Comparación de algoritmos meta-heurísticos para la programación de tareas en la nube

Ruben T. Lima[1] y Carlos V. Rivero[2]

***Abstract*— In cloud computing, task scheduling is one of the most difficult problems to overcome properly and there exists many approaches that can solve this problem, but to find the best solution and improve the datacenter flow. In this paper, we add three algoritms based on ant colony system methods to compare with TSDQ-FLPSO and TSDQ-SAPSO algorithms.**

I. Introducción

La computación en la nube es un área nueva que nos provee de diversas facilidades como el alquiler de un servidor o servidores de manera rápida para que el desarrollo de un servicio web o una aplicación en la nube. La computación en la nube usa diversos recursos dinámicos como centros de datos de manera distribuida o otros recursos computacionales interconectados. Uno de los problemas principales para mejorar el rendimiento es la programación de datos de tal forma que se mejora la calidad del servicio para el cliente (la rapidez al ser atendidos), el tiempo de respuesta en el sistema de cada tarea y el uso apropiado de recursos para optimizar el sistema. La programación de tareas es un problema difícil de resolver(NP-Completo), ya que se cuenta con diversas variables como las características de cada tarea y el sistema donde se ejecuta, el cual presenta capacidades heterogéneas entre sus servidores.

Entre las soluciones a la programación de tareas, encontramos el uso de algoritmos meta-heurísticos, los cuales nos proveen de un acercamiento a una posible solución, ya que se basan en supuestos aleatorios para llevar a una solución aceptable según las heurísticas utilizadas. Las siguientes secciones se dividen en lo siguiente: segunda sección, define lo algoritmos utilizados en todo el proceso; tercera sección, la metodología o funcionamiento del sistema; cuarta sección, experimentación y comparación de resultados; quinta sección, conclusiones del proyecto; sexta sección, trabajos futuros.

II. Algoritmos

II-A. Algoritmos de Colonias de Hormigas

II-A.1. Sistema de colonia de hormigas: Ant System (AS) se basa en el comportamiento colectivo de las hormigas en la búsqueda de alimentos para subsistir.

Resulta fascinante entender como animales casi ciegos, moviéndose aproximadamente al azar, pueden encontrar el camino más corto desde su nido hasta la fuente de alimentos y regresar. Para esto, cuando una hormiga se mueve,deja una señal odorífera, depositando una substancia denominada feromona, para que las demás puedan seguirla.

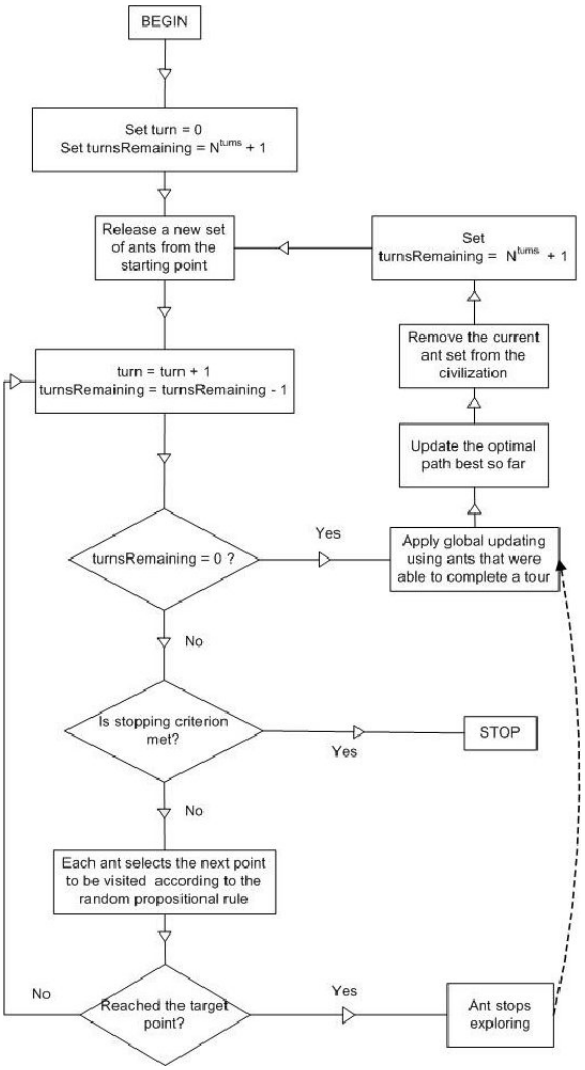

Fig. 1. Ant System algoritmo

II-A.2. Sistema de hormigas elitistas: Es la primera mejora de *Ant System* presentada por Doringo [1].

