم دراسات مرجعية.
- دراسة الخوارزميات السابقة وتوضيح دراسة الباحث الحالية.
- استخدام برمجيات متخصصة لبناء محاكي للخوارزمية المُقترحة.
- الحصول على النتائج 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، حيث يتم اتخاذ القرار خلال زمن التنفيذ. بينما في الجدولة الستاتيكية فإن جميع تلك المعلومات معروفة بشكل مسبق، حيث يتم اتخاذ القرار خلال زمن التنفيذ.

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

من الخوارزميات الديناميكية Batch Mode Mapping Heuristics [14] Dynamic Mapping Heuristics [12] Dynamic Scheduling Cycle Strategy [17] dynamic scheduling method [2].

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

يتضمن الصنف الأول جدولة المهام متعددة المعالجات باستخدام الخوارزميات الجينية GA Task Scheduling Multiprocessor [8] نهج المعرفة الوراثية المعززة Knowledge-Augmented Genetic Approach [4] مشكلة فضاء الخوارزمية الجينية Problem-Space Genetic Algorithm (PSGA) [6].

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

تضم الخوارزميات المعتمدة على الحدس ثلاثة فئات رئيسية: جدولة القائمة، العنقدة، والازدواجية.

من خوارزميات الجدولة الستاتيكة المعتمدة على الحدس الأساسية:

Heterogeneous Earliest Finish Time [19] Critical Path On a Processor [19] Standard Deviation-Based Algorithm for Task Scheduling [15] Predict Earliest Finish Time [1], Longest Dynamic Critical Path (LDCP) [5] Heterogeneous Critical Parent Trees (HCPT) [7] High-Performance Task Scheduling (HPS) [10] low complexity Performance Effective Task Scheduling (PETS) [11] Heterogeneous Earliest Finish with Duplicator (HEFD) [18] and Selective Duplication Algorithm [3].

للجدولة بالعنقدة قيود كلما زاد عدم التجانس، بينما تسبب الجدولة التي تستند على تكرار المهمة تعقيد زمني كبير، كما أن تكرار المهمة يستهلك طاقة معالج أكبر، وهذا لا يسبب فقط استهلاكاً أكثر في الطاقة بل استهلاك الموارد المشتركة من قبل المعالجات التي تستخدم لتنفيذ مهام أخرى.

تضمن خوارزمية جدولة القائمة جدول زمني أكثر كفاءة نسبياً مع تعقيد زمني تربيعي. تُعتبر خوارزمية الجدولة HEFT أكثر شيوعاً واستخداماً إذ تنتج طول جدول زمني قصير [19].

7- مفاهيم

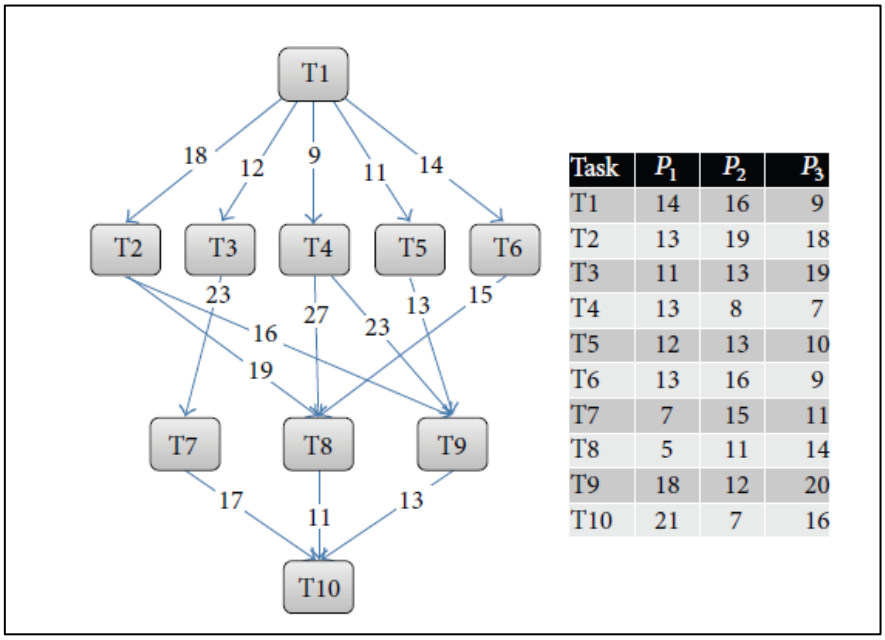
يتألف نموذج جدولة المهام من: التطبيق-بيئة حساب الهدف-معايير الأداء. يمكن وصف التطبيق بمخطط بياني لا دوري مباشر DAG,G=(V,E) Directed Acyclic Graph حيث أن:

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

يبين الشكل (1) مثالاً عن مخطط DAG حيث 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 أخلاف تسمى مهمة الخروج. يُلحق المخطط بمصفوفة $W = T_i \times P_j$.

