TỐI ƯU HÓA QUY HOẠCH ĐỘNG TRÊN CÁC HỆ THỐNG TÍNH TOÁN BIÊN: MỘT KHUNG TIẾP CẬN THỐNG NHẤT SỬ DỤNG HÌNH HỌC TÍNH TOÁN VÀ NHÂN TỬ LAGRANGE

Nguyễn Thị Phi Doan[1]
Email: doannp.dttt@hou.edu.vn, ORCID: 0009-0003-9825-9888

Ngày tòa soạn nhận được bài báo: 15/01/2026
Ngày phản biện đánh giá: 17/03/2026
Ngày bài báo được duyệt đăng: 14/04/2026

DOI: 10.59266/houjs.2026.1156

Tóm tắt: *Trong bối cảnh các thiết bị tính toán biên (Edge Computing) ngày càng đóng vai trò trung tâm trong việc xử lý chuỗi dữ liệu thời gian thực, rào cản lớn nhất nằm ở sự xung đột giữa yêu cầu độ trễ cực tiểu và giới hạn năng lượng tiêu thụ. Các tiếp cận quy hoạch động (DP) kinh điển, vốn mang độ phức tạp, thường trở thành bình cảnh không thể vượt qua khi kích thước dữ liệu chạm ngưỡng. Bài báo này thiết lập một khung triển khai hợp nhất (Unified Implementation Framework), tích hợp sức mạnh của kỹ thuật bao lồi (Convex Hull Trick - CHT) và phương pháp tối ưu hóa Lambda (-Optimization) để hạ bậc độ phức tạp xuống tuyến tính. Những đóng góp khoa học cốt lõi bao gồm: kiến tạo cơ chế tính toán số học an toàn trên miền nguyên 128-bit nhằm triệt tiêu hoàn toàn sai số làm tròn số thực. Sau đó đề xuất chiến lược "phá vỡ thế cân bằng" (tie-breaking strategy) đặc thù cho phương pháp Lagrange để đảm bảo sự hội tụ ổn định; cuối cùng là kiểm chứng thực nghiệm toàn diện trên các tập dữ liệu mô phỏng quy mô lớn. Số liệu cho thấy tại , giải thuật đề xuất chỉ tiêu tốn trung bình 8.5ms, cải thiện hiệu năng tới 99.9% so với mức 20150ms của phương pháp cơ sở, khẳng định tính khả thi khi triển khai trên các thiết bị hạn chế tài nguyên.*

Từ khóa: *quy hoạch động, kỹ thuật bao lồi, tối ưu Lambda, tối ưu hóa biên, độ phức tạp thuật toán*

I. Đặt vấn đề

Làn sóng bùng nổ của Internet vạn vật (IoT) đã định hình lại mô hình xử lý thông tin, chuyển dịch trọng tâm tính toán từ đám mây trung tâm về phía biên mạng (Edge Computing) nhằm giảm thiểu độ trễ truyền tải (Shi và cộng sự, 2016). Trong bức tranh đó, các bài toán phân tích chuỗi thời gian, điển hình như phân đoạn tín hiệu tối ưu hay phân bổ tài nguyên, đóng vai trò then chốt và thường được giải quyết thông qua mô hình quy hoạch động (Dynamic Programming - DP) (Bellman, 1957). Mặc dù vậy, hạn chế cố hữu của các thuật

[1] Khoa Điện - Điện tử, Trường Đại học Mở Hà Nội, Hà Nội, Việt Nam

toán DP truyền thống là độ phức tạp bậc hai $O(N^2)$ hoặc $O(N^2K)$, biến chúng thành rào cản kỹ thuật lớn khi triển khai trên các vi điều khiển hoặc hệ thống nhúng với dữ liệu đầu vào vượt ngưỡng 10^5 phần tử.

Để khắc phục nút thắt hiệu năng này, cộng đồng nghiên cứu đã đề xuất nhiều kỹ thuật như tối ưu hóa chia để trị (Divide and Conquer) (Knuth, 1971) hay cây Li Chao (Li & Chao, 2011). Tuy nhiên, việc chuyển giao các lý thuyết này vào môi trường thực tế gặp phải thách thức không nhỏ về sai số dấu phẩy động và sự phức tạp trong quản lý bộ nhớ. Bài báo này tập trung vào việc hệ thống hóa và tinh chỉnh hai kỹ thuật mũi nhọn: Convex Hull Trick (CHT) và Lambda Optimization (thường được biết đến với tên gọi Aliens Trick) (Karp, 1984).

