Joint Task Offloading and Resource Optimization in NOMA-Based Vehicular Edge Computing: A Game-Theoretic DRL Approach

Abstract :

-VEC đang trở thành mô hình đầy hứa hẹn cho sự phát triển của giao thông thông minh thế hệ mới . Tuy nhiên việc thực thi thực tế gặp nhiều khó khăn thách thức do tài nguyên hạn chế và nhu cầu truyền tải cao .

-Mô hình xây dựng trên NOMA : nơi các nút biên không đồng nhất hợp tác để xử lý tác vụ thời gian thực

Chúng tôi xây dựng một **mô hình truyền dẫn V2I (vehicle-to-infrastructure)** bằng cách xem xét cả **nhiễu trong-biên (intra-edge)** và **nhiễu giữa các biên (inter-edge)**, đồng thời hình thành một **bài toán tối ưu hợp tác tài nguyên (CRO – Cooperative Resource Optimization)** bằng cách **tối ưu đồng thời việc offloading tác vụ và phân bổ tài nguyên**, nhằm mục tiêu **tối đa hóa tỷ lệ dịch vụ (service ratio)**.

Hơn nữa, chúng tôi phân rã CRO thành hai bài toán con: **offloading tác vụ** và **phân bổ tài nguyên**.

* Cụ thể, bài toán con offloading tác vụ được mô hình hóa thành một **trò chơi tiềm năng chính xác (Exact Potential Game - EPG)**, và một giải pháp **MAD4PG (Multi-Agent Distributed Distributional Deep Deterministic Policy Gradient)** được đề xuất để đạt tới **Nash equilibrium**.
* Bài toán con phân bổ tài nguyên được tách thành hai bài toán tối ưu lồi độc lập, và một nghiệm tối ưu được tìm ra bằng **phương pháp lặp dựa trên gradient** và **điều kiện KKT**.

Cuối cùng, chúng tôi xây dựng mô hình mô phỏng dựa trên **dữ liệu quỹ đạo phương tiện thực tế** và thực hiện đánh giá hiệu năng toàn diện, qua đó khẳng định tính ưu việt của giải pháp đề xuất.

### 1. Giới thiệu

Những tiến bộ gần đây trong mạng xe tự hành đã mở đường cho sự phát triển của các hệ thống giao thông thông minh (ITSs) thế hệ mới như lái xe tự động hợp tác [1] và hệ thống không gian mạng trên phương tiện (vehicular cyber-physical systems) [2]. Tuy nhiên, các ứng dụng này đòi hỏi khả năng truyền dữ liệu khối lượng lớn và tính toán tác vụ cường độ cao. Ví dụ, các phương tiện hiện đại như **Tesla Model X** đã được trang bị tám camera, 12 radar siêu âm và một radar sóng milimet, dẫn đến yêu cầu tính toán dữ liệu ngày càng gia tăng. Trong khi đó, tài nguyên truyền thông và tính toán trong mạng xe tự hành lại hạn chế, khiến việc hỗ trợ các ứng dụng thời gian thực trở thành thách thức không nhỏ. Rõ ràng, việc nghiên cứu các giải pháp *offloading tác vụ thời gian thực* và *phân bổ tài nguyên không đồng nhất* trong mạng xe tự hành là vô cùng cấp thiết.

**Tính toán biên trên phương tiện (Vehicular Edge Computing – VEC)** [3] gần đây đã nổi lên như một mô hình đầy hứa hẹn để hỗ trợ xử lý tác vụ ngay tại biên của mạng xe tự hành. Nhiều nghiên cứu [4–9] đã tập trung vào VEC, trong đó các nút biên như trạm gốc 5G và các RSU (roadside units) được tích hợp năng lực tính toán, có thể xử lý tác vụ được các phương tiện tải lên thông qua liên lạc **V2I (vehicle-to-infrastructure)**. Tuy nhiên, các nghiên cứu này chưa khai thác công nghệ **đa truy nhập phi trực giao (NOMA)** [10] nhằm tăng cường dung lượng mạng. Một số nghiên cứu đã tích hợp NOMA trong truyền thông xe tự hành [11–14], cho phép các phương tiện truyền dữ liệu đến nút biên trên cùng băng tần nhưng với mức công suất khác nhau. Tuy nhiên, các nghiên cứu đó mới chỉ xem xét kịch bản một nút biên duy nhất và chưa xử lý được vấn đề phân bổ tài nguyên cũng như giao thoa giữa nhiều nút biên.

Để nâng cao độ tin cậy hệ thống, một số nghiên cứu đã quan tâm đến bài toán phân bổ tài nguyên nhằm chống lại ảnh hưởng của kênh V2I biến động theo thời gian và sự thay đổi động của tài nguyên tính toán sẵn có trong VEC [15–20]. Tuy nhiên, chưa có nghiên cứu nào khai thác **hiệu ứng cộng hưởng giữa offloading tác vụ thời gian thực và phân bổ tài nguyên truyền thông/tính toán**. Một số công trình thiết kế cơ chế phân bổ đồng thời cho cả tài nguyên truyền thông và tính toán để nâng cao hiệu suất sử dụng [21–23]. Một số ít tài liệu đã xây dựng mô hình tối ưu hóa kết hợp bằng cách tích hợp offloading tác vụ và phân bổ tài nguyên [24, 25]. Tuy nhiên, phần lớn các nghiên cứu hiện nay dựa trên **lập lịch tập trung**, điều này có thể cản trở khả năng mở rộng hệ thống trong mạng xe tự hành.

