تحسين أداء جدولة مهام الحوسبة السحابية في الخوارزمية التنافسية ICA باستخدام خوارزمية الكائنات المتكافلة SOS

بشار محسن إبراهيم [1] د. رأفة خازم[2] د.رؤى ونوس[3]

ملخص

تحصص الحوسبة السحابية الموارد المتوفرة ليتم تنفيذ المهام المطلوبة خلال أقل زمن تنفيذ واستخدام فعال للموارد. ويكون دور نظام الجدولة هو اتخاذ القرار المناسب الذي يضمن تنفيذ جميع المهام بحيث يكون مدى الاستفادة من الموارد عالياً ويكون زمن التنفيذ أقل مايمكن.

يقوم هذا البحث بتطبيق خوارزميتَين من خوارزميات الأمثَلة في جدولة مجموعة من المهام على آلات افتراضية متوفرة في نظام سحابي، هاتان الخوارزميتان هما خوارزمية البحث في الكائنات المتكافلة SOS (Symbiotic Organisms Search) وقد يُطلق عليها اسم خوارزمية التكافل البيئي، وخوارزمية التنافس بين المستعمرين ICA (Imperialist Competitive Algorithm) وقد يُطلَق عليها اسم خوارزمية التنافس الإمبريالي، وهما تهدفان للبحث عن حل أمثلي يتمثل بجدولة كل المهام على الآلات الافتراضية المتوفرة بزمن تنفيذ كلي أصغري Makespan علماً بأنَّ هاتين الخوارزميتين تندرجان ضمن الخوارزميات فائقة الاستدلال.

تمت نمذجة كل خوارزمية ومحاكاتها باستخدام بيئة المحاكاة cloudsim واستخدام لغة java من خلال بيئة التطوير Netbeans، وبعد ذلك تم اقتراح خوارزمية جديدة تحمل اسم ICA-SOS باعتماد خوارزمية SOS في مرحلتين من مراحل خوارزمية ICA بهدف الاستفادة من سرعة تقاربها نحو الحل الأمثلي والإبقاء على عملية المنافسة للاستيلاء على المستعمرات الجديدة في خوارزمية ICA، ثم الوصول إلى حل أمثلي أفضل من الحل الذي تعطيه خوارزمية ICA وحدها.

بينتْ النتائج – بعد تنجيز الخوارزمية المقترحة وتجريبها– تحسين الحل بنسبة 25% مقارنة

[1] طالب دراسات عليا (ماجستير) – هندسة الحواسيب وشبكاتها – كلية الهندسة الميكانيكية والكهربائية – جامعة دمشق – دمشق – سورية.

[2] أستاذ – قسم هندسة الحواسيب والأتمتة – كلية الهندسة الميكانيكية والكهربائية – جامعة دمشق – دمشق – سورية.

[3] باحث – المعهد العالي للعلوم التطبيقية والتكنولوجيا – دمشق – سورية.

بخوارزمية ICA.

الكلمات المفتاحية: الحوسبة السحابية، جدولة المهام، خوارزمية SOS، خوارزمية ICA.

Abstract

The goal of scheduling in cloud computing is to allocate the available resources to execute required tasks with low execution time and effective use of resources. The role of the scheduling system is to make the appropriate decision for implementation of all tasks with high utilization of resources and low execution time, to achieve this we used optimization algorithms in this research.

This research implements two optimization algorithms in scheduling a set of tasks on virtual machines available in a cloud system. These two algorithms are the SOS (Symbiotic Organisms Search), and the ICA (Imperialist Competitive Algorithm), they aim to search for an optimal solution by scheduling all tasks on virtual machines available with a minimum Makespan, knowing that these algorithms are among the metahuristic algorithms.

Each algorithm was modeled and simulated using the cloudsim simulation environment and using the java language through the development environment Netbeans, and then a new algorithm called ICA-SOS was proposed by integrating the two algorithms by adopting the SOS algorithm in two stages of the ICA algorithm in order to take SOS advantage speed and maintain On the competition to capture new colonies in the ICA algorithm, and thus reach an optimal solution that is better than the solution given by ICA algorithm.

The results of the new ICA-SOS algorithm showed improvement rates for ICA algorithm, ICA was improved by 25%.

Keywords: cloud computing, task scheduling, Symbiotic Organism Search, SOS, Imperialist Competitive Algorithm, ICA.

1 مقدمة

ظهرت العديد من الأبحاث المتعلقة بالبحث في البيانات والاستفادة من الشبكة بالشكل الأمثل، تماشياً مع التطور السريع في مجال الحوسبة واستخدام شبكة الانترنت وكمية البيانات الضخمة المتبادلة بشكل يومي على الشبكة، وبعد ظهور مفهوم الحوسبة السحابية كان لابد من البحث في هذا المجال وتطويره بما يلائم متطلبات العصر. تعتمد الفكرة الأساسية للحوسبة السحابية على تقديم الخدمات للزبائن عن طريق شبكة الانترنت، ومع زيادة أعداد المستخدمين وزيادة الطلب على الخدمات، زادت معها المشكلات المتعلقة بجدولة هذه الطلبات وتنظيمها على المخدمات للوصول إلى أعلى استفادة وبزمن استجابة أقل مايمكن. وما زال البحث في تطوير خوارزميات للجدولة في الحوسبة السحابية الشغل الشاغل للعديد من الباحثين. ظهرَ مصطلح الحوسبة السحابية في أعمال وشركات مختلفة قبل أن تقوم شركة Amazon عام 2002 بإطلاق سحابتها الأولى Amazon Web Services وفي عام 2006 أطلقت الشركة سحابتها الثانية وأسمتها AC2 لتقديم الخدمات التجارية على شبكة الانترنت، وفي عام 2009 ظهرت السحابة الأشهر التي نستخدمها بشكل يومي، وهي سحابة Google متمثلة بتطبيقاتها المتعددة التي تقدم خدمات عديدة للمستخدمين.

يعتمد نموذج الحوسبة السحابية على تقديم كل شيء ليكون خدمة Everything as a service وتنقسم هذه الخدمات إلى ثلاث طبقات رئيسية هي:

- طبقة البرامج كونها خدمة Software as a service (SAAS): هي أعلى مستوى في السحابة، تهتم بالتطبيقات المتعلقة بالمستخدم النهائي، مثل أنظمة البريد الالكتروني، تطبيقات إدارة علاقات العميل، البرمجيات المشتركة، أنظمة إدارة سير العمل.
- طبقة المنصّات كونها خدمة Platform as a service (PAAS): هي الطبقة الثانية من طبقات الحوسبة السحابية، تتألف بشكل أساسي من مكتبات وبرامج وسيطة، وتحديثات وأدوات يحتاجها المطورون في تحديث الطبقة الأولى، وتستفيد هذه الطبقة من طبقة IAAS مثل

البيئات الافتراضية التي توفرها تلك الطبقة وكذلك البرمجيات المطورة في المصادر الإفتراضية لطبقة IAAS.

- طبقة البنية التحتية كونها خدمة Infrastructure as a service (IAAS): هي الطبقة الأخيرة من طبقات الحوسبة السحابية، يشار إليها أحياناً بطبقة الأجهزة كونها خدمة Hardware as a service (HAAS) تتضمن خدمات التخزين والنسخ الإحتياطية وقواعد البيانات والأمن. توفّر هذه الطبقة البنية التحتية للأجهزة، فبدلاً من شراء المخدمات والبرمجيات يقوم العملاء بشراء هذه المصادر كونها خدمة مستقلة، ويتم حساب التكلفة على أساس المصادر المستخدمة. إن هذه الطبقة تستخدم تكنولوجيا الحوسبة الإفتراضية بشكل كبير Virtualization technology حيث تساعد على توفير الطاقة والتكلفة والمساحة في قواعد البيانات، ولأنها تحوي موارد السحابة سيتم تخصيص هذه الموارد، وجدولة المهام الواجب تنفيذها اعتماداً على خوارزميات الجدولة المدروسة. يدل مصطلح الجدولة في الحوسبة السحابية على عملية تحصيص الموارد المتاحة لتقوم بتنفيذ المهام المطلوبة، مع مراعاة الكفاءة الأعلى لاستخدام الموارد وبزمن تنفيذ أقل، حيث يتوفر العديد من الموارد في النظام السحابي وبالمقابل هناك العديد من المهام الواجب تنفيذها على هذه الموارد، وهنا يأتي دور نظام الجدولة لاتخاذ القرار المناسب الذي يضمن تنفيذ جميع المهام باستخدام أمثل للموارد وبزمن تنفيذ أقل مايمكن، من هنا كان لخوارزميات الأمثَلة دور أساسي للوصول إلى هذا الهدف، وكان المجال واسعاً لاستخدام هذه الخوارزميات وتطويرها في مجال الجدولة في الحوسبة السحابية.