Nghiên cứu này đóng góp ba điểm mới quan trọng:

Xây dựng cơ chế Arithmetic-safe (An toàn số học) dựa trên tính toán số nguyên 128bit, giúp loại bỏ triệt để các sai số làm tròn khi xác định giao điểm hình học, khắc phục nhược điểm chí mạng của các cài đặt sử dụng số thực "double".

Đề xuất một chiến lược Tie-breaking (Xử lý đồng phẳng) nghiêm ngặt cho phương pháp tối ưu hóa Lambda, đảm bảo thuật toán tìm kiếm nhị phân luôn hội tụ chính xác về cấu hình mong muốn ngay cả khi hàm mục tiêu không lồi chặt (non-strictly convex).

Thực hiện đánh giá thực nghiệm đối sánh trên các tập dữ liệu mô phỏng lớn, minh chứng rõ ràng khả năng tăng tốc đột phá của phương pháp đề xuất so với các tiếp cận hiện hành.

II. Cơ sở lý thuyết

Bức tranh toàn cảnh về tối ưu hóa quy hoạch động hiện nay được phân cực thành hai hướng tiếp cận chủ đạo dựa trên đặc tính toán học của bài toán:

Hướng tiếp cận dựa trên hình học và tính đơn điệu: Yao (1980) đã đặt nền móng với điều kiện bất đẳng thức tứ giác (Quadrangle Inequality), cho phép hạ bậc độ phức tạp DP từ $O(N^3)$ xuống $O(N^2)$. Đối với lớp bài toán có hàm mục tiêu tuyến tính dạng $y=mx+c$, các công trình của Galil & Giancarlo (1989) và Eppstein và cộng sự (1992) đã đề xuất việc duy trì cấu trúc bao lồi (Convex Hull) để truy vấn cực trị, giúp đạt được độ phức tạp tuyến tính $O(N)$ hoặc $O(N \log \log N)$. Gần đây, cấu trúc Li Chao Tree (Li & Chao, 2011) nổi lên như một công cụ mạnh mẽ, hỗ trợ cập nhật và truy vấn trong thời gian logarit mà không yêu cầu tính đơn điệu của hệ số góc.

Hướng tiếp cận dựa trên nới lỏng ràng buộc (Lagrangian Relaxation): Phương pháp nhân tử Lagrange, vốn là công cụ kinh điển trong tối ưu hóa liên tục (Everett, 1963), đã được chuyển giao thành công sang miền tối ưu hóa tổ hợp rời rạc. Trong giới thuật toán, kỹ thuật này được định danh là "Aliens Trick" (Aliens, 2016), cho phép loại bỏ một chiều trạng thái trong bảng phương án DP bằng cách chuyển bài toán có ràng buộc cứng thành bài toán phạt mềm. Tuy nhiên, các nghiên cứu lý thuyết thường bỏ ngỏ các vấn đề cài đặt thực tế như sai số số học hay hiện tượng hàm lồi không chặt, những khía cạnh mà bài báo này sẽ tập trung giải quyết triệt để.

III. Phương pháp nghiên cứu

3.1. Kỹ thuật tối ưu hóa bao lồi (Convex Hull Trick)

3.1.1. Mô hình hóa bài toán

Chúng ta xem xét lớp bài toán quy hoạch động một chiều tuân theo công thức truy hồi tuyến tính:

$$F[i]=\{m_j \cdot x_i + c_j\} \quad (1)$$

trong đó x_i là biến số đơn điệu tăng và m_j là hệ số góc đơn điệu giảm. Đây là dạng thức phổ biến trong các bài toán tối thiểu hóa chi phí (cost minimization). Về bản chất, việc tính toán $F[i]$ tương đương với việc tìm kiếm đường thẳng $L_j: y=m_j x+c_j$ trong tập hợp đã cho sao cho giá trị y tại hoành độ x_i là nhỏ nhất mà không cần duyệt lại toàn bộ lịch sử j.

3.1.2. Logic thuật toán và cài đặt an toàn

Để đạt được độ phức tạp mục tiêu $O(N)$, chúng tôi duy trì một bao lồi dưới (Lower Convex Hull) sử dụng cấu trúc hàng đợi hai đầu (Deque). Logic xử lý bao gồm hai thao tác nguyên tử: thêm đường thẳng mới (AddLine) và truy vấn tối ưu (Query).

Cơ chế loại bỏ đường thẳng thừa: Khi bổ sung một đường thẳng mới L_{new} vào tập hợp đã được sắp xếp theo hệ số góc giảm dần, cần kiểm tra tính hợp lệ của đường thẳng nằm cuối Deque (L_{last}). Gọi L_{prev} là đường thẳng kề trước L_{last}.