**Học tăng cường sâu đa tác tử (Multi-Agent Deep Reinforcement Learning – MADRL)** [26] đã nổi lên như một giải pháp phân tán cho các ứng dụng trên xe tự hành [27–29]. Ngoài ra, một số công trình đã kết hợp học tăng cường và lý thuyết trò chơi [30–32] để giải quyết các bài toán tối ưu phức tạp. Tuy nhiên, chưa có giải pháp nào có thể áp dụng trực tiếp vào mạng xe tự hành để giải quyết đồng thời bài toán *offloading tác vụ thời gian thực* và *phân bổ tài nguyên không đồng nhất*.

Xuất phát từ những động lực trên, bài báo này đề xuất một giải pháp lập lịch phân tán cho *offloading tác vụ và phân bổ tài nguyên* dựa trên **thuật toán MAD4PG (multi-agent distributed distributional deep deterministic policy gradient)** và **lý thuyết trò chơi tiềm năng (potential game theory)**. Cụ thể, chúng tôi trước hết mô hình hóa quá trình ra quyết định offloading tác vụ dưới dạng một **trò chơi tiềm năng chính xác (exact potential game – EPG)** [33], trong đó tồn tại và hội tụ đến **cân bằng Nash (NE)** nhờ một hàm tiềm năng được thiết kế. Tại đây, các nút biên đóng vai trò là những người chơi hợp lý, với mục tiêu tối đa hóa lợi ích của mình (tức là **tỷ lệ phục vụ tác vụ thời gian thực – số tác vụ hoàn thành đúng hạn chia cho tổng số tác vụ**). Theo lý thuyết trò chơi tiềm năng, NE có thể đạt được bằng cách tối đa hóa hàm tiềm năng của mỗi nút biên. Do đó, hàm tiềm năng này rất phù hợp để sử dụng làm **hàm thưởng** cho các nút biên trong thuật toán MAD4PG được đề xuất.

Phần bài toán phân bổ tài nguyên còn lại được chia thành **hai bài toán lồi độc lập**, và một nghiệm tối ưu được đưa ra dựa trên **phương pháp lặp dựa trên gradient** và **điều kiện Karush–Kuhn–Tucker (KKT)** [34].

Bài báo này là nỗ lực đầu tiên nhằm nghiên cứu kết hợp *offloading tác vụ thời gian thực* và *phân bổ tài nguyên không đồng nhất* trong VEC dựa trên NOMA bằng cách giải quyết ba thách thức:

1. **Liên kết V2I uplink** chịu giao thoa từ các phương tiện dùng chung kênh, trong đó mức ảnh hưởng phụ thuộc vào công suất truyền mà nút biên phân bổ.
2. **Sự mất cân bằng nghiêm trọng về phân bố khối lượng công việc** giữa các nút biên khác nhau do sự thay đổi động theo thời gian của các tác vụ tính toán chuyên sâu và nhạy trễ.
3. Việc để các nút biên tự đưa ra quyết định offloading và phân bổ tài nguyên **một cách độc lập, hiệu quả nhưng chỉ với thông tin cục bộ** là một thách thức phi tầm thường.

Do đó, việc nghiên cứu một phương pháp **phân tán, hiệu quả cho offloading tác vụ thời gian thực và phân bổ tài nguyên không đồng nhất trong VEC dựa trên NOMA** là vừa cấp thiết vừa mang tính thách thức.

#### Đóng góp chính của bài báo:

* Đề xuất kiến trúc VEC dựa trên NOMA, trong đó các phương tiện chia sẻ cùng băng tần và truyền dữ liệu đến nút biên với công suất truyền được phân bổ. Các tác vụ phát sinh ngẫu nhiên trên phương tiện, có yêu cầu tính toán và deadline khác nhau, có thể được tải lên nút biên để xử lý qua V2I. Nút biên với năng lực tính toán không đồng nhất (tần số xung nhịp CPU khác nhau) có thể xử lý tác vụ cục bộ hoặc di trú tác vụ sang nút biên lân cận thông qua kết nối có dây.
* Xây dựng bài toán tối ưu hóa tài nguyên hợp tác (**Cooperative Resource Optimization – CRO**), kết hợp offloading tác vụ và phân bổ tài nguyên truyền thông/tính toán nhằm tối đa hóa tỷ lệ phục vụ. Cụ thể, xây dựng mô hình truyền dẫn V2I, trong đó cả nhiễu nội-biên (intra-edge) và liên-biên (inter-edge) đều được mô tả theo nguyên tắc NOMA; đồng thời xây dựng mô hình offloading tác vụ xét đến sự hợp tác giữa các nút biên dị thể.
* Phân rã CRO thành hai bài toán con: offloading tác vụ và phân bổ tài nguyên. Bài toán con đầu được mô hình hóa dưới dạng một trò chơi phi hợp tác giữa các nút biên, được chứng minh là **EPG có tồn tại và hội tụ NE**. Sau đó, thiết kế thuật toán **MAD4PG**, phiên bản đa tác tử của **D4PG** [35], để đạt NE, trong đó các nút biên đóng vai trò tác tử độc lập, sử dụng tiềm năng đạt được làm phần thưởng. Tiếp theo, chia bài toán phân bổ tài nguyên thành hai bài toán lồi độc lập và tìm nghiệm tối ưu bằng phương pháp lặp dựa trên gradient và điều kiện KKT.
* Xây dựng mô hình mô phỏng dựa trên quỹ đạo thực tế của phương tiện. Ngoài các chỉ số chính (tổng reward và tỷ lệ phục vụ trung bình), đề xuất thêm bốn chỉ số: thời gian xử lý trung bình, thời gian phục vụ trung bình, tiềm năng đạt được trung bình, và tỷ lệ xử lý cục bộ so với di trú. Đồng thời, triển khai các thuật toán đề xuất cũng như bốn giải pháp so sánh: **ORM** (chỉ phân bổ tài nguyên tối ưu + di trú tác vụ), **ORL** (chỉ phân bổ tài nguyên tối ưu + xử lý cục bộ), **D4PG** [35] (phân bổ tài nguyên + offloading tập trung), và **MADDPG** [28] (offloading đa tác tử dựa trên phân bổ tài nguyên tối ưu). Kết quả mô phỏng khẳng định rõ ràng tính ưu việt của thuật toán đề xuất.