2 الهدف من البحث

إن زيادة أعداد المستخدمين وزيادة الطلب على الخدمات، زادت معها المشكلات المتعلقة بجدولة هذه الطلبات وتنظيمها على المخدمات للوصول إلى أعلى استفادة وبزمن استجابة أقل مايمكن، ومن هنا يعتبر زمن تنفيذ المهام على الموارد المتاحة من أجهزة افتراضية من أهم المعاملات التي لا بد من تحسينها وإبقائها في الحدود الدنيا، لذلك لم يوفر الباحثون طريقة لتحسين هذا المعامل ومازال البحث والتطوير مستمراً لاسيما مع التطور السريع للخدمات المتاحة على شبكة الانترنت، وأيضاً زيادة

الطلب على الخدمات وزيادة أعداد المستخدمين، وهذا يقابله تضخم المهام الواجب تنفيذها على المخدمات وبأطوال مختلفة.

كان لخوارزميات البحث الرياضية التقليدية دوراً أساسياً في كثير من مشكلات التحسين، لكن مع ازدياد فضاء البحث كما في حالة النظام السحابي لم تعد الخوارزميات التقليدية قادرة على التوصل للحل الأمثل، من هنا كان التوجه لاستخدام خوارزميات قادرة على مسح كامل فضاء البحث هو المنقذ للعديد من مشكلات التحسين، ومع ظهور خوارزميات رياضية مستمدة من الطبيعة توجه العديد من الباحثين لاستخدامها ونمذجتها بما يلائم مسألة البحث، فمنهم من استخدم خوارزمية مستعمرة النمل ACO للبحث عن أقصر مسار أو لاستخدامها في البحث ضمن الشبكة، واستخدم آخرون الخوارزمية الجينية GA بهدف تطوير أجيال جديدة تمتلك تكيف أعلى مع المسألة المطروحة وبالتالي التوصل لحل أمثلي، وتجاوز بعض الباحثين هذه الخوارزميات إلى اقتراح خوارزميات جديدة تعتمد على الدمج بين خوارزميتين أو أكثر بهدف الاستفادة من ميزات كل خوارزمية وبالنتيجة التوصل إلى حل أمثلي يكون أفضل من الحل الذي تصل إليه كل خوارزمية على حدة، من هنا كان التوجه في هذا البحث لإيجاد خوارزمية تكون قادرة على إيجاد حل أمثلي بمسح كامل فضاء البحث وأيضاً تعتمد على تقسيم فضاء البحث بهدف التوصل لأفضل حل. إن هدفنا في هذا البحث هو إيجاد الحل الأمثلي لجدولة عدة مهام ضمن البيئة السحابية بحيث يكون الزمن الكلي Makespan لتنفيذ المهام على الأجهزة الافتراضية المتاحة أقل مايمكن.

تم اختيار خوارزميتين هما خوارزمية البحث في الكائنات المتكافلة SOS التي تتميز بتقاربها السريع من الحل الأمثلي، وأيضاً خوارزمية التنافس بين المستعمرين ICA التي تتميز بالتوصل لحل أمثلي أفضل مع ازدياد عمليات التنافس، سيتم الدمج بين هاتين الخوارزميتين بهدف التوصل إلى زمن تنفيذ كلي أمثلي أفضل من الزمن الذي تصل له خوارزمية التنافس بين المستعمرين وحدها ICA.

3 الدراسات المرجعية

تطرقت دراسات عديدة لموضوع جدولة المهام في الحوسبة السحابية، واعتمد بعضها على الدمج بين الخوارزميات المتاحة للتوصل لنتائج أفضل.

درسَ [7] موضوع الجدولة بالاعتماد على الدمج بين خوارزمية مستعمرة النمل ACO وخوارزمية SOS حيث تم معالجة مشكلة ضعف فعالية جدولة المهام عند استخدام خوارزمية النمل وحدها، بالاستفادة من سرعة تقارب خوارزمية SOS، كما قدّمَ باحثون في [8] دراسة جدولة المهام في الحوسبة السحابية لحل مشكلة ضعف فعالية جدولة المهام باستخدام خوارزمية ACO وحدها حيث تم دمج خوارزمية ICA للاستفادة من عملية التنافس بين الإمبرياليين للوصول لحل أمثلي أفضل من الحل الذي تم التوصل له باستخدام خوارزمية ACO وحدها، أما [2] فقد قاموا باستخدام خوارزمية SOS وحدها ونمذجتها لحل مسألة الجدولة في الحوسبة السحابية، وذلك للاستفادة من سرعة تقاربها من الحل الأمثلي في حالة حجم المهام الكبير، ثم قدم [5] دراسة استخدما فيها خوارزمية التنافس الإمبريالي ICA لحل مسألة جدولة المهام في الحوسبة السحابية والاستفادة من عملية المنافسة بين الإمبرياليين.

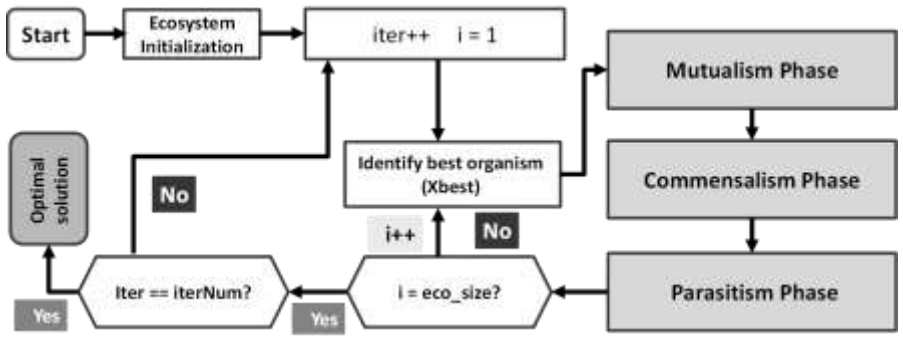
4 خوارزمية البحث في الكائنات المتكافلة SOS

تعتمد خوارزمية SOS على محاكاة سلوك الكائنات الحية في الطبيعة، وهي من الخوارزميات فائقة الاستدلال metaheuristic تُستخدم لحل مشكلات التحسين العددي، وتبدأ الخوارزمية بمجموعة أولية تسمى النظام البيئي.

يتم إنشاء مجموعة من الكائنات في النظام البيئي الأولي بشكل عشوائي في مساحة البحث، يمثل كل كائن حي حلاً مرشحاً واحداً للمشكلة المقابلة، ويرتبط كل كائن حي في النظام البيئي بقيمة تكيف معينة fitness value التي تعكس درجة التكيف مع الهدف المنشود، ويتم في هذه الخوارزمية التحكم في توليد الحلول الجديدة من خلال محاكاة التفاعل بين كائنين في النظام البيئي، وتمر

الخوارزمية بثلاث مراحل هي مرحلة التبادل Mutualism Phase ومرحلة التعايش Commensalism Phase ومرحلة التطفل Parasitism Phase تشبه نموذج التكافل البيئي في العالم الحقيقي [1].

يوضح الشكل 4-1مخطط خوارزمية SOS.



الشكل 4-1 مراحل خوارزمية SOS

يدلّ i في الشكل 4-1 لعدد الكائنات في النظام البيئي، ويتم تحديده كدخل للخوارزمية، ويدل iter على عدد التكرارات، ويتم تحديده كدخل للخوارزمية، بحيث يتم في كل تكرار المرور على كل الكائنات في النظام البيئي ecosystem.