La idea esta basada en mejorar los pesos dados para los caminos que recorrerán las hormigas. Además la hormiga que tenga mejor recorrido llega a depositar mayor feromona en cada iteración. En cuanto a eficacia presenta mejores resultados que AS simple.

Se lo dedicamos a nuestro querido profesor Alvaro M. Aliaga

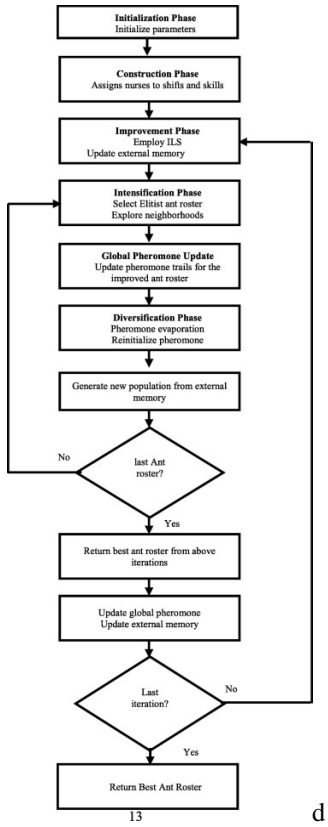

Fig. 2. Ant System Elitist algoritmo

II-A.3. Sistema de hormigas basadas en rankings: Ant System Ranking fue propuesta por Bullnhemiern [2].
La idea principal era que cada hormiga deposita una cantidad de feromona la cual va disminuyendo con su ranking. Además al igual que ASe la mejor hormiga deposita mayor cantidad de feromona en cada iteración

II-B. Algoritmo WTO-PSO

Waiting time Optimization Algorithm - PSO es un algoritmo el cual esta realizado para mejorar los tiempos de espera según el tipo de entrada en este caso es un algoritmo PSO modificado para poder dar el mejor tiempo de espera para las diferentes tareas de entrada.

II-C. Optimización por enjambre de partículas

Un algoritmos Basado en Cumulo de Partículas o Particle Swarm Optimization es una técnica meta-heurística basada en poblaciones e inspirada en el comportamiento del vuelo de aves y el movimiento de los bancos de peces.
El algoritmo analiza la interrelación entre los individuos con los integrantes de los grupos.
PSO fue originalmente desarrollado por el psicólogo sociologo Jammes kennedy y por el ingeniero electrónico Russell Eberhart en 1995 [3].
En el algoritmo cada individuo puede modificar su propia opinión basándose en tres factores:

- Su conocimiento sobre el entorno.
- Su conocimiento histórico o experiencias anteriores.
- El conocimiento histórico o experiencias anteriores de los vecinos.

El algoritmo PSO explica el espacio de soluciones y encuentra soluciones de buena calidad. Eventualmente encuentra el óptimo del problema.

II-D. Algoritmo TSDQ

Dynamic Decision Queues Algorithm(TSDQ) es un algoritmo usado que realiza decisiones con los mejores caminos posibles para crear un óptimo número de colas, lo que garantiza el schedule de tareas eficientemente.
Para estos casos se considera el máximo de colas generadas como el número de número de máquinas virtuales.

II-E. Algoritmo SAPSO

Es un algoritmo optimizado de PSO basado en Self-Adaptative algorithm, en el cual tanto las posiciones y las velocidades de las partículas son modificadas para tener un gran numero de posiciones o metrópolis de posiciones las cuales serán escogidas mediante un método de reemplazo. [4]