| Nhem nghiên cứu | Hướng nghiên cứu | Phương pháp / Thuật toán | Hạn chế |
| --- | --- | --- | --- |
| **Liu et al. [4]** | Phát tán dữ liệu hợp tác trong kiến trúc end–edge–cloud | Thuật toán dựa trên *clique* để lập lịch mã hóa & phát tán dữ liệu | Chỉ dựa trên OMA, chưa khai thác NOMA |
| **Dai et al. [5]** | Kiến trúc VEC cho adaptive-bitrate streaming | Edge cache + dịch vụ truyền tải cho các file chunk nhiều mức chất lượng | Giới hạn trong OMA |
| **Zhang et al. [6]** | Socially-aware vehicular edge caching | Điều phối động cache dựa trên sở thích & dịch vụ | Không xét NOMA |
| **Liu et al. [7]** | Kiến trúc VEC hai tầng (cloud, edge tĩnh, edge di động) | Phân tầng xử lý tác vụ thời gian thực | Vẫn dựa trên OMA |
| **Liu et al. [8]** | Tận dụng caching + network coding trong VEC | Thuật toán tiến hóa lai (memetic algorithm) | Không áp dụng NOMA |
| **Xu et al. [9]** | Chiến lược cảnh báo va chạm | Hiệu chỉnh quỹ đạo + xét trễ & mất gói V2I | Không xét offloading hoặc tài nguyên |
| **Patel et al. [11]** | NOMA trong mạng xe | Đánh giá dung lượng kênh (so sánh NOMA–OMA) | Chỉ mức đánh giá, chưa xét offloading |
| **Zhang et al. [12]** | NOMA + phân bổ tài nguyên tập trung | Matching graph-based + non-coop game power control | Giới hạn single-edge node |
| **Zhu et al. [13]** | Phân bổ công suất tối ưu trong NOMA | Xét tác vụ ngẫu nhiên & kênh biến động | Không giải quyết multi-edge |
| **Liu et al. [14]** | NOMA trong mạng xe tự lái | ADMM-based power allocation | Không xét can nhiễu giữa edge |
| **Liu et al. [15]** | Offloading đa chu kỳ thời gian thực | Distributed method: mobility, resources, deadline-aware | Không xét kết hợp tài nguyên |
| **Liu et al. [16]** | Offloading tối ưu đa mục tiêu | ADMM + PSO, giảm delay/energy/cost | Không xét phối hợp tài nguyên |
| **Chen et al. [17]** | Offloading với khôi phục lỗi | Giảm năng lượng, rút ngắn thời gian | Không xét communication-computation synergy |
| **Liu et al. [18]** | Lập lịch phát tán dữ liệu thời gian thực | Heuristic scheduling (xét freshness & deadline) | Giới hạn trong data dissemination |
| **Xu et al. [19]** | Phân bổ subchannel | Incentive-based probability update (potential game) | Không xét offloading thực tế |
| **Liu et al. [20]** | Phát tán dữ liệu hợp tác | Greedy method trong hybrid vehicular env | Không xét offloading + tài nguyên |
| **Cui et al. [21]** | Phân bổ tài nguyên joint (comm+comp) | Multi-objective RL, giảm delay | Centralized, chưa scale lớn |
| **Han et al. [22]** | Độ tin cậy tính toán & truyền thông | Dynamic programming | Chưa xét offloading |
| **Xu et al. [23]** | Phân bổ tài nguyên joint | Contract theory | Centralized, chưa mở rộng |
| **Dai et al. [24]** | Offloading dữ liệu (server dị thể) | Asynchronous DRL | Overhead cao khi scale lớn |
| **Dai et al. [25]** | Offloading xác suất | Probabilistic allocation ở edge node | Chưa phù hợp mạng quy mô lớn |
| **Alam et al. [27]** | MADRL cho offloading động | MADRLHA (Hungarian-based) | Chưa xét phối hợp tài nguyên |
| **Zhang et al. [28]** | Edge resource allocation | MADDPG để giảm cost dưới delay constraint | Centralized, chưa mở rộng |
| **Nie et al. [29]** | UAV-enabled VEC | Multi-agent Federated RL (MAFRL) | Chưa xét offloading + tài nguyên joint |
| **Zhang et al. [30]** | RL + Game theory | Actor–critic như game 2 người | Mới lý thuyết, chưa áp dụng VEC |
| **Albaba et al. [31]** | DQN + Game theory | Hierarchical game cho prediction hành vi lái xe | Không xét MEC/VEC |
| **Rajeswaran et al. [32]** | Model-based RL như Stackelberg game | Policy player vs Model player | Chưa áp dụng VEC thực tế |

