BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

Đoàn Hoà Khiêm

Mô hình hoá điện toán sương mù - đám mây và bài toán lập lịch tác vụ nhiều ràng buộc

Chuyên ngành: Khoa học dữ liệu

LUẬN VĂN THẠC SĨ KHOA HỌC
KHOA HỌC DỮ LIỆU

NGƯỜI HƯỚNG DẪN KHOA HỌC:
PGS. TS. Nguyễn Bình Minh

HÀ NỘI - 2021

Lời cam đoan

Tôi - Đoàn Hoà Khiêm - cam kết luận văn này là công trình nghiên cứu của bản thân tôi, dưới sự hướng dẫn của PGS. TS. Nguyễn Bình Minh. Các kết quả công bố trong báo cáo này là trung thực, không phải là sao chép của bất kỳ một cá nhân, hoặc tổ chức đã được công bố nào khác. Tất cả các trích dẫn được tham chiếu rõ ràng.

Ngày 19 tháng 5 năm 2021

Tác giả luận văn:

Xác nhận của người hướng dẫn

Lời cảm ơn

Đầu tiên, tôi xin được gửi lời cảm ơn chân thành đến các thầy giáo, cô giáo thuộc trường đại học Bách Khoa Hà Nội. Đặc biệt là các thầy giáo, cô giáo thuộc Viện Công nghệ Thông tin và Truyền thông. Chính các thầy cô giáo đã trang bị cho tôi những kiến thức quý báu trong thời gian tôi học tập và nghiên cứu tại trường. Đồng thời tôi cũng xin được gửi lời cảm ơn đặc biệt đến PGS. TS. Nguyễn Bình Minh. Thầy cô là người đã chỉ dẫn tận tình, cho tôi những kinh nghiệm quý báu để tôi có thể hoàn thành luận văn tốt nghiệp này. Thầy cô luôn động viên, giúp đỡ tôi trong những thời điểm khó khăn nhất.

Tôi xin gửi lời cảm ơn tới gia đình và bạn bè. Lời động viên tinh thần từ gia đình và bạn bè luôn là động lực để tôi tiến lên phía trước.

Học viên: Đoàn Hoà Khiêm, mã số CA190045, lớp Khoa học dữ liệu, khoá 2019A.

Tóm tắt nội dung

Điện toán sương mù (fog computing) lần đầu tiên được Cisco định nghĩa vào năm 2015 [1]. Đến nay, điện toán sương mù đã trở thành từ khoá đầy hứa hẹn được cộng đồng tìm kiếm và giới học thuật dành nhiều thời gian nghiên cứu, thử nghiệm. Tuy nhiên công nghệ này vẫn chưa thực sự trở thành một tiêu chuẩn để áp dụng vào vận hành sản xuất. Chính vì vậy đây vẫn còn là đề tài mở cho các cá nhân, tổ chức tham gia nghiên cứu để mô hình này ngày một hoàn thiện hơn. Luận văn mong muốn đóng góp một phần sức lực vào sự nghiệp nghiên cứu và phát triển khoa học, kỹ thuật của Việt Nam nói riêng cũng như toàn thế giới nói chung. Ý tưởng chính là đề xuất mô hình điện toán sương mù - đám mây với nhiều ràng buộc và bài toán lập lịch trên đó. Áp dụng, thử nghiệm các giải thuật metaheuristic nhằm giải bài toán lập lịch và đánh giá mô hình đã đề xuất.

Mục lục

Danh sách hình vẽ

Danh sách bảng

Danh sách từ viết tắt

ACO	Ant Colony Optimization
BLA	Bee Life Algorithm
GA	Genetic Algorithm
GSA	Gravitational Search Algorithm
IoT	Internet of Things
PSO	Particle Swarm Optimization
QoS	Quality of Service
WOA	Whale Optimization Algorithm
RR	Round-Robin

Danh sách các kí hiệu

T	Tập hợp các tác vụ
t_i	Tác vụ thứ i
D	Tập hợp các thiết bị đầu cuối
d_i	Thiết bị thứ i
$P_r^{d_i}$	Lượng dữ liệu được sinh ra tại thiết bị d_i cần được xử lý
$P_s^{d_i}$	Lượng dữ liệu được sinh ra tại thiết bị d_i cần được lưu trữ
$Q_r^{d_i}$	Lượng dữ liệu được sinh ra tại thiết bị d_i cần được xử lý tại đám mây
$Q_s^{d_i}$	Lượng dữ liệu được sinh ra tại thiết bị d_i cần được lưu trữ tại đám mây
$P_r^{d_i} - Q_r^{d_i}$	Lượng dữ liệu được sinh ra tại thiết bị d_i cần được xử lý tại sương mù
$P_s^{d_i} - Q_s^{d_i}$	Lượng dữ liệu được sinh ra tại thiết bị d_i cần được lưu trữ tại sương mù
EXT	Tổng thời gian xử lý tất cả các tác vụ
τ	Thời gian sống của dữ liệu
ψ_{df}	Điện năng truyền tin
ψ_{cp}	Điện năng tính toán
ψ_{st}	Điện năng lưu trữ
δ_{pr}	Độ trễ truyền tin
δ_{pr}	Độ trễ xử lý
θ_{df}	Chi phí truyền tin
θ_{cp}	Chi phí tính toán
θ_{st}	Chi phí lưu trữ

1. Tổng quan

Ngày nay, có hàng tỷ thiết bị được kết nối Internet và điều này dẫn đến thúc đẩy nhiều tiến bộ vượt bậc trong sự phát triển của công nghệ điện tử và viễn thông trong những năm gần đây. Kết quả là cho ra đời nhiều loại thiết bị có khả năng giao tiếp và kết nối rất mạnh mẽ đã thu hút các ngành công nghiệp áp dụng công nghệ vào hoạt động kinh doanh, sản xuất hằng ngày của họ nhằm tăng hiệu suất công việc. Ngoài lĩnh vực công nghiệp còn có các lĩnh vực khác như dịch vụ hỗ trợ sinh hoạt, dịch vụ công cộng, v.v... có nhu cầu lớn về phát triển công nghệ thông tin và truyền thông. Do đó, cần có một mô hình mới trong giao tiếp máy với máy - Machine-to-Machine (M2M) cho phép kết nối vạn vật (Things) vào mạng lưới Internet toàn cầu. Mô hình này được biết đến với thuật ngữ IoT (Internet of Things).

IoT là mạng lưới các vật thể vật lý hay "vạn vật" được nhúng các thiết bị điện tử, phần mềm, các cảm biến và các kết nối cho phép chúng cung cấp các dịch vụ và mang lại nhiều giá trị bằng cách trao đổi dữ liệu với nhà sản xuất, các bên điều hành và/hoặc các thiết bị được kết nối khác thông qua các giao thức truyền thông tin mà không cần có sự vận hành của con người.

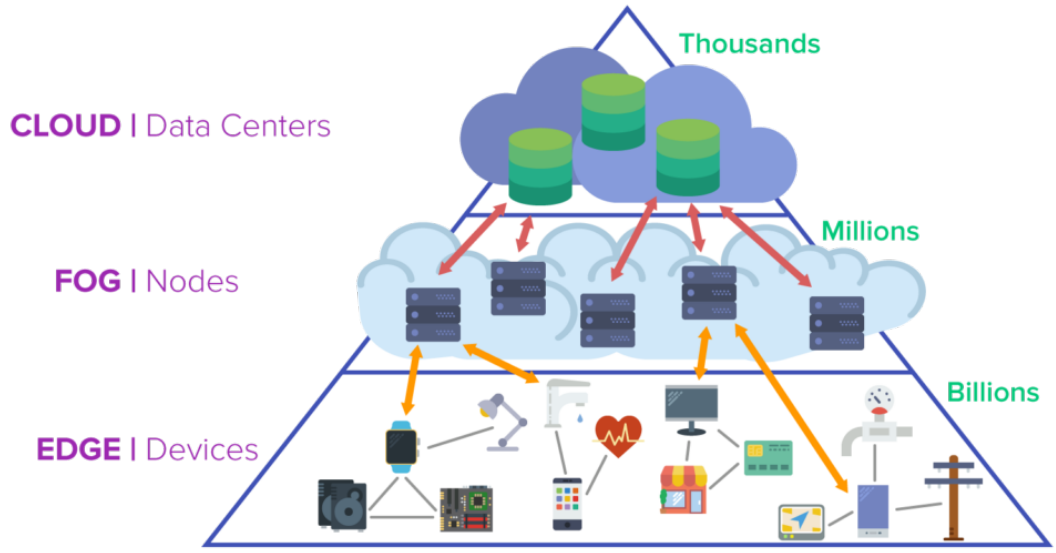
Bằng cách kết nối hàng tỷ, thậm chí hàng nghìn tỷ thiết bị với Internet, chúng ta thấy rằng có rất nhiều ứng dụng đang được sử dụng bởi chính phủ, các ngành công nghiệp hay truy cập công cộng, v.v... Lấy ví dụ ứng dụng hệ thống giao thông thông minh (Intelligent Transport System - ITS) giám sát giao thông trong thành phố bằng các cảm biến không dây hoặc thông qua các video giám sát và gửi thông tin đến người dùng trên thiết bị di động của họ với sự trợ giúp của hệ thống

định vị toàn cầu (GPS) để giúp người dùng phòng tránh kẹt xe và phòng ngừa tai nạn. Đây chỉ là một ví dụ ứng dụng trong số rất rất nhiều ứng dụng khác như nhà thông minh hay chăm sóc sức khoẻ. Một lượng lớn dữ liệu đang được tạo ra bởi hàng tỷ thiết bị được kết nối và truyền qua mạng Internet.

Trong IoT, các thiết bị đầu cuối đóng vai trò quan trọng là các thiết bị thụ cảm môi trường và sinh ra dữ liệu. Tuy nhiên do khả năng tính toán yếu kém của chúng, các thiết bị này thường gửi dữ liệu tới các nút đám mây để lưu trữ, xử lý, phân tích cũng như đưa ra các quyết định điều khiển. Một trong những lợi ích của việc tích hợp này là sự linh hoạt mà người dùng có được khi truy cập các dịch vụ được cung cấp thông qua giao diện web. Điều này cũng mang lại sự linh hoạt cho các nhà cung cấp dịch vụ M2M để dễ dàng đem dịch vụ của mình tới nhiều khách hàng hơn. Điện toán đám mây thường là mô hình cho phép sử dụng mạng thuận tiện, tuỳ theo nhu cầu của người sử dụng về một nhóm tài nguyên máy tính (mạng, máy chủ, lưu trữ, tính toán, ứng dụng, dịch vụ...) một cách nhanh chóng. Tuy nhiên do sự không đồng nhất và kiểm soát lỏng lẻo của Internet, nhiều vấn đề phát sinh đặc biệt là chất lượng dịch vụ khó có thể đảm bảo được. Các ứng dụng thời gian thực có người dùng tương tác trực tiếp chịu ảnh hưởng xấu bởi độ trễ mạng. Một điểm nữa mà điện toán đám mây phải đối mặt là bảo mật và quyền riêng tư. Dữ liệu người dùng nằm trên các đám mây công cộng có nguy cơ bị xâm phạm tính toàn vẹn và tính bảo mật. Các hệ thống đám mây có thể bị tấn công bằng một số phương pháp khác nhau. Như vậy có thể thấy các hệ thống đám mây phải đứng trước nhiều mối đe doạ bảo mật khác nhau do chúng được triển khai trên Internet.

Khắc phục những nhược điểm trên, điện toán sương mù ra đời, được lưu trú tại giữa các thiết bị cuối và đám mây. Các nút sương mù có thể là các switch, gateway, các máy tính nhỏ (mini-computer) hay chính những chiếc điện thoại di động. Chúng có một số chức năng như truyền tải dữ liệu từ các thiết bị đầu cuối lên các đám mây, thu thập, lưu trữ và xử lý dữ liệu đối với các tác vụ yêu cầu được thực hiện trong thời gian thực, có độ trễ thấp. Điện toán sương mù cho phép IoT có tính di động cao với phân bố rộng khắp, thêm vào đó là khả năng nhận biết vị trí và độ trễ thấp. Hơn thế, bằng việc kết hợp cả với các đám mây đã có, ta có được hệ

thống điện toán vô cùng mạnh mẽ, điện toán sương mù - đám mây. Tầng sương mù giúp các hệ thống tiết kiệm băng thông bằng cách giảm thiểu dữ liệu được truyền lên các đám mây. Trong khi đó, các dịch vụ đám mây là những cơ sở hạ tầng máy tính tập trung có khả năng tính toán với hiệu quả cao và lưu trữ khổng lồ. Mặc dù phải đối mặt với vấn đề nghiêm trọng về độ trễ, các đám mây vẫn là thành phần không thể thiếu được nhờ khả năng tính toán cực kỳ mạnh mẽ.



Hình 1.1: Điện toán sương mù - đám mây
(https://cs.uni-paderborn.de/fileadmin/informatik/fg/ce/Teaching/Project_groups/EML.pdf)

Với các thành phần đã được mô tả trên (thiết bị đầu cuối, tầng sương mù và tầng đám mây), việc lập lịch tác vụ cho các yêu cầu xử lý dữ liệu trong các hệ thống sương mù - đám mây trở thành một bài toán thách thức và phức tạp. Lý do chính là sự khác biệt về khả năng tính toán cũng như lưu trữ của các nút sương mù và đám mây. Do sự khác biệt của mỗi loại tính toán, có nhiều yếu tố ảnh hưởng đến thiết kế và hoạt động của môi trường sương mù - đám mây. Ví dụ, các nút sương mù thường có nguồn năng lượng bị giới hạn (được cung cấp bởi pin, ắc quy...); chúng có thể xử lý dữ liệu rất nhanh nhờ khoảng cách ngắn giữa chúng và các thiết bị đầu cuối. Ngược lại, mặc dù các đám mây cung cấp khả năng thực hiện các tác vụ với hiệu năng cao, độ trễ trong việc truyền dữ liệu lại đáng để cân nhắc. Bên cạnh đó, chi phí vận hành, sử dụng dịch vụ là yếu tố rất được người sử dụng và nhà cung cấp quan tâm. Người dùng cần sử dụng các dịch vụ với chi phí nhỏ nhất còn nhà cung cấp dịch vụ không chỉ chi phí vận hành mà còn là sử dụng sao cho phù hợp,

tiết kiệm với hạ tầng hiện có. Theo đó, chiến lược lập lịch trong hệ thống cần phải thoả mãn được nhiều yêu cầu, ràng buộc để hệ thống hoạt động hiệu quả, tối ưu. Bởi vì vậy, bài toán lập lịch là một bài toán quan trọng trong bất kỳ hệ thống điện toán chia sẻ tài nguyên nào.

Luận văn này thực hiện mô hình hoá hệ thống sương mù - đám mây với các ràng buộc khác nhau về môi trường. Theo cách này, luận văn xem xét ba ràng buộc chính gồm: điện năng tiêu thụ, độ trễ dịch vụ và chi phí sử dụng dựa trên các chức năng mà các nút sương mù, đám mây thực hiện (truyền dữ liệu, tính toán, lưu trữ và xử lý). Luận văn cũng định nghĩa một hàm mục tiêu được dùng để tối ưu hoá bài toán lập lịch dựa trên mô hình sương mù - đám mây đã đề xuất. Chúng tôi đã thực hiện một số thử nghiệm với các phương pháp metaheuristic nhằm chứng minh tính đúng đắn của mô hình này. Cùng với đó trình bày mối quan hệ giữa ba ràng buộc trong quá trình tối ưu hoá. Mặc dù có một số nghiên cứu về vấn đề lập lịch tác vụ trong mô trường sương mù - đám mây, vẫn còn chưa có nghiên cứu nào xem xét cả ba ràng buộc nêu trên.

Những kết quả của luận văn chính là những kết quả đã được chấp nhận và công bố tại hội nghị quốc tế IEEE International Symposium on Network Computing and Applications (IEEE NCA) lần thứ 19 vào tháng 11/2020 [2].

Đóng góp của nghiên cứu:

1. Mô hình hoá môi trường nhiều ràng buộc của các hệ thống điện toán sương mù - đám mây gồm điện năng tiêu thụ, độ trễ dịch vụ và chi phí.
2. Đề xuất sử dụng các giải thuật metaheuristic để giải bài toán lập lịch trong mô hình sương mù - đám mây đã đề xuất.
3. Giả lập hệ thống sương mù - đám mây như đã đề xuất để đánh giá các chiến lược lập lịch tác vụ trên đó.
4. Thử nghiệm với một số giải thuật metaheuristic cho bài toán lập lịch trong môi trường nhiều ràng buộc của hệ thống sương mù - đám mây.

Cấu trúc luận văn:

1. Phần đầu là tổng quan, động lực và những đóng góp của luận văn.
2. Phần hai sẽ trình bày các khái niệm, cơ sở lý thuyết, các nghiên cứu liên quan về điện toán đám mây, điện toán sương mù, ưu nhược điểm của chúng cũng như bài toán lập lịch. Đây là nền tảng kiến thức phục vụ cho đồ án.
3. Phần ba đưa ra đề xuất về mô hình điện toán sương mù - đám mây cùng các ràng buộc.
4. Phần bốn sẽ thử nghiệm mô hình đã đề xuất, so sánh, đánh giá các kết quả thu được.
5. Phần năm sẽ đưa ra kết luận cho nghiên cứu, trình bày ưu, nhược điểm của mô hình và đưa ra hướng phát triển trong tương lai.

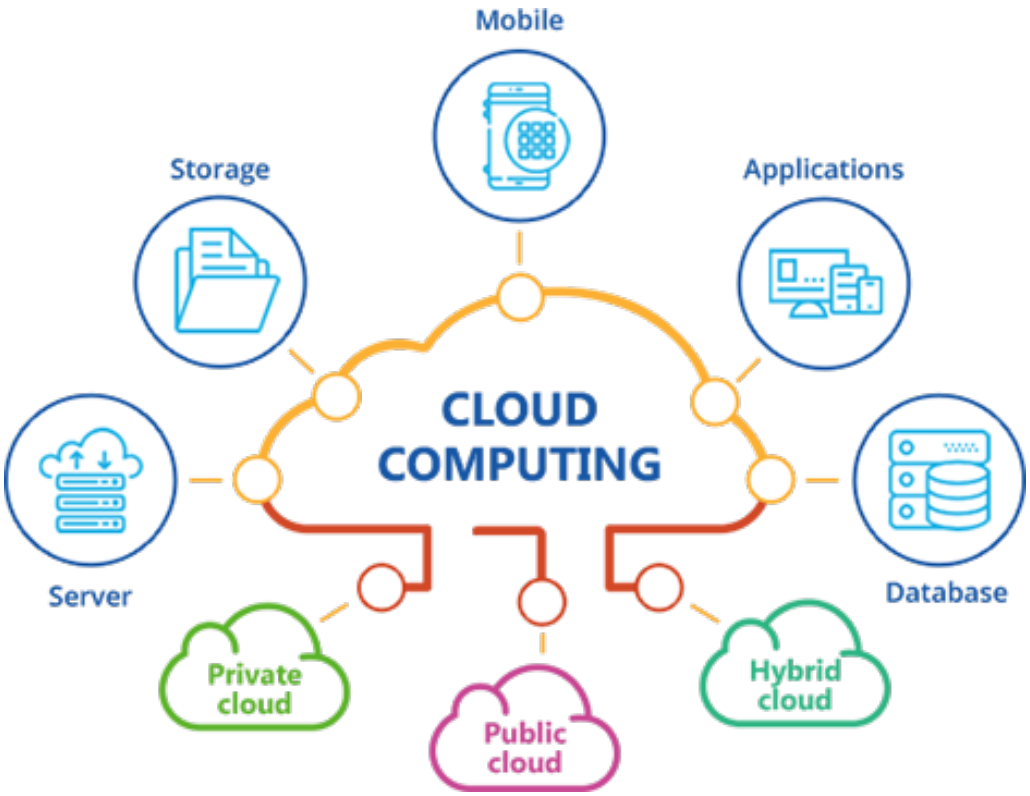
2. Cơ sở lý thuyết và nghiên cứu liên quan

2.1 Điện toán đám mây

2.1.1 Định nghĩa

Thuật ngữ điện toán đám mây (Cloud computing) ra đời từ những năm 2007 với mục đích khái quát lại hướng đi của cơ sở hạ tầng thông tin đã và đang diễn ra từ nhiều năm qua. Theo Viện Tiêu chuẩn và Công nghệ quốc gia Mỹ [1], điện toán đám mây là mô hình cho phép truy cập, sử dụng tài nguyên máy tính và có thể tuỳ chỉnh được (ví dụ như mạng, máy chủ, lưu trữ, các ứng dụng và dịch vụ) tuỳ theo nhu cầu một cách thuận tiện, đồng thời cho phép cung cấp và giải phóng chúng một cách nhanh chóng, giảm thiểu tối đa sự thao tác của quản trị viên. Với các dịch vụ sẵn có trên Internet, doanh nghiệp không phải mua và duy trì hạ tầng cũng như phần mềm mà chỉ cần tập trung vào kinh doanh, nghiệp vụ riêng bởi đã có các nhà cung cấp dịch vụ điện toán đám mây lo cơ sở hạ tầng và công nghệ thông tin thay họ. Đa số người dùng Internet đã tiếp cận những dịch vụ đám mây phổ biến như mạng xã hội, thư điện tử, bản đồ số...