4.1 مرحلة التبادل Mutualism phase

تتلخص هذه المرحلة بعلاقة مستفيد – مستفيد، حيث يتم التفاعل بين كائنين في النظام البيئي وينتج عن هذا التفاعل منفعة متبادلة للطرفين، مثل العلاقة بين الأزهار والنحل في الطبيعة، حيث تستفيد الأزهار من النحل بأنه يساعدها في عملية التلقيح بنقل حبات الطلع من زهرة لأخرى، وأيضاً يستفيد النحل من هذه العلاقة بأنه يتغذى على الأزهار ويقوم بإنتاج العسل.

بفرض Xi هو كائن حي ضمن النظام البيئي، يتم اختيار كائن حي آخر بشكل عشوائي Xj للتفاعل مع Xi، ينخرط كلا الكائنين في علاقة متبادلة بهدف زيادة ميزة البقاء المتبادل. يمثل كل من Xi وXj حلاً مرشحاً يتم الاستبدال بهِ حلاً آخر بالإعتماد على المعادلات التالية التي تحسب الحلول المرشحة الجديدة بناءً على التعايش المتبادل بين الكائنين:

$$X_{inew} = X_i + rand(0,1) * (X_{best} - MutualVector * BF1) \quad (1)$$

$$X_{jnew} = X_j + rand(0,1) * (X_{best} - MutualVector * BF2) \quad (2)$$

$$MutualVector = \frac{X_i + X_j}{2} \quad (3)$$

إذا كانت درجة تكيف X_{inew} أعلى من درجة تكيف X_i يتم الاستبدال بقيمة X_i قيمةً جديدةً X_{inew} وإلا تبقى القيمة السابقة ل X_i كما هي، وكذلك الأمر بالنسبة للكائن X_j [1].

تُحدد التوابع BF1 و BF2 فيما إذا كان الكائن الحي يستفيد جزئياً أو كلياً من التفاعل حيث الاستفادة متبادلة، تكون قيمة كل منهما إما 1 أو 2، يتم اختيار هذه القيمة بشكل عشوائي من أجل كل كائن حي، فهذه التوابع تعتمد على كمية الفائدة التي يحصل عليها كل من الكائنين [1].

يمثل Xbest الحل المرشح الأفضل الذي يمتلك أعلى درجة تكيف، ويمثل الهدف الذي تسعى بقية الكائنات للوصول إلى درجة تكيفه، حيث تحاول بقية الكائنات في النظام البيئي زيادة درجة تكيفها للبقاء على قيد الحياة، وفقاً لنظرية داروين "ستسود الكائنات الصالحة فقط" [1].

4.2 مرحلة التعايش Commensalism phase

تتلخص هذه المرحلة بعلاقة مستفيد – محايد، حيث يستفيد أحد الطرفين من العلاقة، بينما لايتأثر الطرف الآخر، مثال العلاقة بين أسماك ريمورا وأسماك القرش، حيث تتغذى أسماك ريمورا على بقايا طعام سمكة القرش فهي بذلك تستفيد وتحافظ على بقائها، بينما سمكة القرش لاتتأثر بذلك ولاتشعر بوجود سمكة الريمورا، تمثل المعادلة التالية حساب كائن جديد بالاستفادة من كائن آخر.

$$X_{inew} = X_i + rand(-1,1) * (X_{best} - X_j) \quad (4)$$

يمثل الجزء (Xbest-Xj) الفائدة التي يقدمها الكائن Xj لمساعدة Xi على البقاء، إذا كانت درجة تكيف Xinew أعلى من درجة تكيف Xi يتم استبدال قيمة Xi بالقيمة الجديدة Xinew وإلا تبقى القيمة السابقة كما هي [1].

4.3 مرحلة التطفل Parasitism phase

تمثل هذه المرحلة علاقة مستفيد – متضرر، حيث يستفيد أحد الطرفين من العلاقة بينما يتضرر الطرف الآخر، مثال على التطفل هو طفيلي البلازموديوم، الذي يستخدم علاقته ببعوضة الأنوفيلة للانتقال بين المضيفين من البشر. بينما ينمو الطفيل ويتكاثر داخل جسم الإنسان، فإن مضيفه البشري يعاني من الملاريا وقد يموت نتيجة لذلك [1].

يتم إعطاء الكائن Xi دوراً مشابهاً لبعوضة الأنوفيلة من خلال إنشاء طفيلي اصطناعي يتم إنشاؤه في مساحة البحث عن طريق تكرار الكائن Xi، ثم تعديل الأبعاد المختارة عشوائياً باستخدام رقم عشوائي، ثم يتم اختيار الكائن Xj بشكل عشوائي من النظام البيئي ويعمل كمضيف لناقل الطفيلي، يحاول الطفيلي استبدال Xj في النظام البيئي، إذا كان لدى الطفيلي قيمة تكيف أفضل فسوف يقتل الكائن Xj ويتولى موقعه في النظام البيئي، أما إذا كانت قيمة تكيف Xj أفضل، فسيتمتع Xj بحصانة من الطفيلي ويبقى Xj في مكانه ويتم إهمال الطفيلي.

يتم توليد الطفيلي بالإعتماد على الكائن Xi من خلال توليد رقم عشوائي وضربه بقيمة Xi ثم اختيار كائن آخر Xj بشكل عشوائي ثم حساب درجة التكيف لكلا الكائنين واختيار الكائن الذي يحقق درجة تكيف أعلى وإهمال الكائن الآخر.

يتم إعادة المراحل السابقة إلى أن ينتهي عدد التكرارات وتنتهي الخوارزمية، مع العلم أن دخل الخوارزمية هو عدد التكرارات وعدد الكائنات.

يتم الحصول في نهاية الخوارزمية على كائن يحقق نسبة تكيف أعلى، يمثل هذا الكائن الحل الأمثلي الذي استطاعت الخوارزمية إيجاده.

5 خوارزمية التنافس بين المستعمرين Imperialist Competitive Algorithm

تم اقتراح هذه الخوارزمية لحل مشاكل الأمثلة العددية [3] وهي مستوحاة من التنافس الإمبريالي، تبدأ الخوارزمية بمجموعة أولية، يطلق على كل عنصر فيها اسم الدولة Country التي تنقسم إلى نوعين، المستعمرات Colonies والإمبرياليين imperialists اللذين يشكلان معاً إمبراطوريات empires.

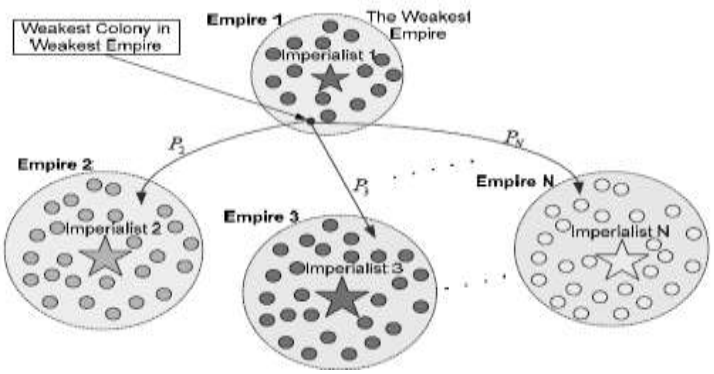
تعتمد فكرة الخوارزمية على المنافسة الإمبريالية بين الإمبراطوريات Imperialistic competition للوصول بالنهاية إلى إمبراطورية واحدة قوية تكون قد سيطرت في مراحل سابقة على جميع مستعمرات الإمبراطوريات الضعيفة.