### **3. Kiến trúc hệ thống**

Trong phần này, chúng tôi đề xuất một kiến trúc dựa trên **NOMA** cho truyền thông và tính toán hợp tác trong mạng VEC. Như minh họa ở Hình 1, các hạ tầng ven đường (ví dụ: ) như trạm gốc 5G và RSU, được trang bị các đơn vị tính toán (CPU chip), đóng vai trò là **nút biên** (edge nodes) nhằm tăng tốc xử lý các tác vụ tính toán được offload không dây từ các phương tiện di động.

Các tác vụ đến ngẫu nhiên tại các phương tiện, có thể bao gồm các dữ liệu khác nhau cần được xử lý. Mỗi phương tiện có thể liên lạc với một nút biên thông qua liên lạc **V2I** trong phạm vi phủ sóng của nó. Sau đó, phương tiện sẽ tải tác vụ lên nút biên gần nhất, trong đó công suất truyền được phân bổ bởi nút biên tương ứng.

Bằng cách sử dụng **mã chồng tín hiệu (superposition coding – SC)** ở phía phương tiện và **khử giao thoa liên tiếp (successive interference cancellation – SIC)** tại các nút biên [36], nhiều phương tiện có thể chia sẻ cùng một băng tần. Cụ thể, tín hiệu của các phương tiện có kênh mạnh sẽ được giải mã và loại bỏ trước, sau đó mới giải mã tín hiệu của các phương tiện có kênh yếu trong hệ thống VEC dựa trên NOMA. Ngoài ra, các nút biên được kết nối với nhau thông qua mạng có dây. Các nút biên quyết định việc thực thi tác vụ tại chỗ hay di chuyển sang nút khác thông qua kết nối có dây. Cuối cùng, các nút biên sẽ phân bổ tài nguyên tính toán để xử lý các tác vụ.

### **Đặc điểm của hệ thống**

1. **Đa dạng yêu cầu tính toán:** Các tác vụ tính toán do phương tiện yêu cầu có kích thước dữ liệu, nhu cầu tài nguyên, và hạn chót (deadline) khác nhau. Vì vậy, **khả năng hoàn thành tác vụ đúng hạn** phụ thuộc vào nút biên được chọn, vốn có năng lực tính toán khác nhau (tốc độ xung nhịp CPU).
2. **Nhiễu truyền thông trong V2I:** Việc tăng công suất truyền của phương tiện có thể cải thiện tốc độ V2I, nhưng đồng thời gây **nhiễu nội biên (intra-edge interference)** và **nhiễu liên biên (inter-edge interference)** cho các kênh V2I khác. Hơn nữa, công suất truyền được cấp phát bởi các nút biên thay đổi theo thời gian và không được chia sẻ lẫn nhau, do đó mỗi nút biên phải quyết định công suất truyền cho phương tiện của mình trong bối cảnh chịu ảnh hưởng của nút biên khác.
3. **Cân bằng tải tính toán:** Do sự đến ngẫu nhiên của tác vụ và sự di chuyển động của phương tiện, khối lượng công việc tại các nút biên có thể mất cân bằng. Nếu một nút biên bị quá tải, việc **di chuyển (migrate) tác vụ** sang nút khác với tài nguyên dư thừa sẽ giúp giảm độ trễ xử lý. Tuy nhiên, việc truyền dữ liệu qua mạng có dây lại kéo dài thời gian phục vụ tác vụ.

### **Ví dụ minh họa**

Trong Hình 1, phương tiện và tải tác vụ thông qua kênh V2I. Vì chất lượng kênh V2I giữa nút biên và phương tiện tốt hơn so với và , nên tín hiệu của được giải mã trước bằng cách coi tín hiệu khác là nhiễu. Sau đó, tín hiệu của sẽ bị loại bỏ khi giải mã tín hiệu của và .

Tuy nhiên, tín hiệu của có thể bị gây nhiễu bởi trong truyền V2I — loại nhiễu này gọi là **nhiễu nội biên (intra-edge interference)** vì cả hai phương tiện cùng trong vùng phủ sóng của . Ngược lại, tín hiệu của có thể bị nhiễu bởi , nhưng từ một nút biên khác — loại này gọi là **nhiễu liên biên (inter-edge interference)**.

Ngoài ra, dễ thấy rằng tải công việc giữa các nút và không đều: có ba tác vụ trong khi chỉ có một tác vụ . Nếu có tài nguyên mạnh hơn , thì nên **di chuyển tác vụ**  từ sang qua mạng có dây để giảm thời gian xử lý.

Như vậy, để tận dụng hiệu quả **truyền thông và tính toán hợp tác** giữa các nút biên, cần có một **cơ chế lập lịch phân tán và hiệu quả** cho offloading tác vụ theo thời gian thực và phân bổ tài nguyên không đồng nhất, nhằm tối ưu hiệu năng toàn hệ thống.

