Performance Evaluation of Heuristic Algorithms for Resource Allocation in Distributed Cloud Computing

Dr. Ali Ismail*
Reham Darweesh**

(Received 9 / 7 / 2024. Accepted 3 / 9 / 2024)

□ ABSTRACT □

Cloud computing is a term that refers to computer resources and systems that are available upon request over the network and that can provide a number of integrated computer services without being restricted by local resources, with the aim of making it easier for the user. It has been proven that the process of optimally allocating resources in the cloud computing environment is NP-complete, hence the need to use heuristic methods.

A number of resource allocation algorithms have been proposed and used to address this problem, but no one has been shown to be absolutely superior to the other. Therefore, five resource allocation algorithms were applied in order to evaluate their performance in terms of overall execution time for a set of tasks (Makespan), Throughput, Response Time, Memory Utilization, CPU Utilization, and BW Utilization.

The evaluated algorithms in this paper in terms of resource allocation performance in cloud computing are MaxMin, MinMin, Honey Bee algorithm, Ant Colony Optimization, and the Particle Swarm Optimization algorithm. The comparisons studied in this article indicate that the Honey Bee algorithm is the most suitable choice for allocating resources in cloud computing if the goal of the computing system is to achieve the highest Throughput with lowest Response Time.

Keywords: cloud computing, resource allocation, Honey Bee algorithm

* **Assistant Professor, Department of System and Computing Network Engineering, Faculty of Informatics Engineering, Tishreen University , Lattakia ,Syria.**
****Postgraduate Student (Master), Department of System and Computing Network Engineering, Faculty of Informatics Engineering, Tishreen University, Lattakia, Syria.**

تقييم أداء الخوارزميات الإرشادية لتخصيص الموارد في الحوسبة السحابية الموزعة

د. علي إسماعيل *

رهام درويش**

(تاريخ الإيداع 9 / 7 / 2024. قُبِل للنشر في 3/ 9 / 2024)

□ ملخّص □

الحوسبة السحابية هو مصطلح يشير إلى موارد وأنظمة الكمبيوتر المتاحة عند الطلب عبر الشبكة والتي يمكن أن توفر عددًا من خدمات الكمبيوتر المتكاملة دون تقييدها بالموارد المحلية، بهدف تسهيل الأمر على المستخدم. وقد ثبت أن عملية تخصيص الموارد بشكل مثالي في بيئة الحوسبة السحابية هي NP–complete، ومن هنا تأتي الحاجة إلى استخدام الأساليب الاستدلالية.

وقد تم اقتراح واستخدام عدد من خوارزميات تخصيص الموارد لمعالجة هذه المشكلة، ولكن لم يثبت أن أي منها متفوق بشكل مطلق على الآخر. لذلك، تم تطبيق خمس خوارزميات لتخصيص الموارد من أجل تقييم أدائها من حيث وقت التنفيذ الإجمالي لمجموعة من المهام (Makespan)، والإنتاجية، ووقت الاستجابة، واستخدام الذاكرة، واستخدام وحدة المعالجة المركزية، واستخدام عرض النطاق الترددي.

الخوارزميات التي تم تقييمها في هذه الورقة من حيث أداء تخصيص الموارد في الحوسبة السحابية هي MaxMin و MinMin و Honey Bee و Ant Colony Optimization و Particle Swarm Optimization. تشير المقارنات التي تمت دراستها في هذه المقالة إلى أن خوارزمية Honey Bee هي الخيار الأكثر ملاءمة لتخصيص الموارد في الحوسبة السحابية إذا كان هدف نظام الحوسبة تحقيق أعلى إنتاجية مع أدنى زمن استجابة.

الكلمات المفتاحية: الحوسبة السحابية، تخصيص الموارد، خوارزمية عسل النحل

* **مدرس – قسم النظم والشبكات الحاسوبية – كلية الهندسة المعلوماتية – جامعة تشرين – اللاذقية – سورية.**

** **طالبة دراسات عليا (ماجستير) – قسم النظم والشبكات الحاسوبية – كلية الهندسة المعلوماتية – جامعة تشرين – اللاذقية – سورية.**

مقدمة:

نتج عن تطور التكنولوجيا وكثرة الاختراعات، وجود العديد من التقنيات التي اختصرت على البشرية الجهد والوقت، ووفرت الكثير من التكاليف والنفقات المادية على الشركات وحتى الأفراد، لدرجة أنك تستطيع تلبية حاجات الأفراد من دون دفع تلك التكاليف الباهظة كما كان الحال سابقًا، ومن تلك التقنيات هي الحوسبة السحابية. تعرف الحوسبة السحابية بأنها نموذج يسمح بالوصول الملائم عند الطلب عبر الشبكة لمجموعة مشتركة من الموارد الحاسوبية (الشبكة، الخوادم، وسائط التخزين، التطبيقات والخدمات) والتي يمكن توفيرها حسب الحاجة وإطلاقها بأقل جهد إداري أو تفاعل مع مزود الخدمة [1].

تمتلك خوارزميات تخصيص الموارد تأثيراً مباشراً على سرعة تنفيذ المهام التي يطلبها المستخدمون وعلى كفاءة استخدام الموارد، إذ ثبت أن عملية تخصيص الموارد في بيئة الحوسبة السحابية هي من التعقيد NP−complete [2]. من الصعب مقارنة خوارزميات تخصيص الموارد الموجودة حالياً في بيئة الحوسبة السحابية وتحديد أياً منها تمثل الخوارزمية الأفضل وذلك بسبب اختلاف أداء كل منها باختلاف البيئات والسيناريوهات المفروضة، بالإضافة إلى تفاوت أداء هذه الخوارزميات وذلك تبعاً لطبيعة العامل الذي تتم دراسته.

تم في هذا البحث اختيار خمس خوارزميات أثبتت كفاءتها في نظام الحوسبة السحابية وهي MaxMin و MinMin و Honey Bee و PSO و ACO . هذه الخوارزميات تم تطبيقها في بيئة متجانسة وهي البيئة التي تكون فيها خصائص جميع الآلات الافتراضية التي يتم إنشاءها ثابتة ولا تتغير بتغير نوع المهمة التي يتم اسنادها إلى هذه الآلة وتم تحقيق هذه العملية باستخدام المحاكي Cloudsim. أظهرت النتائج تفوّق خوارزمية عسل النحل على الخوارزميات المدروسة الأخرى من حيث أغلب المعايير.