[1] https://csrc.nist.gov/publications/detail/sp/800-145/final



Hình 2.1: Điện toán đám mây

(https://viettelidc.com.vn/tin-tuc/bao-cao-ve-thi-truong-dien-toan-dam-may-tai-viet-nam-nam-2020)

2.1.2 Các đặc trưng

Tính linh hoạt của điện toán đám mây là phân phát tài nguyên tuỳ theo nhu cầu. Điều này tạo ra khả năng mềm dẻo, thuận lợi cho việc sử dụng tài nguyên của hệ thống, loại bỏ sự ràng buộc phải đầu tư phần cứng cụ thể cho mỗi nhiệm vụ. Trước khi có điện toán đám mây, các trang web hoặc ứng dụng được chạy trên một máy chủ cụ thể hoạt động trong một hệ thống. Với sự ra đời của điện toán đám mây, các tài nguyên được hợp nhất và sử dụng như kho chung. Cấu hình hợp nhất này cung cấp một môi trường ở đó các ứng dụng thực hiện một cách độc lập mà không quan tâm đến bất kỳ cấu hình cụ thể nào. Viện Tiêu chuẩn và Công nghệ cũng định nghĩa năm đặc trưng cốt lõi của mô hình điện toán đám mây:

- Dịch vụ cung cấp theo nhu cầu (On-demand self-service): Người dùng tự mua, tự thuê, tự cấu hình triển khai các dịch vụ điện toán đám mây theo các chuẩn định sẵn (template) mà không cần tới sự trợ giúp của bộ phận IT. Để làm được điều này, các nhà cung cấp hạ tầng phải tạo ra các chuẩn định sẵn từ trước.

Các chuẩn này chứa các cấu hình được định nghĩa trước, căn cứ vào đó người dùng sẽ tuỳ chỉnh thêm và cài đặt các dịch vụ theo nhu cầu.

- Truy cập mạng băng thông cao (Broad network access): Tài nguyên tính toán luôn sẵn sàng ở toàn mạng và được truy cập thông qua các chuẩn mạng.

- Tài nguyên không giới hạn (Resource pooling): Nhà cung cấp dự trữ các tài nguyên tính toán để chia sẻ cho nhiều người sử dụng bằng mô hình multi-tenant, tự động cung cấp các tài nguyên vật lý hoặc ảo theo nhu cầu của họ.

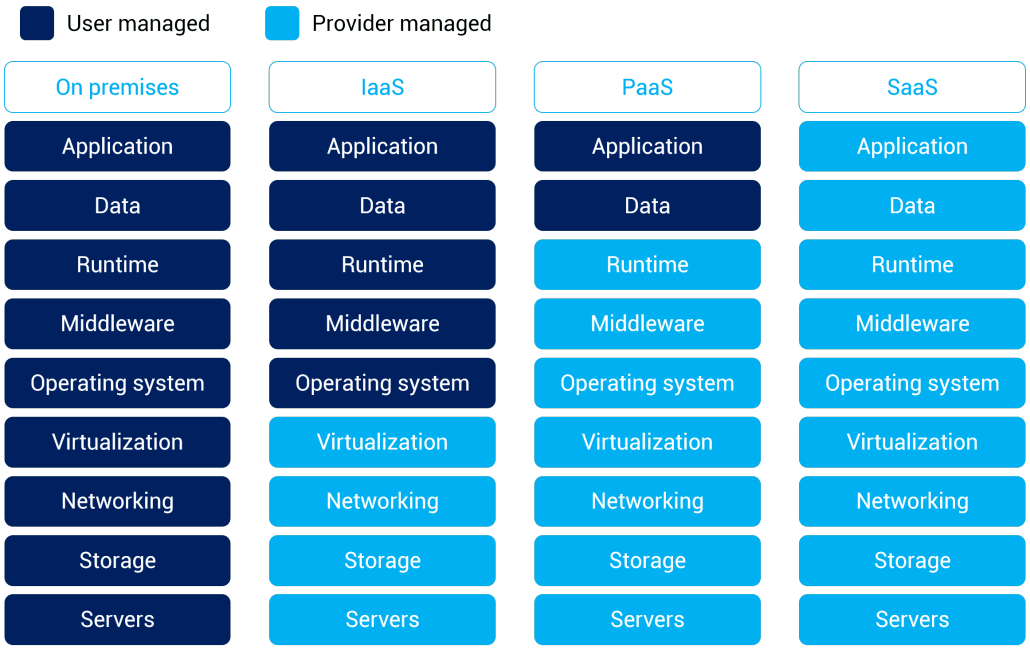
- Cung cấp tài nguyên mềm dẻo (Rapid Elasticity): Tài nguyên tính toán được cung cấp và giải phóng một cách mềm dẻo tự động, tự mở rộng hoặc thu nhỏ lại. Với người sử dụng thì tài nguyên tính toán cung cấp gần như không giới hạn, ở bất cứ nơi đâu và bất cứ lúc nào.

- Dịch vụ đo lường (Measured Service): Các hệ thống điện toán đám mây tự động quản trị và tối ưu tài nguyên sử dụng bằng cách tận dụng năng lực đo đạc ở mức trừu tượng phù hợp với các dịch vụ. Lượng sử dụng tài nguyên được giám sát, điều khiển và thống kê hoàn toàn trong suốt với các nhà cung cấp và người sử dụng.

2.1.3 Mô hình dịch vụ

Dịch vụ điện toán đám mây rất đa dạng và bao gồm tất cả các lớp dịch vụ điện toán từ cung cấp năng lực tính toán trên máy chủ có hiệu suất cao hay các máy chủ ảo, không gian lưu trữ dữ liệu, hay một hệ điều hành, một công cụ lập trình hay một ứng dụng kế toán... Các dịch vụ cũng được phân loại khá đa dạng nhưng các mô hình dịch vụ điện toán đám mây phổ biến nhất có thể được phân thành ba nhóm:

- Hạ tầng như dịch vụ (Infrastructure as a service - IaaS): Đây là mô hình điện toán đám mây cơ bản nhất, các nhà phát hành IaaS cung cấp các hạ tầng tính toán, bộ nhớ lưu trữ được ảo hoá trên nền tảng hypervisor như Xen, Oracle



Hình 2.2: Mô hình dịch vụ điện toán đám mây

(https://www.alibabacloud.com/knowledge/what-is-paas)

Virtualbox, KVM, VMware,... Hệ thống các máy chủ ảo bên trong điện toán đám mây có thể phục vụ một số lượng lớn khách hàng thông qua các máy khách chạy hệ điều hành ảo trên cùng một máy chủ và có khả năng tăng giảm theo nhu cầu thường xuyên của khách hàng. Ngoài ra, IaaS còn cung cấp một số tài nguyên khác như thư viện virtual-image, lưu trữ khối (block storage), lưu trữ đối tượng (object storage), tường lửa ảo, mạng nội bộ ảo (Virtual LAN). Khách hàng sử dụng dịch vụ IaaS thông qua mạng Internet hoặc qua mạng LAN ảo. Để triển khai ứng dụng, người dùng sẽ phải cài đặt hệ điều hành và các ứng dụng của họ lên hệ thống điện toán đám mây. Với mô hình này, người sử dụng tự cập nhật, vá lỗi cho các phần mềm đã cài đặt. Nhà cung cấp dịch vụ sẽ tính tiền dựa trên lượng tài nguyên được cấp và tiêu thụ.

- Nền tảng như dịch vụ (Platform as a service - PaaS): Mô hình PaaS là mô hình mà ở đó nhà cung cấp dịch vụ đưa ra môi trường triển khai ứng dụng cho các nhà lập trình viên bao gồm hệ điều hành, môi trường thực thi ngôn ngữ lập trình, cơ sở dữ liệu và máy chủ web. Các nhà phát triển ứng dụng có thể cài đặt, triển khai giải pháp phần mềm của họ trên nền tảng điện toán đám mây mà không quan tâm tới các chi phí bản quyền hay quản lý các phần mềm bên dưới. Cụ thể với các nhà cung cấp như Microsoft Azure, Google App Engine,

hạ tầng tính toán và lưu trữ được khả mở một cách tự động sao cho phù hợp với nhu cầu của người sử dụng khi mà người dùng điện toán đám mây không phải thao tác một cách thủ công.

- Phần mềm như dịch vụ (Software as a service - SaaS): Trong mô hình SaaS, người sử dụng được truy cập tới ứng dụng và cơ sở dữ liệu. Nhà cung cấp dịch vụ quản lý hạ tầng và nền tảng chạy ứng dụng đó. SaaS hay còn gọi là phần mềm theo nhu cầu và được trả phí theo mức sử dụng hoặc theo đơn hàng. Với mô hình này, nhà cung cấp sẽ cài đặt và quản trị ứng dụng trên môi trường đám mây và người sử dụng sẽ truy cập thông qua giao diện máy khách như trình duyệt web, ứng dụng desktop,... Người dùng không cần phải quản lý hạ tầng đám mây và nền tảng ứng dụng đang chạy. Điều này đã tối thiểu hoá chi phí cài đặt và chạy ứng dụng trên máy tính cá nhân, đồng thời đơn giản hoá các thao tác bảo trì và hỗ trợ. Sự khác biệt giữa ứng dụng điện toán đám mây và ứng dụng truyền thống nằm ở tính chất khả mở. Điều này đạt được nhờ việc tạo ra các bản sao tác vụ chạy trên nhiều máy ảo khác nhau tại cùng một thời điểm để đạt được yêu cầu nghiệp vụ. Các bộ cân bằng tải sẽ phân phối công việc tới các máy chủ ảo, hoàn toàn trong suốt với người sử dụng, vốn dĩ truy cập theo một điểm duy nhất (URL,...). Nhằm phục vụ lượng lớn khách hàng, các ứng dụng điện toán đám mây có thể được multi-tenent, nghĩa là một máy tính sẽ phục vụ nhiều hơn một nhóm người sử dụng.

2.1.4 Mô hình triển khai

Tuỳ thuộc vào hình thức cung cấp và nhu cầu sử dụng mà điện toán đám mây lại được triển khai theo bốn mô hình phổ biến nhất hiện nay:

1. Đám mây nội bộ (Private cloud) là hạ tầng đám mây được điều hành trong nội bộ công ty. Việc triển khai đám mây nội bộ yêu cầu một nguồn lực đáng kể và yêu cầu tổ chức đánh giá một cách kỹ lưỡng các tài nguyên hiện có. Dự án đám mây nội bộ nếu được triển khai đúng hướng sẽ nâng cao nghiệp vụ nhưng ẩn bên trong mỗi bước lại chứa những rủi ro về vấn đề bảo mật cần được giải

quyết để ngăn chặn các lỗ hổng nghiêm trọng. Các trung tâm dữ liệu thường đòi hỏi rất nhiều vốn: chi phí hạ tầng phần cứng, chi phí bảo vệ,... Bên cạnh đó, chúng luôn cần được bảo trì và cập nhật một cách định kỳ, kéo theo một khoản phí bổ sung. Quản lý đám mây nội bộ yêu cầu công cụ phần mềm để giúp cho nhà quản trị dễ dàng kiểm soát, theo dõi và quản lý.

2. Đám mây công cộng (Public cloud) là mô hình các dịch vụ được cung cấp thông qua mạng. Các dịch vụ đám mây công cộng có thể miễn phí. Về mặt lý thuyết thì không có sự khác biệt nào giữa kiến trúc đám mây nội bộ và đám mây công cộng, tuy nhiên, vấn đề bảo mật cần được quan tâm cho các dịch vụ (các ứng dụng, lưu trữ và các tài nguyên khác) sẵn có cung cấp cho người dùng đại chúng và khi kết nối thông qua mạng không an toàn. Thông thường các nhà cung cấp dịch vụ đám mây công cộng như Amazon AWS, Microsoft và Google sở hữu, vận hành cơ sở hạ tầng và truy cập thông qua Internet.

3. Đám mây cộng đồng (Community cloud) chia sẻ hạ tầng giữa một số tổ chức từ một cộng đồng cụ thể có các mối quan tâm chung (ví dụ như một nhóm ngành nghề lớn). Chi phí trải đều trên tập người dùng và ít hơn so với đám mây nội bộ, nhiều hơn so với đám mây công cộng.

4. Đám mây lai (Hybrid cloud) là một tổ hợp của hai hay nhiều đám mây (nội bộ, công cộng hay cộng đồng) nhằm tận dụng lợi thế của các mô hình này đem lại. Công ty Gartner định nghĩa một dịch vụ đám mây lai như dịch vụ điện toán đám mây bao gồm sự kết hợp của các dịch vụ đám mây nội bộ, công cộng từ các nhà cung cấp dịch vụ khác nhau. Mô hình này giúp nhà phát triển ứng dụng dễ dàng mở rộng số lượng cũng như năng lực tính toán của một dịch vụ đám mây nhờ vào việc tích hợp, thống nhất hoặc tuỳ biến các dịch vụ khác nhau. Một ví dụ khác áp dụng đám mây lai khi các tổ chức IT sử dụng đám mây công cộng để đáp ứng nhu cầu tính toán tạm thời khi tài nguyên của họ không đáp ứng đủ. Năng lực này cho phép đám mây lai dễ dàng triển khai ứng dụng lên các đám mây khác nhau. Thuật ngữ "cloud bursting" ám chỉ mô hình triển khai ứng dụng có thể chạy trên đám mây cá nhân hoặc trung tâm dữ liệu và "burst" lên đám mây công cộng khi mà nhu cầu tính toán tăng lên. Ưu điểm

của cloud bursting và mô hình đám mây lai đem lại cho tổ chức có thể chỉ chi trả các tài nguyên phát sinh khi họ cần sử dụng.

2.2 Điện toán sương mù - đám mây

2.2.1 Định nghĩa

Ngày nay, Internet of Things (IoT) hay Mạng lưới vạn vật kết nối Internet đang ngày càng phát triển. Với IoT, việc kết nối Internet được mở rộng từ các thiết bị thông minh truyền thống như điện thoại di động, máy tính bảng cho tới vô số các thiết bị khác như các cảm biến, máy móc hay xe cộ... mang tới rất nhiều các dịch vụ khác nhau như chăm sóc sức khoẻ, y tế, điều khiển giao thông, tự động hoá, quản lý nguồn năng lượng, v.v... Khi mọi thứ kết nối Internet sinh ra một lượng vô cùng lớn dữ liệu và những dữ liệu đó cần phải được lưu trữ, xử lý và phân tích để có được những thông tin hữu ích mà người dùng cần. Tuy nhiên, ngay cả những thiết bị thông minh nhất vẫn không thể xử lý nổi chúng. Điện toán đám mây (cloud computing) được biết đến là những trung tâm dữ liệu lớn bằng việc cung cấp, chia sẻ tài nguyên một cách linh hoạt tới người dùng thông qua các cơ chế ảo hoá được xem là nền tảng vững chắc để hỗ trợ sự phát triển của IoT. Các giới hạn của các thiết bị thông minh như thời lượng pin, sức mạnh xử lý, khả năng lưu trữ được giảm thiểu bằng việc đưa các tác vụ tiêu tốn thời gian và tài nguyên lên các đám mây, các thiết bị IoT chỉ còn cần xử lý các tác vụ đơn giản.

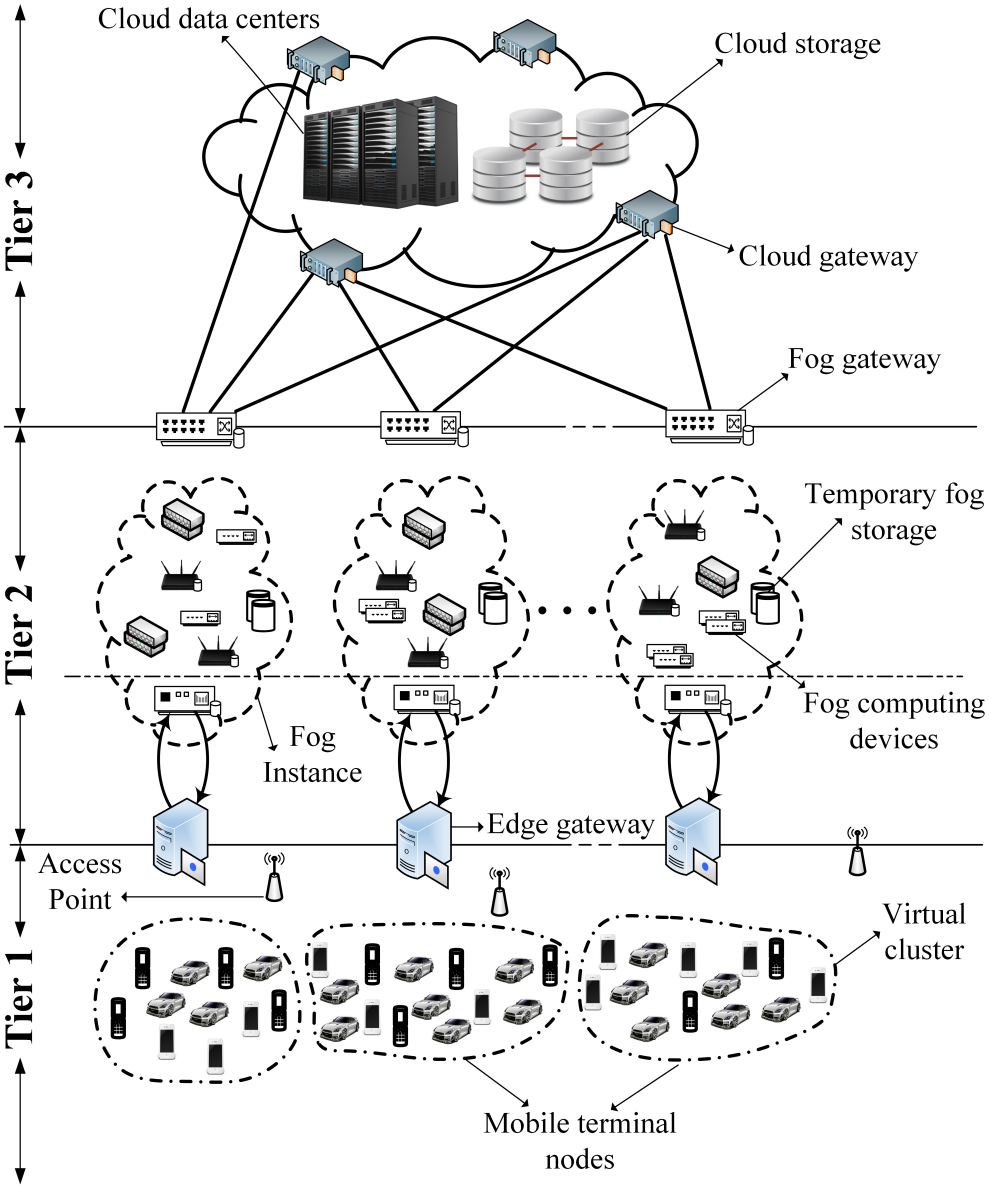
Số lượng các thiết bị IoT đã tăng trưởng 31% mỗi năm lên đến 8,4 tỷ thiết bị vào năm 2017 và theo Gartner dự đoán có khoảng 20,4 tỷ thiết bị kết nối Internet vào năm 2020 [2]. Với sự bùng nổ số lượng thiết bị được kết nối, kiến trúc đám mây với việc xử lý tập trung, tính toán và tài nguyên lưu trữ được thực hiện trên một vài trung tâm dữ liệu lớn sẽ không còn có thể đáp ứng được yêu cầu của các ứng dụng IoT. Hơn thế nữa, các thiết bị IoT thường ở rất xa các đám mây không thể tránh

[2]https://www.gartner.com/en/newsroom/press-releases/2017-02-07-gartner-says-8-billion-connected-things-will-be-in-use-in-2017-up-31-percent-from-2016

khỏi sự tắc nghẽn mạng khi một lượng dữ liệu rất lớn được sinh ra liên tục. Độ trễ đường truyền gây ảnh hưởng nghiêm trọng tới trải nghiệm người dùng. Điều này trở thành động lực cho một ý tưởng mở rộng khả năng tính toán của đám mây tới các biên của mạng (Edge of network).

Kết quả là điện toán sương mù (Fog computing) được đề xuất lần đầu tiên bởi Cisco [1]. Thay vì đưa tất cả các xử lý lên đám mây, nhiều các tác vụ có thể được xử lý ngay gần nơi dữ liệu được sinh ra, trên các nút sương mù. Các dịch vụ được lưu trữ tại biên của mạng hoặc thậm chí là tại các thiết bị đầu cuối như đầu thu truyền hình hay điểm truy cập mạng. Bất kỳ thiết bị nào có khả năng xử lý tính toán, lưu trữ và kết nối tới mạng đều có thể được xem là nút sương mù như là: các thiết bị điều khiển, switch, router, các server nhúng hay camera giám sát an ninh,... Đặc điểm phân biệt của sương mù là nó ra rất gần với người dùng cuối, phân bố địa lý dày đặc và hỗ trợ cả các thiết bị di động. Bằng cách này, sương mù giảm thiểu độ trễ, cải thiện chất lượng dịch vụ, dẫn đến nâng cao trải nghiệm người dùng. Điện toán sương mù hỗ trợ các ứng dụng IoT yêu cầu thời gian thực hay độ trễ có thể dự đoán được (tự động hoá công nghiệp, giao thông vận tải, mạng lưới cảm biến,...).

Việc đặt các tài nguyên tại các biên của mạng giúp giảm thời gian truyền dữ liệu, tuy nhiên, khả năng xử lý của các nút sương mù là có hạn, chỉ những tác vụ nhỏ và nhạy cảm với độ trễ mới được ưu tiên xử lý trên tầng sương mù, các đám mây vẫn đóng vai trò quan trọng cho các tác vụ có thể chấp nhận độ trễ và có quy mô lớn. Cuối cùng, điện toán sương mù cùng với các đám mây kết hợp trở thành một mô hình điện toán mới, điện toán sương mù - đám mây (Fog - cloud computing).