👉 Như vậy:

* **Offloading** = quyết định tải tác vụ sang đâu (nút nào, tại chỗ hay di chuyển).
* **Phân bổ (allocation)** = chia công suất truyền + tài nguyên CPU ở các nút biên.
* **NOMA** = cho phép nhiều xe dùng chung băng tần nhờ SC + SIC, nhưng phải xử lý tốt nhiễu nội biên/liên biên.

# 4. Mô hình bài toán (Problem Formulation)

## 4.1. Kiến thức tiền đề (Preliminary)

Tập các **khe thời gian rời rạc** được ký hiệu , với là số khe thời gian. Tập **phương tiện** ký hiệu , và **vị trí** của xe tại thời điểm ký hiệu .

**Xác suất sinh tác vụ** của xe tại thời điểm ký hiệu . Tập **tác vụ** do xe yêu cầu ký hiệu . **Tác vụ** do xe yêu cầu ở thời điểm , ký hiệu , được đặc trưng bởi bộ ba

trong đó là **kích thước dữ liệu**, là **số chu kỳ CPU để xử lý mỗi bit dữ liệu**, và là **hạn chót (deadline)**.

Tập **nút biên** ký hiệu . Mỗi **nút biên** được đặc trưng bởi bộ bốn

trong đó là **công suất tối đa** cho liên lạc V2I, là **tần số xung nhịp tính toán (CPU clock)**, là **bán kính/miền phủ liên lạc V2I**, và là **vị trí**. **Tốc độ truyền có dây** giữa các nút biên ký hiệu .

**Khoảng cách** giữa xe và nút biên tại thời điểm ký hiệu . Tập **xe nằm trong vùng phủ** của nút biên tại thời điểm là

**Băng thông** V2I ký hiệu .

Bảng 1 liệt kê các ký hiệu chính (dịch ở cuối mục).

## 4.2. Mô hình truyền dẫn V2I (V2I Transmission Model)

Mô hình V2I được minh hoạ ở Hình 2(a); **nhiễu nội biên** và **nhiễu liên biên** được mô hình theo nguyên tắc NOMA.

**Công suất truyền** của xe được **nút biên** cấp tại thời điểm ký hiệu . Tổng công suất cấp phát không vượt quá công suất V2I tối đa của nút :

**Hệ số kênh** giữa xe và nút tại thời điểm ký hiệu , được tính theo [37]:

trong đó là fading quy mô nhỏ Rayleigh, và là **số mũ suy hao đường truyền** (path-loss exponent) quy mô lớn.

Tập **các xe có điều kiện kênh tức thời kém hơn** xe tại thời điểm ký hiệu :

Sau khi cấp phát công suất cho từng xe , **tín hiệu thu** tại nút biên là [10]:

trong đó là **tín hiệu của xe**  và là **nhiễu AWGN**.

Theo NOMA, nút **khử giao thoa liên tiếp (SIC)** các tín hiệu của những xe có điều kiện kênh **tốt hơn** xe trước khi giải mã xe . Do đó, **SINR** giữa xe và nút tại thời điểm là

Vậy **thời gian tải lên (uploading time)** của tác vụ do xe gửi tới nút :

trong đó là kích thước dữ liệu của và là băng thông V2I (Hz).

## 4.3. Mô hình offloading tác vụ (Task Offloading Model)

Tập **tác vụ tải lên** bởi các xe trong vùng phủ của nút tại thời điểm :

Như Hình 2(b), mỗi có thể **xử lý cục bộ** tại hoặc **di trú** sang nút biên khác.

Ký hiệu **biến offloading** cho biết tác vụ của xe có được **offload** đến nút ở thời điểm hay không. Mỗi tác vụ **chỉ offload tới đúng một nút**:

Tập **tác vụ được offload** (bao gồm tác vụ địa phương và tác vụ di trú tới ):

**Tài nguyên tính toán** (tốc độ CPU) cấp cho tại nút ký hiệu . Tổng tài nguyên cấp phát **không vượt quá** năng lực tính toán của :

**Thời gian thực thi** của tại là

với là kích thước dữ liệu và là số chu kỳ/bit của .

Nếu **được xử lý ở nút khác** (không trùng nút nơi dữ liệu được tải lên), tác vụ **chưa thể chạy** cho đến khi dữ liệu được **truyền có dây** đến nút offload. **Thời gian truyền có dây** từ nút (gửi) đến nút (nhận) đối với ký hiệu :

trong đó là khoảng cách giữa hai nút biên và , là hằng số **giảm theo khoảng cách** (distance discount), và là **tốc độ đường truyền có dây**.

**Thời gian xử lý** tổng cộng của tại thời điểm (ở nút đích offload) ký hiệu :

Tức là thời gian xử lý gồm **thời gian truyền có dây** (nếu có) và **thời gian thực thi**, phụ thuộc vào quyết định offload.

## 4.4. Bài toán tối ưu hoá tài nguyên hợp tác (Cooperative Resource Optimization – CRO)

**Thời gian phục vụ** (service time) của gồm **thời gian tải lên** và **thời gian xử lý**, ký hiệu :

Tác vụ được coi là **phục vụ thành công** nếu .