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، عدد المعالجات n=3، 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)

يتم حساب الزمن المتوسط لإنجاز المهمة t_i وفق العلاقة التالية:

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

تمثل $c_{i,j}$ في الشكل(1) زمن الاتصال بين المهمة t_i والمهمة t_j. عندما تنفذ المهمتان على نفس المعالج فإن زمن الاتصال يصبح صفر 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:

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

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.

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

حيث إن $T_{Avl}(p_j)$ الزمن الأبكر عندما يكون المعالج p_j جاهز. بينما $pred(t_i)$ 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. زمن الاتصال $c_{m,i}$ صفر عندما تنفذ المهمة السلف على نفس المعالج p_j. لمهمة الدخول فإن:

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

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

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

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

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

- **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:

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

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

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 [13]:

طول الجدولة النسبي SLR [1]:

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

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

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 [9] حيث تم استدعاء المكتبة المتوفرة وتضمينها في برنامج Visual Studio باستخدام لغة C++ مع العلم أن الذي يحدد شكل المخطط المتولد المحددات التالية:

- **n:** عدد العقد في المخطط (هذا يعني عدد المهام).
- **محدد الشكل fat:** يؤثر هذا المحدد على ارتفاع وعرض المخطط. 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، حيث أن القيمة المنخفضة تؤدي إلى مسارات أقل والمرتفعة تُنتج العديد من المسارات.
- **محدد التناسق regularity:** يُحدد هذا المحدد تناسق عدد المهام في كل مستوى. تنتج القيمة المنخفضة مخطط بعدد مهام متباين في كل مستوى، بينما تؤدي القيمة المرتفعة لعدد مهام متماثل في كل مستوى.
- **درجة القفزjump:** يُشير هذا المحدد 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 ل 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 التالية:

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

- تم استخدام القيم التالية 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

تُعتبر الخوارزمية المقترحة عبارة عن نظام هجين من عدة خوارزميات سابقة مع إضافة بعض التحسينات حيث تم استخدام تابع الرتبة وكذلك مضاعفة مهمة الدخول والإدخال لفتحات الزمن المثالية من خوارزمية HSIP [21] وإدخال فكرة المسار الحرج من خوارزمية CPOP [20] والمضاعفة للمهام من خوارزمية HEFT [20] وPEFT [21] وSDBATS [21]. لذلك تم اصطلاح تسمية الخوارزمية المُقترحة AHPSHC اختصاراً ل SDBATS HEFT CPOP Advanced HSIP PEFT.

تعتمد خوارزمية AHPSHC على تابع الرتبة التصاعدية الذي ينطلق في حساب أولوية المهام من مهمة النهاية إلى مهمة البداية (علماً أن هذا التابع مُغاير لجميع التوابع المستخدمة سابقاً لحساب الرتبة من حيث اعتماده على جداء متوسط كلفة حساب المهام في الانحراف المعياري وأكبر قيمة لمجموع زمن الاتصال مع الأخلاف ورتبتها) وهو نفس التابع المستخدم في خوارزمية HSIP.

$$rank_u(t_i) = max_{t_{j\in succ(t_i)}}\{\overline{w}_i \times \sigma_i + occw(t_i) + rank_u(t_j)\} \quad (9)$$

$$rank_u(t_{exit}) = \overline{w_{exit}} \times \sigma_{exit} \quad (10)$$

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 نفس المعالج فإن كلفة الاتصال صفر 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 تسمية كل مهمة من المسار الحرج بالمهمة ذات الأهمية الشديدة VIT.

تم اصطلاح تسمية السلف الأكبر Maximum parent MP لمهمة ما T_k بين مجموعة أسلافها إذا كان المجموع: EFT (T_k , P_j) + c(T_k, T_i) أكبر ما يمكن بين الأسلاف.