أهمية البحث وأهدافه:

إن سوق العمل في مجال الحوسبة السحابية كبير جداً وهو في ازدياد. ويعدّ هذا القطاع في الجمهورية العربية السورية ضعيف جداً ولكن لا مفر منه لذلك لابد من التهيئة له بشكل جيد من خلال تجهيز مهندسين مختصين بالأنظمة السحابية. كما يشكل تخصيص الموارد بكفاءة عالية أحد التحديات المفتوحة في الحوسبة السحابية. وتعد الحجر الأساس ضمن السحابة، فهي ضمن طبقة البنية التحتية ولذا فإن أي تحسّن في هذه الطبقة سينعكس على باقي الطبقات.

منهجية البحث:

يتناول هذا البحث عملية تخصيص الموارد والتي تعني عملية اسناد الموارد للتطبيقات السحابية عبر الانترنت مع مراعاة البنية التحتية المتاحة واتفاقيات مستوى الخدمة والتكلفة واستهلاك الطاقة. استخدم هذا البحث المحاكي Cloudsim في عملية المحاكاة إذ أجريت عدّة سيناريوهات لتقييم أداء خوارزميات التخصيص المدروسة وتحليل النتائج التي تم التوصل إليها.

1- الدراسات المرجعيّة:

لقد تطرقت العديد من الأبحاث إلى خوارزميات تخصيص الموارد المختلفة وتم اقتراح العديد من الخوارزميات ليتم استخدامها في بيئة الحوسبة السحابية. إذ اقترحت [3] نموذجاً هجيناً يستخدم الخوارزمية الجينية وتقنية الغابة العشوائية. تقوم الخوارزمية الجينية بإنشاء مجموعة البيانات المطلوبة لتدريب خوارزمية الغابة العشوائية.

تشتمل مجموعة بيانات التدريب على تخصيص أو تعيين الأجهزة الافتراضية للأجهزة المادية. تظهر النتائج أن النموذج المقترح موفر للطاقة من خلال تقليل استهلاك الطاقة ووقت التنفيذ ومتوسط وقت البدء ووقت الانتهاء مقارنة بالطرائق ACO و PSO و GA.

عرّف [4] خوارزمية تخصيص وجدولة الموارد (QRAS) المعتمدة على جودة الخدمة. تستخدم هذه الخوارزمية تقنية تحسين مستعمرة النمل (ACO). تقدم الخوارزمية المقترحة جدولة فعالة لأعباء العمل من خلال استخدام الحلول المثلى الناتجة عن محاكاة سلوك البحث عن الطعام للنمل. تمت مقارنة أداء الخوارزمية المقترحة مع خوارزميات MINMIN و MCT و ACO من ناحية تكلفة التنفيذ واستخدام الموارد. نستخلص من هذه الدراسة استخدام العديد من الدراسات خوارزمية ACO من أجل تخصيص الموارد في بيئة الحوسبة السحابية وتحقيقها نتائج مرضية.

اقترح [5] خوارزمية هجينة تعمل على الاستفادة من الخوارزميات الجينية والشبكات العصبية لتحسين الجدولة. تصنف هذه الطريقة المهام باستخدام تصنيف مهام الشبكة العصبية (N2TC) وترسل المهام المختارة إلى مهمة الخوارزمية الجينية (GATA) لتخصيص الموارد. تظهر النتائج تفوّق النهج المقترح على الأساليب FIFO و SJF في وقت التنفيذ، والتكاليف، ووقت الاستجابة. نستخلص من هذه الدراسة أهمية دمج خوارزميات التحسين الإرشادية مع الشبكات العصبية أو غيرها من تقنيات الذكاء الصنعي للتغلّب على نقاط الضعف في الخوارزمية.

تطرّق [6] إلى تقنية (PRA–ABC) لتخصيص الموارد والجدولة على أساس الأولوية باستخدام تحسين مستعمرة النحل الاصطناعية ((ABC) في البيئة السحابية. في البداية، يتم التنبؤ بعبء عمل الخادم ومتطلبات الموارد للمستخدمين بوساطة النحل من خلال مراقبة استخدامات الموارد السابقة وحجم الجهاز الافتراضي المخصص. وباستخدام عبء العمل المتوقع هذا، يتم تقدير وقت الانتهاء المتوقع لكل خادم. ثم يتم تصنيف المهام التي تطلب الموارد بناءً على الموعد النهائي ومتطلبات الموارد. وتصنيف الخوادم بناءً على عبء العمل ووقت الانتهاء المتوقع. تخصّص كل فئة من المهام لفئة الخوادم المعنية. تم تنفيذ النهج المقترح في بيئة Cloudsim ومقارنته بتقنية Suprex من حيث استخدام الموارد، والنسبة المئوية للموارد المخصصة بنجاح، والنسبة المئوية للمواعيد النهائية الفائتة، ومتوسط عبء العمل على الخادم وما إلى ذلك. نستخلص من هذه الدراسة استخدام العديد من الدراسات خوارزمية ABC من أجل تخصيص الموارد في بيئة الحوسبة السحابية وتحقيقها نتائج مرضية.

يوضح [7] أن خوارزمية PSO تعطي تكلفة أقل وتنفيذ سريع مقارنة بخوارزمية GA. ولكن توفر خوارزمية GA استهلاكًا أقل بقليل للطاقة مقارنة بخوارزمية PSO. نستخلص من هذه الدراسة استخدام العديد من الدراسات خوارزمية PSO من أجل تخصيص الموارد في بيئة الحوسبة السحابية وتحقيقها نتائج مرضية.

اقترح [8] خوارزمية تحسين عسل النحل النمل المعدلة (MAHO) لتحقيق موازنة حمل فعالة للمهام على الأجهزة الافتراضية عن طريق تهجين بين خوارزمية تحسين مستعمرة النمل (ACO) وخوارزمية موازنة حمل نحل العسل (HBL) في البيئة السحابية. تمت مقارنة الخوارزمية المقترحة مع الخوارزميات RR و SJF و HBL و ACO باستخدام المحاكي Cloudsim واستنتج أن MAHO يوفر كفاءة أفضل عن طريق تقليل ال Makespan. نستخلص من هذه الدراسة أهمية دمج خوارزميات التحسين الموجودة حاليا للحصول على خوارزمية هجينة تستفيد من نقاط القوة لهذه الخوارزميات.

يقدم [9] خوارزمية Dragonfly–PSO الهجينة إذ يستفيد النموذج المقترح من نقاط القوة في كلا الخوارزميتين PSO و Dragonfly. تمت مقارنة الخوارزمية المقترحة مع الخوارزميتين PSO و Dragonfly باستخدام المحاكي Cloudsim. يوفر النهج الهجين المقترح وقت استجابة أفضل.