Hình 2.3: Kiến trúc điện toán sương mù - đám mây [3]

Điện toán sương mù - đám mây có nhiều ưu điểm gồm độ trễ thấp, giảm thiểu lưu lượng mạng, hiệu quả về năng lượng, tuy nhiên mô hình mới này cũng có những thách thức. Một trong số chúng là phân bổ tài nguyên và lập lịch cho các tác vụ với nhiều nút xử lý gồm có nút đám mây và nút sương mù. Mục tiêu của việc lập lịch hướng tới lợi ích cho người dùng cũng như các nhà cung cấp dịch vụ. Về phía người sử dụng, đó là các tiêu chí về thời gian, chi phí. Về phía nhà cung cấp dịch vụ, đó là cân bằng tải và hiệu quả về năng lượng. Để đảm bảo chất lượng dịch vụ, thời gian phản hồi đóng vai trò quan trọng ảnh hưởng trực tiếp tới trải nghiệm

người dùng. Bên cạnh đó, chi phí cũng là một khía cạnh mà bất kỳ người dùng nào cũng quan tâm.

2.2.2 So sánh điện toán đám mây và điện toán sương mù

Bảng 2.1 dưới đây trình bày tóm tắt những khác biệt của điện toán đám mây và điện toán sương mù. Nhờ kết hợp cả điện toán đám mây và điện toán sương mù mà điện toán sương mù - đám mây có thể phát huy được ưu điểm và hạn chế các nhược điểm của cả hai mô hình đó.

Dễ dàng nhận thấy nhất là độ trễ của điện toán sương mù nhỏ hơn đám mây rất nhiều do điện toán sương mù nằm ở gần các thiết bị IoT, nơi sinh ra dữ liệu và đi qua ít các điểm trung gian. Nhưng trái lại, chúng ta không thể kỳ vọng khả năng xử lý của sương mù sẽ so sánh được với các đám mây được đặt tại các trung tâm dữ liệu với phần cứng mạnh mẽ, hiện đại. Về phân bố địa lý, các đám mây nằm tập trung trong các trung tâm dữ liệu chuyên biệt, cần máy lạnh để hoạt động; sương mù nằm rải rác trên biên của mạng, có thể ở bất cứ đâu trong nhà hay ngoài trời, có hệ thống làm mát hoặc không. Về an ninh, các đám mây được truy cập qua Internet, nơi không thể loại trừ trường hợp các tin tặc truy cập trái phép vào dữ liệu cá nhân hay việc nghe trộm trên đường truyền do dữ liệu phải đi qua rất nhiều nút. Với sương mù, mọi truy cập từ bên ngoài vào đều bị hạn chế tuỳ theo mỗi chính sách riêng. Với sương mù, các dịch vụ có thể tự biết vị trí địa lý của chúng nhờ vào sự phân tán theo địa lý của chính các nút sương mù.

Bảng 2.1: Điện toán đám mây và điện toán sương mù

Thuộc tính	Điện toán đám mây	Điện toán sương mù
Độ trễ	Cao	Thấp
Thời gian xử lý	Nhỏ	Lớn
Phân bố địa lý	Tập trung	Phân tán
Vị trí của dịch vụ	Trên Internet	Tại biên của mạng nội bộ
Khoảng cách tới máy chủ	Nhiều bước nhảy	Một bước nhảy
An ninh	Kém an toàn	An toàn hơn
Số lượng các nút máy chủ	Ít	Rất nhiều
Hỗ trợ di động	Hạn chế	Hỗ trợ
Nhận biết về vị trí địa lý	Không	Có
Giao tiếp giữa các thiết bị	Qua Internet	Trực tiếp từ các nút biên thông qua đa dạng các giao thức và chuẩn khác nhau
Xử lý dữ liệu	Từ xa nguồn thông tin và mang tính dài hạn	Gần với nguồn thông tin và mang tính ngắn hạn
Môi trường làm việc	Tại các trung tâm dữ liệu có máy lạnh	Ngoài trời (Đường phố, vườn hoa,...) hoặc trong nhà (Cửa hàng, khách sạn,...)

2.2.3 Ứng dụng của điện toán sương mù

- Điện lưới thông minh: Điện lưới thông minh là mạng lưới phân bố năng lượng điện thế hệ tiếp theo. Điện lưới thông minh bao gồm đường dây tải điện, trạm biến áp, máy biến áp, v.v... Nó sử dụng các luồng năng lượng và dữ liệu hai chiều để tạo ra một mạng lưới phân phối năng lượng tăng cường tự động và phân tán. Điện lưới thông minh cung cấp khả năng phân phối năng lượng rõ ràng giúp các nhà cung cấp và khách hàng có thể theo dõi và kiểm soát giá

cả, sản xuất và tiêu thụ của họ trong thời gian thực [4]. Trong môi trường dữ liệu lớn, hàng triệu công tơ điện thông minh được lắp đặt trong nhà của khách hàng. Tại các xử lý ở biên, tầng sương mù thu thập, xử lý và lọc các thông tin ở cục bộ và các thông tin được lưu trữ lâu dài sẽ được gửi về các đám mây [5].

- Ô tô tự hành: Sự ra đời của ô tô bán tự hành và ô tô tự hành sẽ làm tăng lượng dữ liệu vốn đã lớn mà các phương tiện tạo ra. Để ô tô hoạt động độc lập đòi hỏi khả năng phân tích cục bộ dữ liệu nhất định trong thời gian thực, chẳng hạn như môi trường xung quanh, điều kiện lái xe và chỉ đường. Các dữ liệu khác có thể cần được gửi lại cho nhà sản xuất để giúp cải thiện việc bảo dưỡng xe hoặc theo dõi việc sử dụng xe. Môi trường điện toán sương mù cho phép xử lý tất cả các nguồn dữ liệu này ở cả biên (trên xe) và ở điểm cuối của nó (nhà sản xuất).

- Thành phố thông minh: Các hệ thống tiện ích đang ngày càng tăng việc sử dụng dữ liệu thời gian thực để tăng hiệu quả cho chúng. Đôi khi dữ liệu này ở các vùng sâu vùng xa, do đó việc xử lý gần nơi tạo ra nó là điều cần thiết. Thêm nữa, dữ liệu cần được tổng hợp từ một số lượng lớn các cảm biến. Kiến trúc điện toán sương mù có thể giải quyết được cả hai vấn đề này.

- Nông nghiệp thông minh: Với sự gia tăng mạnh mẽ của các thiết bị hỗ trợ IoT và lợi ích mà chúng mang lại, nông nghiệp thông minh trở thành lĩnh vực thích hợp cho các nhà cung cấp dịch vụ IoT. Người nông dân sẽ chuyển dần sang sử dụng các phương pháp canh tác thông minh và tạo ra lượng dữ liệu lớn từ các cảm biến đặt trong đất, nước, không khí như cảm biến độ ẩm, nhiệt độ, nồng độ khoáng chất, ánh sáng môi trường,...

- Chăm sóc sức khoẻ: Các dịch vụ và ứng dụng chăm sóc sức khoẻ thường bị phản hồi trễ và là nơi tạo ra các thông tin bí mật của bệnh nhân. Dữ liệu được sinh ra gồm cả các dữ liệu nhạy cảm và cá nhân. Tương tự thế, dữ liệu về vị trí đôi khi cũng là nhạy cảm trong một số tình huống. Sự mất ổn định và độ trễ gia tăng có thể gây ra nhiều vấn đề khác nhau trong các ứng dụng chăm sóc sức khoẻ từ xa và y tế từ xa. Những tình huống như vậy có thể làm cho điện toán sương mù trở thành một mô hình thích hợp trong các tình huống chăm sóc

sức khoẻ [6]. Điện toán sương mù đóng một vai trò quan trọng trong dịch vụ y tế khẩn cấp với ít hạn chế về độ trễ liên quan đến các thiết bị y tế cấy ghép, liên lạc cứu thương hay truy cập di động vào các tệp y tế của bệnh nhân [7]. Tác giả của [8] đã đề xuất một hệ thống cho bệnh nhân đột quỵ. Hệ thống này sử dụng điện toán sương mù để phát hiện, dự đoán và ngăn chặn người bệnh bị ngã. Họ sử dụng thuật toán dự đoán ngã trên các thiết bị biên và tài nguyên đám mây và so sánh kết quả của hệ thống đã đề xuất với các hướng tiếp cận khác. Họ đưa ra kết luận hệ thống này có thời gian phản hồi ngắn hơn và tiêu thụ ít năng lượng hơn so với các phương pháp sử dụng đám mây.

- Thực tế tăng cường (Augmented reality - AR): Thực tế tăng cường là khả năng bao phủ một lớp kỹ thuật số vào thế giới thực. Thông tin về thực tế tăng cường yêu cầu độ trễ thấp và tốc độ xử lý cao để cung cấp thông tin đúng như những gì được chỉ ra bởi vị trí của người dùng [9]. Các ứng dụng của thực tế tăng cường rất nhạy cảm với độ trễ, chỉ một chút chậm chễ trong phản hồi có thể làm hỏng các trải nghiệm của người sử dụng. Do đó, điện toán sương mù có thể trở thành một nhân tố bắt buộc trong lĩnh vực thực tế tăng cường [10].

- Hệ thống điều khiển giao thông: Trong hệ thống điều khiển giao thông, các máy quay video phát hiện ra đèn hiệu của xe cứu thương có thể tự động thay đổi đèn đường và mở đường cho xe. Đèn đường thông minh hợp tác cục bộ cùng với các cảm biến và xác định sự xuất hiện của người đi bộ và người đi xe đạp đồng thời đánh giá khoảng cách và vận tốc của các phương tiện đang đến gần. Bên cạnh đó, đèn chiếu sáng thông minh sẽ tự động được bật khi cảm biến phát hiện có chuyển động và tắt khi phương tiện đã đi qua. Hệ thống điều khiển giao thông là hữu ích, giảm thiểu tai nạn, duy trì lưu lượng phương tiện ổn định và thu thập dữ liệu liên quan để đánh giá và cải thiện hiệu suất hệ thống [11].

2.3 Bài toán lập lịch

2.3.1 Định nghĩa

Bài toán lập lịch có thể được định nghĩa là một bài toán tìm kiếm chuỗi tối ưu để thực hiện một tập các hoạt động chịu tác động của một tập các ràng buộc cần phải được thoả mãn. Người lập lịch thường cố gắng thử để sử dụng tối ưu các cá thể, máy móc và tối thiểu thời gian cần thiết để hoàn thành toàn bộ các công việc. Vì thế bài toán lập lịch là một bài toán rất khó để có thể giải quyết được. Hiện nay có nhiều kỹ thuật để giải bài toán lập lịch. Những kỹ thuật đó bao gồm: cách tiếp cận trí tuệ nhân tạo như hệ thống cơ sở tri thức (knowledge-based systems), hệ chuyên gia, mạng Neuron và các cách tiếp cận của các nghiên cứu hoạt động: lập trình tính toán, lập trình động, tìm kiếm nhánh và đường biên, kỹ thuật mô phỏng, tìm kiếm Tabu hay phương pháp nút cổ chai...

2.3.2 Đặc trưng

Khi nghiên cứu về bài toán lập lịch cần làm rõ các đặc trưng cơ bản của bài toán. Các đặc trưng cơ bản đó bao gồm: tài nguyên, tác vụ, ràng buộc và mục tiêu.

1. Tài nguyên: đó là các nguồn dữ liệu đầu vào của bài toán. Các tài nguyên này có thể phục hồi hoặc không.
2. Tác vụ: được đánh giá qua các tiêu chuẩn thực hiện như thời gian thực hiện, chi phí, mức tiêu thụ nguồn tài nguyên.
3. Ràng buộc: đây là những điều kiện cần thoả mãn để bài toán có thể đưa ra lời giải tốt nhất.
4. Mục tiêu: đánh giá độ tối ưu của lời giải bài toán. Khi các mục tiêu được thoả mãn thì các ràng buộc cũng phải được thoả mãn.

Vấn đề ràng buộc tài nguyên trong bài toán lập lịch rất đa dạng và phức tạp. Thông thường các vấn đề liên quan đến ràng buộc về tài nguyên trong bài toán lập lịch thường xảy ra trong tình huống lượng tài nguyên khan hiếm, hạn chế, cùng với đó là lượng thời gian hoàn thành công việc. Các công việc có thể được thực hiện theo một số phương pháp khác nhau nhưng có điểm chung đó là được xác định bởi một quy tắc ràng buộc, các công việc được thực hiện theo dây chuyền, một số công việc chỉ được thực hiện sau khi đã thực hiện xong một số công việc khác.

2.3.3 Một số hướng tiếp cận và thuật toán

Độ phức tạp để giải những bài toán lập lịch tăng theo hàm số mũ với độ tăng của số lượng các biến, do đó thời gian để tìm ra lời giải tối ưu là rất lớn. Thực vậy, bài toán lập lịch đã được chứng minh là thuộc lớp NP-đầy đủ [12].

Giải thuật vét cạn

Vét cạn là một trong những giải thuật giải bài toán tối ưu. Giải thuật vét cạn là tìm phương án tối ưu của bài toán bằng cách lựa chọn một phương án trong tập hợp tất cả các phương án của bài toán để tìm ra phương án tối ưu. Trong nhiều bài toán, không gian các phương án có thể quá lớn. Do vậy, khi áp dụng thuật toán vét cạn không đảm bảo về thời gian cũng như kỹ thuật. Trong quá trình duyệt ta luôn giữ lại một phương án là phương án mẫu, là phương án có giá nhỏ nhất tại thời điểm đó.

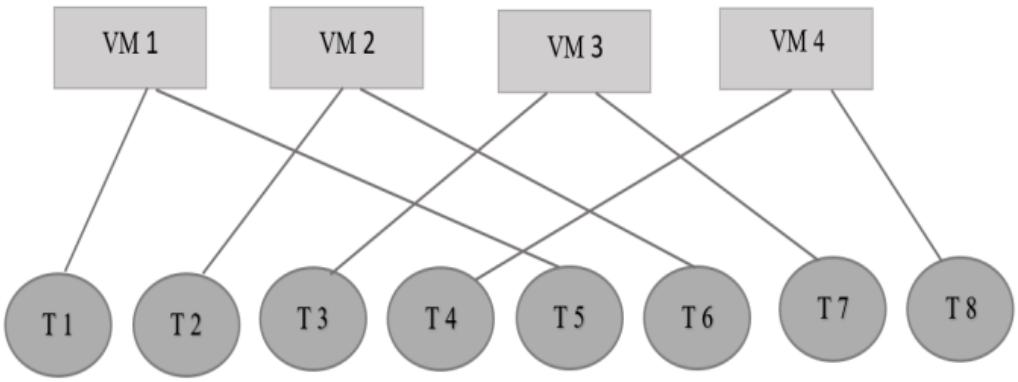
Phương pháp nhánh cận

Phương pháp đánh giá nhánh cận là phương pháp tính giá trị của phương án ngay trong quá trình xây dựng các thành phần của phương án, có nghĩa là ta sẽ tính xem việc xây dựng phương án theo hướng đó có thể tốt hơn phương án mẫu hay không. Nếu không tốt hơn ta lựa chọn hướng khác. Bằng cách này ta đã hạn

chế được nhiều phương án mà chắc rằng trong đó không chứa phương án tối ưu. Một yêu cầu đặt ra là tính toán nhánh cận như thế nào để có thể hạn chế tối đa các phương án phải duyệt.

Giải thuật Round Robin

Giải thuật Round Robin hay còn được gọi là giải thuật định thời luân phiên là một trong những giải thuật lập lịch cũ nhất, phổ biến và đơn giản nhất được thiết kế đặc biệt cho các hệ thống chia sẻ thời gian nhằm tăng hiệu quả của hệ thống [13]. Các tác vụ sẽ được đi tới các bộ xử lý một cách tuần tự, vòng tròn. Với đặc điểm coi mỗi tác vụ đều là độc lập, bình đẳng, Round Robin dễ dàng được cài đặt, thực thi để lập lịch cho hệ điều hành, điều phối CPU hay chuyển gói mạng, cân bằng tải, lập lịch trên điện toán đám mây. Bên cạnh có cũng có nhiều nghiên cứu để cải tiến, tối ưu hoá lại giải thuật này như Round Robin nâng cao, Round Robin có trọng số... [14], [15], [16]



Hình 2.4: Giải thuật Round Robin trong điện toán đám mây [13]

Giải thuật chia để trị

Đây là kỹ thuật từ trên xuống (top - down), có thể nói rằng đây là kỹ thuật quan trọng, được áp dụng rộng rãi nhất để thiết kế các giải thuật có hiệu quả. Nội dung của thuật toán là: Để giải một bài toán kích thước *n*, ta chia bài toán đã cho thành một số bài toán con có kích thước nhỏ hơn. Giải các bài toán con này rồi

tổng hợp kết quả lại để được lời giải của bài toán ban đầu. Đối với các bài toán con, chúng ta lại sử dụng kỹ thuật chia để trị để có được các bài toán kích thước nhỏ hơn nữa. Quá trình trên sẽ dẫn đến những bài toán mà lời giải của chúng là hiển nhiên hoặc dễ dàng thực hiện, ta gọi các bài toán này là bài toán cơ sở.

Tóm lại kỹ thuật chia để trị bao gồm hai quá trình: Phân tích bài toán đã cho thành các bài toán cơ sở và tổng hợp kết quả từ bài toán cơ sở để có lời giải của bài toán ban đầu. Tuy nhiên đối với một số bài toán thì quá trình phân tích đã chứa đựng việc tổng hợp kết quả, do đó nếu chúng ta đã giải xong các bài toán cơ sở thì bài toán ban đầu cũng đã được giải quyết. Ngược lại, có những bài toán mà quá trình phân tích thì đơn giản nhưng việc tổng hợp kết quả lại rất khó khăn. Hai giải thuật sắp xếp MergeSort và QuickSort thực chất là đã sử dụng kỹ thuật chia để trị. Với MergeSort, để sắp xếp một danh sách L gồm n phần tử, chúng ta chia L thành hai danh sách con L_1 và L_2, mỗi danh sách có $n/2$ phần tử. Sắp xếp L_1, L_2 và trộn hai danh sách đã được sắp này để được một danh sách có thứ tự. Quá trình phân tích ở đây là quá trình chia đôi một danh sách, quá trình này sẽ dẫn đến bài toán sắp xếp một danh sách có độ dài bằng 1, đây chính là bài toán cơ sở vì việc sắp xếp danh sách này là "không làm gì cả". Việc tổng hợp các kết quả ở đây là trộn hai danh sách đã được sắp xếp để được một danh sách có thứ tự.

Với QuickSort, để sắp xếp một danh sách gồm n phần tử, ta tìm một giá trị khoá và phân hoạch danh sách đã cho thành hai danh sách con "bên trái" và "bên phải". Sắp xếp "bên trái" và "bên phải" thì ta được danh sách có thứ tự. Quá trình phân chia sẽ dẫn đến các bài toán sắp xếp một danh sách chỉ gồm một phần tử hoặc bao gồm nhiều phần tử có khoá bằng nhau, đó chính là các bài toán cơ sở vì bản thân chúng đã có thứ tự rồi. Ở đây chúng ta cũng không có việc tổng hợp kết quả một cách tường minh vì việc đó đã được thực hiện trong quá trình phân hoạch.

Giải thuật heuristic

Là các giải thuật dựa trên phân tích yếu tố cụ thể gắn với bài toán để xác định phương hướng tìm kiếm lời giải. Nó thể hiện cách giải bài toán với các đặc

tính sau:

- Thường tìm được lời giải tốt (nhưng không chắc là lời giải tốt nhất).
- Giải bài toán theo giải thuật heuristic thường dễ dàng và nhanh chóng đưa ra kết quả hơn so với giải thuật tối ưu. Vì vậy chi phí thấp hơn.
- Giải thuật heuristic thường thể hiện khá tự nhiên, gần gũi với cách suy nghĩ và hành động của con người.

Có nhiều phương pháp để xây dựng một giải thuật heuristic, trong đó người ta thường dựa vào một số nguyên lý cơ sở như sau:

- Nguyên lý vét cạn thông minh: Trong một bài toán tìm kiếm nào đó, khi không gian tìm kiếm lớn, ta thường tìm cách giới hạn lại không gian tìm kiếm hoặc thực hiện một số kiểu dò tìm đặc biệt dựa vào đặc thù của bài toán để nhanh chóng tìm ra mục tiêu.
- Nguyên lý tham lam (Greedy): Lấy tiêu chuẩn tối ưu (trên phạm vi toàn cục) của bài toán để làm tiêu chuẩn chọn lựa hành động cho phạm vi cục bộ của từng bước (hay từng giai đoạn) trong quá trình tìm kiếm lời giải.
- Nguyên lý thứ tự: Thực hiện hành động dựa trên một cấu trúc thứ tự hợp lý của không gian khảo sát nhằm nhanh chóng đạt được một lời giải tốt.
- Nguyên lý hướng đích (hàm heuristic): Trong việc xây dựng các giải thuật heuristic, người ta thường dùng các hàm heuristic. Đó là các hàm đánh giá thô, giá trị của hàm phụ thuộc vào trạng thái hiện tại của bài toán tại mỗi bước giải. Nhờ giá trị này, ta có thể chọn được cách hành động tương đối hợp lý trong từng bước của giải thuật.