تم اصطلاح تسمية 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 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 أياً منهما للجدولة.

استراتيجية مضاعفة مهمة الإدخال:

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 نفس الوقت، لذلك لا حاجة لأخذ مشكلة الحمل الزائد للمعالج بعين الاعتبار. 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 حمل زائد 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 التاليين:

- 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 التالية 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:

- 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 عدة خوارزميات جدولة. 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 ITS $\leq w_{i,j}$.
- 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 ل ITS.
- إذا هناك العديد من فتحات ITS المحققة للشرط في البند الثاني اختر ITS ذو أصغر EFT.

كود 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 scheduling 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

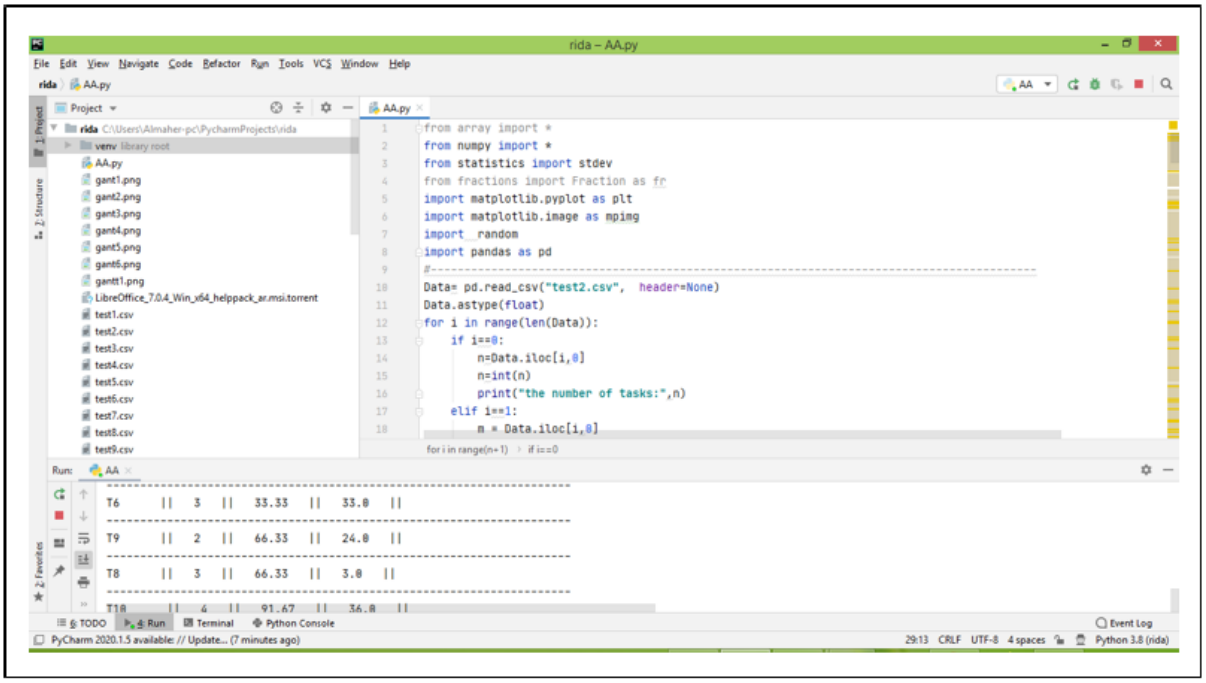
الدخل: المخطط والجدول المُرفق معه.

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 تحقُّق 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.

11- بناء محاكي لخوارزمية AHPSHC:

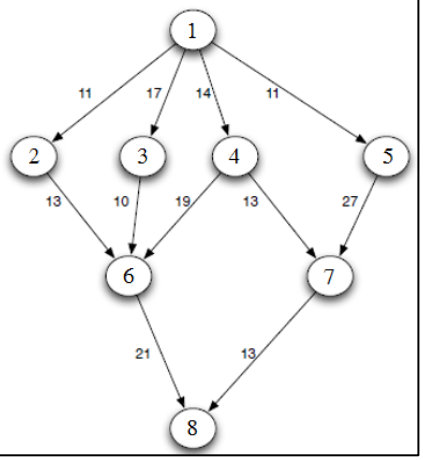
تم بناء محاكي الخوارزمية المقترحة باستخدام لغة البرمجة Python وبيئة العمل PyCharm Community Edition 2020.1 x64 كما هو مُوضح في الشكل (2).