تبدأ الخوارزمية بتعريف فضاء الحلول population الذي يمثل مجموعة الدول countries بحيث تنقسم الدول إلى إمبرياليين imperialists ومستعمرات Colonies يتم تصنيف الدولة كإمبريالية حسب أفضليتها وطاقتها، وتكون أفضل الدول هي إمبريالية imperialist وماتبقى تصنف كمستعمَرة Colony حيث كل مجموعة مستعمَرات تتبع لإمبريالية ما، يمثل الشكل 5-1 الإمبريالي بنجمة والمستعمرة بدائرة، ويمثل كل من الإمبريالي والمستعمرَة دولة، أما الإمبراطورية فهي مجموعة مستعمَرات وإمبريالي واحد.

يتم تقسيم المستعمرات على الإمبرياليين وفقاً لطاقتها، تشكل المستعمرات والإمبرياليين ما يسمى إمبراطورية empire، ولكل إمبراطورية طاقة كلية تعتمد على طاقة الدولة الإمبريالية وطاقة مستعمراتها.

وبعد تقسيم جميع المستعمرات على الإمبرياليين، تبدأ هذه المستعمرات بالتحرك نحو دولتهم الإمبريالية ذات الصلة، ثم تبدأ المنافسة الإمبريالية imperialistic competition بين كل الإمبراطوريات، بحيث كل إمبراطورية لاتتمكن من زيادة طاقتها (أو على الأقل تمنع نقصان طاقتها)

سيتم استبعادها من المنافسة [3]، وستؤدي المنافسة الإمبريالية تدريجياً إلى زيادة قوة الإمبراطوريات القوية ونقصان قوة الإمبراطوريات الأضعف بحيث تفقد الإمبراطوريات الضعيفة قوتها وتنهار، وبالنهاية ينتج إمبراطورية واحدة يكون للإمبريالي فيها الطاقة الأكبر وهو يمثل الحل الذي تجده الخوارزمية.

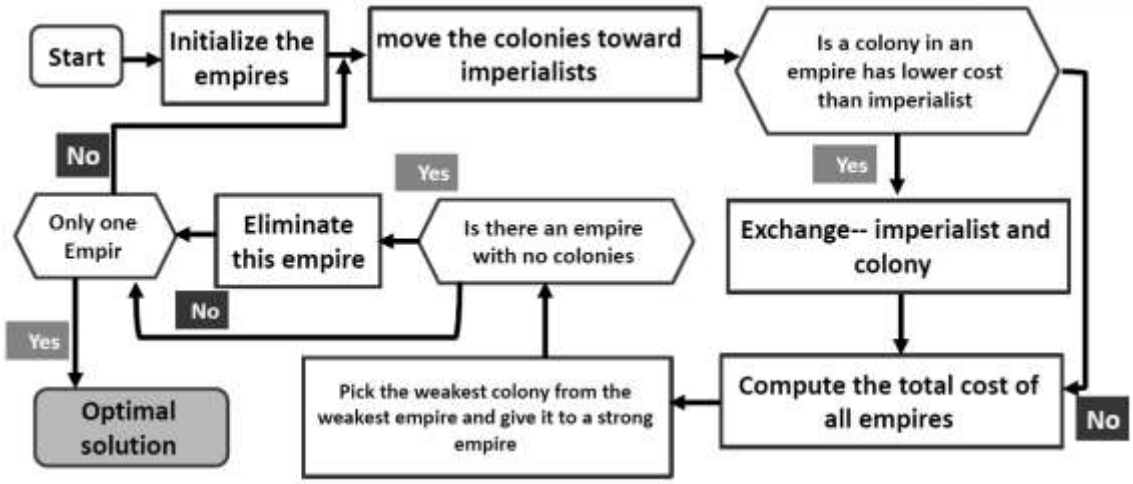


الشكل 5-1 الإمبرياليون والمستعمرات التابعة لهم والإمبراطوريات

من نقاط ضعف هذه الخوارزمية تقاربها السريع من local optima الذي يمثل حلاً مرشحاً لكن ليس هو الحل النهائي global optima لكنها تستطيع الخروج منه بعد عدة تكرارات في مرحلة التنافس الإمبريالي، وقد يؤدي ذلك إلى استهلاك وقت للوصول إلى إمبراطورية واحدة.

يُعدُّ الوصول إلى الحل الأمثلي المحلي local optima فخاً أو مصيدةً traps لأغلب خوارزميات البحث التقليدية، وبهذا تتغلب خوارزميات البحث ال metaheuristic بقدرتها على الهروب من الأفخاخ بسبب اعتمادها على قيم عشوائية وعلى عدد من التكرارات [4].

يوضح الشكل 5-2 مراحل خوارزمية ICA [4].



الشكل 5- 2 مخطط خوارزمية ICA

5.1.1 مراحل خوارزمية ICA

- **تهيئة الإمبراطوريات وتوزيع المستعمرات عليها**

يتم تشكيل إمبراطوريات عددها N_{imp} تتألف كل إمبراطورية من عدة دول، العدد الكلي للدول هو N_{pop} وهو يمثل حجم فضاء الحلول الأولي، يُعدُّ كل من N_{imp} و N_{pop} محددات دخل الخوارزمية.

تكون الدولة إما إمبريالي imperialist وإما مستعمرة colony وتمتلك كل إمبراطورية إمبريالي واحد وعدة مستعمرات تتبع لهذا الإمبريالي، بالتالي عدد الإمبرياليين يساوي عدد الإمبراطوريات N_{imp} أما عدد المستعمرات الكلي N_{col} يساوي الفرق بين عدد الدول وعدد الإمبرياليين:

$$N_{col} = N_{pop} - N_{imp}$$

يتم توزيع المستعمرات على الإمبرياليين استناداً إلى طاقة الإمبريالي:

$$C_n = c_n - max_i\{c_i\} \quad (9)$$

حيث c_n هو قيمة تابع الكلفة للإمبريالي n أما C_n فهو قيمة تابع الكلفة للإمبريالي n بعد التقييس.

إنَّ تابع الكلفة في مسألة الجدولة في هذا البحث هو التابع الذي يحسب زمن التنفيذ الكلي Makespan للإمبريالي n، كما توضح المعادلة:

$$Makespan_n = makespan_n - max_i\{makespan_i\} \quad (10)$$

حيث يتم حساب زمن التنفيذ لكل مستعمرة من المستعمرات التابعة للإمبريالي n ثم حساب القيمة العظمى max لقيم أزمنة المستعمرات، ثم طرحها من قيمة زمن التنفيذ للإمبريالي n.

$$p_n = \left| \frac{C_n}{\sum_{i=1}^{N_{imp}} C_i} \right| \quad (11)$$

تمثل p_n طاقة الإمبريالي التي سيتم على أساسها حساب عدد المستعمرات التابعة لهذا الإمبريالي كما يلي:

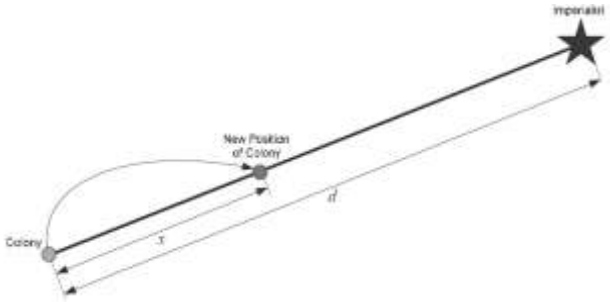
$$N.C_n = round\{p_n.N_{col}\} \quad (12)$$

تمثل $N.C_n$ عدد المستعمرات الأولي في الإمبراطورية n ذات الإمبريالي n

تمثل N_{col} عدد المستعمرات الكلي.

- **تحرك المستعمرات باتجاه الإمبريالي (Assimilating)**

تبدأ المستعمرات في كل إمبراطورية بالتحرك نحو الإمبريالي في هذه الإمبراطورية كما يوضح الشكل 5−3، قد تتحرك المستعمرة إلى موضع جديد تكون قيمة الكلفة عند هذا الموضع أقل(أفضل) من قيمة الكلفة للإمبريالي[3].