2- المحاكي Cloud Simulator (Cloudsim)[10] :

ال Cloudsim عبارة عن مجموعة من صفوف Java التي توفر محاكاة لمفاهيم الحوسبة السحابية ويتضمن إطار عمل Cloudsim ما يأتي [10]:

- المناطق (Regions): يتم تقسيم المناطق جغرافيًا لتوفير الموارد في مواقع مختلفة.
- مركز البيانات(Datacenter): مجموعة من المضيفين (hosts) أو الخوادم التي توفر خدمة البنية التحتية.
- المضيفون(Hosts): هو الكيان المادي الذي يعد موردًا للمهام.
- Cloudlet: عبارة عن مجموعة من الطلبات المقدمة من المستخدم.
- وسيط الخدمة (Service Broker): يقرر أي جهاز افتراضي سيقدم الخدمة المطلوبة.
- تخصيص VM(VM allocation): تمثل هذه السياسات نموذجًا لتخصيص الموارد للمهام.
- جدولة VM (VM scheduler): تحاكي سياسات الجدولة الخاصة بـ Cloudsim تخصيص الموارد ل cloudlets.

أما بالنسبة للمنهجية المستخدمة لإجراء المحاكاة في ال Cloudsimهي بالشكل الاتي:

1. تهيئة الأجهزة الافتراضية وال cloudlet:

تتم إضافة كل جهاز افتراضي بخصائص مثل معرف الجهاز الافتراضي وعدد المعالجات والذاكرة. وبالمثل، يتم إنشاء cloudlets بالمعرف والطول وحجم الملف.

2. تعيين ال cloudlets إلى الأجهزة الافتراضية:

يتم تعيين ال cloudlets للأجهزة الافتراضية وفقًا لمتطلباتها، مثل متطلبات التخزين وعرض النطاق الترددي وما إلى ذلك باستخدام تقنيات حوسبة مختلفة مثل الخوارزمية الجينية والخوارزمية المستوحاة من نحل العسل.

2. المحاكاة: البدء في محاكاة تنفيذ ال cloudlets

3. حساب بارامترات الأداء: مثل Makespan ووقت الاستجابة ووقت الانتظار والحمل والطاقة وفقًا لدوال الهدف.

4. التحليل: تحليل النتيجة من خلال مقارنة البارامترات.

3- خوارزميات تخصيص الموارد:

تخصيص الموارد هو عملية تعيين الموارد المتاحة للتطبيقات السحابية عبر الانترنت مع مراعاة البنية التحتية المتاحة واتفاقيات مستوى الخدمة والتكلفة وعوامل الطاقة. أي أن مزود الخدمة السحابية يدير الموارد وفقاً لطريقة التسعير عند الطلب مع ضمان جودة الخدمة ورضى المستخدم [11].

- تحسين سرب الجسيمات (PSO):

تحسين سرب الجسيمات هو أسلوب تحسين قائم على ذكاء السرب مستوحى من السلوك الجماعي لأسراب الطيور وأسراب الأسماك. يتم استخدام PSO بشكل شائع لمهام مثل وضع VM، وموازنة التحميل، وتحسين الطاقة في الحوسبة السحابية [12]. تحاكي خوارزميات PSO حركة الجزيئات في مساحة حل متعددة الأبعاد، وتقوم بتعديل مواقعها بشكل متكرر بناءً على أفضل الحلول المحلية والعالمية للبحث عن الحلول المثلى.

يبدأ PSO بتكوين عدد من الجسيمات لتكوين سرب يسافر في فضاء المشكلة بحثًا عن الحل الأمثل. لكل جسيم في هذه الخوارزمية موقعًا في الفضاء ذي البعد D، حيث يمكن لكل عنصر أن يأخذ القيمة الثنائية 0 أو 1. بالإضافة إلى ذلك، كل جسيم له سرعة ذات بعد D، حيث يكون كل عنصر في الفضاء [0، 1]. يتم حساب السرعات على أنها

احتمالات تتغير خلال الزمن الذي تتحرك فيه الجسيمات في الفضاء، حيث تتغير كل قيمة سرعة من حالة إلى أخرى. تتلخّص خطوات الخوارزمية على الشكل التالي [13]:

```
1: procedure BINARY PSO(nodesList)
2:     CalculateExecTimes();
3:     initSwarm();
4:     initGlobalBest();
5:     for i ← 0 → numberIterations do
6:         for j ← 0 → numberParticles do
7:             calculateInertiaValue();
8:             calculateNewVelocities();
9:             calculateNewPositions();
10:            calculateFitnessValue();
11:            evaluateSolution();
12:            updateParticleMemory();
13:            updateGlobalBest();
14:        end for
15:    end for
16: end procedure
```

في كل تكرار، يتم تحديث السرعة، vi، والموقع، pi، لكل جسيم بشكل عشوائي من خلال الجمع بين الحل الحالي للجسيم، وأفضل حل محلي للجسيم، وأفضل حل شامل من بين جميع الجسيمات. العلاقات الرياضية المستخدمة هي (1) و(2) حيث ω هو ثابت القصور الذاتي، و c1 و c2 يمثلان الثوابت المعرفية والاجتماعية التي عادة ما تكون حوالي 2.00، و r1 و r2 أرقام عشوائية. من أجل تحديث سرعات ومواضع كل جسيم، يتم استخدام (3) و(4) لإضافة اللاخطية حيث pi هو المكون i للجسيم p و r هو رقم عشوائي. تشير العلاقات (1) –(4) إلى أن سرعة الجيران والسرعة الحالية للجسيم نفسه تساهم في تحديد الموقع التالي للجسيم.

$$v_i = \omega v_i + c_1 r_1 \cdot (\hat{p}_i - p_i) + c_2 r_2 \cdot (\hat{g} - p_i) \quad (1)$$

$$p_i = p_i + v_i \quad (2)$$

$$s(p_i) = \frac{1}{1 + \exp(-p_i)} \quad (3)$$

$$p_i = \begin{cases} 0, & s(p_i) \le r \\ 1, & \text{Otherwise} \end{cases} \quad (4)$$

- تحسين مستعمرة النمل (ACO):

تحسين مستعمرة النمل هو أسلوب تحسين مستوحى من الطبيعة يعتمد على سلوك البحث عن الطعام لمستعمرات النمل. في الحوسبة السحابية، يتم استخدام خوارزميات ACO لمهام مثل جدولة المهام والتوجيه وتحسين الشبكة. تحاكي خوارزميات ACO سلوك البحث التعاوني للنمل، وتقوم بشكل متكرر ببناء مسارات الحلول بناءً على مسارات الفورمون والمعلومات الإرشادية للعثور على حلول مثالية أو شبه مثالية [12]. تلخّص الخطوات التالية [14] خوارزمية ACO:

Algorithm 1: ACO algorithm
Input: List of Cloudlet (Tasks) and List of VMs Output:
The best solution for tasks allocation on VMs Steps: 1. Initialize:
Set Current_iteration_t = 1.
Set Current_optimal_solution=null.
Set Initial value τij (t) =c for each path between tasks and VMs.
2. Place m ants on the starting VMs randomly.
3. for k: = 1 to m do
Place the starting VM of the k-th ant in tabuk.
Do ants_trip while not all ants end their trips Every ant chooses the VM for the next task according to Equation 1.

$$p_{ij}^{k}(t) = \begin{cases} \dfrac{[\tau_{ij}(t)]^{\alpha} * [\eta_{ij}]^{\beta}}{\sum_{s \in allowed_k} [\tau_{is}(t)]^{\alpha} * [\eta_{is}]^{\beta}} & if\ j \in allowed_k \\ 0 & otherwise \end{cases} \quad (1)$$

Insert the selected VM to tabuk.
End Do
4. for k: =1 to m do
Compute the length Lk of the tour described by the k-th ant according to Equation 4.

$$L^{k}(t) = \arg\max_{j \in J} \{sum_{i \in IJ}(d_{ij})\} \quad (4)$$

Update the current_optimal_solution with the best-founded solution 5.
For every edge (i, j), apply the local pheromone according to Equation 5.

$$\tau_{ij}(t) = (1-\rho)\tau_{ij}(t) + \Delta\tau_{ij}(t) \quad (5)$$

6. Apply global pheromone update according to Equation 7.

$$\tau_{ij}(t) = \tau_{ij}(t) + \frac{Q}{L^{+}} \quad if\ (i,j) \in T^{+} \quad (7)$$

7. Increment Current_iteration_t by one.
8. If (Current_iteration_t < tmax)
Empty all tabu lists.
Goto step 2
Else
Print current_optimal_solution.
End If
9. Return

ملاحظات عن الخوارزمية السابقة:

- **τij(t)** هي كمية الفيرمون الافتراضية للحافة التي تربط المهمة i بالجهاز الافتراضي j . يُفترض أن تكون الكمية الأولية **τ0** من الفيرومون على الحواف ثابتة موجبة صغيرة (توزيع متجانس للفيرومون في الوقت t=0).

- **Allowed k** هي الآلات الافتراضية المسموح بها لـ ant k في الخطوة التالية
- **ηij=1/dij** يمكن حساب **dij** الذي يعبر عن وقت التنفيذ المتوقع ووقت نقل المهمة i على VMj باستخدام المعادلة 2.

$$d_{ij} = \frac{TL\_Task_i}{Pe\_num_j * Pe\_mips_j} + \frac{InputFileSize}{VM\_bw_j} \qquad (2)$$

TL_Taski الطول الإجمالي للمهمة التي تم إرسالها إلى VMj

Pe_numj عدد معالجات VMj

Pe_mipsj هو MIPS لكل معالج من معالجات VMj

InputFileSize طول المهمة قبل التنفيذ

VM_bwj هي قدرة عرض النطاق الترددي للاتصالات الخاصة بـ VMj

- ***α* و*β*** تستخدم للتحكم في الوزن النسبي لمسار الفيرومون ومعلومات الارشاد على التوالي
- بعد اكتمال الجولة، تضع كل نملة k كمية من الفيرومون يتم حسابها بواسطة المعادلة 3 على كل حافة (i, j) استخدمتها.

$$\Delta\tau_{ij}^{k}(t) = \begin{cases} \dfrac{Q}{L^{k}(t)} & if\ (i,j) \in T^{k}(t) \\ 0 & if\ (i,j) \notin T^{k}(t) \end{cases} \qquad (3)$$

***Tk*(*t*)** هي الجولة التي قامت بها النملة k عند التكرار t

***Lk*(*t*)** طولها (المدة المتوقعة لهذه الجولة)

- ***Ij*** مجموعة المهام المعينة لـ VMj
- ρ معدل تبخر المسار
- يتم حسابها باستخدام المعادلة 6

$$\Delta\tau_{ij}(t) = \sum_{k=1}^{m} \Delta_{\tau_{ij}}^{k}(t) \qquad (6)$$

- **T+** أفضل جولة تم ايجادها منذ بداية التجربة
- **L+** طول أفضل جولة
- تحسين نحل العسل (HBO):

تعد خوارزمية نحل العسل الاصطناعية أيضًا خوارزمية مستوحاة من الطبيعة بشكل جيد للغاية والتي يمكن استخدامها لجدولة الموارد في البيئة السحابية بكفاءة. تتمثل الميزات الرئيسية لخوارزمية نحل العسل الاصطناعية في أنها تقوم بشكل أساسي بفحص جميع الخوادم الموجودة وتقدم دائمًا الطلب القادم إلى جهاز افتراضي معين بحيث لا يتم تحميله بشكل زائد فهي تقوم بشكل عام بحساب مدى ملاءمة كل خادم وتحاول تخصيص أفضل خادم ممكن للعملية. في هذه الخوارزمية هناك ثلاثة أنواع من النحل: النشط، غير النشط، والكشاف. يتم تحديد أعداد كل نوع من هذه

الأنواع من النحل في المرحلة الأولية للخوارزمية. في البداية، تقوم الخوارزمية بإنشاء مجموعة من النحل، لكل منها حل عشوائي. تتبع الخوارزمية أفضل حل إجمالي تم العثور عليه من قبل أي نحلة وأفضل حل هنا يرتبط بطول Makespan. تتم فهرسة التكرارات بواسطة t ، حيث tmax هو الحد الأقصى لعدد التكرارات المسموح بها. في كل تكرار، تتم معالجة كل نوع من النحل بالطريقة المناسبة. يتم استخدام tmax للتحكم في عدد مرات تكرار روتين الحل؛ يبين الشكل التالي خطوات الخوارزمية [15]:

```
Input: List of Cloudlet (Tasks) and List of VMs
Output: the best solution for taks alloction on VMs
Steps:
1. Initialize:
    Set value of parameters Number_of_Bees, Number_of_Inactive, Number_of_active, Number_of_Scout,
      Max_number_of_Vists, Prob_Mistake, Prob_Presuasion, tmax.
    Set Current_iteration_t=1.
    Set Global_Best_Solution=null.
2. Generate Random Solution for each bee
3. Update Current_Best_Solution
4. For k :=1 to Number_of_Bees
     IF k is an ActiveBee
        Perform ProcessActiveBee()
     ElseIF k is an ScoutBee
        Perform ProcessScoutBee()
     Else
       Perform ProcessInactiveBee ()
     End IF
5. Increment Current_iteration_t by one.
6. If (Current_iteration_t < t_max)
    Goto step 4
  Else
    Print Global_Best_solution.
  End If
Stop
```

- خوارزمية MaxMin :

تعتمد هذه الخوارزمية على تحديد المهمة ذات أقصى وقت للتنفيذ وتخصيصها للمورد الذي يستغرق أقل وقت لتنفيذها. تُعطى خوارزمية Max–Min الأولوية للمهام الأكبر[16].