الشكل (2) مُحاكي خوارزمية AHPSHC

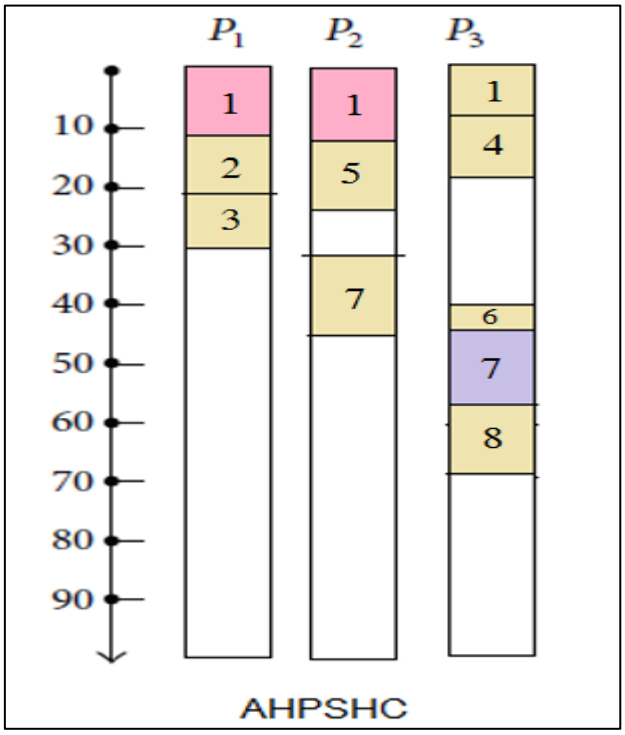
لنطبق الخوارزمية المقترحة على مخطط DAG في الشكل (3) [16]:

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



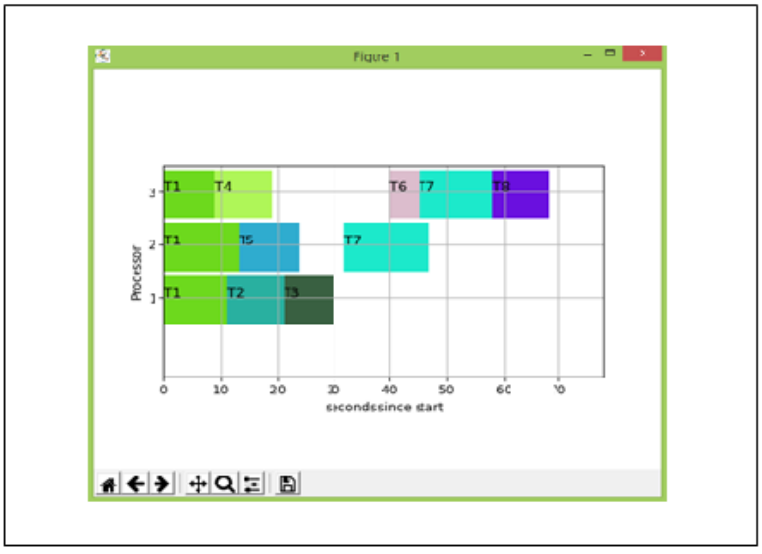
الشكل (3) مخطط بياني لا دوري مباشر (DAG)

النتائج حسابياً حسب الشكل (4):



الشكل (4) نتائج خوارزمية AHPSHC حسابياً

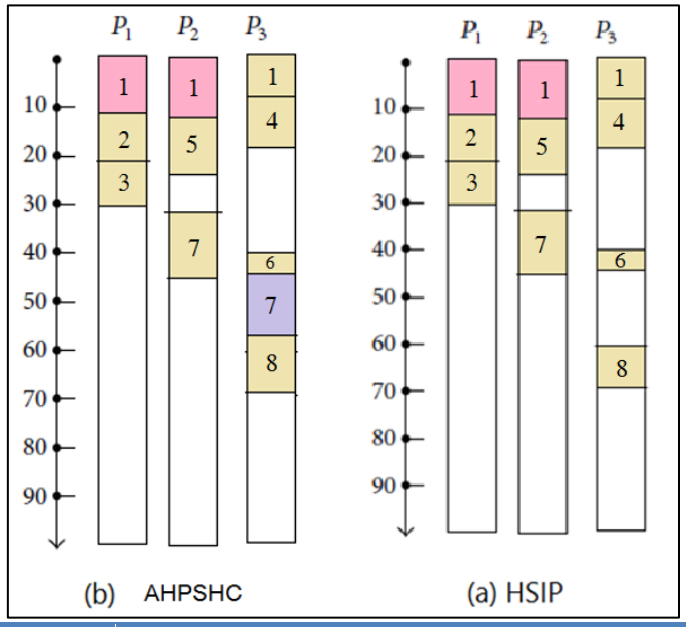
النتائج باستخدام المحاكي حسب الشكل (5):