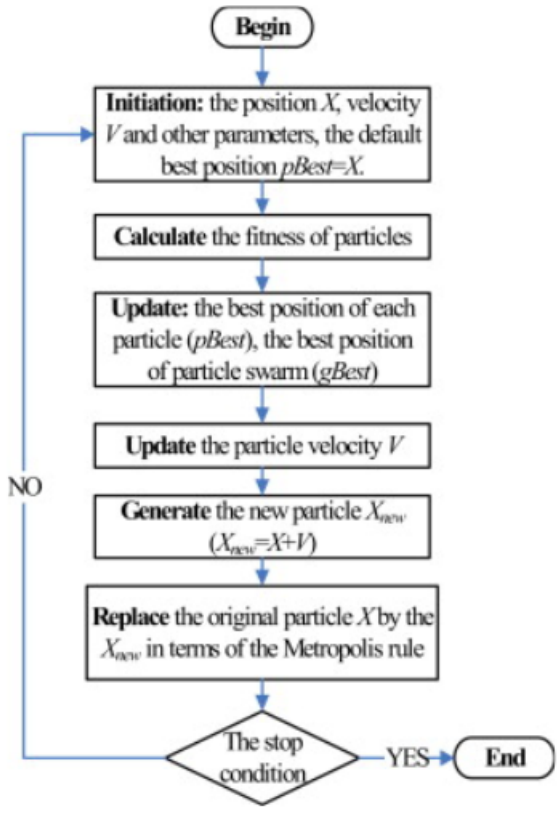

Fig. 3. SAPSO algoritmo

II-F. Algoritmo FLPSO

FLPSO es un algoritmo basado en lógica difusa en el cual los pesos de los valores serán calculados mediante esta lógica a su vez el fitness también es calculado de la misma manera, para Fuzzy logic es necesario discretizar los datos obtenidos

en este caso los datos están normalizados para tener una distribución de datos mas exacta. [5]

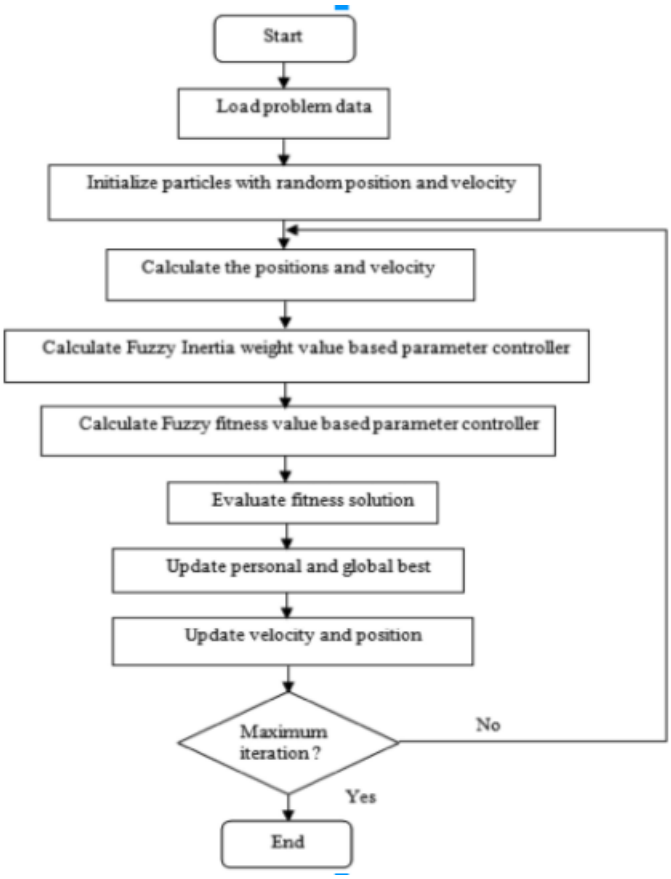

Fig. 4. FLPSO algoritmo

III. Metodología

El sistema actual propuesto para la resolución de problemas esta compuesto de tres fases principales, todas ellas orientadas a ofrecer un mejor servicio en la nube de parte del proveedor. La primera consiste en el ordenamiento de tareas para reducir el tiempo de espera de los clientes, por lo cual utilizamos un algoritmo para la optimización del tiempo de espera5. En esta etapa se considera la cantidad de instrucciones por cada trabajo como factor principal en el ordenamiento, de tal manera que esto ayude a mejorar la calidad del servicio, reducir el tiempo de espera y mejorar el rendimiento en las siguientes fases del proceso.

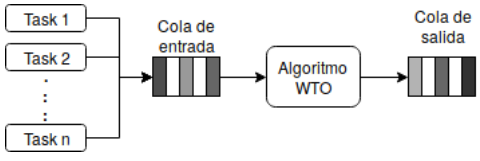

Fig. 5. Algoritmo para la optimización del tiempo de espera

Para la segunda fase, luego de obtener una cola ordenada, se consideran las características del sistema al cual se van a dirigir estas tareas, por la cuál se crean colas independientes de manera que se genera una posible estrategia diferenciando cada cola por un limite específico y dinámico a través de un algoritmo de despacho dinámico de colas6. El número de colas máximo es igual a la cantidad de máquinas virtuales disponibles.