- خوارزمية MinMin:

يتم حساب الحد الأدنى لوقت الاكتمال لكل مهمة يتم جدولتها ويتم تحديد المهمة ذات الحد الأدنى لوقت الاكتمال وتعيينها للموارد التي يمكنها تنفيذ المهمة بأقل Makespan [16].

4- القسم العملي:

تم في مجموعات التجارب التي تمت دراستها تقييم أداء خمس خوارزميات أثبتت كفاءتها في نظام الحوسبة السحابية وهذه الخوارزميات هي MaxMin و MinMin و Honey Bee و PSO و ACO؛ من حيث عدة معايير هي زمن التنفيذ الكلي لمجموعة من المهام Makespan وزمن الاستجابة Response Time والإنتاجية Throughput واستخدام الذاكرة Memory Utilization واستخدام وحدة المعالجة المركزية CPU Utilization واستخدام عرض النطاق الترددي BW Utilization. وذلك بهدف الوصول إلى معرفة الخوارزمية الأفضل الواجب تطبيقها في نظام الحوسبة السحابية حيث تمت المقارنة ضمن الظروف نفسها لجميع الخوارزميات المدروسة مع تغيير عدد المهام الواردة لتقييم أداء كل خوارزمية عند الأحمال الخفيفة والمتوسطة والثقيلة.

تمت دراسة هذه الخوارزميات في المحاكي CloudSim3.0 بالاعتماد على قيم المتحولات المبينة في الجدول(1) لخوارزمية PSO [13] وفي الجدول (2) لخوارزمية ACO [14] وفي الجدول (3) لخوارزمية Honey Bee [15].

في البيئة المتجانسة تم توحيد الخصائص لكل الآلات الافتراضية التي تم إنشاؤها في النظام. تم دراسة كل من هذه الخوارزميات تحت عدد متغير من المهام الواردة وفقاً لكل معيار من المعايير السابقة. يوضح الجدول (4) السيناريوهات المدروسة.

الجدول (1): قيم المتحولات المستخدمة في دراسة خوارزمية PSO:

Parameter	Value
Number of particles	100
Number of iterations.	1000
Inertia min.	0.1
Inertia max.	0.9
Cognitive/local coefficient.	1.49445
Social/global coefficient.	1.49445
Contriction factor	5
Method used to update inertia.	Linearly Decreasing Inertai Weight method.
Method used to update particle position.	Sigmoid

الجدول (2) قيم المتحولات المستخدمة في دراسة خوارزمية ACO

Parameter	Value
Number of ants in the ant colony.	8
Number of iterations for the ant colony optimization.	5
Initial pheromone level on the trail.	1
Importance of pheromone in decision-making (alpha).	3
Importance of distance in decision-making (beta).	2
Total pheromone deposited by ants (Q).	8
Rate of pheromone evaporation (rho).	0.01

الجدول (3) قيم المتحولات المستخدمة في دراسة خوارزمية Honey Bee

Parameter	Value
Threshold	1

الجدول (4) السيناريوهات المدروسة

		number of tasks	task length (random)	pes	filesize
group 1	scenario 1	100	[10000,20000]	4	10
group 2	Scenario2	500	[10000,20000]	4	10
group 3	Scenario3	1000	[10000,20000]	4	10

5- معايير تقييم الأداء Performance Metrics:

تم تقييم أداء الخوارزميات المذكورة من خلال حساب عدة معايير[17] توضح أداء هذه الخوارزميات وكانت هذه المعايير هي زمن التنفيذ الكلي لمجموعة من المهام وزمن الاستجابة والإنتاجية واستخدام الذاكرة واستخدام وحدة المعالجة المركزية واستخدام عرض النطاق الترددي.

- الإنتاجية:

وهي عبارة عن إجمالي عدد المهام التي يتم تنفيذها بنجاح خلال فترة زمنية معينة في الحوسبة السحابية. يمكن أن يعطي هذا المقياس نظرة شاملة لأداء النظام، لكنه لا يضمن تقليل وقت الاستجابة والتكلفة. وتعطى الإنتاجية بالقانون:

Throughput = Sum of (Data Transferred / (Cloudlet End Time - Cloudlet Start Time) for each cloudlet

- زمن الاستجابة:

يشير وقت الاستجابة إلى الزمن الممتد منذ وصول المهمة عند قبول التحميل إلى إرجاع الإجابة المقابلة للمستخدم. وهو نسبة الفرق بين وقت انتهاء عبء العمل ووقت بدء تنفيذ عبء العمل إلى عدد أحمال العمل.

وتعطى قيمة زمن الاستجابة بالقانون التالي:

Response Time = Sum of (Finish Time - Submission Time) for each cloudlet / Number of cloudlets

- زمن التنفيذ الكلي:

وهو إجمالي الوقت المستغرق لمعالجة مجموعة من المهام لتنفيذها بالكامل. يمكن أن يتم تصغير Makespan عن طريق تعيين مجموعة المهام لمجموعة من الأجهزة الافتراضية، ولا يهم ترتيب تنفيذ المهام في الأجهزة الافتراضية.

يتم حسابه عن طريق حساب زمن الانتهاء الكلي لآخر مهمة يتم تنفيذها. وتعطى قيمة زمن التنفيذ الكلي بالقانون التالي:

Makespan = max (cloudlet.getFinishTime () for cloudlet in sr.recievedCloudlets)

- استخدام وحدة المعالجة المركزية:

يمثل استخدام الموارد كثافة الحمل على جهاز افتراضي، ويمكن الإشارة إليه بنسبة مئوية. وكتعريف آخر، فإن استخدام الموارد هو نسبة الوقت الفعلي الذي يقضيه المورد في تنفيذ عبء العمل. في مجال السحابة، الاستخدام هو إجمالي كمية الموارد المستهلكة فعليًا في مراكز البيانات. الهدف من استخدام الموارد بفعالية وهو زيادة إيرادات وأرباح مزود الموارد إلى الحد الأقصى مع الحفاظ على رضا المستخدمين.