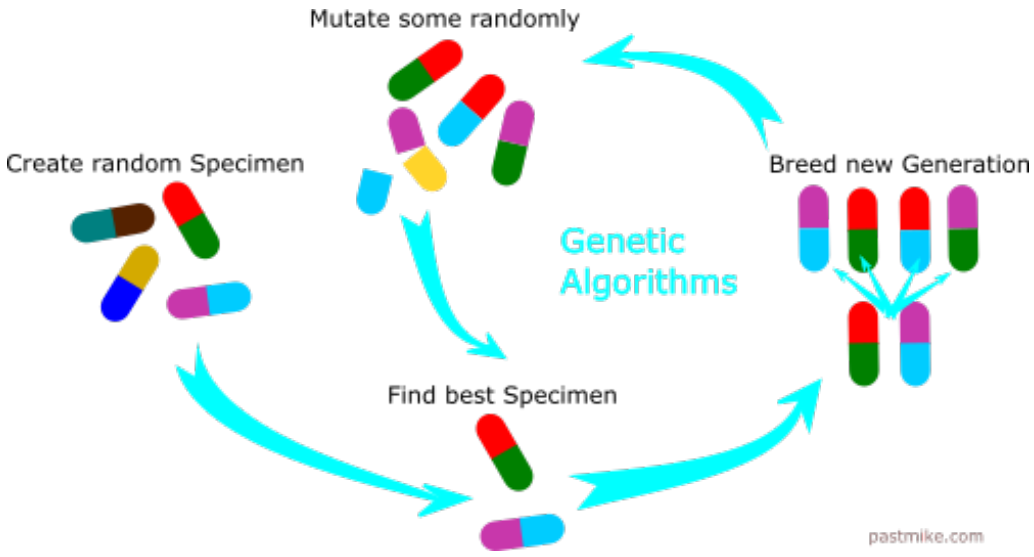
Giải thuật metaheuristic

Metaheuristic là loại heuristic tổng quát có thể áp dụng cho nhiều lớp bài toán, thường nó là họ của những giải thuật tìm kiếm như: giải thuật mô phỏng

luyện kim, giải thuật di truyền, giải thuật bầy đàn,... Chúng ta có thể sẽ sử dụng metaheuristic để giải bài toán NP-đầy đủ, tuy nhiên metaheuristic cũng như một số giải thuật khác cố gắng tìm lời giải tối ưu của bài toán trong thời gian đa thức, nhưng nó không đảm bảo rằng sẽ tìm ra lời giải tối ưu.

Giải thuật di truyền

Giải thuật di truyền (Genetic Algorithm - GA) là một thuật toán lấy ý tưởng từ thuyết tiến hoá của Charles Darwin dựa trên việc quan sát quá trình tiến hoá trong tự nhiên. Các nguyên lý cơ bản của giải thuật di truyền được tác giả John H. Holland công bố lần đầu tiên vào năm 1962. Sau đó, các nền tảng toán học của giải thuật lần đầu tiên được công bố vào năm 1975 trong cuốn sách "Adaptation in Natural and Artificial Systems" cũng của tác giả [17] [18]. Có thể nói Holland là người đi tiên phong nghiên cứu trong lĩnh vực giải thuật di truyền cùng với các tác giả Goldbeg, Beglay...



Hình 2.5: Giải thuật di truyền
(https://pastmike.com/what-is-a-genetic-algorithm/)

Giải thuật di truyền là một giải thuật dựa trên cơ chế của chọn lọc tiến hoá trong tự nhiên: "Trong mọi thế hệ, một tập mới các sinh vật được tạo ra bằng cách lai ghép những nhân tố thích nghi nhất với môi trường của những sinh vật trong thế hệ cũ cùng với sự xuất hiện đột biến ngẫu nhiên của các cá thể trong thế hệ mới".

Vận dụng cơ chế đó, giải thuật di truyền được bắt đầu với một quần thể ngẫu nhiên có *n* chuỗi, rồi sao chép các chuỗi theo khuynh hướng hướng đến cái tốt, ghép cặp và đổi các chuỗi con thành phần, thỉnh thoảng làm đột biến giá trị bit.

Theo đề xuất ban đầu của giáo sư John Holland, một vấn đề, bài toán đặt ra sẽ được mã hóa thành các chuỗi bit với chiều dài cố định. Nói một cách chính xác là các thông số của bài toán sẽ được chuyển đổi và biểu diễn lại dưới dạng các chuỗi nhị phân. Các thông số này có thể là các biến của một hàm hoặc hệ số của một biểu thức toán học. Người ta gọi các chuỗi bit này là mã genome ứng với mỗi cá thể, các genome đều có cùng chiều dài. Nói ngắn gọn, một lời giải sẽ được biểu diễn bằng một chuỗi bit, cũng giống như mỗi cá thể đều được quy định bằng gen của cá thể đó vậy. Như vậy, đối với thuật giải di truyền, một cá thể chỉ có một gen duy nhất và một gen cũng chỉ phục vụ cho một cá thể duy nhất.

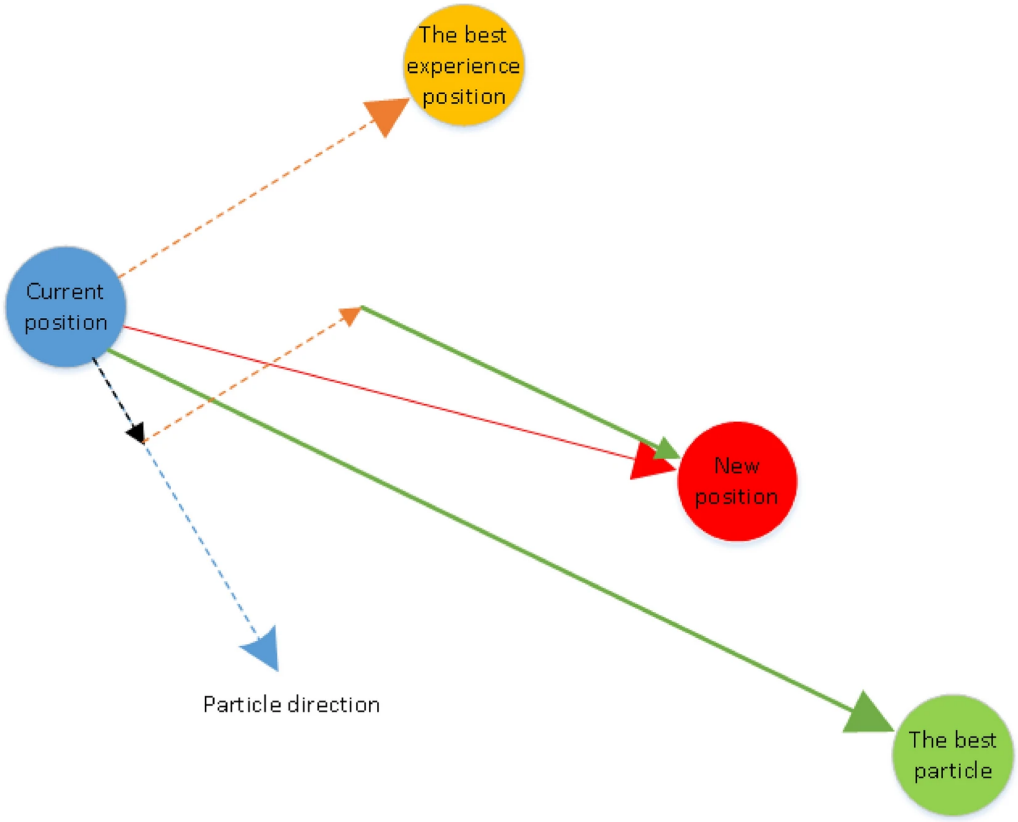
Ban đầu, phát sinh một tập hợp các chuỗi bit ngẫu nhiên. Tập các cá thể này được gọi là quần thể ban đầu (initial population). Sau đó, xác định một giá trị gọi là độ thích nghi - fitness, chính là độ "tốt" của lời giải. Để cải thiện tính thích nghi của quần thể, có hai thao tác:

- Đầu tiên là sao chép nguyên mẫu một nhóm các cá thể tốt từ thế hệ trước đưa sang thế hệ sau. Thao tác này đảm bảo độ thích nghi của thế hệ sau luôn được giữ ở mức độ hợp lý. Các cá thể được chọn thường là các các thể có độ thích nghi cao nhất.
- Thứ hai là tạo ra các các thể mới bằng cách thực hiện các thao tác *sinh sản* trên một số cá thể được chọn từ thế hệ trước: lai tạo (crossover) và đột biến (mutation).

Thế hệ mới tạo ra lại được xử lý như thế hệ trước (xác định độ thích nghi và tạo thế hệ mới) cho đến khi cá thể đạt được giải pháp mong muốn hoặc đến thời gian giới hạn.

Giải thuật tối ưu bầy đàn

Giải thuật tối ưu bầy đàn (Particle swarm optimization - PSO) là một trong những thuật toán xây dựng dựa trên khái niệm trí tuệ bầy đàn để tìm kiếm lời giải cho các bài toán tối ưu hoá trên một không gian tìm kiếm nào đó. Phương pháp tối ưu bầy đàn là một dạng của thuật toán tiến hoá quần thể, với sự tương tác giữa các cá thể trong một quần thể để khám phá một không gian tìm kiếm. PSO là kết quả của sự mô hình hoá việc đàn chim bay đi tìm kiếm thức ăn cho nên nó thường được xếp vào các thuật toán có sử dụng trí tuệ bầy đàn, được giới thiệu vào năm 1995 tại một hội nghị của IEEE bởi James Kennedy và Russel C. Eberhart [19].



Hình 2.6: Một điểm tìm kiếm bằng PSO [20]

PSO được khởi tạo bằng một nhóm cá thể ngẫu nhiên và sau đó tìm nghiệm tối ưu bằng cách cập nhật các thế hệ. Trong mỗi thế hệ, vị trí của mỗi cá thể được cập nhật theo hai vị trí tốt nhất. Giá trị thứ nhất là vị trí tốt nhất mà nó đã từng đạt được cho tới thời điểm hiện tại, gọi là P_{best}. Một nghiệm tối ưu khác mà cá thể này bám theo là nghiệm tối ưu toàn cục G_{best}, đó là vị trí tốt nhất trong tất cả quá trình tìm kiếm cả quần thể từ trước tới thời điểm hiện tại. Nói cách khác, mỗi cá

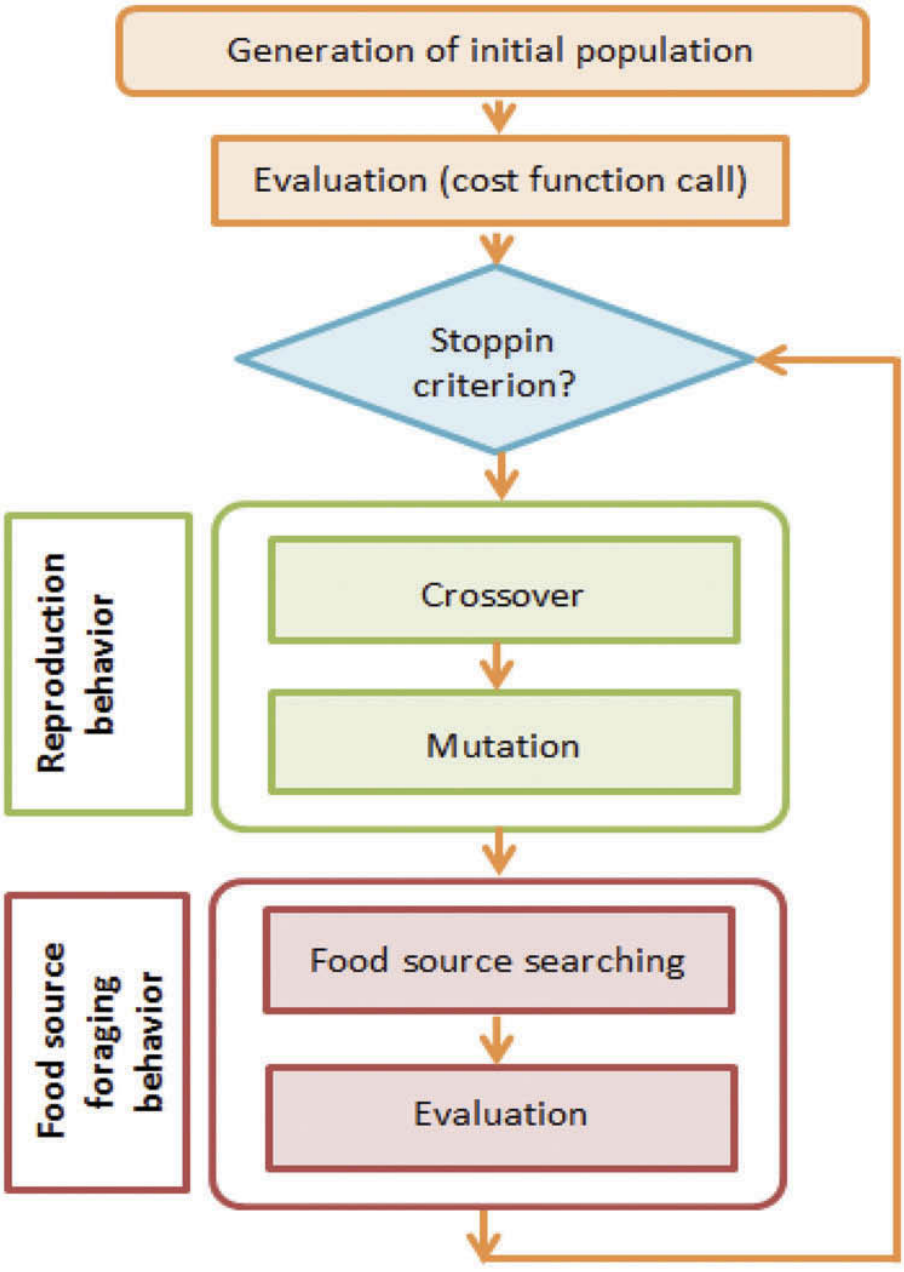
thể trong quần thể cập nhật vị trí của nó theo vị trí tốt nhất của nó và của cả quần thể tính tới thời điểm hiện tại.

Giải thuật đàn ong

Giải thuật đàn ong (Bees Life Algorithm - BLA) là một giải thuật meta-heuristic mới cho những bài toán tối ưu nâng cao được đề xuất bởi Bitam và các cộng sự [21]. BLA đã được áp dụng thành công để giải các bài toán NP-khó:

- Bài toán định tuyến đa hướng cho mạng xe cộ.
- Phân cụm văn bản.
- Lựa chọn dịch vụ cho điện toán đám mây.
- Bài toán lập lịch cho điện toán đám mây.

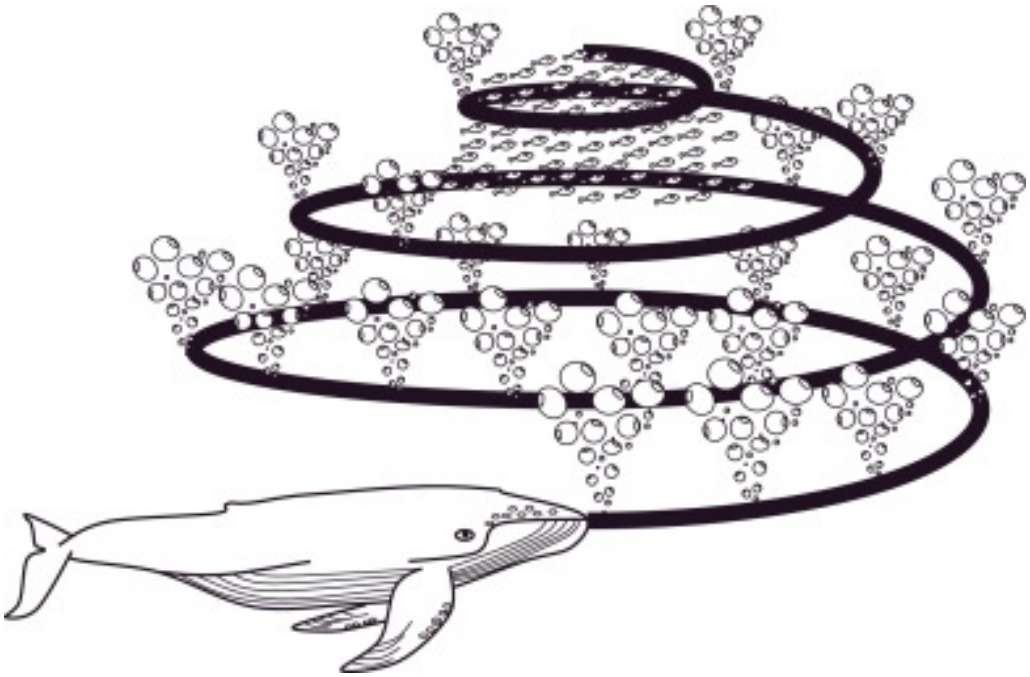
Ý tưởng của BLA dựa trên hai hành vi chính của loài ong là sinh sản và tìm kiếm thức ăn. Đầu tiên với N cá thể ong ban đầu được khởi tạo và được đánh giá độ thích nghi. Sau khi đã sắp xếp, ong thích nghi nhất sẽ là ong chúa, D ong tiếp theo là ong đực và W ong còn lại là những ong thợ. Quần thể ong thay đổi qua mỗi thế hệ, vòng đời của ong gồm hai hành vi: Thứ nhất, ong chúa ghép đôi với các ong đực để sinh ra ấu trùng bởi lai tạo và đột biến. Đánh giá mỗi cá thể để tìm ra ong tốt nhất sẽ thay thế ong chúa, D ong tiếp theo sẽ là ong đực và W ong thợ. Thứ hai là tìm kiếm thức ăn, các ong thợ sẽ tìm kiếm thức ăn trong các khu vực của nó cũng là tìm kiếm cục bộ để mỗi cá thể trở nên tốt hơn so với chính nó lúc ban đầu. Việc đánh giá tiếp tục được thực hiện để có ong tốt nhất là ong chúa, D ong tiếp theo là ong đực và W ong thợ. Quá trình này được lặp đi lặp lại cho tới khi điều kiện dừng được thoả mãn.



Hình 2.7: Sơ đồ BLA [21]

Giải thuật cá voi

Giải thuật cá voi (Whale Optimization Algorithm - WOA) là một giải thuật metaheuristic được giới thiệu lần đầu năm 2016 dựa trên hành vi săn mồi của cá voi lưng gù [22]. Hành vi săn mồi đặc biệt này được gọi là phương pháp kiếm ăn bằng lưới bong bóng. Người ta quan sát thấy rằng chúng tạo ra các bong bóng đặc biệt theo đường tròn hoặc hình số "9" như được minh hoạ trong hình 2.8.



Hình 2.8: Cá voi kiếm ăn bằng lưới bong bóng [22]

Giải thuật này gồm một số bước:

- Bẫy con mồi: Những con cá voi nhận thấy vị trí của con mồi và bao vây chúng. Bởi vì vị trí trong không gian tìm kiếm không được biết trước, giải thuật WOA giả định rằng lời giải tốt nhất hiện tại là con mồi mục tiêu gần với giá trị tối ưu nhất. Sau khi cá voi tốt nhất được xác định, những con cá khác sẽ cố gắng cập nhật vị trí để hướng tới con tốt nhất.
- Tấn công bằng lưới bong bóng (Pha khai thác): Ở pha này, những con cá voi sẽ thu hẹp vòng vây bằng cách bơi thành những hình tròn đặc biệt hay số "9" nhằm tìm ra điểm tối ưu cục bộ gần với con cá đang là tốt nhất.

- Tìm kiếm con mồi (Pha khám phá): Những con cá voi tìm kiếm ngẫu nhiên theo vị trí của chúng sao cho càng xa những con cá khác càng tốt. Đối lập với pha khai thác, chúng ta cập nhật vị trí của những con cá bằng cách chọn ngẫu nhiên thay vì vị trí tốt nhất của một cá thể nào đó. Theo cách này cho phép WOA thực hiện tìm kiếm toàn cục.

Giải thuật WOA bắt đầu với một tập lời giải ngẫu nhiên. Tại mỗi vòng lặp, các cá thể cập nhật vị trí của nó hoặc ngẫu nhiên hoặc theo vị trí tốt nhất từ trước tới nay. Quá trình này không ngừng giúp WOA thu được lời giải tốt hơn. Cuối cùng WOA sẽ kết thúc khi giải thuật thoả mãn được điều kiện dừng.

2.3.4 Bài toán lập lịch trong điện toán sương mù - đám mây

Trong điện toán đám mây và điện toán sương mù, có rất nhiều giải thuật metaheuristic đã được áp dụng hiệu quả cho bài toán lập lịch. Có một số giải thuật metaheuristic nổi tiếng có thể kể đến như: Giải thuật di truyền (Genetic Algorithm - GA) [23] và giải thuật bầy đàn (Particle Swarm Optimization - PSO) [24]. Gần đây có giải thuật cá voi (Whale Optimization Algorithm - WOA) [22] cũng đã trở thành một phương pháp điển hình chứng minh hiệu năng cạnh tranh khi so sánh với những phương pháp khác nói trên. Theo môi trường chạy của các chiến lược lập lịch tác vụ, luận văn phân loại các nghiên cứu đã có thành ba nhóm gồm các hệ thống đám mây, hệ thống sương mù và hệ thống sương mù - đám mây.

Nhiều nghiên cứu đã giải quyết với các bài toán lập lịch tác vụ và phân bổ tài nguyên cho đám mây sử dụng các phương pháp khác nhau. Trong [25], [26], và [27], các tác giả đã đề xuất triển khai các kỹ thuật mờ và học máy để phân tích và dự đoán khối lượng công việc nhằm đưa ra các quyết định mở rộng, co giãn tài nguyên. Trong khi đó, hàng đợi và lý thuyết trò chơi đã được áp dụng cho các bài toán lập lịch hệ thống [28], [29]. Các tác giả của [30], [31], [32], [33] đề xuất chiến lược quản lý các công việc cần xử lý sử dụng các phương pháp PSO, GA và WOA.