الشكل 5-3 تحرك المستعمرة نحو الإمبريالي

يتم نمذجة هذه الحركة رياضياً باختيار المتحول العشوائي x الذي يتم حسابه باستخدام تابع توزيع عشوائي وليكن U حيث

$$x \sim U(0, \beta * d)$$

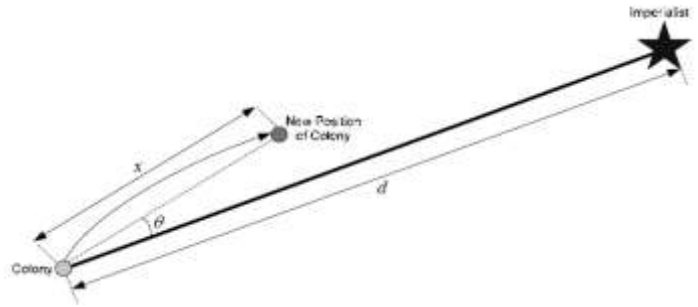
تمثل d المسافة بين المستعمرة والإمبريالي.

تمثل β مقدار أكبر من الواحد (يساوي تقريباً 2).

قد تتحرك المستعمرة باتجاه الإمبريالي وفق زاوية θ يتم توليدها وفق تابع توزيع عشوائي

$$\theta \sim U(-\gamma, \gamma)$$

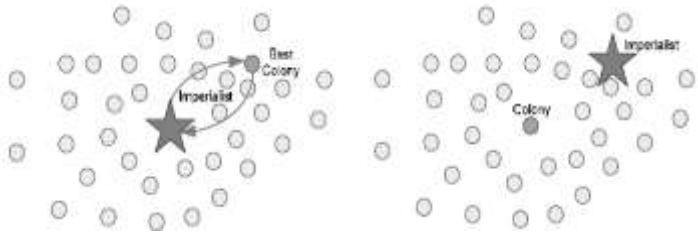
حيث $\gamma \sim \frac{\pi}{4}$ كما يوضح الشكل 5-4.



الشكل 5-4 تحرك المستعمرة نحو الإمبريالي باتجاه عشوائي

- **تبديل مواقع المستعمرة والإمبريالي (Exchanging)**

عندما تتحرك المستعمرة إلى موضع جديد قيمة تابع الكلفة عنده أقل(أفضل) من قيمة تابع الكلفة للإمبريالي، يتم تبديل مواضع المستعمرة والإمبريالي، وتصبح المستعمرة إمبريالياً، ويصبح الإمبريالي القديم مستعمرةً كما يوضح الشكل 5-5، وتتابع الخوارزمية خطواتها مع الإمبريالي الجديد [3].



الشكل 5-5 تبديل مواضع المستعمرة والإمبريالي

- **حساب الطاقة الكلية للإمبراطورية**

طاقة الإمبراطوية تتأثر بشكل رئيسي بطاقة الإمبريالي، وتؤدي بقية المستعمرات دوراً صغيراً في حساب طاقة الإمبراطورية [3] كما توضح المعادلة:

$$T.C_n = Cost(Imperialist_n) + \xi mean\{Cost(Colonies\ of\ empire_n)\} \quad (13)$$

تمثل $T.C_n$ الكلفة الكلية للإمبراطورية n ويمثل ξ مقداراً صغيراً موجباً أصغر من الواحد، يعبر عن كمية تأثير كلفة المستعمرات في الكلفة الكلية للإمبراطورية.

- **التنافس الإمبريالي (Imperialistic competition)**

يتم في هذه المرحلة التنافس بين الإمبراطوريات للحصول على مستعمرات، يتم اختيار الإمبراطورية الأضعف والاستيلاء على أضعف مستعمرة فيها من قِبل إمبراطوية قوية، حيث تتنافس الإمبراطوريات للحصول على المستعمرة الضعيفة.

إن تحديد الإمبراطوريات القوية يتم بحساب إمكانية استيلاء كل إمبراطورية، يتم أولاً حساب كلفة الإمبراطورية n

$$N.T.C_n = T.C_n - max_i\{T.C_i\} \quad (14)$$

يمثل $N.T.C_n$ قيمة تابع الكلفة للإمبراطورية n بعد التقييس.

يمثل $T.C_n$ قيمة تابع الكلفة للإمبراطورية n

يتم بعدها حساب إمكانية استيلاء كل إمبراطورية كما في المعادلة:

$$p_{p_n} = \left| \frac{N.T.C_n}{\sum_{i=1}^{N_{imp}} N.T.C_i} \right| \quad (15)$$

يتم بعدها تقسيم المستعمرات على الإمبراطوريات بحسب إمكانية استيلاء كل إمبراطورية لذلك يتم حساب الشعاع P

$$P = \left[p_{p_1},\ p_{p_2}, p_{p_3}, \dots, p_{p_{N_{imp}}}\right] \quad (16)$$

ثم يتم إنشاء الشعاع R بحجم الشعاع P نفسه ويحتوي كل مسقط فيه قيمة عشوائية بين الصفر والواحد:

$$R = \left[r_1,\ r_2, r_3, \dots, r_{N_{imp}}\right] \quad (17)$$

$$r_1,\ r_2, r_3, \dots, r_{N_{imp}} \sim U(0,1)$$

ثم يتم تشكيل شعاع الفرق بين الشعاعين السابقين:

$$D = P - R = \left[p_{p_1} - r_1,\quad p_{p_2} - r_2,\quad p_{p_3} - r_3,\ \dots,\quad p_{p_{N_{imp}}} - r_{N_{imp}}\right]$$

$$D = \left[D_1,\ D_2, D_3, \dots, D_{N_{imp}}\right] \quad (18)$$

القيمة الأكبر من D ذات الدليل n توافق الإمبراطورية ذات الدليل نفسه والتي تمتلك أعلى إحتمالية استيلاء على المستعمرة الأضعف في الإمبراطورية الأضعف[3].

يتم إعادة حساب الإمبراطورية الأضعف والإمبراطورية ذات احتمال الاستيلاء الأعلى وتستمر المنافسة.

- **إهمال الإمبراطورية الأضعف**

بعد عدة عمليات استيلاء قد تفقد إمبراطورية ما جميع مستعمراتها وعندها يتم استبعاد هذه الإمبراطورية من المنافسة ويتناقص عدد الإمبراطوريات، ثم يتم الرجوع إلى مرحلة تحرك المستعمرات باتجاه الإمبريالي في كل إمبراطورية وتستمر المنافسة، إلا إذا بقي إمبراطورية واحدة فقط عندها تتوقف الخوارزمية.

- **توقف الخوارزمية عند الوصول لإمبراطورية واحدة**

تتوقف خوارزمية التنافس الإمبريالي عند الوصول إلى إمبراطورية واحدة، ويكون الحل الأمثلي الذي وصلت إليه الخوارزمية هو الإمبريالي في هذه الإمبراطورية الوحيدة، والذي يمتلك أفضل قيمة لتابع الكلفة.

في حالة الجدولة يمثل الإمبريالي في الإمبراطورية الوحيدة بعد توقف الخوارزمية الحل الأمثلي الذي وصلت إليه الخوارزمية، وهو عبارة عن مجموعة الثنائيات التي تمثل إسناد كل مهمة إلى جهاز إفتراضي معين بحيث يكون زمن التنفيذ الكلي لمجموعة الإسنادات أصغر مايمكن.

6 استخدام خوارزمية ICA في جدولة المهام في الحوسبة السحابية

إن الحل الذي يتم البحث عنه هو تشكيلة من الإسنادات، يتم إسناد كل مهمة c_i إلى جهاز افتراضي ما v_j

بفرض أن الأرقام المعرفة للمهام هي 0,1,2,3,4 وأن الأرقام المعرفة للأجهزة الافتراضية هي 0,1,2,3

زمن التنفيذ الكلي Makespan هو مجموع أزمنة التنفيذ لكل ثنائية في الحل المقترح (الدولة country)، بفرض أن عدد الثنائيات k يكون زمن التنفيذ الكلي:

$$\text{Makespan} = \sum_{i=1}^{k} ETCi$$

يمثل ETC الزمن المتوقع لتنفيذ مهمة (Cloudlet) على جهاز افتراضي (VM)

$$ETC(i,j) = \frac{Length\ of\ cloudlet(i)}{MIPS\ of\ VM(j)} \quad \text{(second * } 10^{-6}\text{)}$$

إن الحل النهائي الذي تتوصل إليه الخوارزمية هو تشكيلة من الثنائيات (دولة country) تحقق أصغر زمن تنفيذ Makespan، وتكون الدولة في هذه الحالة هي الإمبريالي في الإمبراطورية، وذلك عند الوصول لإمبراطورية واحدة وعندها تتوقف الخوارزمية.