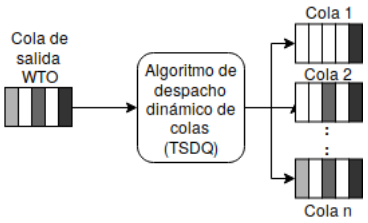

Fig. 6. Algoritmo de despacho dinámico de colas

Finalmente, luego de obtener las colas diferenciadas se utilizan los algoritmos SAPSO o FLPSO para la obtención de una respuesta final. Las fases anteriores sirven para dividir mejor el problema y generar resultados diferentes.

IV. Experimentos y Resultados

Para los experimentos se uso:

- Intel core i7-4700MQ CPU 2.40GHz $\times$ 8
- 12 GB de memoria RAM.
- ubuntu 18.04.1

En los experimentos se realizaron pruebas de makespan, eficiencia y eficacia para cada algoritmo, para cada prueba se usaron los mismos datos de tareas, maquinas virtuales y numero de procesos(en el caso de makespan los resultados son diferentes debido a la dependencia de numero de tareas y el peso de cada una). Los algoritmos implementados fueron:

- Optimización por enjambre de partículas(PSO).
- Colonia de Hormigas(Colonia1).
- Colonia de Hormigas Elitista(Colonia2).
- Colonia de Hormigas por Ranking(Colonia3).

Estos algoritmos fueron usados para facilitar el ordenamiento antes de realizar las colas de tareas. El numero de iteraciones utilizada para cada algoritmos fue de 50. La salida obtenida es usada por el Algoritmo TSDQ el cual realiza la partición de las colas. Las cuales sirven como entradas para los algoritmos FLPSO y SAPSO.

Los resultados obtenidos de makespan están mostrados en las Figuras 7 y 8

En ambas figuras se puede ver que para pocas tareas el makespan de los algoritmos resulta igual, pero cuando el numero de tareas va aumentado el makespan de PSO ya sea con SAPSO o FLPSO tiende a tener resultados menos favorables a diferencia de los algoritmos con colonia de hormigas.

En cuanto a las pruebas de eficiencia presentadas en las Figuras 9 y 10

Según los datos obtenidos el algoritmos colonia de hormigas elitista(Colonia2) tanto con SAPSO como FLPSO resulta teniendo problemas en cuanto a eficiencia se trata por otro lado el algoritmo de colonias Simple presente mejores resultados en cuanto a eficiencia se trata.