Một số nghiên cứu đã cố gắng mô hình hoá môi trường sương mù với các

đặc tính của nó. Trong [34] và [35], các tác giả chỉ ra rằng các đặc điểm của mô hình tính toán này cùng với một loạt các ràng buộc như là độ trễ, vị trí, phân bố về mặt địa lý, tính di động hay không đồng nhất, và các quyền truy cập chính vào thiết bị cuối. Tuy nhiên, họ lại chưa đưa ra bất cứ giải pháp nào cho bài toán lập lịch tác vụ. Trong [36], các tác giả viết về khả năng phục hồi tính toán trong môi trường hiệu năng cao và phân tán. Trong [37], các tác giả giới thiệu lập lịch tác vụ có sự ưu tiên trong hệ thống sương mù sử dụng hàng đợi. Các tác giả của [38] khám phá công nghệ container để phân bổ tài nguyên tầng sương mù cho các tác vụ đi tới. Trong công trình [39], các tác giả áp dụng PSO, PSO nhị phân và giải thuật đàn dơi (Bat algorithm) cho việc phân bổ tài nguyên trong điện toán sương mù. Giải thuật tìm kiếm trọng lực (Gravitational Search Algorithm - GSA) được sử dụng trong công trình [40] để giải quyết bài toán độ trễ trong môi trường này.

Trong môi trường IoT, với sự tích hợp của điện toán sương mù và đám mây, việc lập lịch các tác vụ xử lý dữ liệu khá phức tạp. Để giải quyết bài toán này, xây dựng môi trường sương mù - đám mây với các ràng buộc khác nhau là một cơ sở tất yếu cho các chiến lược lập lịch tác vụ. Các tác giả của [3] đã công thức hoá mô hình hoạt động cho các hệ thống sương mù - đám mây. Công trình này xem xét ba ràng buộc môi trường cùng lúc gồm: điện năng tiêu thụ, độ trễ dịch vụ và chi phí. Tuy nhiên, các tác giả của công trình này lại chưa sử dụng bất kỳ phương pháp nào cho mô hình họ đề xuất. Trong [41] và [42], các tác giả đã giải quyết vấn đề phân bổ tài nguyên và lập lịch tác vụ sử dụng một số cơ chế tiêu chuẩn như là Round-Robin, Min-Min hay Max-Min. Round-Robin là giải thuật lập lịch cũ nhất, đơn giản nhất khi các tác vụ được gán cho từng nút theo thứ tự vòng tròn mà không có độ ưu tiên nào. Min-Min nhiều phức tạp hơn Round-Robin. Nó bắt đầu với một tập tác vụ chưa được gán. Min-Min có hai pha: Trong pha thứ nhất, tập hợp của các thời gian hoàn thành dự kiến tối thiểu (tác vụ có thời gian hoàn thành dự kiến sớm nhất trên các máy tương ứng) cho mỗi tác vụ được tìm ra. Trong pha thứ hai, tác vụ với tổng thời gian toàn hành dự kiến tối thiểu từ tập hợp đó sẽ được chọn và gán cho tài nguyên tương ứng. Tác vụ này sau đó được xoá bỏ khỏi tập hợp và quá trình cứ lặp lại cho đến khi tất cả các tác vụ đã được ánh xạ tới tài nguyên hợp lý. Cũng tương tự Min-Min, ngoại trừ pha thứ hai, Max-Min gán các tác vụ với thời

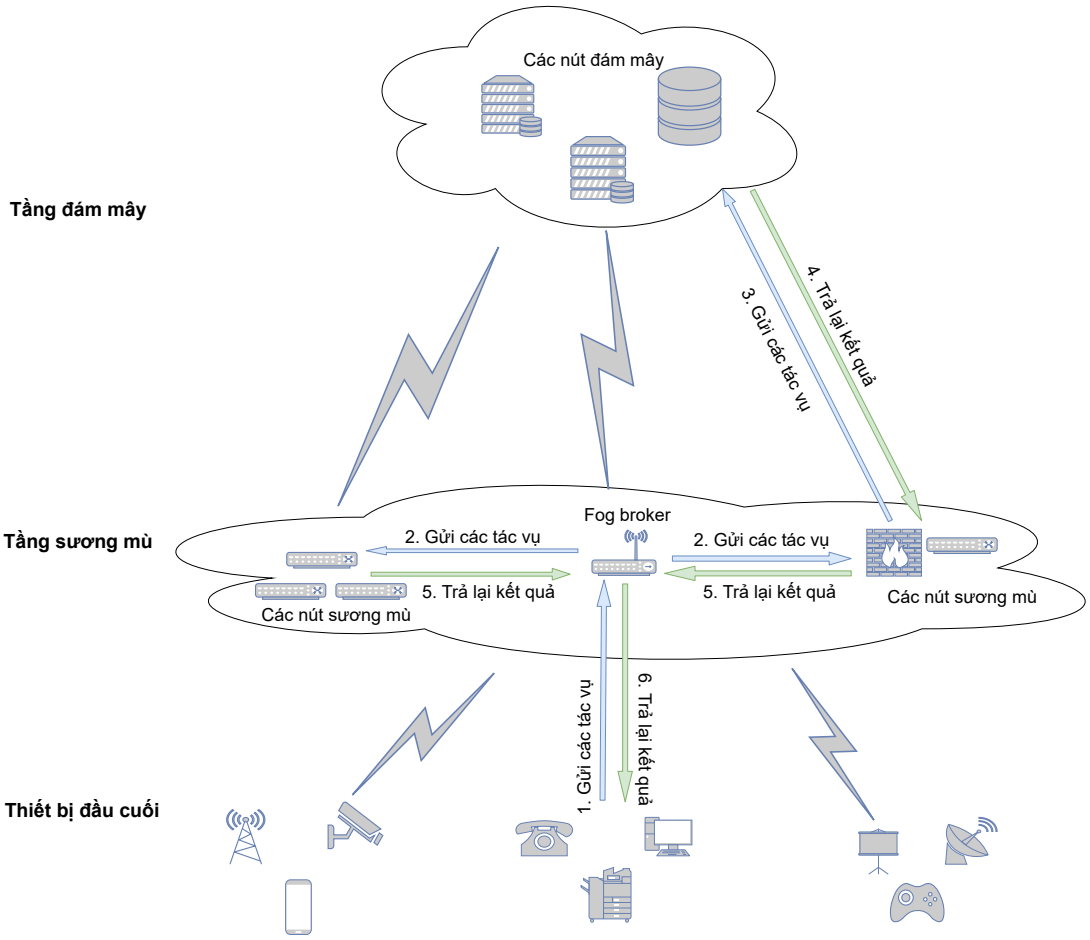
gian hoàn thành dự kiến lớn nhất vào các tài nguyên tương ứng. Trong khi đó, các giải thuật metaheuristic được sử dụng rộng rãi cho bài toán lập lịch công việc trong điện toán sương mù - đám mây. Các tác giả của [43] đề xuất sử dụng giải thuật đàn kiến (Ant Colony Optimization - ACO) để cân bằng khối lượng công việc trong môi trường sương mù - đám mây cho các hệ thống lưới thông minh. Trong [44], bài toán của việc lập lịch tác vụ trong hệ thống sương mù - đám mây được mô hình hoá để cân bằng giữa thời gian xử lý và chi phí về tiền (hai ràng buộc). Trong [21], các tác giả sử dụng giải thuật đàn ong (Bee Life Algorithm - BLA) để tối ưu hoá thời gian thực thi và bộ nhớ được sử dụng (hai ràng buộc). Giải thuật PSO được đề xuất sử dụng trong [45] để tối ưu thời gian hoàn thành tác vụ trong môi trường sương mù - đám mây (một ràng buộc). Trong [46] các tác giả đã giới thiệu một giải thuật dựa trên phương pháp tiến hoá để tối ưu tương quan giữa tổng thời gian xử lý và chi phí về tiền (hai ràng buộc). Tuy nhiên vẫn chưa có nghiên cứu nào xem xét các ràng buộc giữa điện năng tiêu thụ, độ trễ dịch vụ và chi phí sử dụng cho các tính năng khác nhau của các nút sương mù/đám mây (truyền dữ liệu, tính toán, lưu trữ dữ liệu) cho bài toán lập lịch sử dụng các phương pháp metaheuristic.

3. Mô hình đề xuất

Trong phần này, chúng ta sẽ thảo luận về định nghĩa bài toán lập lịch tác vụ trong môi trường điện toán sương mù - đám mây và ba ràng buộc như đã trình bày ở các mục trên cùng với các giải thích và công thức của chúng.

3.1 Bài toán lập lịch trong điện toán sương mù - đám mây

Luận văn xem xét hệ thống sương mù - đám mây gồm ba tầng: thiết bị đầu cuối, sương mù và đám mây. Hình 3.1 thể hiện kiến trúc của một hệ thống sương mù - đám mây cùng với luồng lập lịch tác vụ. Các thiết bị đầu cuối gửi các tác vụ của chúng lên một Fog broker là nơi các lịch sẽ được quyết định cho các tác vụ sẽ được chạy trên nút sương mù hay đám mây nào. Trong thiết kế này, các nút sương mù truyền dữ liệu từ thiết bị đầu cuối tới đám mây, đồng thời cũng xử lý và lưu trữ cho các tính năng thông minh yêu cầu thời gian thực và phản hồi với độ trễ thấp. Các nút cũng giúp hệ thống tiết kiệm băng thông bằng việc giảm thiểu lượng dữ liệu được gửi tới tầng đám mây. Các nút đám mây là những máy chủ hiện đại hoặc là các trung tâm dữ liệu cho phép khả năng lưu trữ và xử lý một khối lượng vô cùng lớn. Chúng đảm nhận việc xử lý các yêu cầu có độ phức tạp. Mặc dù đối mặt với vấn đề nghiêm trọng về thời gian trễ, các đám mây vẫn đóng một vai trò không thể bỏ được nhờ vào khả năng tính toán to lớn. Trong hệ thống này, khả năng tính toán được cung cấp bởi các nút sương mù và đám mây mà các nút sương mù có khả năng tính toán và lưu trữ dữ liệu yếu hơn các nút đám mây rất nhiều. Tuy nhiên, các



Hình 3.1: Kiến trúc hệ thống

nút sương mù lại có chi phí rẻ hơn rất nhiều cũng như độ trễ truyền thông nhỏ hơn các nút đám mây. Các yêu cầu xử lý dữ liệu và các phản hồi được gửi đi hoặc nhận lại từ các thiết bị đầu cuối tới các nút sương mù, đám mây và ngược lại được chia thành một tập hợp n tác vụ nhỏ và độc lập $T = \{t_1, t_2, ...t_n\}$. Mục tiêu lập lịch tác vụ là tối ưu hoá điện năng tiêu thụ, độ trễ truyền tin và tổng chi phí để hoàn thành tất cả các tác vụ trong các nút sương mù và đám mây sao cho hiệu quả nhất.

3.2 Yếu tố về điện năng tiêu thụ

Chúng ta định nghĩa D là tập hợp gồm các thiết bị đầu cuối có d_i là thiết bị thứ i. $P_r^{d_i}$ và $P_s^{d_i}$ lần lượt là lượng dữ liệu được sinh ra tại thiết bị d_i cần được xử lý và lưu trữ. $Q_r^{d_i}$ và $Q_s^{d_i}$ là lượng dữ liệu được sinh ra tại thiết bị d_i cần được xử lý và

lưu trữ tại các nút đám mây. Do đó, $P_r^{d_i} - Q_r^{d_i}$ và $P_s^{d_i} - Q_s^{d_i}$ là lượng dữ liệu tương ứng được xử lý và lưu trữ tại các nút sương mù được sinh ra từ thiết bị d_i. Gọi EXT là thời gian tất cả các tác vụ được thực hiện hoàn thành trong hệ thống. Công thức 3.1 tính toán tổng năng lượng tiêu thụ ψ để hoàn thành tất cả các tác vụ và được chia ra thành ba phần:

1. Điện năng tiêu thụ do việc truyền dữ liệu (ψ_{df}) giữa các tầng với nhau,
2. Điện năng tiêu thụ phục vụ tính toán (ψ_{cp}),
3. Điện năng tiêu thụ phục vụ lưu trữ (ψ_{st}).

Các điện năng tiêu thụ cho việc truyền dữ liệu, tính toán và lưu trữ trên các nút sương mù được xác định theo thứ tự bởi $\psi_{df}^{fog}(t)$, $\psi_{cp}^{fog}(t)$ và $\psi_{st}^{fog}(t)$. Tương tự vậy, với các dữ liệu có nhu cầu được xử lý tại các nút đám mây bằng những phân tích phức tạp và yêu cầu thời gian lưu trữ lâu dài sẽ cần mức năng lượng được tính bởi $\psi_{df}^{cloud}(t)$, $\psi_{cp}^{cloud}(t)$ và $\psi_{st}^{cloud}(t)$ cho việc truyền tải dữ liệu, tính toán và lưu trữ. Tổng điện năng tiêu thụ sẽ trong hệ thống sương mù - đám mây sẽ được biểu diễn bởi công thức dưới đây:

$$\begin{aligned}\psi &= \sum_{t=1}^{EXT} \left[\psi_{df}(t) + \psi_{cp}(t) + \psi_{st}(t)\right] \\ &= \sum_{t=1}^{EXT} \left[\psi_{df}^{fog}(t) + \psi_{df}^{cloud}(t) + \psi_{cp}^{fog}(t) + \psi_{cp}^{cloud}(t) + \psi_{st}^{fog}(t) + \psi_{st}^{cloud}(t)\right]\end{aligned} \tag{3.1}$$

3.2.1 Truyền tải dữ liệu

Năng lượng để truyền dữ liệu trong tầng sương mù $\psi_{df}^{fog}(t)$ và trong đám mây $\psi_{df}^{cloud}(t)$ được xác định bởi công thức 3.2 và 3.3 dựa trên số byte mà dữ liệu được truyền đi bởi các nút sương mù, đám mây.

$$\psi_{df}^{fog}(t) = \gamma_{idle}^{fog} + \gamma^{fog} \sum_{i=1}^{D} \left[P_r^{d_i}(t) - Q_r^{d_i}(t) + P_s^{d_i}(t) - Q_s^{d_i}(t)\right] \tag{3.2}$$

$$\psi_{df}^{cloud}(t) = (\gamma_{idle}^{fog} + \gamma_{idle}^{cloud}) + (\gamma^{fog} + \gamma^{cloud}) \sum_{i=1}^{D} [Q_r^{d_i}(t) + Q_s^{d_i}(t)] \quad (3.3)$$

với γ_{idle}^{fog} và γ_{idle}^{cloud} là mức năng lượng cần thiết cho các nút sương mù, đám mây ở trong thái tạm dừng. Trong khi đó, γ^{fog}, γ^{cloud} là năng lượng cần thiết để truyền đi một byte dữ liệu từ thiết bị đầu cuối tới một nút sương mù và một nút đám mây.

3.2.2 Xử lý

Điện năng tiêu thụ của việc xử lý một tác vụ trên một nút phụ thuộc vào lượng dữ liệu đi tới và lượng dữ liệu đã lưu trữ sẵn tại đó vì các nút sử dụng cả hai để xử lý công việc của nó. Dữ liệu càng cũ sẽ càng có ít ảnh hưởng tới các xử lý hiện tại nên nó sẽ càng ít ảnh hưởng tới điện năng tiêu thụ. Chính vì vậy ảnh hưởng của dữ liệu sẽ tỉ lệ nghịch tới thời gian tồn tại của nó và nghiên cứu chọn $\frac{1}{2^{(t-j)}}$ là hàm nghịch biến được sử dụng. Bởi vì các nút sương mù có khả năng lưu trữ hạn chế, nó không thể lưu trữ dữ liệu trong thời gian dài. Chính vì vậy, các nút sương mù có thêm một tham số τ là thời gian tồn tại của dữ liệu. Với các nút đám mây có sức mạnh lưu trữ to lớn, dữ liệu có thể tồn tại mãi mãi trên đây. Vì vậy τ được bỏ qua trên các nút đám mây. Điện năng tiêu thụ dùng để tính toán bởi các nút sương mù và đám mây được xác định bằng công thức 3.4 và 3.5.

$$\psi_{cp}^{fog}(t) = \beta_{idle}^{fog} + \beta^{fog} \sum_{i=1}^{D} \left\{ P_r^{d_i}(t) - Q_r^{d_i}(t) + \sum_{j=t-\tau}^{t-1} \frac{1}{2^{(t-j)}} [P_s^{d_i}(j) - Q_s^{d_i}(j)] \right\} \quad (3.4)$$

$$\psi_{cp}^{cloud}(t) = \beta_{idle}^{cloud} + \beta^{cloud} \sum_{i=1}^{D} [Q_r^{d_i}(t) + \sum_{j=1}^{t-1} \frac{1}{2^{(t-j)}} Q_s^{d_i}(j)] \quad (3.5)$$

với β^{fog} và β^{cloud} là điện năng cần thiết để xử lý một byte dữ liệu trên nút sương mù và đám mây. β_{idle}^{fog} và β_{idle}^{cloud} là điện năng tiêu thụ của các nút ở trạng thái ngủ.

3.2.3 Lưu trữ

Tương tự điện năng tiêu thụ cho công việc tính toán đã trình bày ở trên, điện năng tiêu thụ cho việc lưu trữ cũng phụ thuộc vào số lượng byte của dữ liệu được lưu trữ tại các nút sương mù hay đám mây. Điện năng tiêu thụ lưu trữ trên các nút lần lượt được tính dựa theo công thức 3.6 và 3.7.

$$\psi_{st}^{fog}(t) = \alpha_{idle}^{fog} + \alpha^{fog} \sum_{i=1}^{D} \sum_{j=t-\tau}^{t} [P_s^{d_i}(j) - Q_s^{d_i}(j)] \tag{3.6}$$

$$\psi_{st}^{cloud} = \alpha_{idle}^{cloud} + \alpha^{cloud} \sum_{i=1}^{D} \sum_{j=1}^{t} Q_s^{d_i}(j) \tag{3.7}$$

với α^{fog} và α^{cloud} là điện năng cần thiết để lưu trữ một byte dữ liệu trên một nút sương mù hay đám mây. α_{idle}^{fog} và α_{idle}^{cloud} là điện năng cần thiết cho các nút sương mù và đám mây ở trạng thái ngủ.

3.3 Yếu tố về độ trễ dịch vụ

Độ trễ dịch vụ cho một yêu cầu xử lý gửi bởi một thiết bị đầu cuối được đo lường bởi thời gian phản hồi của nó. Do đó, độ trễ được xác định bởi các độ trễ truyền tin và độ trễ xử lý cho một yêu cầu xử lý bằng công thức 3.8. Tương tự điện năng tiêu thụ, độ trễ truyền tin và xử lý dữ liệu được xem xét cho các nút sương mù và đám mây. Do đó độ trễ để hoàn thành tất cả các tác vụ được tính như sau:

$$\begin{aligned} \delta &= \sum_{t=1}^{EXE} [\delta_{tr}(t) + \delta_{pr}(t)] \\ &= \sum_{t=1}^{EXE} [\delta_{tr}^{fog}(t) + \delta_{tr}^{cloud}(t) + \delta_{pr}^{fog}(t) + \delta_{pr}^{cloud}(t)] \end{aligned} \tag{3.8}$$

3.3.1 Độ trễ truyền tin

Gọi δ_{df} và δ_{fc} là độ trễ trung bình để truyền một byte từ thiết bị đầu cuối tới các nút sương mù và từ các nút sương mù tới các nút đám mây. Độ trễ truyền tin trên các nút sương mù và đám mây được tính bởi công thức 3.9 và 3.10.

$$\delta_{tr}^{fog}(t) = \delta_{df} \sum_{i=1}^{D} [P_r^{d_i}(t) - Q_r^{d_i}(t) + P_s^{d_i}(t) - Q_s^{d_i}(t)] \tag{3.9}$$

$$\delta_{tr}^{cloud}(t) = (\delta_{df} + \delta_{fc}) \sum_{i=1}^{D} [Q_r^{d_i}(t) + Q_s^{d_i}(t)] \tag{3.10}$$

3.3.2 Độ trễ xử lý

Độ trễ xử lý của một tác vụ trên một nút sương mù phụ thuộc vào độ trễ để xử lý dữ liệu đi tới từ các thiết bị đầu cuối và các dữ liệu được lưu trữ tạm thời trên nút đó của các tác vụ đã được xử lý trong khoảng thời gian τ trước đó. Mặt khác, độ trễ xử lý của một tác vụ trên một nút đám mây phụ thuộc vào dữ liệu đi tới và dữ liệu của các tác vụ đã được lưu trữ trước đó kể từ thời điểm bắt đầu. Các số liệu này được tính bằng các công thức dưới đây:

$$\delta_{pr}^{fog}(t) = \lambda^{fog} \sum_{i=1}^{D} \left\{ P_r^{d_i}(t) - Q_r^{d_i}(t) + \sum_{j=t-\tau}^{t-1} \frac{1}{2^{(t-j)}} [P_s^{d_i}(j) - Q_s^{d_i}(j)] \right\} \tag{3.11}$$

$$\delta_{pr}^{cloud}(t) = \lambda^{cloud} \sum_{i=1}^{V} [Q_r^{d_i}(t) + \sum_{j=1}^{t-1} \frac{1}{2^{(t-j)}} Q_s^{d_i}(j)] \tag{3.12}$$

với λ^{fog} và λ^{cloud} là độ trễ trung bình để xử lý một byte trên một nút sương mù và một nút đám mây.