في كل عمليات التبديل والتعديل على الحل، مايتم تعديله هو الجهاز الافتراضي الذي سينفذ المهمة، أي المسقط الثاني من الثنائية فقط.

دخل الخوارزمية هو عدد الإمبراطوريات N_{imp} وعدد الدول N_{pop}

7 الخوارزمية المقترحة في جدولة المهام ICA-SOS

بعد دراسة كل من الخوارزميتين كلٌّ على حدة، وجدنا أن خوارزمية SOS تتقارب من الحل الأمثلي بشكل أسرع حيث تصل إلى حل أمثلي بعد عدد محدد من التكرارات، ولاتصل لحل أفضل حتى لو

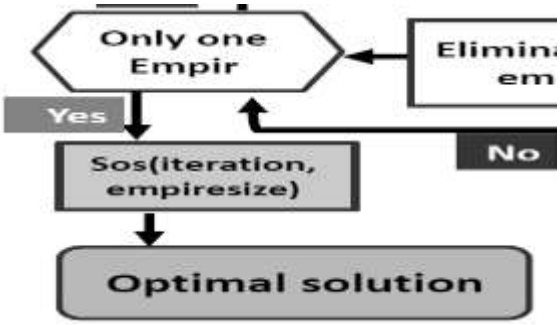
ازداد عدد التكرارات عن هذا الحد، وبالمقابل فإن خوارزمية ICA تأخذ وقتاً لإنهاء عمليات التنافس الإمبريالي ولاسيما عندما تكون الإمبراطوريات متقاربة في طاقتها.

مما سبق يمكننا أن نتوقع بأن إدراج خوارزمية SOS في مرحلة تقارب المستعمرة من الإمبريالي ضمن خوارزمية ICA، على اعتبار أن الإمبراطورية هي فضاء حلول أولي لخوارزمية SOS فإن هذا الإدراج سيحسن من خوارزمية ICA.

إن إدراج خوارزمية SOS بعد مرحلة تهيئة الإمبراطوريات وقبل عملية تقارب المستعمرة من الإمبريالي يساعد على تجهيز امبراطورية محسّنة أساساً قبل أن يتم اختبار تقارب المستعمرات، بحيث يكون دخل Input الخوارزمية التكافلية SOS هو عدد التكرارات Iteration وهو مقدار يتم تحديده عند البداية كدخل للخوارزمية المقترحة، أما عدد الكائنات (مقدار الدخل الثاني للخوارزمية SOS) هو عدد الدول في الإمبراطورية (empireSize) التي سيتم تطبيق الخوارزمية التكافلية SOS عليها.

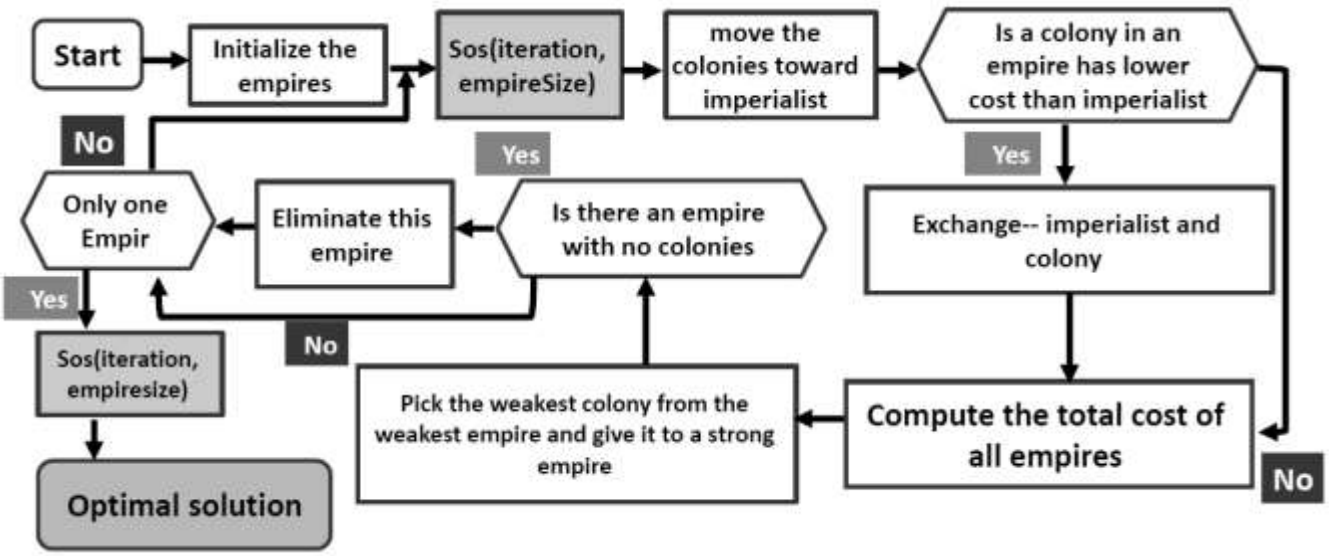
بالتالي قد لاتحتاج الإمبراطورية المحسّنة إلى عمليات تبديل بين المستعمرات والإمبريالي، وهذا يؤدي إلى توفير وقت في البحث بكل خوارزمية، يبقى هذا الوقت أكبر من الوقت الذي استهلكته خوارزمية SOS التي تم حقنها لأنها تتقارب بشكل سريع من الحل، وهنا في حالة الدمج لم نكرر SOS أكثر من ثلاث مرات –كما ستظهر النتائج– في كل عملية حقن.

كما أن الإمبراطورية الأخيرة الوحيدة التي تصل إليها خوارزمية ICA يمكن عدُّها فضاء حلول أولي لخوارزمية SOS.



حيث تم حقن خوارزمية SOS بعد التوصل لإمبراطورية واحدة وقبل التوقف لاختيار الحل الأمثلي Optimal Solution بحيث يكون عدد التكرارات Iteration لخوارزمية SOS هو عدد التكرارات الذي تم تحديده عند البدء كدخل للخوارزمية المقترحة ICA-SOS، أما عدد الكائنات هو عدد الدول في الإمبراطورية الأخيرة التي بقيت بعد انتهاء عمليات المنافسة. إن الحل الأمثلي Optimal Solution هو الدولة (مجموعة الثنائيات) التي تمثل بالنسبة لخوارزمية SOS الكائن الحي الذي يمتلك أعلى درجة تكيف بالنسبة لزمن التنفيذ، بحيث يكون زمن تنفيذ هذه التشكيلة من الثنائيات أصغر ما يمكن.

إذاً يمكن صياغة خطوات الخوارزمية الجديدة ICA-SOS كما يوضح الشكل 7-1.



الشكل 7-1 مخطط الخوارزمية الهجينة ICA-SOS

دخل الخوارزمية ICA-SOS هو عدد التكرارات وعدد الإمبراطوريات وعدد الدول.

يتم تلخيص الخوارزمية الهجينة ICA-SOS بالخطوات الرئيسية التالية:

1. تهيئة الإمبراطوريات وتوزيع المستعمرات عليها.
2. تطبيق مراحل SOS على كل إمبراطورية بعدد تكرارات يتم تحديده كدخل للخوارزمية.
3. تبديل مواقع المستعمرة والإمبريالي (Exchanging).
4. حساب الطاقة الكلية للإمبراطورية.
5. التنافس الإمبريالي (Imperialistic competition).
6. إهمال الإمبراطورية التي لاتمتلك مستعمرات وإلا العودة للخطوة 2
7. توقف الخوارزمية عند الوصول لإمبراطورية واحدة.
8. تطبيق مراحل SOS على الإمبراطورية الأخيرة بعدد تكرارات يتم تحديده عند البدء.