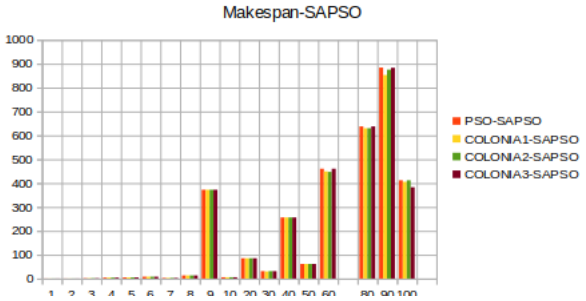

Fig. 7. Makespan de SAPSO usando cada uno de los algoritmos de ordenamiento

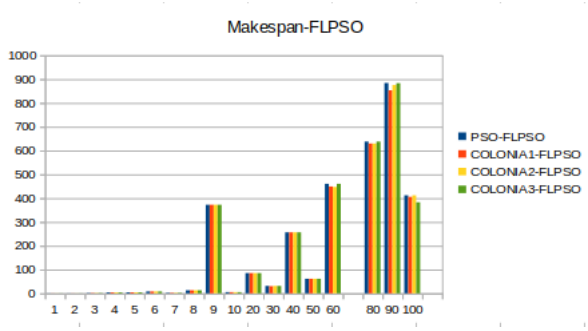

Fig. 8. Makespan de FLPSO usando cada uno de los algoritmos de ordenamiento

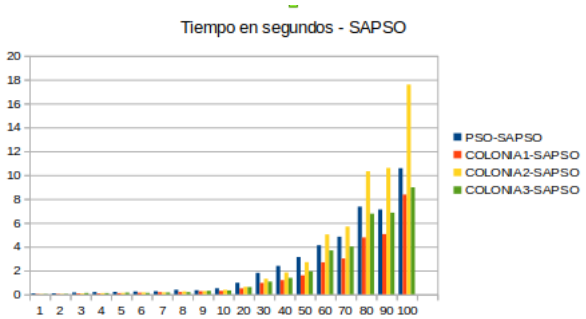

Fig. 9. Pruebas de eficiencia con SAPSO

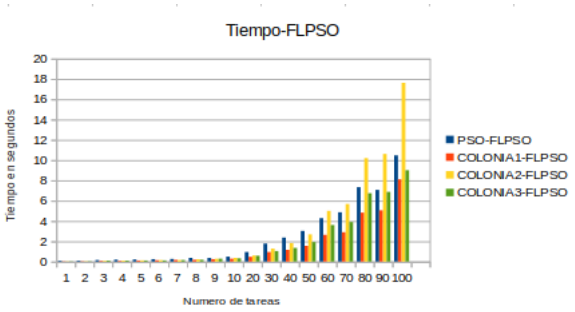

Fig. 10. Pruebas de eficiencia con FLPSO

Para culminar se realizo pruebas de los algoritmos ya planteados para ver cual de ellos mostraba mejores resultados en cuanto al ordenamiento de tareas los cuales se muestran en la figura 11.

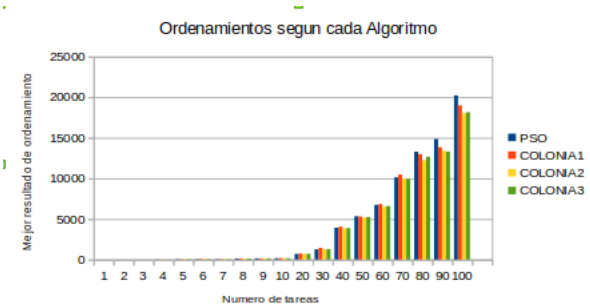

Fig. 11. Resultados de ordenamiento con dependencia al numero de tareas

En los resultados se puede apreciar que el algoritmo de Colonia de Hormigas elitista tiene mejores resultados en cuanto a ordenamiento se refiere contrastado a PSO que en la mitad de los casos llega a tener el resultado menos favorable.

V. Conclusiones

. Como conclusión, al comparar cada uno de los gráficos obtenidos se puede decir que a pesar que el algoritmo de Colonia de Hormigas elitista presente mejores resultados en ordenamiento, este solo es eficaz para casos con pocas tareas, ya sea por que el makespan formado por el mismo es alto o bien por la eficiencia obtenida. A pesar de que el algoritmo de Colonia de Hormigas Simple tiende a ser el peor en eficacia de ordenamiento entre los tipos de colonia, este resulta siendo mas eficaz y al unirse ya sea con SAPSO o FLPSO este presente mejores resultados de makespan ya sea con cantidades de grandes de tarea o cantidades pequeñas.

VI. Trabajos futuros

Como trabajo futuro se tiene pensado en la utilización de paralelismo para mejorar los tiempos de cada uno de los algoritmos, debido a ser algoritmos meta-heurísticos la paralelización de dichos algoritmos resultaría mas sencillos de realizar.

References

[1] Marco Dorigo. Optimization, learning and natural algorithms. *PhD Thesis, Politecnico di Milano*, 1992.

[2] Bernd Bullnheimer, Richard F Hartl, and Christine Strauss. An improved ant system algorithm for thevehicle routing problem. *Annals of operations research*, 89:319–328, 1999.

[3] Russell Eberhart and James Kennedy. A new optimizer using particle swarm theory. In *Micro Machine and Human Science, 1995. MHS'95., Proceedings of the Sixth International Symposium on*, pages 39–43. IEEE, 1995.

[4] Hicham Ben Alla, Said Ben Alla, Abdellah Touhafi, and Abdellah Ezzati. A novel task scheduling approach based on dynamic queues and hybrid meta-heuristic algorithms for cloud computing environment. *Cluster Computing*, 21(4):1797–1820, 2018.

[5] Hicham Ben Alla, Said Ben Alla, Abdellah Ezzati, and Ahmed Mouhsen. A novel architecture with dynamic queues based on fuzzy logic and particle swarm optimization algorithm for task scheduling in cloud computing. *Lecture Notes in Electrical Engineering Advances in Ubiquitous Networking 2*, page 205–217, Apr 2016.