الشكل (5) نتائج خوارزمية AHPSHC باستخدام المحاكي

وهذا مطابق للنتائج الحسابية.

المقارنة مع خوارزمية HSIP حسب الشكل (6):

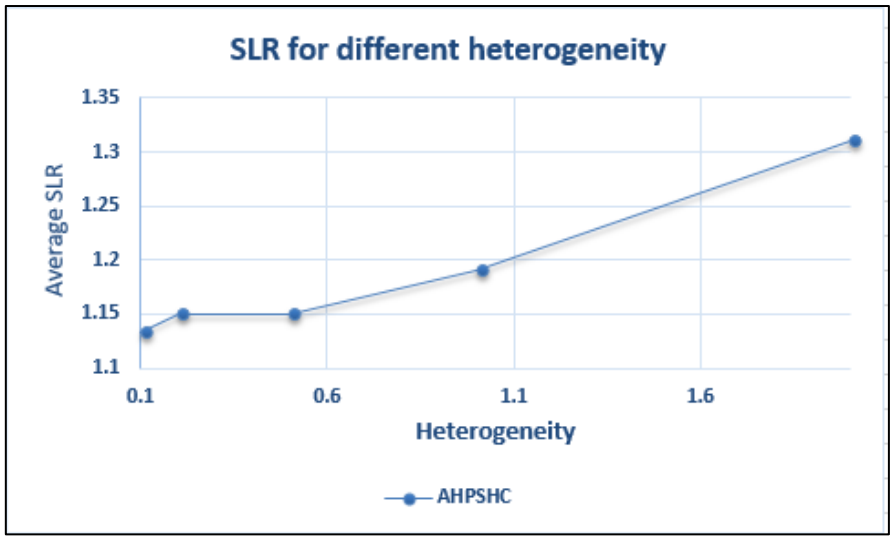


	AHPSHC	HSIP
Makespan	68	**70**

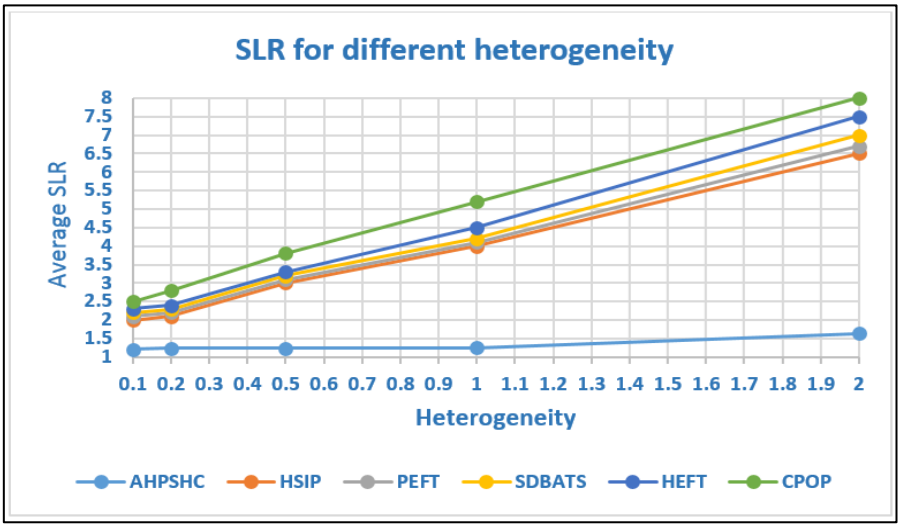
الشكل (6) مقارنة بين خوارزمية HSIP وخوارزمية AHPSHC

12-الاستنتاجات:

تم توليد عدة مخططات من أجل كل قيمة ل β فكانت النتائج كما هو مُوضح في الشكل (7) حيث نلاحظ أن خوارزمية AHPSHC أبدت أداء أفضل من سابقاتها كما هو مبين في الشكل (8) في الأوساط مختلفة التجانس حيث أعطت طول جدولة نسبي أقل.