8 التطبيق العملي

تم استخدام بيئة المحاكاة cloudsim وتضمين المكتبات الخاصة بها على بيئة التطوير Netbeans IDE 8.2، تم كتابة الكود البرمجي بلغة Java version 1.8.0_251

تم تهيئة بيئة المحاكاة بالقيم الموضحة في الجدول 8-1 والجدول 8-2 والجدول 8-3 قبل البدء بتجريب الخوارزميات وتسجيل النتائج:

مراكز البيانات Datacenters:

الجدول 8-1 مواصفات مراكز البيانات في بيئة المحاكاة

عدد المضيفين في كل مركز Hosts	سرعة تنفيذ التعليمات Million Instructions Per Second MIPS
2	1000000

المضيفون الفيزيائيون Hosts:

الجدول 8-2 مواصفات كل مضيف فيزيائي في مركز البيانات

رقم تسلسلي للمضيف Host	الذاكرة RAM (GB)	التخزين Storage (TB)	التردد Bandwidth (GB/s)	طريقة جدولة الآلات الإفتراضية VM Scheduling	هرمية النظام system architecture	نظام التشغيل operating system	عدد النوى Cores
1	20	1	10	time-shared	x86	Linux	2
2	20	1	10	time-shared	x86	Linux	4

الآلات الافتراضية Virtual Machines:

الجدول 8-3 مواصفات كل آلة افتراضية في مركز البيانات

الذاكرة RAM (GB)	التردد Bandwidth (GB/s)	عدد المعالجات	سرعة تنفيذ التعليمات MIPS
0.5	1	1	من 1000 إلى 10000

تم تجريب الخوارزميات على مسألتين:

المسألة الأولى: جدولة 1000 مهمة على 125 جهاز افتراضي

المهام التي سيتم جدولتها ليست متساوية جميعها بالطول، كما يوضح الجدول 8-4:

الجدول 8-4 الأرقام المعرفة للمهام وأطوال المهام.

Cloudlet_ID	0	1	2	3	4	5	...	24	25	26	...	999
Length	1000	1100	1200	1300	1400	1500	...	3400	1000	1100	...	3400

وكذلك الأجهزة الإفتراضية VMs ليست متساوية جميعها بعدد التعليمات التي تنفذها بالثانية (مضروباً بمليون) كما يوضح الجدول 8-5:

الجدول 8-5 الأرقام المعرفة للأجهزة الإفتراضية وعدد التعليمات.

VM_ID	0	1	2	3	4	5	6	7	8	9	10	11	...	124
MIPS	50	150	250	350	450	550	650	750	850	950	50	150	...	450

<u>الحل باستخدام خوارزمية ICA</u> دخل الخوارزمية: عدد الدول Countries وعدد الإمبراطوريات Empires

يوضح الجدول 8-6 نتائج تنفيذ الخوارزمية.

الجدول 8-6 نتائج الخوارزمية ICA لحل المسألة الأولى

Countries (input)	Empires (input)	Competitive count (عدد مرات الاستيلاء على مستعمرة- حجم المنافسة)	Makespan (ms)
10	3	151	4.259
10	3	610	3.989
10	3	623	4.014
10	3	141	3.955
10	3	43	5.501
10	3	721	3.116
10	4	149	4.356
10	4	111	5.491
10	4	127	4.979
10	4	163	5.311
10	4	174	4.080
10	5	54	4.899
10	5	9	6.637
10	5	115	4.972
10	5	79	6.188
10	5	43	5.788

بملاحظة قيم الجدول 8-6 نجد بأن زيادة حجم المنافسة يزيد من تحسن الحل أي يعطينا زمن تنفيذ أفضل (أصغر) ومن هنا نقول بأن كل ما كانت المنافسة كبيرة كل ما كان الحل أفضل، وذلك بفرض عدم الوقوع بفخ الحل الأمثلي المحلي Local Optima لأن هذه الخوارزمية ICA قد تقع بفخ الحل الأمثلي المحلي، لكنها تستطيع الخروج منه بسبب كونها من الخوارزميات فائقة الاستدلال وتعتمد على قيم عشوائية.

إن حجم المنافسة هو مقدار تم حسابه بشكل برمجي بعد تشغيل الخوارزمية حيث يتم زيادة عداد يبدأ من الصفر بمقدار واحد عند كل عملية استيلاء على مستعمرة تتبع لإمبراطورية ضعيفة من قبل

إمبراطورية قوية، وعندما تتوقف الخوارزمية يكون قيمة العداد هي قيمة حجم المنافسة المسجلة في الجداول.

الحل باستخدام الخوارزمية الهجينة ICA-SOS

دخل الخوارزمية: عدد الدول Countries وعدد الإمبراطوريات Empires وعدد التكرارات Iterations لخوارزمية SOS ضمن مرحلة تقارب المستعمرة وفي الإمبراطورية الأخيرة.

يوضح الجدول 8–7 نتائج تنفيذ الخوارزمية.

الجدول 8–7 نتائج الخوارزمية ICA-SOS لحل المسألة الأولى

Countries (input)	Empires (input)	Iterations (input)	Competitive count (عدد مرات الاستيلاء على مستعمرة– حجم المنافسة)	Makespan (ms)
10	3	1	11	3.752
10	3	3	329	2.316
10	3	5	545	2.316
10	4	1	81	2.583
10	4	3	13	3.884
10	4	3	22	2.981
10	4	3	8	2.524
10	4	5	19	2.559
15	4	2	4034	2.316

بملاحظة نتائج الخوارزمية الجديدة نجد أنها استطاعت الوصول لحل أمثلي يقارب ال 2.316 ms لجدولة 1000 مهمة على 125 جهازاً افتراضياً ولم تصل إلى حل أفضل منه مع تغيير عدد التكرارات وعدد الإمبراطوريات وعدد المدن، كما نلاحظ أن الخوارزمية وصلت لهذا الحل بعدد

تكرارات لم يتجاوز ثلاثة تكرارات فقط في كل مرة تم حقن خوارزمية SOS ضمن مرحلة من مراحل خوارزمية ICA، كما أن زيادة حجم المنافسة يعطينا تحسن إضافي للوصول للحل النهائي

تجميع النتائج السابقة لحساب نسبة التحسين في الجدول 8-8:

الجدول 8-8 نسبة التحسين في خوارزمية ICA للمسألة الأولى

ICA_Makespane Empires=3,Countries=10 (ms)	ICA_SOS_Makespane Empires=3,Countries=10, Iterations=3	ICA Optimization
3.116	2.316	(100 – (2.316/3.116)*100) = 25.67 %

إن التحسن على أداء ICA يدل على مدى الاستفادة من تقارب SOS من الحل، فإذا اعتبرنا الإمبراطورية الأخيرة التي توقفت عندها ICA هي فضاء حلول عشوائي لخوارزمية SOS أي هي أول خطوة من خوارزمية SOS فإننا نأخذ هذا الفضاء (المحسّن بالأساس بسبب عمليات المنافسة) ونحسنه باستخدام SOS وبعدد تكرارات لم يتجاوز ثلاثة تكرارات.

المسألة الثانية: جدولة عدد من المهام (تبدأ ب 100 وتصل إلى 1000 مهمة) على 20 جهازاً افتراضياً

تم توليد أطوال المهام بشكل عشوائي بين القيمتين 1000 و 10000 ثم اعتماد هذه العينة من أطوال المهام واختبار الخوارزميات عليها.

<u>الحل باستخدام خوارزمية ICA</u>

تم تطبيق المعايير المستخدمة في الورقة البحثية [2] وتنفيذ الخوارزمية ICA التي تم نمذجتها في هذا البحث، وتم التوصل للنتائج الموضحة في

الجدول ***8-9*** .