ومنه فإن استخدام وحدة المعالجة هو المعدل بين الانشغال الكلي لوحدة المعالجة المركزية للألة الافتراضية بالنسبة لزمن التنفيذ الكلي. وتعطى قيمة استخدام وحدة المعالجة المركزية بالقانون التالي:

CPU Utilization = (Total CPU Usage / Number of Cloudlets) * 100

- استخدام الذاكرة:

وهو المعدل بين الانشغال الكلي لذاكرة الألة الافتراضية بالنسبة لزمن التنفيذ الكلي. وتعطى قيمة استخدام الذاكرة بالقانون التالي:

RAM Utilization = (Total RAM Usage / Number of Cloudlets) * 100

- استخدام عرض النطاق الترددي:

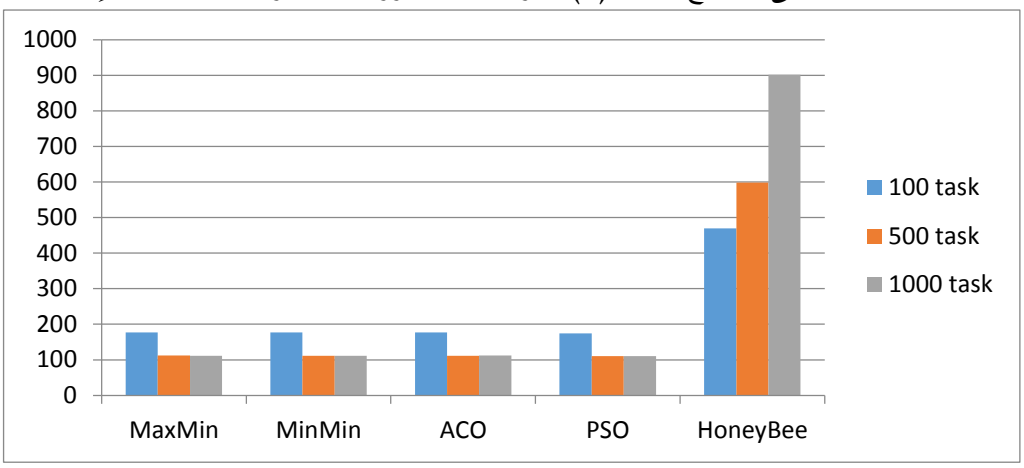
وهو المعدل بين الانشغال الكلي لعرض النطاق الترددي للألة الافتراضية بالنسبة لزمن التنفيذ الكلي

وتعطى قيمة استخدام عرض النطاق الترددي بالقانون التالي:

Bandwidth Utilization = (Total Bandwidth Usage / Number of Cloudlets) * 100

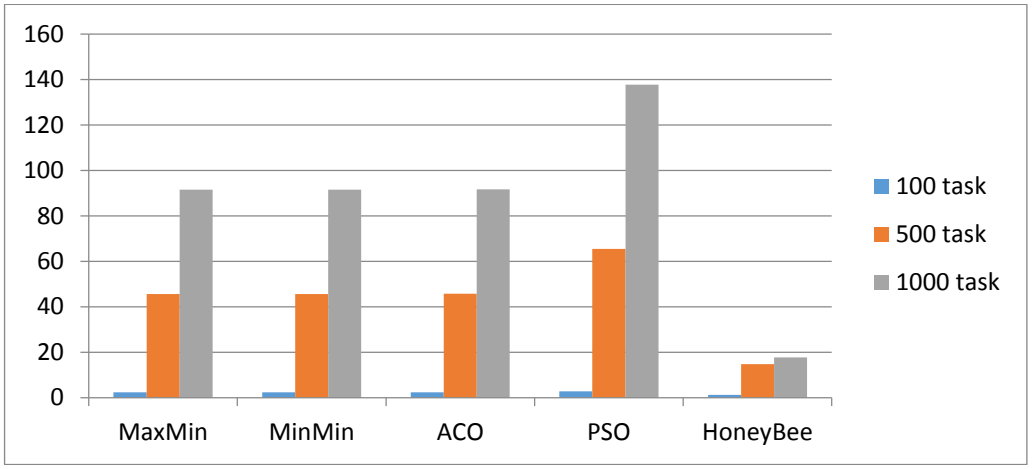
النتائج والمناقشة:

- **التجربة الأولى:** يوضح الشكل (1) مقارنة بين الخوارزميات المدروسة من ناحية الإنتاجية



الشكل(1) نتائج مقارنة الخوارزميات من حيث معيار الإنتاجية

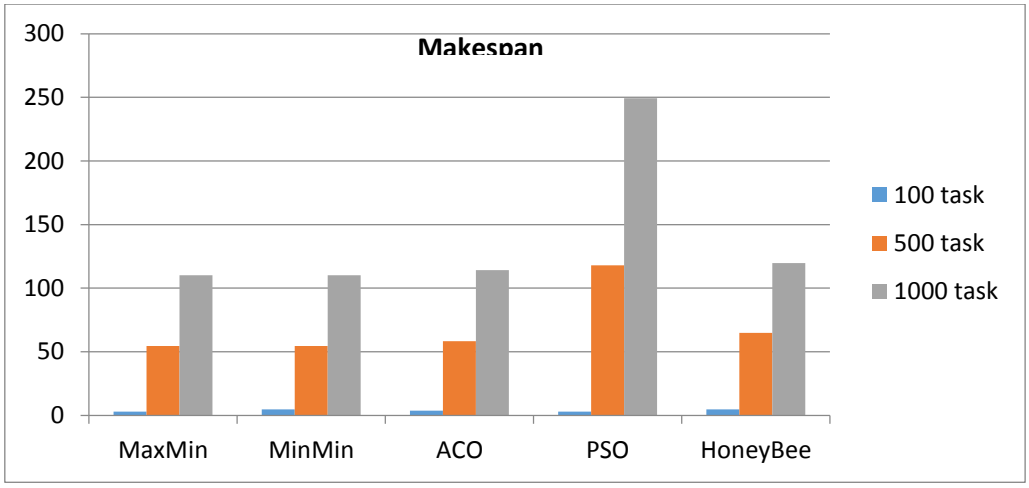
- نلاحظ من الشكل (1) تفوق خوارزمية Honey Bee على الخوارزميات الأخرى من حيث الإنتاجية من أجل عدد المهام 100و500 و1000. والسبب هو التصميم اللامركزي لخوارزمية Honey Bee الذي يمكّن من التوازي والتزامن في تخصيص الموارد مما يساعد على الحفاظ على الإنتاجية حتى مع زيادة عدد المهام.
- **التجربة الثانية:** يوضح الشكل(2) مقارنة بين الخوارزميات المدروسة من ناحية زمن الاستجابة



الشكل(2) نتائج مقارنة الخوارزميات من أجل معيار زمن الاستجابة

نلاحظ من الشكل (2) أن خوارزمية HoneyBee لها أقل وقت استجابة من الخوارزميات الأخرى من أجل عدد المهام 100 و500 و1000. ويرجع السبب في تفوق هذه الخوارزمية على غيرها من الخوارزميات المدروسة إلى حقيقة أن خوارزمية نحل العسل تأخذ دائمًا الحل الأنسب. حيث أنه في نحل العسل، يتم حساب الحل الأنسب من خلال حساب الملاءمة العامة، وهذا يساعد النحل المستكشف مرة أخرى في تحقيق النتائج المرجوة، أي وقت معالجة إجمالي أفضل.