**Tỷ lệ phục vụ** (service ratio) của nút tại thời điểm , ký hiệu , là tỷ lệ giữa **số tác vụ phục vụ thành công** và **tổng số tác vụ được yêu cầu** trong vùng phủ của :

trong đó là số tác vụ trong vùng phủ của , và là **hàm chỉ báo** (bằng 1 nếu điều kiện đúng, ngược lại bằng 0).

Gọi nghiệm xác định , trong đó:

Mục tiêu của bài báo là **tối đa hóa tổng tỷ lệ phục vụ** của các nút biên trong toàn bộ giai đoạn lập lịch, bằng cách **tối ưu kết hợp** quyết định offloading và **phân bổ tài nguyên dị thể** (truyền thông & tính toán) trong VEC dựa trên NOMA. Bài toán CRO được phát biểu:

* **C1**: tổng công suất truyền cấp phát bởi mỗi nút biên **không vượt** công suất V2I tối đa.
* **C2**: tổng tài nguyên tính toán cấp phát **không vượt** năng lực tính toán của nút biên.
* **C3–C4**: biến offloading là nhị phân 0–1 và **mỗi tác vụ chỉ được offload tới đúng một nút**.

### Bảng 1. Tóm tắt ký hiệu chính

* : tập khe thời gian rời rạc
* : tập phương tiện
* : tập nút biên;
* : xác suất sinh tác vụ của xe tại thời điểm
* : tập tác vụ của xe
* : tác vụ tại thời điểm (kích thước dữ liệu, chu kỳ/bit, deadline)
* : vị trí xe tại thời điểm
* : công suất V2I tối đa của nút
* : tần số CPU của nút
* : bán kính phủ V2I của nút
* : vị trí nút
* : khoảng cách xe–nút tại thời điểm
* : tốc độ truyền có dây giữa các nút biên
* : băng thông V2I
* : công suất truyền xe (được nút cấp) tại thời điểm
* : biến nhị phân offload tới nút
* : tài nguyên CPU nút cấp cho tác vụ của tại thời điểm
* : tập xe trong vùng phủ của tại thời điểm
* : tập xe có kênh **kém hơn** xe tại thời điểm
* : tập tác vụ do các xe trong vùng tải lên tại thời điểm
* : tập tác vụ **được offload** tại nút ở thời điểm

## 5. Giải pháp đề xuất

Như thể hiện trong Hình 3, bài toán **CRO** (computation-resource optimization) được giải thông qua việc tách thành hai bài toán con, đó là **task offloading (P1)** và **resource allocation (P2)**. Cụ thể:

* **P1** được mô hình hóa như một **trò chơi phi hợp tác (non-cooperative game)** giữa các edge nodes, và được chứng minh là một **Exact Potential Game (EPG)** với sự tồn tại và hội tụ tới **Nash Equilibrium (NE)**. Để giải P1, nhóm tác giả thiết kế **MAD4PG** (multi-agent deep deterministic policy gradient) triển khai tại mỗi edge node nhằm tìm được NE cho bài toán offloading.
* **P2** được chia thành **hai bài toán tối ưu lồi độc lập**. Nhóm tác giả dẫn xuất nghiệm tối ưu cho việc phân bổ tài nguyên không đồng nhất (heterogeneous resource allocation) dựa trên **phương pháp lặp theo gradient và điều kiện KKT** để xử lý P2.

Sự tương tác giữa hai bài toán con được mô tả như sau:

1. Đầu tiên, các quyết định offloading được xác định trước nhờ MAD4PG, với đầu vào là quan sát cục bộ của hệ thống.
2. Sau đó, việc phân bổ tài nguyên (power & CPU) được tính toán bằng nghiệm tối ưu, dựa trên quyết định offloading đã có.
3. Trong môi trường VEC dựa trên NOMA, **hành động kết hợp** của cả hai (offloading + resource allocation) sẽ được dùng để tính toán **reward** cho edge node thông qua hàm potential đã thiết kế.
4. Quy trình này tiếp tục lặp cho đến khi việc huấn luyện MAD4PG hoàn tất.

### 5.1. Phân rã bài toán

Trong mục này, ta phân rã CRO thành nhiều bài toán nhỏ ở mỗi **time slot**. Do các biến tại thời điểm **độc lập lẫn nhau**, và 4 ràng buộc (constraints) cũng **tách biệt** vì không có biến trùng lặp, nên bài toán có thể được tách thành 2 bài toán con như sau:

#### (1) **Task Offloading (P1)**

Bài toán con P1 với biến liên quan đến **các quyết định offloading của edge nodes**, được mô hình hóa như sau:

với các ràng buộc:

* **C5:**
* **C6:**

Trong đó, mỗi xe chỉ có thể offload đến **duy nhất 1 edge node**.

👉 Nhóm tác giả mô hình P1 như một trò chơi phi hợp tác , trong đó:

* : tập các edge nodes (người chơi).
* : không gian chiến lược (tập hợp tất cả các chiến lược khả dĩ của các edge node).
* : **hàm lợi ích (utility function)** của edge node .

**Định nghĩa 1:** Hàm utility được định nghĩa là **tổng tỉ lệ dịch vụ đạt được** (achieved service ratio) của edge nodes dưới chiến lược :

Nhóm tác giả chứng minh trò chơi này là **Exact Potential Game (EPG)** với hàm potential như sau:

→ Tồn tại ít nhất một **NE thuần chiến lược (pure-strategy NE)**, và quá trình lặp sẽ hội tụ trong số bước hữu hạn (finite steps). Ngoài ra, để giảm chi phí tính toán, nhóm sử dụng khái niệm **ε-equilibrium** (xấp xỉ NE trong sai số ε).