الجدول 8-9 نتائج خوارزمية ICA لحل المسألة الثالثة

Countries (input)	Empires (input)	Cloudlets	Makespan ($s * 10^{-6}$)	Competitive count (عدد مرات الاستيلاء على مستعمرة- حجم المنافسة)
10	3	100	67.22	969
10	3	200	138.60	854
10	3	300	215.89	429
10	3	400	254.64	781
10	3	500	377.58	463
10	3	600	515.92	455
10	3	700	626.14	170
10	3	700	488.50	467
10	3	700	445.07	587
10	3	700	583.18	316
10	3	800	559.76	197
10	3	800	594.90	133
10	3	800	682.43	117
10	3	900	570.01	321
10	3	900	552.41	400
10	3	1000	703.40	569
10	3	1000	768.03	172
10	3	1000	705.57	328
10	3	1000	641.07	1310

في

الجدول ***8-9*** تم إعادة تشغيل الخوارزمية أكثر من مرة وسجلنا في كل مرة قيمة زمن التنفيذ Makespan وحجم المنافسة وذلك عند عدد المهام 700 و800 و900 و1000 بهدف اختبار مدى كفاءة الخوارزمية عند عدد المهام الكبير، لاحظنا زيادة تحسن الحل مع ازدياد حجم المنافسة.

الحل باستخدام خوارزمية ICA-SOS

باستخدام المعايير نفسها توصلنا للنتائج التالية:

الجدول 8-10 نتائج خوارزمية ICA-SOS لحل المسألة الثالثة

Countries (input)	Empires (input)	Iterations (input)	Cloudlets	Makespan ($s * 10^{-6}$)	Competitive count (عدد مرات الاستيلاء على مستعمرة- حجم المنافسة)
10	3	3	100	48.73	114
10	3	3	200	99.96	47
10	3	3	300	153.70	32
10	3	3	400	209.94	18
10	3	3	500	268.67	60
10	3	3	600	329.91	78
10	3	3	700	370.82	86
10	3	3	800	413.62	41
10	3	3	900	458.28	130
10	3	3	1000	504.83	182

في الجدول 8–10 نلاحظ بأن الخوارزمية الهجينة ICA-SOS لم تتطلب حجم منافسة كبير – مقارنة مع خوارزمية ICA وحدها– كي تصل لحل أمثلي أفضل من الحل الذي توصلت له خوارزمية ICA وحدها، ومن هنا يمكن القول بأنه إضافة لتحسن الحل، تمكنت الخوارزمية الهجينة من توفير الجهد في عمليات المنافسة وأيضاً الإبتعاد عن الوقوع بفخ الحل الأمثلي المحلي Local Optima

نسب التحسين في خوارزمية ICA.

الجدول 8– 11نسب التحسين في خوارزمية ICA للمسألة الثالثة

Cloudlets	ICA Makespan ($s * 10^{-6}$)	ICA Competitive count	ICA-SOS Makespan ($s * 10^{-6}$)	ICA-SOS Competitive count	ICA Optimization
100	67.22	969	48.73	114	27.51%
200	138.60	854	99.96	47	27.88 %
300	215.89	429	153.70	32	28.81 %
400	254.64	781	209.94	18	17.55 %
500	377.58	463	268.67	60	28.84 %
600	515.92	455	329.91	78	36.05 %
700	445.07	587	370.82	86	16.68 %
800	594.90	133	413.62	41	30.47 %
900	552.41	400	458.28	130	17.04 %
1000	641.07	1310	504.83	182	21.25 %
					Average = 25.21 %

تم في الجدول 8–11 تجميع نتائج تحسّن زمن التنفيذ Makespan باستخدام الخوارزمية الهجينة ICA-SOS مقارنة بخوارزمية ICA مع اختلاف عدد المهام، نلاحظ أن نسبة التحسن تبقى أكبر

من 20% عند عدد المهام الكبير ومع ازدياد حجم المنافسة، وأن الخوارزمية الهجينة لم تتطلب حجم منافسة كبير، وهذا يدلنا على تأثير خوارزمية SOS في الحد من عمليات المنافسة المطلوبة لتوقف الخوارزمية.

9 النتائج والآفاق المستقبلية

تم في هذا البحث اقتراح خوارزمية جدولة جديدة ونمذجتها بالإعتماد على خوارزميتين من خوارزميات الأمثَلة العددية فائقة الاستدلال Metahuristic هما خوارزمية البحث في الكائنات المتكافلة SOS و خوارزمية التنافس بين المستعمرين ICA حيث تم نمذجة كل من الخوارزميتين في بيئة المحاكاة cloudsim وباستخدام لغة java ثم اختُبرَت الخوارزميتان على مسألة جدولة مجموعة من المهام في بيئة سحابية مكونة من مجموعة من الآلات الافتراضية بحيث يكون زمن التنفيذ أمثلياً.

أعطت الخوارزمية ICA حلاً أمثلياً، إلا أن الخوارزمية المقترحة أعطت حلاً أمثلياً أفضل بنسبة 25% مقارنة بخوارزمية ICA.

البحث العلمي واسع ولا يتوقف حيث يمكن تطبيق الخوارزمية المقترحة في البيئة السحابية لتحسين موازنة الحمل أو تحسين إحدى الخوارزميتين ودمجها مع الأخرى، ثم اختبار الخوارزمية الناتجة في أمثَلة زمن التنفيذ أو موازنة الحمل.

10 المراجع

[1] Cheng, M. Y., & Prayogo, D. (2014). Symbiotic organisms search: a new metaheuristic optimization algorithm. Computers & Structures, 139, 98-112.

[2] Abdullahi, M., & Ngadi, M. A. (2016). Symbiotic organism search optimization based task scheduling in cloud computing environment. Future Generation Computer Systems, 56, 640-650.

[3] Arshad, R., & Rafeh, R. (2015, November). Deadline-constrained workflow scheduling using imperialist competitive algorithm on infrastructure as a service clouds. In 2015 2nd International Conference on Knowledge-Based Engineering and Innovation (KBEI) (pp. 835-842). IEEE.

[4] Atashpaz-Gargari, E., & Lucas, C. (2007, September). Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition. In 2007 IEEE congress on evolutionary computation (pp. 4661-4667). Ieee.

[5] Habibi, M., & Navimipour, N. J. (2016). Multi-objective task scheduling in cloud computing using an imperialist competitive algorithm. IJACSA) International Journal of Advanced Computer Science and Applications, 7(5), 289-293.

[6] Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A., & Buyya, R. (2011). CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Software: Practice and experience, 41(1), 23-50.

[7] Li, Y. (2019). ACO-SOS-based task scheduling in cloud computing. International Journal of Performability Engineering, 15(9), 2534.

[8] Li, Y., & Yao, Y. (2019). Scheduling Algorithm for a Task under Cloud Computing. International Journal of Performability Engineering, 15(8), 2081.

[9] "Main Categories of Optimization Algorithms." 1/5/2022. https://www.al-roomi.org/.

[10] "Optimization."1/5/2022. https://www.al-roomi.org/.

[11] Patidar, S., Rane, D., & Jain, P. (2012, January). A survey paper on cloud computing. In 2012 second international conference on advanced computing & communication technologies (pp. 394-398). IEEE.

[12] Sharma, M., & Verma, A. (2017, February). Energy-aware discrete symbiotic organism search optimization algorithm for task scheduling in a cloud environment. In 2017 4th International Conference on Signal Processing and Integrated Networks (SPIN) (pp. 513-518). IEEE.

[13] Miao, F., Zhou, Y., & Luo, Q. (2019). Complex-valued encoding symbiotic organisms search algorithm for global optimization. Knowledge and Information Systems, 58(1), 209-248.

[14] Arshad, R., & Rafeh, R. (2015, November). Deadline-constrained workflow scheduling using imperialist competitive algorithm on infrastructure as a service clouds. In 2015 2nd International Conference on Knowledge-Based Engineering and Innovation (KBEI) (pp. 835-842). IEEE.

[15] Kashikolaei, S. M. G., Hosseinabadi, A. A. R., Saemi, B., Shareh, M. B., Sangaiah, A. K., & Bian, G. B. (2020). An enhancement of task scheduling in cloud computing based on imperialist competitive algorithm and firefly algorithm. The Journal of Supercomputing, 76(8), 6302-6329.

[16] Ruder, S. (2016). An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747.