3.4 Yếu tố về chi phí

Một khía cạnh khác cần tối ưu trong hệ thống sương mù - đám mây là tổng chi phí cần thiết để hoàn thành tất cả các tác vụ được sinh ra. Khi một tác vụ được thực thi trong hệ thống sương mù - đám mây, có một lượng chi phí cho các công việc vận chuyển dữ liệu, xử lý và lưu trữ được tiêu tốn. Ta gọi các chi phí này trên các nút sương mù lần lượt là θ_{df}^{fog}, θ_{cp}^{fog} và θ_{st}^{fog}. Các chi phí trên các nút đám mây là θ_{df}^{cloud}, θ_{cp}^{cloud} và θ_{st}^{cloud}. Do đó, tổng chi phí để hoàn thành tất cả các tác vụ được tính như sau:

$$\begin{aligned}\theta &= \sum_{t=1}^{EXT} [\theta_{df}(t) + \theta_{cp}(t) + \theta_{st}(t)] \\ &= \sum_{t=1}^{EXT} [\theta_{df}^{fog}(t) + \theta_{df}^{cloud}(t) + \theta_{cp}^{fog}(t) + \theta_{cp}^{cloud}(t) + \theta_{st}^{fog}(t) + \theta_{st}^{cloud}(t)]\end{aligned} \tag{3.13}$$

3.4.1 Truyền tải dữ liệu

Công thức 3.14 tính chi phí cho việc vận chuyển dữ liệu trên các nút sương mù. Trong khi đó, chi phí cho việc chuyển dữ liệu trên các nút đám mây được tính bằng công thức 3.15 dựa trên tổng số lượng byte dữ liệu được truyền đi trên các nút sương mù và đám mây.

$$\theta_{df}^{fog}(t) = \sigma_{idle}^{fog} + \sigma^{fog} \sum_{i=1}^{D} [P_r^{d_i}(t) - Q_r^{d_i}(t) + P_s^{d_i}(t) - Q_s^{d_i}(t)] \tag{3.14}$$

$$\theta_{df}^{cloud}(t) = (\sigma_{idle}^{fog} + \sigma_{idle}^{cloud}) + (\sigma^{fog} + \sigma^{cloud}) \sum_{i=1}^{D} [Q_r^{d_i}(t) + Q_s^{d_i}(t)] \tag{3.15}$$

với σ^{fog} và σ^{cloud} tương ứng là các chi phí cần thiết để truyền một byte dữ liệu trên các nút sương mù và đám mây. σ_{idle}^{fog} và σ_{idle}^{cloud} là chi phí cần được yêu cầu cho các nút sương mù và đám mây khi ở trạng thái ngủ.

3.4.2 Xử lý

Chi phí xử lý phụ thuộc vào việc tính toán cần thiết và lượng dữ liệu được lưu trữ tạm thời trên các nút sương mù. Vì vậy, chi phí cho các nút sương mù và đám mây để xử lý tất cả các tác vụ tại thời điểm t được tính bằng công thức 3.16 và 3.17.

$$\theta_{cp}^{fog}(t) = \pi_{idle}^{fog} + \pi^{fog} \sum_{i=1}^{D} \left\{ P_r^{d_i} - Q_r^{d_i} + \sum_{j=t-\tau}^{t-1} \frac{1}{2^{(t-j)}} [P_s^{d_i}(j) - Q_s^{d_i}(j)] \right\} \tag{3.16}$$

$$\theta_{cp}^{cloud}(t) = \pi_{idle}^{cloud} + \pi^{cloud} \sum_{i=1}^{D} [Q_r^{d_i} + \sum_{j=1}^{t-1} \frac{1}{2^{(t-j)}} Q_s^{d_i}(j)] \tag{3.17}$$

với π là chi phí trung bình để xử lý một byte dữ liệu và π_{idle} là chi phí của các nút sương mù và đám mây trong trạng thái ngủ.

3.4.3 Lưu trữ

Chi phí cho việc lưu trữ dữ liệu được tính dựa trên tổng thời gian và lượng dữ liệu được lưu trữ trên các nút sương mù và đám mây. Các chi phí này xác định như sau:

$$\theta_{st}^{fog}(t) = \omega_{idle}^{fog} + \omega^{fog} \sum_{i=1}^{D} \sum_{j=t-\tau}^{t} [P_s^{d_i}(t) - Q_s^{d_i}(j)] \tag{3.18}$$

$$\theta_{st}^{cloud}(t) = \omega_{idle}^{cloud} + \omega^{cloud} \sum_{i=1}^{D} \sum_{j=1}^{t} Q_s^{d_i}(j) \tag{3.19}$$

với ω là chi phí trung bình để lưu trữ một byte dữ liệu và ω_{idle} là chi phí của các nút sương mù và đám mây khi chúng không lưu trữ bất kỳ dữ liệu nào.

3.5 Định nghĩa hàm mục tiêu

Trong mô hình môi trường sương mù - đám mây với nhiều ràng buộc, hàm mục tiêu được sử dụng nhằm tối ưu các yếu tố điện năng tiêu thụ (ψ), độ trễ dịch vụ (δ) và chi phí sử dụng (θ). Hàm F này được định nghĩa như sau:

$$F = \alpha * \frac{\min(\psi)}{\psi} + \beta * \frac{\min(\delta)}{\delta} + (1 - \alpha - \beta) * \frac{\min(\theta)}{\theta} \tag{3.20}$$

công thức 3.20 được xây dựng tương tự như trong nghiên cứu [47] và [46] với $\alpha, \beta \in [0, 1]$ là hệ số cân bằng giữa điện năng tiêu thụ, độ trễ dịch vụ và chi phí sử dụng. $\min(\psi)$, $\min(\delta)$, và $\min(\theta)$ là các giá trị nhỏ nhất của điện năng tiêu thụ, độ trễ dịch vụ và chi phí sử dụng tương ứng. Sử dụng phép chia giúp chúng ta loại bỏ được thứ nguyên của các thành phần đồng thời đưa giá trị của chúng về cùng một khoảng.

Việc đi tìm các giá trị tối ưu một bài toán NP-đầy đủ [12] và thời gian để tìm ra lời giải là rất lớn. Với các giải thuật metaheuristic, chúng ta dễ dàng tìm được lời giải tốt (tuy không phải tốt nhất) gần với các giá trị tối ưu này trong thời gian ngắn. Chính vì vậy, trong thiết kế này, $\min(\psi)$, $\min(\delta)$ và $\min(\theta)$ là giá trị được tối ưu riêng lẻ nhờ sử dụng các giải thuật metaheuristic. Chúng tôi chạy các giả lập nhiều lần với các thuật toán khác nhau và lấy ra giá trị nhỏ nhất của chúng. Điều này không thực sự tìm được cực tiểu toàn cục của chúng nhưng lại giúp chúng ta dễ dàng so sánh sự hiệu quả giữa các giải thuật với nhau. Trong trường hợp α và β tăng, chiến lược lập lịch của chúng ta sẽ tập trung hơn vào cực tiểu điện năng tiêu thụ và độ trễ dịch vụ. Nếu α và β tiến gần về không, chiến lược sẽ tập trung vào tối ưu chi phí sử dụng hơn là những tiêu chí khác. Theo cách này, mục tiêu của hàm F là đi tìm lời giải tối ưu sao cho điện năng tiêu thụ, độ trễ dịch vụ và chi phí sử dụng gần với giá trị cực tiểu nhất. Kết quả là hàm mục tiêu F càng lớn, lời giải thu được càng gần với điểm tối ưu.

Do đó, bài toán lập lịch tác vụ trong hệ thống sương mù - đám mây được công thức hoá như sau:

Đầu vào: $T = \{T_1, T_2, ..., T_n\}$, với T là một tập hợp các tác vụ độc lập;
Đầu ra: Là sự phân công các tác vụ được xử lý tại các nút sương mù và đám mây;
Mục tiêu: Cực đại hoá hàm mục tiêu F: $F \rightarrow Max$.

4. Thử nghiệm và đánh giá

4.1 Cài đặt thử nghiệm

Trong các thử nghiệm, chúng tôi giả lập một hệ thống sương mù - đám mây với hai mươi nút sương mù và năm nút đám mây. Do khả năng về tài nguyên khác nhau, các nút sương mù và đám mây có các cấu hình và thông số đa dạng như được mô tả trong các bảng 4.1, 4.2 và 4.3. Các đặc trưng của mỗi nút (ví dụ như lượng dữ liệu được sinh ra, tuổi đời của dữ liệu, điện năng tiêu thụ...) được sinh ra ngẫu nhiên trong các khoảng giá trị được cho trong các bảng. Các khoảng này đã được tham khảo từ một số nghiên cứu trước đó như [3], [48] hay [46]. Chúng tôi cũng xem xét hai trạng thái của các nút sương mù và đám mây gồm trạng thái ngủ và trạng thái hoạt động là hai trạng thái trong thực tế của hệ thống sương mù - đám mây. Một nút được đặt về trạng thái ngủ nếu nó không xử lý, lưu trữ hay vận chuyển dữ liệu. Trái lại, một nút được thiết lập về trạng thái hoạt động nếu nó đang tiến hành xử lý hay lưu trữ hay vận chuyển dữ liệu. Thời gian sống của dữ liệu được lưu trữ trên các nút sương mù được ký hiệu bởi tham số τ được sinh ngẫu nhiên trong khoảng $[0, 20]$. Trong khi đó, các nút đám mây lưu trữ dữ liệu lâu dài mà không cần quan tâm tới thời gian sống.

Chúng tôi sinh ra mười tập dữ liệu từ 50 đến 500 tác vụ để kiểm định hệ thống sương mù - đám mây được giả lập. Chúng tôi tìm lời giải tốt nhất cho bài toán lập lịch tác vụ bằng cách sử dụng các phương pháp metaheuristic đối với mỗi tập dữ liệu được kiểm thử. Đối với thiết kế của mô hình hệ thống sương mù - đám mây, chúng tôi quan sát mối quan hệ giữa ba ràng buộc môi trường là điện năng

Bảng 4.1: Thiết lập môi trường các tác vụ

Tham số	Ký hiệu	Phạm vi giá trị	Đơn vị
Dữ liệu được sinh ra tại các thiết bị	$P_r^{d_i}$ $P_s^{d_i}$	$[0.2 * 10^6, 20 * 10^6]$	byte
Dữ liệu được xử lý tại đám mây	$Q_r^{d_i}$ $Q_r^{d_i}$	$[0.1 * 10^6, 10 * 10^6]$	

Bảng 4.2: Thiết lập môi trường các nút sương mù

Tham số	Ký hiệu	Phạm vi giá trị	Đơn vị
Năng lượng tiêu thụ truyền tin	γ^{fog} γ_{idle}^{fog}	$[5 * 10^{-8}, 5 * 10^{-6}]$ $[25, 75]$	W/byte W
Năng lượng tiêu thụ tính toán	β^{fog} β_{idle}^{fog}	$[5 * 10^{-7}, 5 * 10^{-4}]$ $[100, 500]$	W/byte W
Năng lượng tiêu thụ lưu trữ	α^{fog} α_{idle}^{fog}	$[5 * 10^{-8}, 5 * 10^{-5}]$ $[10, 100]$	W/byte W
Độ trễ truyền tin	δ_{df}	$[0.005, 0.05]$	s/byte
Độ trễ xử lý	λ^{fog}	$[10^{-7}, 10^{-6}]$	
Chi phí truyền tin	σ^{fog} σ_{idle}^{fog}	$[5 * 10^{-10}, 5 * 10^{-9}]$ $[0.001, 0.01]$	\$/byte \$/s
Chi phí tính toán	π^{fog} π_{idle}^{fog}	$[1.8 * 10^{-15}, 1.8 * 10^{-14}]$ $[1.8 * 10^{-7}, 5.5 * 10^{-7}]$	\$/byte \$/s
Chi phí lưu trữ	ω^{fog} ω_{idle}^{fog}	$[10^{-16}, 10^{-15}]$ $[10^{-8}, 2 * 10^{-8}]$	\$/byte \$/s

Bảng 4.3: Thiết lập môi trường các nút đám mây

Tham số	Ký hiệu	Phạm vi giá trị	Đơn vị
Năng lượng tiêu thụ truyền tin	γ^{cloud}	$[5 * 10^{-7}, 5 * 10^{-5}]$	W/byte
	γ^{cloud}_{idle}	$[100, 200]$	W
Năng lượng tiêu thụ tính toán	β^{cloud}	$[10^{-9}, 10^{-6}]$	W/byte
	β^{cloud}_{idle}	$[50, 200]$	W
Năng lượng tiêu thụ lưu trữ	α^{cloud}	$[5 * 10^{-9}, 5 * 10^{-6}]$	W/byte
	α^{cloud}_{idle}	$[30, 200]$	W
Độ trễ truyền tin	δ_{fc}	$[0.2, 5.0]$	s/byte
Độ trễ xử lý	λ^{cloud}	$[10^{-8}, 10^{-7}]$	
Chi phí truyền tin	σ^{cloud}	$[5 * 10^{-9}, 5 * 10^{-8}]$	\$/byte
	σ^{cloud}_{idle}	$[0.02, 0.1]$	\$/s
Chi phí tính toán	π^{cloud}	$[3.6 * 10^{-15}, 3.6 * 10^{-14}]$	\$/byte
	π^{cloud}_{idle}	$[3.6 * 10^{-7}, 1.1 * 10^{-6}]$	\$/s
Chi phí lưu trữ	ω^{cloud}	$[2 * 10^{-16}, 2 * 10^{-15}]$	\$/byte
	ω^{cloud}_{idle}	$[2 * 10^{-8}, 4 * 10^{-8}]$	\$/s

Bảng 4.4: Tham số các giải thuật metaheuristic

Tham số		BLA	GA	PSO	WOA
Số lần chạy		10	10	10	10
Kích thước quần thể (N)	Ong chúa	1	100	100	100
	Ong đực	39			
	Ong thợ	60			
Tỷ lệ lai ghép		0.9	0.9	$c_1 = c_2 = 1.5$	$a = 2 \rightarrow 0$
Tỷ lệ đột biến		0.05	0.05	$w = 0.9 \rightarrow 0.1$	
Số thế hệ		200	200	200	200

tiêu thụ, độ trễ dịch vụ và chi phí trong giả lập của chúng tôi. Kiểm thử theo hướng tiếp cận bằng các phương pháp metaheuristic cho bài toán lập lịch trong điện toán sương mù - đám mây, chúng tôi đánh giá một số giải thuật gồm GA, PSO, BLA và WOA. Chúng tôi cũng so sánh kết quả đạt được với cơ chế Round-Robin để đánh giá sự hiệu quả. Các giải thuật được kiểm thử là những giải thuật nổi tiếng và đã được sử dụng trong nhiều lĩnh vực trước đó như đã đề cập trong chương 3 trước đó.

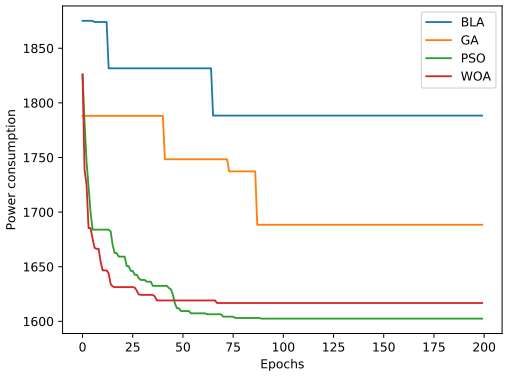
Theo công thức 3.20, đầu tiên chúng tôi sử dụng một thuật toán metaheuristic để tìm các giá trị cực tiểu cho mỗi ràng buộc trong mỗi tập dữ liệu được kiểm thử. Sau đó, các giá trị thu được sẽ được sử dụng để thực hiện tiến trình lập lịch tác vụ của hệ thống sương mù - đám mây. Để đánh giá cơ chế Round-Robin, độ phù hợp được xác định như sau: Các giá trị thu được bởi Round-Robin và các giải thuật metaheuristic trước đó được sử dụng cùng nhau trong công thức 3.20. Bảng 4.4 thể hiện các tham số cài đặt của các phương pháp metaheuristic trong các thử nghiệm. Chúng tôi đã chạy bốn giải thuật là BLA, GA, PSO và WOA để giải bài toán lập lịch tác vụ. Với mỗi giải thuật, chúng tôi thử nghiệm với tối đa số thế hệ là 200 trên quần thể gồm 100 cá thể mỗi thế hệ. Trong trường hợp của BLA và GA, chúng tôi thiết lập tỉ lệ ghép đôi là 0,9 và tỉ lệ đột biến là 0,05. Với PSO, chúng tôi thiết lập hệ số quán tính w giảm gần từ 0,9 đến 0,1 để cân bằng giữa tìm kiếm toàn cục và tìm kiếm cục bộ qua mỗi vòng lặp. Chúng tôi cũng thiết lập các hệ số gia tốc c_1 và c_2 là $1,5$. Giá trị c_1 là mức độ ảnh hưởng của vị trí tốt nhất trong quần thể và c_2 là mức độ ảnh hưởng của tất cả vị trí của cả quần thể. Cuối cùng với WOA, chúng tôi thiết lập a giảm dần từ 2 về 0 để cân bằng giữa pha thăm dò và pha khai thác.

4.2 Tối ưu hoá các ràng buộc

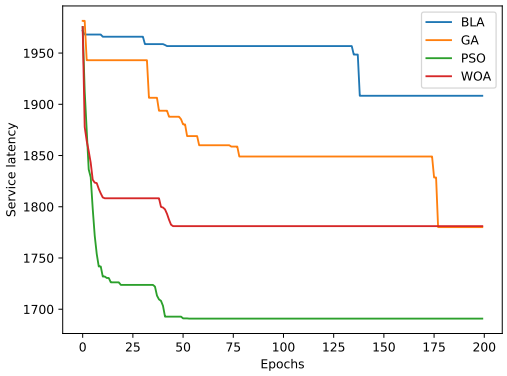
Mục tiêu của bài kiểm thử này là đánh giá khả năng tìm thấy các giá trị cực tiểu cho mỗi ràng buộc ψ, δ và θ theo như công thức 3.20 với các bộ dữ liệu khác nhau. Do đó, chúng tôi thực hiện kiểm thử với tất cả các phương pháp metaheuristic đã đề cập trong mục trước. Bảng 4.5 thể hiện các giá trị nhỏ nhất thu được của mỗi ràng buộc. Trong khi đó, hình 4.1 minh hoạ quá trình tối ưu của ba ràng buộc trong

trường hợp sử dụng 200 tác vụ. Giá trị cực tiểu của ràng buộc về điện năng tiêu thụ (được thể hiện trong hình 4.1a) đạt 1602,49 *Wh* sau 90 vòng lặp. Tương tự thế, các giá trị cực tiểu của độ trễ dịch vụ (hình 4.1b) và chi phí (hình 4.1c) tương ứng là 1690,83 *s* và 673,48 $ sau 54 và 98 vòng lặp.

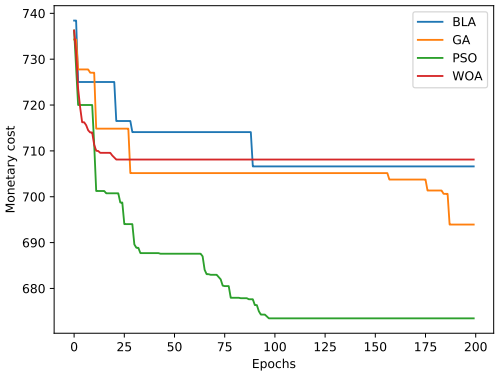
Thông qua những kết quả của các thử nghiệm này, chúng ta có thể quan sát các giá trị cực tiểu của các giải thuật metaheuristic cho bài toán tối ưu đơn ràng buộc. Các giá trị thu được của quá trình này sẽ được sử dụng để tính hàm F theo công thức 3.20 và được mô tả trong các thử nghiệm sau đây.