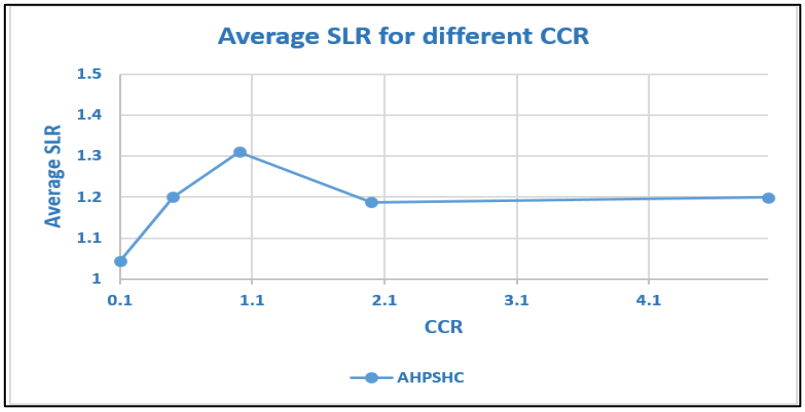


الشكل (7) طول الجدول النسبي في خوارزمية AHPSHC في الأوساط مختلفة التجانس

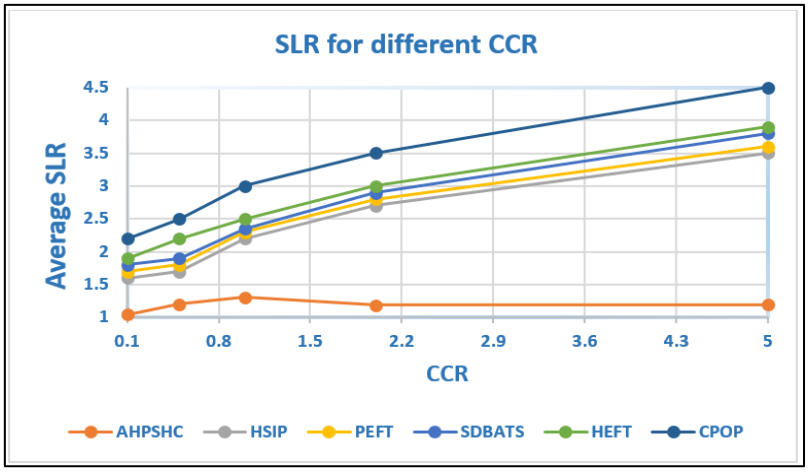


الشكل (8) مقارنة بين طول الجدولة النسبي في الأوساط مختلفة التجانس مع الخوارزميات السابقة

كما تم توليد عدة مخططات من أجل كل قيمة ل CCR فكانت النتائج حسب الشكل (9) حيث نلاحظ أن خوارزمية AHPSHC أبدت أداء أفضل من سابقاتها كما هو مبين في الشكل (10) حيث أعطت طول جدولة نسبي أقل.



الشكل (9) طول الجدول النسبي في خوارزمية AHPSHC نسبة إلى محدد نسبة كلفة الاتصالات للحسابات



الشكل (10) مقارنة بين طول الجدول النسبي في خوارزمية AHPSHC نسبة إلى محدد نسبة كلفة الاتصالات للحسابات مع الخوارزميات السابقة

13- الخلاصة:

يؤثر محدد عدم التجانس وكذلك محدد نسبة كلفة الاتصالات للحسابات في الأوساط مختلفة التجانس على جدولة المهام المترابطة بشكل كبير من حيث طول الجدولة النسبي، وقد أبدت خوارزمية AHPSHC المقترحة أداء أفضل من الخوارزميات السابقة في تحسين طول الجدولة النسبي نسبةً إلى المحددين السابقين.

14- التوصيات والمقترحات.

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

قائمة المراجع.

[1] Arabnejad, H., & Barbosa, J. G. (2013). List scheduling algorithm for heterogeneous systems by an optimistic cost table. IEEE Transactions on Parallel and Distributed Systems, 25(3), 682-694.