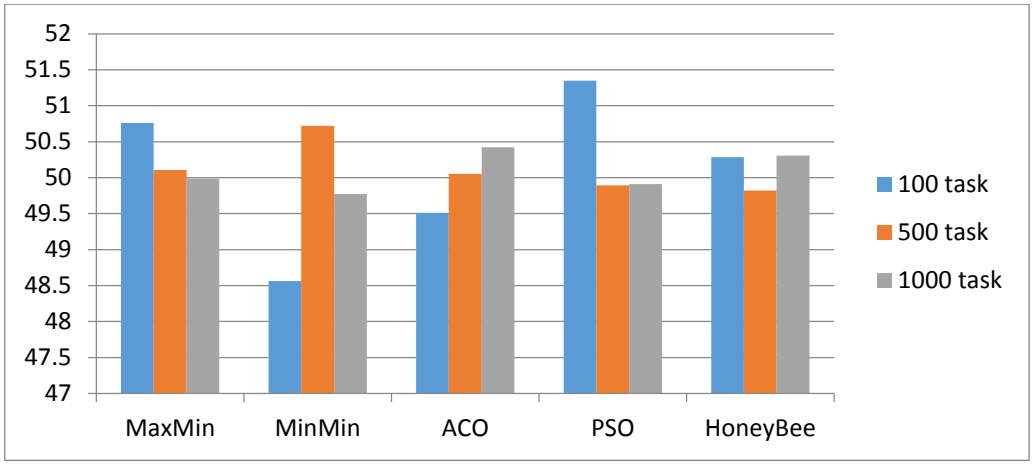
- **التجربة الثالثة:** يوضح الشكل (3) مقارنة بين الخوارزميات المدروسة من ناحية زمن التنفيذ الكلي



الشكل(3) نتائج مقارنة الخوارزميات من أجل معيار Makespan

نلاحظ من الشكل (3) أن الخوارزميات Honey Bee وACO وMimMin وMaxMin متقاربة من أجل 100 و500 و1000 مهمة؛ في حين أعطت خوارزمية PSO أكبر زمن تنفيذ كلي وبالتالي هي الأسوأ وذلك نتيجة لوقوعها في مشكلة الأمثليّة المحليّة.

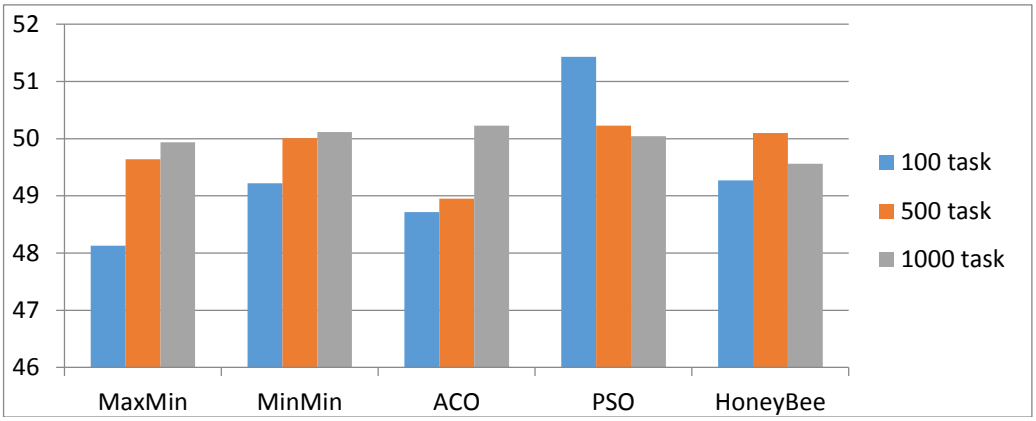
- **التجربة الرابعة:** يوضح الشكل (4) مقارنة بين الخوارزميات المدروسة من ناحية استخدام وحدة المعالجة المركزية



الشكل(4) نتائج مقارنة الخوارزميات من أجل معيار استخدام وحدة المعالجة

نلاحظ من الشكل (4) ان استخدام وحدة المعالجة المركزية يبقى قريباً من 50% بالنسبة لمعظم الخوارزميات من أجل عدد المهام المختلف. وهذه النتيجة مرغوبة لأنها تشير إلى أن وحدات المعالجة المركزية ليست محملة بالكامل ولا تعاني من نقص الاستخدام.

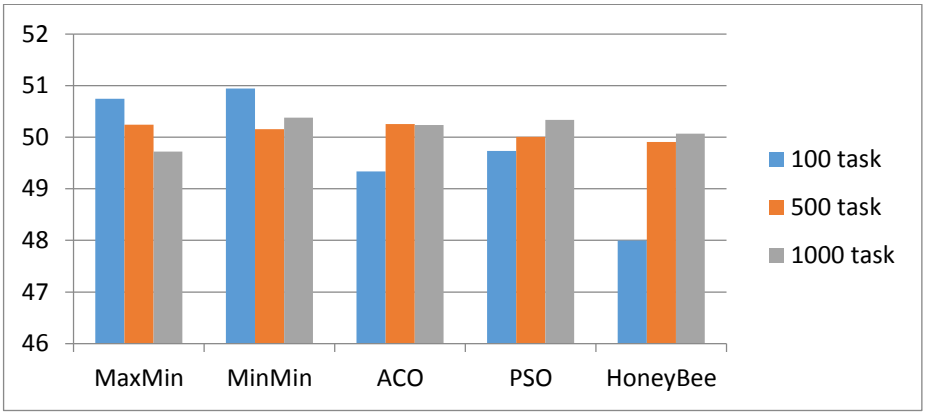
التجربة الخامسة: يوضح الشكل (5) مقارنة بين الخوارزميات المدروسة من ناحية استخدام الذاكرة



الشكل(5) نتائج مقارنة الخوارزميات من أجل معيار استخدام الذاكرة

نلاحظ من الشكل (5) أن استخدام الذاكرة يبقى قريباً نسبياً من 50% بالنسبة لمعظم الخوارزميات من اجل عدد المهام المختلف. وهذا يشير إلى استخدام متوازن للذاكرة أي يجنّب الاستخدام غير الكافي أو تخصيص الذاكرة المفرط.