#### (2) **Resource Allocation (P2)**

Bài toán con P2 liên quan đến việc phân bổ **công suất truyền (P^t)** và **tài nguyên tính toán (C^t)**:

với các ràng buộc:

* **C7:**
* **C8:**

👉 Vì biến và độc lập, nên P2 tiếp tục được chia thành 2 bài toán nhỏ:

##### (a) Transmission Power Allocation (P3 → P6)

Mục tiêu: phân bổ công suất truyền giữa các xe trong cùng edge node.

s.t. .

* Sau khi tách riêng theo từng edge node, ta có P4.
* Do P4 **không lồi** (nonconvex) vì có giao thoa (interference), tác giả áp dụng **xấp xỉ convex**:
  + Xây dựng **cận dưới (lower bound)** cho hàm mục tiêu (Eq. 23–25).
  + Biến đổi log công suất thành biến mới .

👉 Cuối cùng, P6 trở thành **bài toán tối ưu lồi chuẩn (convex optimization problem)**, có thể giải bằng phương pháp chuẩn (gradient descent).

##### (b) Computation Resource Allocation (P7 → P8)

Mục tiêu: phân bổ CPU giữa các task trong cùng edge node.

s.t. .

* Sau khi chia nhỏ theo từng edge node, ta có P8:

s.t. .

👉 Đây là **bài toán convex**, có thể giải trực tiếp bằng phương pháp KKT.

✅ **Tóm lại:**

* **P1 (Offloading):** giải bằng **game theory + MAD4PG** để tìm NE.
* **P2 (Resource Allocation):** giải bằng **convex optimization (gradient descent + KKT)** cho power và CPU.
* **Kết hợp:** hai bước xen kẽ, lặp lại cho đến khi chính sách MAD4PG hội tụ.

## 5.2. MAD4PG cho Offloading tác vụ (Dịch & giải thích)

MAD4PG gồm các thành phần chính: một learner (trình học chung), nhiều distributed actors (tác tử phân tán), môi trường VEC dựa trên NOMA, và một replay buffer để lưu trải nghiệm.

### 1) **Trạng thái hệ thống (System State)**

Quan sát cục bộ của nút biên tại thời điểm được ký hiệu:

* : chỉ số nút biên.
* : chỉ số time-slot.
* : tập khoảng cách từ nút tới từng xe .
* : các tập kích thước dữ liệu, yêu cầu chu kỳ CPU, và deadline của các tác vụ đang trong vùng phủ của . Toàn bộ trạng thái hệ thống tại thời là .

### 2) **Không gian hành động (Action Space)**

Hành động của nút biên tại thời là các quyết định offloading cho từng tác vụ do xe yêu cầu:

với biểu diễn việc tác vụ có được offload đến nút hay không. Tập các hành động toàn hệ là .

### 3) **Hàm thưởng (Reward Function)**

Mỗi edge node tối đa hóa utility của mình; trong mô hình trò chơi của bài báo, **reward hệ thống tại thời**  được định nghĩa là utility tổng thể:

(trong đó là service ratio của nút lúc ).

Tiếp đó, **hàm potential** của trò chơi được dùng làm reward cục bộ cho mỗi nút. Cụ thể, reward mà nút nhận khi thực hiện hành động trong trạng thái được định nghĩa bằng chênh lệch đóng góp của nó:

trong đó là reward hệ thống khi **loại bỏ đóng góp của nút**  (thiết lập tập hành động của là null). Tập reward toàn hệ .

Mỗi nút tối đa hóa **expected return**:

với là hệ số chiết khấu.

### 4) **Khởi tạo mạng và replay buffer**

* Tham số của mạng policy (actor) và critic (Q-network) được khởi tạo ngẫu nhiên tại learner: ký hiệu (policy/local actor) và (critic).
* Tham số mạng mục tiêu (target networks) khởi tạo bằng cách copy: .
* Replay buffer được khởi tạo với kích thước tối đa .

### 5) **Thuật toán tổng quát (Algorithm 1 — MAD4PG)**

(Bản dịch tóm tắt các bước trong Algorithm 1)

1. Khởi tạo trọng số mạng.
2. Khởi tạo replay buffer .
3. Khởi chạy distributed actors; sao chép trọng số mạng từ learner sang các actors.
4. Lặp cho tới max-iteration:
   * Trong mỗi epoch: lặp qua các time-slot :
     + Với mỗi edge node : lấy minibatch transitions (mỗi transition dài ) từ ; xây dựng target distributions; tính loss cho policy và critic; cập nhật local policy & critic; cập nhật target networks theo chu kỳ ; sao chép trọng số sang actors theo chu kỳ .

(Đây là khung tiêu chuẩn của D4PG mở rộng cho multi-agent, với replay buffer và cập nhật target networks.)

### 6) **Các distributed actors (Algorithm 2)**

Mỗi actor thực hiện:

1. Khởi tạo noise process để khám phá.
2. Nhận trạng thái ban đầu .
3. Với mỗi time-slot :
   * Mỗi edge node nhận quan sát cục bộ .
   * Chọn hành động:
   * hoặc tổng quát (với hệ số khám phá).
   * Thực thi trong môi trường NOMA-VEC, nhận reward và trạng thái tiếp theo .
   * Lưu transition vào replay buffer .
4. Thực hiện cho đến khi learner hoàn tất.