[2] Barbosa, J. G., & Moreira, B. (2011). Dynamic scheduling of a batch of parallel task jobs on heterogeneous clusters. Parallel computing, 37(8), 428-438.

[3] Bansal, S., Kumar, P., & Singh, K. (2003). An improved duplication strategy for scheduling precedence constrained graphs in multiprocessor systems. IEEE Transactions on Parallel and Distributed Systems, 14(6), 533-544.

[4] Correa, R. C., Ferreira, A., & Rebreyend, P. (1996, October). Integrating list heuristics into genetic algorithms for multiprocessor scheduling. In Proceedings of SPDP'96: 8th IEEE Symposium on Parallel and Distributed Processing (pp. 462-469). IEEE.

[5] Daoud, M. I., & Kharma, N. (2008). A high performance algorithm for static task scheduling in heterogeneous distributed computing systems. Journal of Parallel and distributed computing, 68(4), 399-409.

[6] Dhodhi, M. K., Ahmad, I., Yatama, A., & Ahmad, I. (2002). An integrated technique for task matching and scheduling onto distributed heterogeneous computing systems. Journal of parallel and distributed computing, 62(9), 1338-1361.

[7] Hagras, T., & Janecek, J. (2003, October). A simple scheduling heuristic for heterogeneous computing environments. In Parallel and Distributed Computing, International Symposium on (pp. 104-104). IEEE Computer Society.

[8] Hou, E. S., Ansari, N., & Ren, H. (1994). A genetic algorithm for multiprocessor scheduling. IEEE Transactions on Parallel and Distributed systems, 5(2), 113-120.

[9]https://github.com/frs69wq/daggen. Accessed in: 14/11/2020.

[10] Ilavarasan, E., Thambidurai, P., & Mahilmannan, R. (2005, October). High performance task scheduling algorithm for heterogeneous computing system. In International conference on algorithms and architectures for parallel processing (pp. 193-203). Springer, Berlin, Heidelberg.

[11] Ilavarasan, E., & Thambidurai, P. (2007). Low complexity performance effective task scheduling algorithm for heterogeneous computing environments. Journal of Computer sciences, 3(2), 94-103.

[12] Kim, J. K., Shivle, S., Siegel, H. J., Maciejewski, A. A., Braun, T. D., Schneider, M., ... & Yellampalli, S. S. (2007). Dynamically mapping tasks with priorities and multiple deadlines in a heterogeneous environment. Journal of parallel and distributed computing, 67(2), 154-169.

[13] Kwok, Y. K., & Ahmad, I. (1998, March). Benchmarking the task graph scheduling algorithms. In Proceedings of the First Merged International Parallel Processing Symposium and Symposium on Parallel and Distributed Processing (pp. 531-537). IEEE.

[14] 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 and distributed computing, 59(2), 107-131.

[15] Munir, E. U., Mohsin, S., Hussain, A., Nisar, M. W., & Ali, S. (2013, May). SDBATS: a novel algorithm for task scheduling in heterogeneous computing systems. In 2013 IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum (pp. 43-53). IEEE.

[16] Soto, C., Santiago, A., Fraire, H., & Dorronsoro, B. (2018, December). Optimal Scheduling for Precedence-Constrained Applications on Heterogeneous Machines. In Int. Conf. Ser. Multidiscipl. Sci.

[17] Sun, W., Zhang, Y., & Inoguchi, Y. (2007). Dynamic task flow scheduling for heterogeneous distributed computing: algorithm and strategy. IEICE transactions on information and systems, 90(4), 736-744.

[18] Tang, X., Li, K., Liao, G., & Li, R. (2010). List scheduling with duplication for heterogeneous computing systems. Journal of parallel and distributed computing, 70(4), 323-329.

[19] Topcuoglu, H., Hariri, S., & Wu, M. Y. (2002). Performance-effective and low-complexity task scheduling for heterogeneous computing. IEEE transactions on parallel and distributed systems, 13(3), 260-274.

[20] Topcuoglu, H., Hariri, S., & Wu, M. Y. (1999, April). Task scheduling algorithms for heterogeneous processors. In Proceedings. Eighth Heterogeneous Computing Workshop (HCW'99) (pp. 3-14). IEEE.

[21] Wang, G., Wang, Y., Liu, H., & Guo, H. (2016). HSIP: A novel task scheduling algorithm for heterogeneous computing. Scientific Programming, 2016.