- **التجربة السادسة:** يوضح الشكل (6) مقارنة بين الخوارزميات المدروسة من ناحية استخدام عرض النطاق الترددي



الشكل(6) نتائج مقارنة الخوارزميات من أجل معيار استخدام عرض النطاق الترددي

نلاحظ من الشكل(6) أن استخدام عرض النطاق الترددي يبقى قريباً من 50% بالنسبة لمعظم الخوارزميات من أجل عدد مهام العمل المختلفة. يشير هذا إلى استخدام متوازن لموارد الشبكة أي يجنّب الاستخدام غير الكافي أو تخصيص عرض النطاق الترددي المفرط الذي يمكن أن يؤدي إلى الازدحام.

الاستنتاجات والتوصيات:

- تحديد أفضل خوارزمية لتخصيص الموارد يعتمد على متطلبات النظام وهدفه، حيث في حالة كان هدف النظام تحقيق أعلى إنتاجية فإن خوارزمية Honey Bee هي الأفضل، أما في حالة كان هدف النظام تحقيق أعلى استخدام للموارد من أجل زيادة أرباح مزوّد الخدمة فإن خوارزمية PSO هي الأفضل.

- على الرغم من مزايا الخوارزميات الإرشادية في إيجاد حلول جيدة للمشاكل التي يصعب حلّها باستخدام الخوارزميات التقليدية فإنها تعاني من بعض السلبيات؛ على سبيل المثال وجدنا أن خوارزمية PSO تعاني من مشكلة الوقوع في الأمثلية المحلية حيث يعلق أفضل جسيم عالمي في الحدود الدنيا المحلية لأنه لا يتم تعديله ديناميكيًا في جميع التكرارات مما ينتج عنه معدل تقارب ضعيف. بينما تعاني خوارزمية Honey Bee من قدرات استغلال ضعيفة حيث أن صيغة تحديث الحل الخاصة بها تغير مكونًا واحدًا فقط من متجه الحل في كل مرة. كما وجدنا أن خوارزمية ACO تفشل في الحفاظ على القيود الزمنية لتنفيذ المهام وذلك لأنها حساسة لخصائص المهمة والعقد.

الأعمال المستقبلية:

- اقتراح خوارزمية هجينة تستفيد من نقاط القوة الموجودة في الأساليب الحالية بهدف الحصول على خوارزمية تعطي نتائج أفضل من الخوارزميات الحالية.
- إجراء تعديلات على بارامترات معينة في الأساليب الحالية للتغلّب على نقاط الضعف فيها.
- دراسة الخوارزميات الحالية والخوارزمية المحسنة الجديدة في بيئة غير متجانسة وهي البيئة التي تتغير فيها خصائص الآلات الافتراضية التي يقوم نظام الحوسبة السحابية بإنشائها تبعاً لطبيعة العمل الوارد وأولويته وحجمه.

References:

[1] Mell P, Grance T. The NIST Definition of Cloud Computing. 2011;
[2] Pandharpatte ProfRM. A Review: Resource Allocation Problem in Cloud Environment. International Journal of Engineering and Technology. 2017 Jun 30;9(3):1695–700.
[3] S MH, Kumar T S, Mustapha SMFDS, Gupta P, Tripathi RP. Hybrid Approach for Resource Allocation in Cloud Infrastructure Using Random Forest and Genetic Algorithm. Aldinucci M, editor. Scientific Programming. 2021 Oct 16;2021:1–10.
[4] Gupta NK, Vashishth TK. advancing resource allocation and scheduling in cloud computing: a novel algorithmic approach. 2023;
[5] Manavi M, Zhang Y, Chen G. Resource Allocation in Cloud Computing Using Genetic Algorithm and Neural Network. 2023;
[6] Sheetal P, Kongara R. Priority based resource allocation and scheduling using artificial bee colony (ABC) optimization for cloud computing systems. International Journal of Innovative Technology and Exploring Engineering. 2019;8(6S).
[7] Saad-Eddine Chafi, Younes Balboul, Fattah M, Mazer S, Moulhime El Bekkali. Enhancing Resource Allocation in Edge and Fog-Cloud Computing with Genetic Algorithm and Particle Swarm Optimization. Intelligent and converged networks. 2023 Dec 1;4(4):273–9. [8] Bayan T, Sarma P. MAHO: Modified Ant Honey Bee Optimization Algorithm for Load Balancing in Cloud Environment. International Journal of Computer Sciences and Engineering. 2020;8(7).
[9] Mohapatra S, Mohanty S, Hriteek Kumar Nayak, Millan Kumar Mallick, Naga V, Khasim Vali Dudekula. DPSO: A Hybrid Approach for Load Balancing using Dragonfly and PSO Algorithm in Cloud Computing Environment. EAI endorsed transactions on internet of things. 2024 Jan 11;10.
[10] Rani E, Kaur H. Study on fundamental usage of CloudSim simulator and algorithms of resource allocation in cloud computing. 2017 Jul 1;
[11]Gong S, Yin B, Zheng Z, Cai KY. Adaptive Multivariable Control for Multiple Resource Allocation of Service-Based Systems in Cloud Computing. IEEE Access.

2019;7:13817–31.
[12] Haki F, Luz A. Optimization Algorithm for Virtual Machine Placement in Cloud Computing. 2024 Feb 8;
[13] Al-Olimat HS, Alam M, Green RC, Lee JH. Cloudlet Scheduling with Particle Swarm Optimization. 2015 Apr 1;
[14] Li K, Xu G, Zhao G, Dong Y, Wang D. Cloud Task Scheduling Based on Load Balancing Ant Colony Optimization. ChinaGrid Annual Conference. 2011 Aug 22;
[15]Tawfik M, Bahgat A, Keshk A, Torkey F. Artificial Bee Colony Algorithm for Cloud Task Scheduling. IJCI International Journal of Computers and Information. 2015 Jun 1;4(1):1–10.
[16] Auna SY, Amburs FU. Efficient Max-Min Algorithm for Scheduling Workflow Tasks in Cloud Environment. International Journal of Information Processing and Communication. 2020 Dec;10(1&2):99–106.
[17] Aslanpour MS, Gill SS, Toosi AN. Performance Evaluation Metrics for Cloud, Fog and Edge Computing: A Review, Taxonomy, Benchmarks and Standards for Future Research. Internet of Things. 2020 Aug;12:100273.