### 7) **Huấn luyện learner (mini-batch, target distribution, loss, gradient)**

* Lấy minibatch transitions, mỗi transition là dãy độ dài : .
* **Target distribution** cho nút :

(trong đó với lấy từ target policy .)

* **Loss của critic**:
* **Gradient policy** (stochastic policy gradient update):
* Cập nhật tham số local policy và critic với learning rates .
* **Cập nhật mạng mục tiêu (soft update)** theo chu kỳ :

(với là hệ số làm mượt; bản gốc ký hiệu là , mình đổi thành để phổ biến hơn.)

* **Đồng bộ actor**: theo chu kỳ , sao chép tham số target policy sang các actor:

**Ghi chú:** đây là kiến trúc D4PG (distributional DDPG) mở rộng cho multi-agent; điểm khác so với DDPG thông thường là sử dụng distributional targets (ở đây biểu diễn bằng target distribution ) và kiến trúc learner + many actors để tăng hiệu quả thu thập dữ liệu.

## 5.3. Phân bổ tài nguyên bằng tối ưu lồi (Dịch & giải thích)

P2 được tách thành hai bài toán độc lập: **phân bổ công suất truyền (transmission power allocation)** và **phân bổ tài nguyên tính toán (computation resource allocation)**.

### 1) **Transmission Power Allocation (Giải P3 → P6 → P9 → cập nhật)**

* Bài toán (sau các biến đổi) đã đưa về dạng tối đa hoá tổng throughput (P4/P5/P6); để giải P6 (convexified form), tác giả dùng **Lagrange dual**.
* **Lagrange function** cho nút :

(trong đó biến , và ràng buộc tổng công suất viết dưới dạng .)

* **Bài toán đối ngẫu (dual)**:
* P9 tách thành **hai lớp**:
  + **Inner layer**: tối đa hoá theo cho cố định.
  + **Outer layer**: tối thiểu hoá theo (cập nhật bằng gradient descent).
* **Cập nhật dual variable (gradient descent)**:

với là bước học nhỏ, số lặp.

* **Giải inner problem**: tìm điểm dừng (stationary point) của theo khi cố định:

(Biểu thức đạo hàm được tính, rồi biến trở lại không gian .)

* Từ đó hình thành **phương trình điểm cố định** và công thức cập nhật công suất của xe (công thức dạng cố định — iterative update):

trong đó:

* Ở đây là tổng **nhiễu nội-biên + nhiễu liên-biên + nhiễu AWGN** tính ở bước lặp .
* Công thức (45) cho một cập nhật theo từng bước lặp; outer loop cập nhật theo (43), inner loop dùng (45) cho tới hội tụ → nghiệm tối ưu (hoặc xấp xỉ) của bài toán công suất.

**Ghi chú:** chi tiết biểu thức đạo hàm ở (44) trong văn bản gốc có nhiều ký hiệu rời rạc; ý chính ở đây là: giải điều kiện stationarity của Lagrangian cho pgt (log-power) → đưa về cập nhật công suất dạng điểm cố định (45), trong đó các tham số là hệ số xấp xỉ/linearization đã định nghĩa trước (Eq. 24 trong phần trước).

### 2) **Computation Resource Allocation (Giải P7 → P8 → P10 bằng KKT)**

* Với P8 (mỗi nút độc lập), ta đặt vấn đề tối ưu phân bổ CPU cho các task đã offload tới với ràng buộc .
* Dùng Lagrange multiplier cho ràng buộc, Lagrangian:
* **Bài toán đối ngẫu P10**:
* Áp dụng **điều kiện KKT** (tại nghiệm tối ưu):
* Giải hệ phương trình KKT thu được nghiệm phân bổ CPU cho mỗi task . Sau giản lược và giải, nghiệm tối ưu có dạng tỉ lệ theo . Cụ thể:

(Đây là nghiệm chuẩn: CPU cycles phân bổ tỉ lệ với — tức task có “khối lượng công việc” lớn hơn (kích thước × chu kỳ/bit) sẽ nhận nhiều CPU hơn theo tỉ lệ căn bậc hai.)

**Phân tích nhanh dẫn tới (49):**

* Mục tiêu P8 là tối thiểu hoá tổng thời gian xử lý với .
* Lagrangian dẫn tới điều kiện: → .
* Từ ràng buộc tổng suy ra . Thay lại, được biểu thức (49).

## Kết luận ngắn gọn về phần 5.2–5.3

* **MAD4PG**: kiến trúc multi-agent dạng learner + nhiều actors; dùng reward = potential contribution (Eq.33) để học policy offloading phân tán, cập nhật bằng distributional D4PG (target distribution, critic loss, policy gradient, soft update).
* **Resource Allocation**: sau khi có quyết định offload , giải P2 bằng chia ra:
  + **Power allocation**: giải đối ngẫu (Lagrange), cập nhật dual bằng gradient, inner-loop giải bằng điểm cố định (iterative update p^t\_{v,e}).
  + **Computation allocation**: P8 là convex; dùng KKT → nghiệm đóng dạng tỉ lệ (49), .