(a) Điện năng tiêu thụ



(b) Độ trễ dịch vụ



(c) Chi phí sử dụng

Hình 4.1: Quá trình tối ưu cho mỗi ràng buộc

Bảng 4.5: Giá trị nhỏ nhất cho mỗi ràng buộc

Số lượng tác vụ	**Điện năng tiêu thụ** (*Wh*)	**Độ trễ dịch vụ** (*s*)	**Chi phí sử dụng** ($)
50	300,65	316,44	127,61
100	692,91	697,29	311,54
150	1135,5	1173,7	443,05
200	1602,49	1690,83	673,48
250	2126,06	2236,4	823,29
300	2701,48	2637,48	965,28
350	3129,01	3229,54	1174,04
400	3655,38	3690,02	1309,28
450	4246,62	4003,53	1455,65
500	4828,81	4717,58	1631,34

4.3 Hướng tiếp cận metaheuristic cho bài toán lập lịch tác vụ

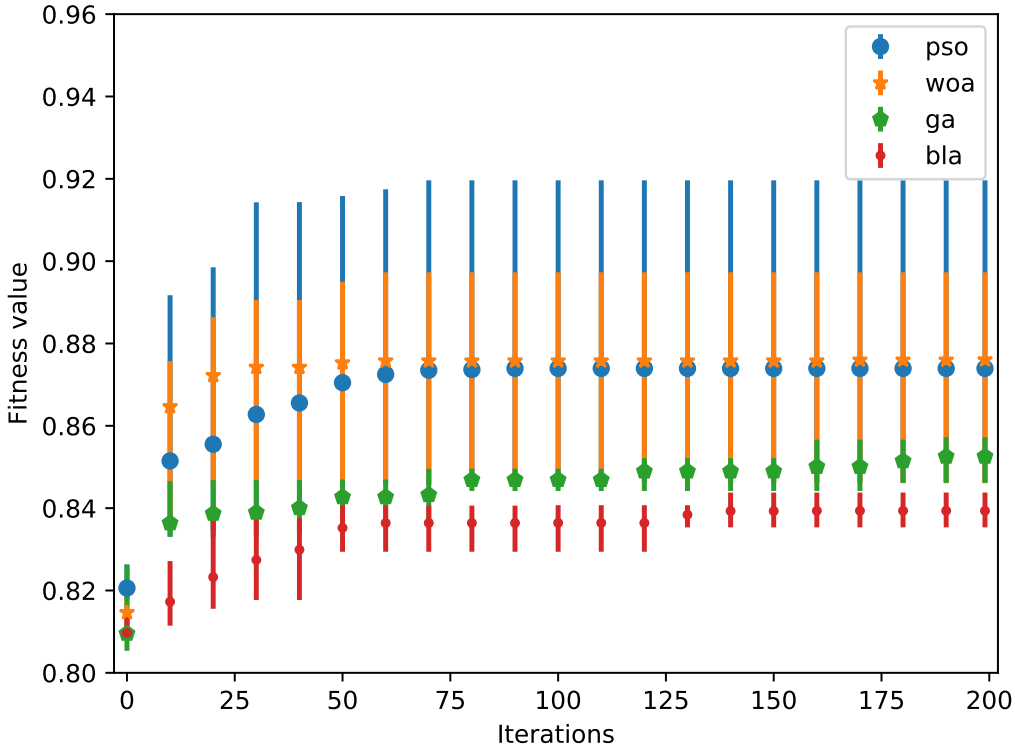
Mục tiêu của thử nghiệm này là đánh giá khả năng hội tụ của các giải thuật metaheuristic khi lập lịch các tác vụ trong môi trường sương mù - đám mây của chúng tôi. Trong hướng đi này, chúng tôi đã thực hiện các cơ chế lập lịch sử dụng GA, PSO, WOA, BLA và Round-Robin với mười bộ dữ liệu (từ 50 đến 500 tác vụ). Chúng tôi xem xét hai kịch bản lập lịch để đánh giá các phương pháp đã sử dụng có hoặc không có giới hạn về thời gian thực thi thuật toán. Bởi vì không hề có bất kỳ tiêu chuẩn nào về cận thời gian cho việc lập lịch trong các hệ thống sương mù - đám mây, chúng tôi đặt 30 giây là giới hạn thời gian cho mỗi lần lập lịch 50 tác vụ. Lưu ý rằng cấu hình thời gian được sử dụng để đánh giá sự hiệu quả của các phương pháp. Trong thực tế, giới hạn thời gian có thể được thay đổi tuỳ theo điều kiện môi trường. Các cấu hình thời gian cho các tập dữ liệu khác nhau được sử dụng trong các thử nghiệm của chúng tôi được trình bày trong bảng 4.6. Những giá trị cận thời gian được sử dụng chỉ cho việc đánh giá các giải thuật metaheuristic.

Trong khi đó, bởi vì Round Robin không chạy bất kỳ cơ chế tối ưu nào, chúng tôi đã không đặt các giá trị cận thời gian cho chiến lược này.

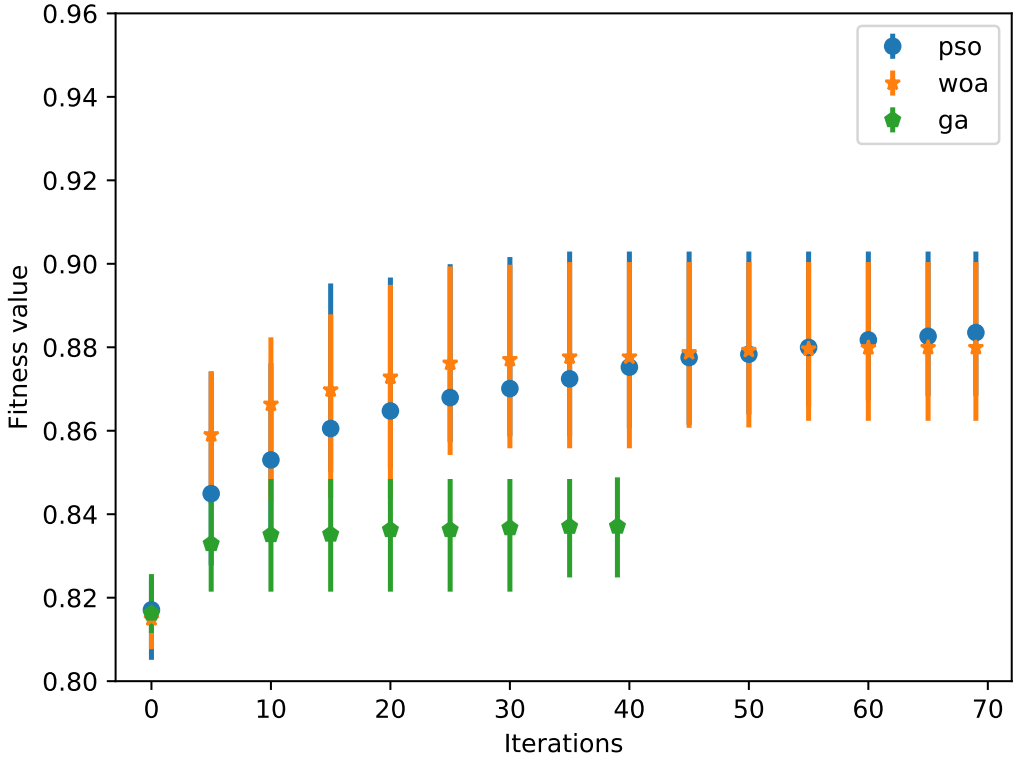
Hình 4.2 và hình 4.3 thể hiện các đầu ra tương ứng thu được trên 150 tác vụ cùng với có và không có điều kiện giới hạn thời gian. Tỉ lệ giữa α, β đã định nghĩa trong công thức 3.20 được đặt bằng $1/3$ (ba ràng buộc là tương đương). Chúng tôi quan sát quần thể của tất cả các phương pháp metaheuristic sau mỗi mười thế hệ trong các thử nghiệm không có giới hạn thời gian và mỗi năm thế hệ trong các thử nghiệm có giới hạn thời gian. Trong các hình này, với mỗi quan sát, trong đó điểm ở giữa đường thể hiện giá trị trung bình, đỉnh và đáy của các đường thể hiện giá trị thích nghi tốt nhất và tệ nhất. Thông qua kết quả đầu ra, các thử nghiệm của chúng tôi chỉ ra rằng các giải thuật metaheuristic được dùng có thể tìm ra được các giá trị tối ưu cho bài toán lập lịch tác vụ. Đặc biệt với các bài kiểm thử không có giới hạn thời gian, PSO mang lại lời giải tốt nhất với đầu ra $0,9196$ chỉ với 70 vòng lặp. Với các giá trị thích nghi trung bình, WOA mang lại giá trị tốt hơn một chút $(0,876)$ so với PSO $(0,8739)$. Trong khi đó, GA và BLA không đủ tốt so với hai giải thuật này.

Trong quá trình thử nghiệm với giới hạn thời gian (hình 4.3), chúng tôi nhận ra rằng BLA cần rất nhiều thời gian cho quá trình tối ưu của nó. Do đó, thời gian thiết yếu để lập lịch tác vụ sử dụng BLA luôn vượt quá các giá trị chặn thời gian. Chính vì vậy, trong các thử nghiệm của chúng tôi, chúng tôi không đánh giá sự hội tụ của BLA với điều kiện chặn về mặt thời gian. Trong thử nghiệm này, PSO và WOA cho thấy chúng có lợi thế về thời gian so với GA. Thật vậy, với các giá trị về cận thời gian giống nhau của 150 tác vụ, PSO và WOA có thể chạy 70 thế hệ trong khi GA chỉ có thể chạy được 40 thế hệ. Một nhận xét quan trọng nữa là PSO và WOA không chỉ có thể chạy được nhiều thế hệ hơn mà còn có thể cho kết quả tốt hơn GA. Cụ thể là với giá trị độ thích nghi trung bình, PSO và WOA có kết quả tốt hơn, $0,883$ và $0,8799$ khi so sánh với các giải thuật khác. Trong khi đó, sau 40 thế hệ, GA cho kết quả lời giải tốt nhất là $0,8362$.

Hình 4.4 cho thấy các giá trị độ thích nghi trung bình qua mười lần chạy của các phương pháp đã được kiểm thử có giới hạn thời gian. Ở đây, chúng tôi

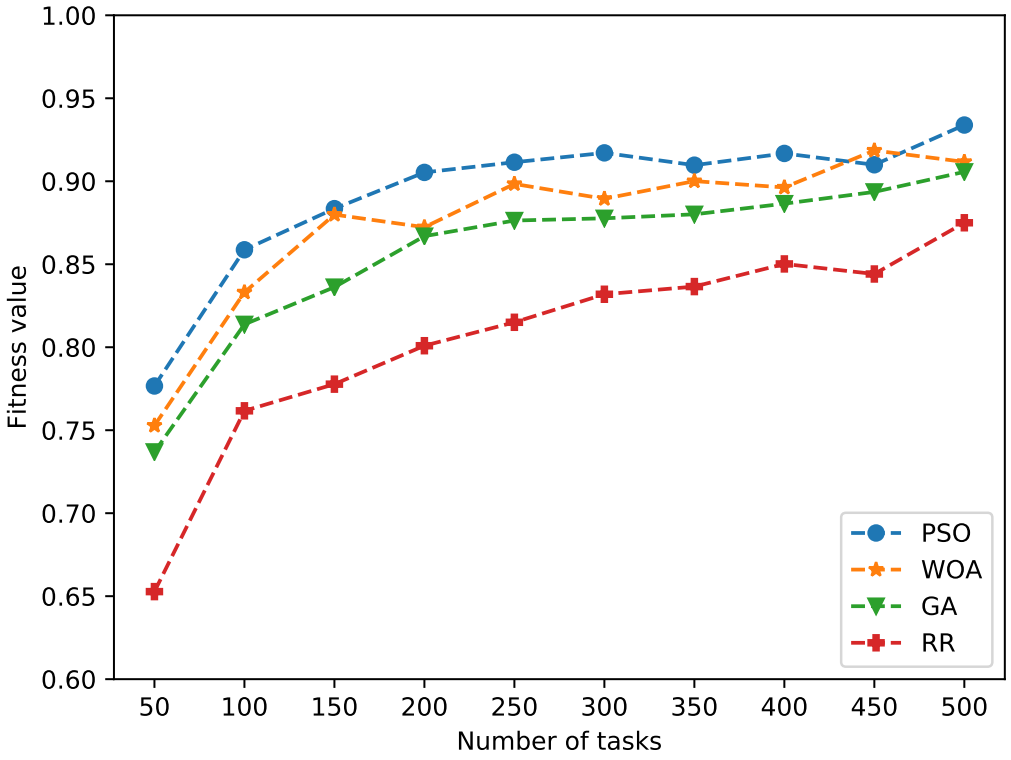


Hình 4.2: Đánh giá sự hội tụ (không giới hạn thời gian)

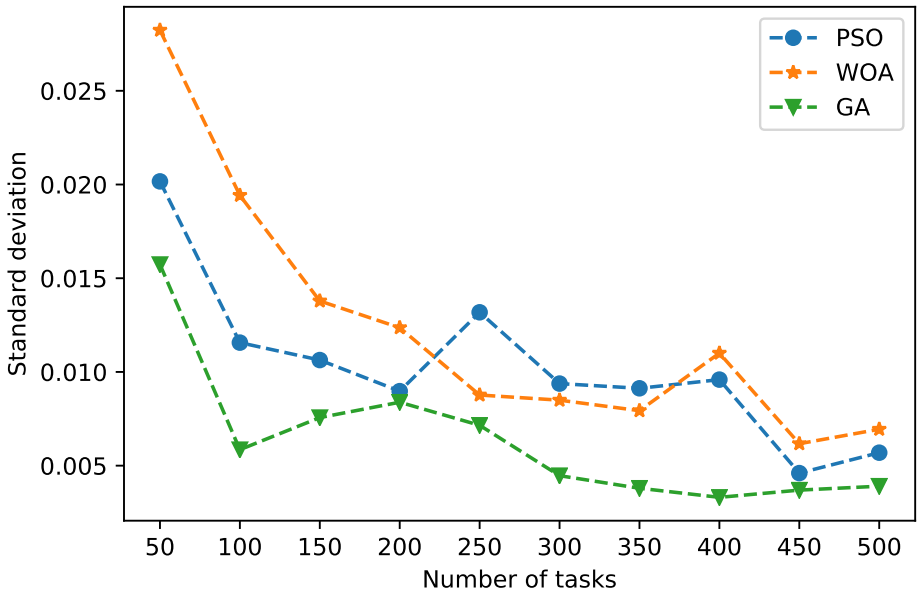


Hình 4.3: Đánh giá sự hội tụ có giới hạn thời gian

cũng đánh giá cơ chế Round-Robin trong so sánh với các giải thuật metaheuristic. $\alpha = \beta = 1/3$ cũng được thiết lập trong bài thử nghiệm này. Kết quả đạt được cho



Hình 4.4: Đánh giá độ thích nghi có giới hạn thời gian



Hình 4.5: Độ lệch chuẩn của độ thích nghi có giới hạn thời gian

thấy rằng việc sử dụng các giải thuật metaheuristic mang lại độ thích nghi tốt hơn so với việc sử dụng Round-Robin (đường màu đỏ), đặc biệt khi so sánh với PSO và WOA (đường màu xanh nước biển và màu cam). Lưu ý rằng trong thử nghiệm này, giá trị độ thích nghi cao hơn cho chiến lược lập lịch tốt hơn. Cụ thể, với 200 tác

vụ, các giá trị độ thích nghi PSO và WOA tương ứng là $0,9053$ và $0,8723$. Trong khi đó, giá trị thu được của cơ chế Round-Robin chỉ là $0,8009$. Hình 4.5 là độ lệch chuẩn của thử nghiệm này (ứng với hình 4.4). Giá trị độ lệch chuẩn thấp cho thấy sự ổn định của các giải thuật, đồng thời, GA là giải thuật có sự ổn định hơn hai giải thuật còn lại.

Bảng 4.7, 4.8, 4.9 thể hiện các giá trị trung bình thu được của các ràng buộc về điện năng tiêu thụ, độ trễ dịch vụ và chi phí sử dụng của mười tập dữ liệu khi lập lịch tác vụ sử dụng PSO, WOA, GA, BLA và Round-Robin với cận thời gian. Chúng tôi thực hiện mười lần cho mỗi tập dữ liệu và phương pháp để tính các giá trị trung bình. Trong các bảng này, giá trị nhỏ hơn là tốt hơn. Có thể nhận thấy một số quan sát quan trọng như sau. BLA cũng cần nhiều thời gian cho mỗi vòng lặp, vì vậy nó không thể đạt được một lời giải tối ưu trong một khoảng thời gian giới hạn (như đã mô tả ở trên). PSO, WOA và GA mang lại những kết quả tốt khi so sánh với cơ chế Round-Robin trong tất cả các trường hợp đã kiểm thử. Với giới hạn về thời gian, PSO và WOA cung cấp sự ổn định nhiều hơn nhờ quá trình tối ưu hoá diễn ra với nhiều lần lặp hơn GA. Những kết quả cuối cùng của các ràng buộc điện năng tiêu thụ, độ trễ dịch vụ và chi phí sử dụng với PSO và WOA cũng cho kết quả tốt hơn các kết quả của GA trên tất cả các tập dữ liệu.

Thông qua các kết quả đánh giá độ thích nghi (đã trình bày phía trên) và việc tối ưu hoá các ràng buộc (Mục 4.2), giải pháp của chúng tôi sử dụng các giải thuật metaheuristic cho bài toán lập lịch tác vụ trong môi trường nhiều ràng buộc đã chứng minh hiệu quả đáng kể của nó khi so sánh với cơ chế truyền thống (ví dụ Round-Robin).

4.4 Mối quan hệ giữa các ràng buộc

Trong thử nghiệm này, chúng tôi sử dụng PSO (giải thuật mang kết quả tối ưu tốt nhất như đã trình bày ở những thử nghiệm trước đó) cho việc lập lịch 150 tác vụ. α và β được thay đổi để đánh giá các ràng buộc về điện năng tiêu thụ, độ

Bảng 4.6: Thiết lập về giới hạn thời gian

Số lượng tác vụ	**Giới hạn thời gian (giây)**
50	30
100	60
150	90
200	120
250	150
300	180
350	210
400	240
450	270
500	300

Bảng 4.7: Giá trị tối ưu của điện năng tiêu thụ

Số lượng tác vụ	**Điện năng tiêu thụ** (*Wh*)				
	PSO	WOA	GA	BLA	RR
50	**386,57**	395,1	395,8	N/A	426,99
100	**818,26**	861,97	851,01	N/A	886,44
150	**1243,58**	1288,03	1346,96	N/A	1396,33
200	**1734,74**	1858,42	1872,74	N/A	1995,91
250	**2303,49**	2407,75	2503,27	N/A	2617,13
300	**2870,41**	2971,48	3015,51	N/A	3120,02
350	**3448,75**	3504,35	3606,59	N/A	3743,02
400	**3985,26**	4013,79	4114,79	N/A	4192,97
450	4589,45	**4513,65**	4712,47	N/A	4945,6
500	**5190,61**	5361,97	5408,55	N/A	5466,25

Bảng 4.8: Giá trị tối ưu của độ trễ dịch vụ

Số lượng tác vụ	**Độ trễ dịch vụ** (*s*)				
	PSO	WOA	GA	BLA	RR
50	**399,12**	406,97	431,68	N/A	485,54
100	875,53	**862,92**	907,89	N/A	964,41
150	1340,1	**1329,48**	1391,03	N/A	1524,12
200	**1933,83**	2027,18	2007,17	N/A	2157,8
250	**2445,13**	2481,51	2532,62	N/A	2709,13
300	**2858,29**	2990,65	3007,8	N/A	3180,14
350	**3506,59**	3570,85	3645,24	N/A	3832,06
400	**3989,62**	4130,27	4171,1	N/A	4300,4
450	4426,11	**4372,02**	4494,24	N/A	4668,95
500	**4958,39**	5076,73	5106,87	N/A	5302,26

Bảng 4.9: Giá trị tối ưu của chi phí sử dụng

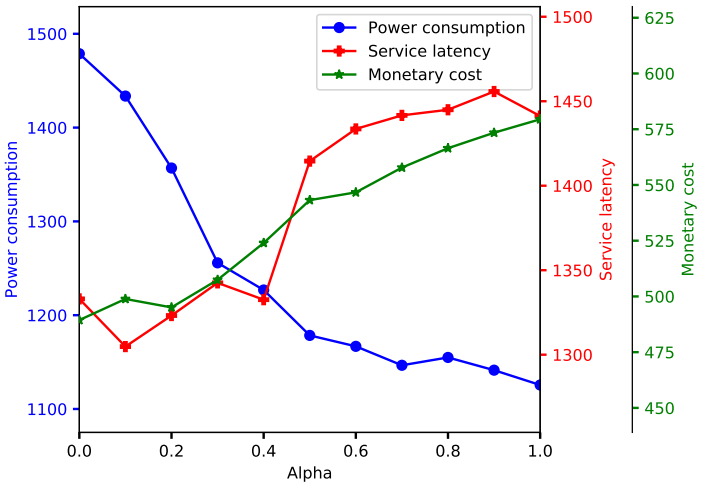
Số lượng Tác vụ	**chi phí sử dụng** ($)				
	PSO	WOA	GA	BLA	RR
50	**168,05**	177,27	177,7	N/A	211,83
100	**333,94**	351,1	362,65	N/A	399,23
150	515,13	**506,12**	538,95	N/A	590,76
200	733,72	**731,54**	746,1	N/A	825,08
250	918,04	**903,81**	918,19	N/A	1019,93
300	**1087,75**	1100,3	1122,05	N/A	1205,79
350	1303,34	**1300,23**	1324,04	N/A	1413,0
400	**1441,68**	1479,87	1476,97	N/A	1595,31
450	**1617,46**	1619,07	1637,8	N/A	1783,41
500	**1773,39**	1802,22	1811,81	N/A	1915,06

trễ dịch vụ và chi phí sử dụng. Hình 4.6 và 4.7 minh hoạ cho các kết quả thu được và cũng là mối liên hệ giữa α và β trong trường hợp lập lịch tác vụ có và không có giới hạn về thời gian.

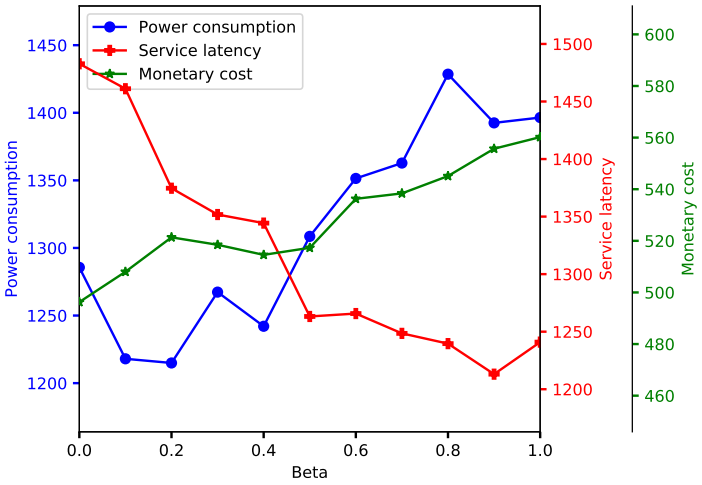
Trong kịch bản kiểm thử thứ nhất, chúng tôi thay đổi dần dần α (hệ số của điện năng tiêu thụ) và thiết lập các giá trị hệ số của độ trễ dịch vụ và chi phí sử dụng là như nhau. Các kết quả của thử nghiệm này được minh hoạ trong hình 4.6a và 4.7a. Trong cả hai trường hợp, khi thay đổi α từ 0 tới 1, điện năng tiêu thụ (đường màu xanh) có xu hướng giảm. Trong khi đó, độ trễ dịch vụ (đường màu đỏ) và chi phí dịch vụ (đường màu xanh lá) tăng lên. Trong kịch bản thứ hai, β được thay đổi từ 0 tới 1 và các giá trị hệ số của điện năng tiêu thụ và chi phí sử dụng được cài đặt như nhau. Những kết quả thu được của việc lập lịch với hạn chế về thời gian được trình bày ở hình 4.6b. Trong đó, độ trễ dịch vụ giảm từ 1482,7879 (s) đến 1240,9322 (s), trong khi đó điện năng tiêu thụ tăng từ 1285,6203 (Wh) đến 1396,4628 (Wh) và chi phí sử dụng tăng từ 496,2646 (\$) đến 560,1893 (\$). Trong trường hợp khác, việc lập lịch tác vụ được giải bằng PSO không có cận thời gian. Độ trễ dịch vụ giảm từ 1460,9753 (s) còn 1231,467 (s), trong khi đó điện năng tiêu thụ tăng từ 1174,3961 (Wh) lên 1451,8769 (Wh) và chi phí sử dụng tăng từ 518,8534 (\$) lên 562,4542 (\$).

Các kết quả thu được của kịch bản thứ ba được thể hiện trong hình 4.7c. Các giá trị hệ số của chi phí sử dụng $(1-\alpha-\beta)$ được thay đổi với α và β như nhau. Khi α và β tăng, hệ số của chi phí sử dụng giảm. Một quan sát có thể nhận thấy từ thử nghiệm này. Giá trị của chi phí sử dụng tỷ lệ thuận với giá trị của α và β, trong khi các giá trị của điện năng tiêu thụ và độ trễ dịch vụ tỷ lệ nghịch với chi phí sử dụng.

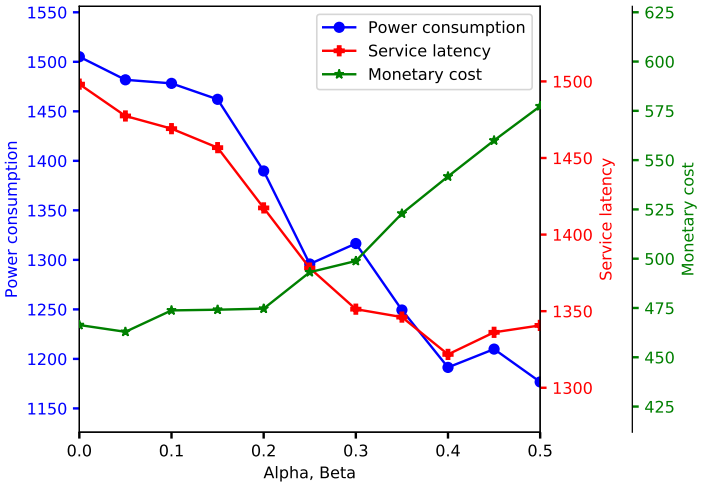
Chúng ta có thể đưa ra một kết luận từ thử nghiệm này như sau. Bằng việc điều chỉnh hệ số cân bằng α và β, mô hình của chúng ta có thể thay đổi linh hoạt đáp ứng các yêu cầu của người dùng khi họ quan tâm đặc biệt tới một ràng buộc nào đó.



(a) Điện năng tiêu thụ

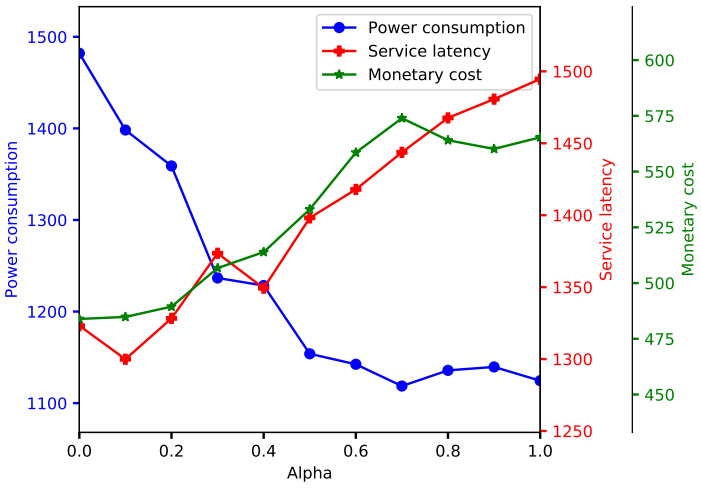


(b) Độ trễ dịch vụ

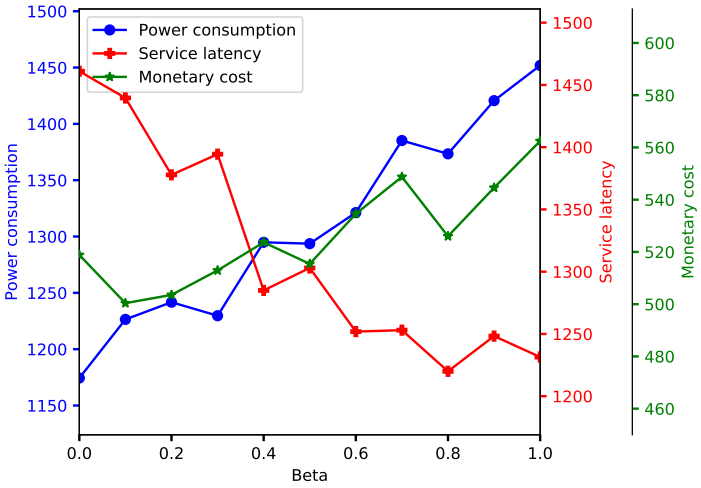


(c) chi phí sử dụng

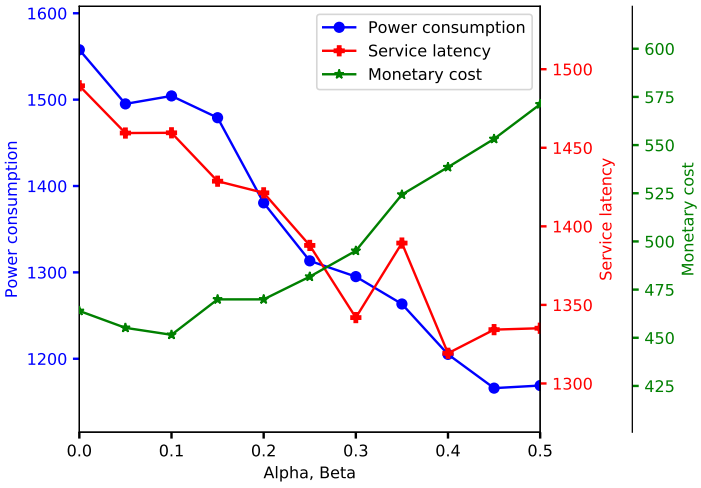
Hình 4.6: Tương quan giữa các ràng buộc với 150 tác vụ có giới hạn thời gian



(a) Điện năng tiêu thụ



(b) Độ trễ dịch vụ



(c) chi phí sử dụng

Hình 4.7: Tương quan giữa các ràng buộc với 150 tác vụ không có giới hạn thời gian

5. Kết luận

Trong luận văn này, chúng tôi đã giải quyết bài toán lập lịch tác vụ trong hệ thống sương mù - đám mây với nhiều các ràng buộc môi trường. Theo hướng này, chúng tôi đã mô hình hoá ba ràng buộc của điện toán sương mù - đám mây gồm có: điện năng tiêu thụ, độ trễ dịch vụ và chi phí sử dụng theo các tính năng của các nút sương mù và đám mây: vận chuyển dữ liệu, tính toán và lưu trữ dữ liệu. Dựa trên mô hình đã đề xuất, tôi đã định nghĩa hàm mục tiêu được sử dụng để tính toán mối quan hệ giữa ba ràng buộc đó. Bài toán lập lịch tác vụ được xem xét như một quá trình tối ưu hoá đa mục tiêu và có thể giải được bằng các phương pháp metaheuristic. Thông qua giả lập, tôi cũng đã chứng minh tính đúng đắn, phù hợp của mô hình sương mù - đám mây nhiều ràng buộc. Tôi cũng đã trình bày tính hiệu quả của của một số giải thuật metaheuristic trong việc lập lịch tác vụ cũng như so sánh với cơ chế truyền thống như Round-Robin.

Những kết quả thu được này cũng chính là những kết quả đã công bố tại hội nghị quốc tế IEEE International Symposium on Network Computing and Applications (IEEE NCA) lần thứ 19 vào tháng 11/2020 [2].

Tuy nhiên, các thử nghiệm này vẫn còn đơn giản khi sinh ngẫu nhiên một tập tác vụ cho trước rồi mới thực hiện lập lịch. Trong tương lai, chúng tôi có kế hoạch xây dựng các thử nghiệm cho phép lập lịch cho các tác vụ được sinh ra liên tục theo thời gian cũng như các tác vụ được vận chuyển qua nhiều nút hơn ở các tầng sương mù và đám mây. Lập lịch với các tác vụ có quan hệ thứ tự, trước sau, tác vụ này cần phải chạy trước tác vụ kia cũng được chúng tôi xem xét. Cùng với đó là công thức hoá thêm các ràng buộc khác cho mô hình sương mù - đám mây

để tiếp cận tới các điều kiện thực tế của các hệ thống sương mù - đám mây. Chúng tôi cũng sẽ cải thiện và đánh giá thêm các phương thức metaheuristic khác cho bài toán lập lịch tác vụ trong môi trường này.

5. Tài liệu tham khảo

[1] Cisco Fog Computing Solutions. “Unleash the power of the Internet of Things”. In: *Cisco Systems Inc* (2015).

[2] Thang Nguyen, Khiem Doan, Giang Nguyen, and Binh Minh Nguyen. “Modeling Multi-constrained Fog-cloud Environment for Task Scheduling Problem”. In: *2020 IEEE 19th International Symposium on Network Computing and Applications (NCA)*. IEEE. 2020, pp. 1–10.

[3] Subhadeep Sarkar, Subarna Chatterjee, and Sudip Misra. “Assessment of the Suitability of Fog Computing in the Context of Internet of Things”. In: *IEEE Transactions on Cloud Computing* 6.1 (2015), pp. 46–59.

[4] Feyza Yildirim Okay and Suat Ozdemir. “A fog computing based smart grid model”. In: *2016 international symposium on networks, computers and communications (ISNCC)*. IEEE. 2016, pp. 1–6.

[5] Rabindra K Barik, Satish Kumar Gudey, Gujji Giridhar Reddy, Meenakshi Pant, Harishchandra Dubey, Kunal Mankodiya, and Vinay Kumar. “FogGrid: Leveraging fog computing for enhanced smart grid network”. In: *2017 14th IEEE India Council International Conference (INDICON)*. IEEE. 2017, pp. 1–6.

[6] Robert Brzoza-Woch, Marek Konieczny, Bartosz Kwolek, Piotr Nawrocki, Tomasz Szydło, and Krzysztof Zieliński. “Holistic approach to urgent computing for flood decision support”. In: *Procedia Computer Science* 51 (2015), pp. 2387–2396.

[7] Xavi Masip-Bruin, Eva Marin-Tordera, Ghazal Tashakor, Admela Jukan, and Guang-Jie Ren. “Foggy clouds and cloudy fogs: a real need for coordinated

management of fog-to-cloud computing systems". In: *IEEE Wireless Communications* 23.5 (2016), pp. 120–128.

[8] Yu Cao, Songqing Chen, Peng Hou, and Donald Brown. "FAST: A fog computing assisted distributed analytics system to monitor fall for stroke mitigation". In: *2015 IEEE international conference on networking, architecture and storage (NAS)*. IEEE. 2015, pp. 2–11.

[9] Yuan Ai, Mugen Peng, and Kecheng Zhang. "Edge computing technologies for Internet of Things: a primer". In: *Digital Communications and Networks* 4.2 (2018), pp. 77–86.

[10] Amir Vahid Dastjerdi and Rajkumar Buyya. "Fog computing: Helping the Internet of Things realize its potential". In: *Computer* 49.8 (2016), pp. 112–116.

[11] Pranali More. "Review of implementing fog computing". In: *International Journal of Research in Engineering and Technology* 4.06 (2015), pp. 335–338.

[12] Jeffrey D. Ullman. "NP-complete scheduling problems". In: *Journal of Computer and System sciences* 10.3 (1975), pp. 384–393.

[13] Taghreed Balharith and Fahd Alhaidari. "Round robin scheduling algorithm in CPU and cloud computing: a review". In: *2019 2nd International Conference on Computer Applications & Information Security (ICCAIS)*. IEEE. 2019, pp. 1–7.

[14] D Chitra Devi and V Rhymend Uthariaraj. "Load balancing in cloud computing environment using improved weighted round robin algorithm for nonpreemptive dependent tasks". In: *The scientific world journal* 2016 (2016).

[15] Jayanti Khatri. "An enhanced Round Robin CPU scheduling algorithm". In: *IOSR Journal of Computer Engineering (IOSR-JCE)* 18.4 (2016), pp. 20–24.

[16] Ibrahim Saidu, Shamala Subramaniam, Azmi Jaafar, and Zuriati Ahmad Zukarnain. "A load-aware weighted round-robin algorithm for IEEE 802.16 networks". In: *EURASIP Journal on Wireless Communications and Networking* 2014.1 (2014), pp. 1–12.

[17] Jeffrey R Sampson. *Adaptation in natural and artificial systems (John H. Holland)*. 1976.

[18] Melanie Mitchell. *An introduction to genetic algorithms*. MIT press, 1998.

[19] James Kennedy and Russell Eberhart. "Particle swarm optimization". In: *Proceedings of ICNN'95-international conference on neural networks*. Vol. 4. IEEE. 1995, pp. 1942–1948.

[20] Meisam Babanezhad, Iman Behroyan, Ali Taghvaie Nakhjiri, Azam Marjani, Mashallah Rezakazemi, Amir Heydarinasab, and Saeed Shirazian. "Investigation on performance of particle swarm optimization (PSO) algorithm based fuzzy inference system (PSOFIS) in a combination of CFD modeling for prediction of fluid flow". In: *Scientific Reports* 11.1 (2021), pp. 1–14.

[21] Salim Bitam, Sherali Zeadally, and Abdelhamid Mellouk. "Fog computing job scheduling optimization based on bees swarm". In: *Enterprise Information Systems* 12.4 (2018), pp. 373–397.

[22] Seyedali Mirjalili and Andrew Lewis. "The whale optimization algorithm". In: *Advances in engineering software* 95 (2016), pp. 51–67.

[23] Lawrence Davis. "Handbook of genetic algorithms". In: (1991).

[24] Qinghai Bai. "Analysis of particle swarm optimization algorithm". In: *Computer and information science* 3.1 (2010), p. 180.

[25] Dang Tran, Nhuan Tran, Giang Nguyen, and Binh Minh Nguyen. "A proactive cloud scaling model based on fuzzy time series and SLA awareness". In: *Procedia Computer Science* 108 (2017), pp. 365–374.

[26] Nhuan Tran, Thang Nguyen, Binh Minh Nguyen, and Giang Nguyen. "A multivariate fuzzy time series resource forecast model for clouds using LSTM and data correlation analysis". In: *Procedia Computer Science* 126 (2018), pp. 636–645.

[27] Amir Vahid Dastjerdi and Rajkumar Buyya. "Compatibility-aware cloud service composition under fuzzy preferences of users". In: *IEEE Transactions on Cloud Computing* 2.1 (2014), pp. 1–13.

[28] Binh Minh Nguyen, Dang Tran, and Giang Nguyen. “Enhancing service capability with multiple finite capacity server queues in cloud data centers”. In: *Cluster Computing* 19.4 (2016), pp. 1747–1767.

[29] Parvathy S Pillai and Shrisha Rao. “Resource allocation in cloud computing using the uncertainty principle of game theory”. In: *IEEE Systems Journal* 10.2 (2014), pp. 637–648.

[30] Thieu Nguyen, Nhuan Tran, Binh Minh Nguyen, and Giang Nguyen. “A resource usage prediction system using functional-link and genetic algorithm neural network for multivariate cloud metrics”. In: *2018 IEEE 11th conference on service-oriented computing and applications (SOCA)*. IEEE. 2018, pp. 49–56.

[31] AI Awad, NA El-Hefnawy, and HM Abdel_kader. “Enhanced particle swarm optimization for task scheduling in cloud computing environments”. In: *Procedia Computer Science* 65 (2015), pp. 920–929.

[32] Sung Ho Jang, Tae Young Kim, Jae Kwon Kim, and Jong Sik Lee. “The study of genetic algorithm-based task scheduling for cloud computing”. In: *International Journal of Control and Automation* 5.4 (2012), pp. 157–162.

[33] Karnam Sreenu and M Sreelatha. “W-Scheduler: whale optimization for task scheduling in cloud computing”. In: *Cluster Computing* (2017), pp. 1–12.

[34] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. “Fog computing and its role in the internet of things”. In: *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*. 2012, pp. 13–16.

[35] Luis M Vaquero and Luis Rodero-Merino. “Finding your way in the fog: Towards a comprehensive definition of fog computing”. In: *ACM SIGCOMM Computer Communication Review* 44.5 (2014), pp. 27–32.

[36] Ladislav Hluchỳ, Giang Nguyen, Ján Astaloš, Viet Tran, Viera Šipková, and Binh Minh Nguyen. “Effective computation resilience in high performance and distributed environments”. In: *Computing and Informatics* 35.6 (2017), pp. 1386–1415.

[37] Tejaswini Choudhari, Melody Moh, and Teng-Sheng Moh. “Prioritized task scheduling in fog computing”. In: *Proceedings of the ACMSE 2018 conference*. 2018, pp. 1–8.

[38] Luxiu Yin, Juan Luo, and Haibo Luo. “Tasks scheduling and resource allocation in fog computing based on containers for smart manufacturing”. In: *IEEE Transactions on Industrial Informatics* 14.10 (2018), pp. 4712–4721.

[39] Sambit Kumar Mishra, Deepak Puthal, Joel JPC Rodrigues, Bibhudatta Sahoo, and Eryk Dutkiewicz. “Sustainable service allocation using a metaheuristic technique in a fog server for industrial applications”. In: *IEEE Transactions on Industrial Informatics* 14.10 (2018), pp. 4497–4506.

[40] Amir Karamoozian, Abdelhakim Hafid, and El Mostapha Aboulhamid. “On the Fog-Cloud Cooperation: How Fog Computing can address latency concerns of IoT applications”. In: *2019 Fourth International Conference on Fog and Mobile Edge Computing (FMEC)*. IEEE. 2019, pp. 166–172.

[41] Lan Wang and Erol Gelenbe. “Adaptive dispatching of tasks in the cloud”. In: *IEEE Transactions on Cloud Computing* 6.1 (2015), pp. 33–45.

[42] Kobra Etminani and M Naghibzadeh. “A min-min max-min selective algorihtm for grid task scheduling”. In: *2007 3rd IEEE/IFIP International Conference in Central Asia on Internet*. IEEE. 2007, pp. 1–7.

[43] Syed Aon Ali Naqvi, Nadeem Javaid, Hanan Butt, Muhammad Babar Kamal, Ali Hamza, and Muhammad Kashif. “Metaheuristic optimization technique for load balancing in cloud-fog environment integrated with smart grid”. In: *International Conference on Network-Based Information Systems*. Springer. 2018, pp. 700–711.

[44] Xuan-Qui Pham and Eui-Nam Huh. “Towards task scheduling in a cloud-fog computing system”. In: *2016 18th Asia-Pacific network operations and management symposium (APNOMS)*. IEEE. 2016, pp. 1–4.

[45] Solmaz Abdi, Seyyed Ahmad Motamedi, and Saeed Sharifian. “Task scheduling using modified PSO algorithm in cloud computing environment”. In: *In-*

ternational conference on machine learning, electrical and mechanical engineering. 2014, pp. 8–9.

[46] Binh Minh Nguyen, Huynh Thi Thanh Binh, Bao Do Son, et al. "Evolutionary algorithms to optimize task scheduling problem for the IoT based bag-of-tasks application in cloud–fog computing environment". In: *Applied Sciences* 9.9 (2019), p. 1730.

[47] Huynh Thi Thanh Binh, Tran The Anh, Do Bao Son, Pham Anh Duc, and Binh Minh Nguyen. "An evolutionary algorithm for solving task scheduling problem in cloud-fog computing environment". In: *Proceedings of the Ninth International Symposium on Information and Communication Technology*. 2018, pp. 397–404.

[48] Federico Larumbe and Brunilde Sanso. "A tabu search algorithm for the location of data centers and software components in green cloud computing networks". In: *IEEE Transactions on cloud computing* 1.1 (2013), pp. 22–35.