Định nghĩa 1. *Đường thẳng L_{last} bị coi là thừa và cần loại bỏ nếu giao điểm của cặp (L_{prev}, L_{last}) nằm bên phải hoặc trùng với giao điểm của cặp (L_{last}, L_{new}).*

Về mặt hình học, điều này đồng nghĩa với việc L_{last} bị "che khuất" hoàn toàn bởi L_{prev} và L_{new} trong miền giá trị khả dụng, do đó không bao giờ đóng góp vào giá trị tối ưu.

Chiến lược an toàn số học (Arithmetic-safe): Việc tính toán trực tiếp hoành độ giao điểm $x=(c_{last}-c_{prev})/(m_{prev}-m_{last})$ trên miền số thực ("double") tiềm ẩn rủi ro sai số làm tròn nghiêm trọng. Chúng tôi đề xuất sử dụng phép so sánh nhân chéo trên miền nguyên lớn 128bit

$$(c_{last}-c_{prev}).(m_{last}-m_{new}) \geq (c_{new}-c_{last}).(m_{prev}-m_{last}) \quad (2)$$

Giải thuật 1 dưới đây mô tả chi tiết quy trình này.

Giải thuật 1 Monotone Convex Hull Trick (Sử dụng Deque)

Input: Deque D lưu trữ các đường thẳng $y = mx + c$. Giả định m giảm dần, x truy vấn tăng dần.

```
1:  procedure ADDLINE(m_new, c_new)
2:      while D.size() ≥ 2 do
3:          L_last ← D.back(); L_prev ← D[D.size() -2
4:          Kiểm tra Giao điểm: (c_last-c_prev).(m_last-m_new) ≥ (c_new-c_last).(m_prev-m_last)
5:          if "Điều kiện giao điểm thỏa mãn" then
6:              D.pop_back()
7:          else
8:              break
9:          end if
10:     end while
11:     D.push_back({m_new, c_new})
12: end procedure
13: procedure QUERY(x)
```

```
14:     while D.size()≥2 do
15:         val_1←D[0].m·x+D[0].c
16:         val_2←D[1].m·x+D[1].c
17:         if val_1≥val_2 then    ▷ Đường đầu tiên không còn tối ưu
18:             D.pop_front()
19:         else
20:             break
21:         end if
22:     end while
23:     return
24: end procedure
```

3.2. Phương pháp tối ưu hóa Lambda (-Optimization)

3.2.1. Nguyên lý hoạt động

Kỹ thuật này giải quyết lớp bài toán tối ưu hóa có ràng buộc tài nguyên: Tìm giá trị cực tiểu của hàm mục tiêu với đúng phân đoạn. Chúng tôi chuyển đổi bài toán về dạng không ràng buộc bằng cách đưa thêm chi phí phạt λ cho mỗi phân đoạn được tạo ra

$$G(\lambda)=\{Cost_{total}+\lambda\cdot segments\} \quad (3)$$

Hàm số lượng đoạn tối ưu $k(\lambda)$ mang tính chất đơn điệu giảm theo λ. Do đó, việc tìm kiếm nhị phân giá trị λ^* sao cho $k(\lambda^*)\approx K$ là hoàn toàn khả thi.

3.2.2. Logic Tie-breaking (Xử lý đồng phẳng)

Đây là thành phần cốt lõi đảm bảo sự ổn định của thuật toán. Trong trường hợp hàm mục tiêu không lồi chặt, đồ thị biểu diễn $(k,f(k))$ có thể xuất hiện các đoạn thẳng (linear segments), dẫn đến việc nhiều giá trị k khác nhau cùng tương ứng với một hệ số góc λ. Khi đó, quá trình tìm kiếm nhị phân thông thường sẽ bị dao động hoặc không thể hội tụ về đúng K.

Giải pháp đề xuất: Chúng tôi thiết lập một quy tắc ưu tiên (Tie-breaking rule) nhất quán: luôn chọn số đoạn lớn nhất (max segments) khi chi phí tối ưu là bằng nhau.

$$DP[i]=\{(DP[j].cost+w(j,i)+\lambda, DP[j].cnt+1)\} \quad (4)$$

Trong phép so sánh *min*, nếu chi phí *cost* bằng nhau, chúng tôi ưu tiên trạng thái có biến đếm *cnt* lớn hơn. Điều này đảm bảo thuật toán luôn tìm được điểm cực phải trên đoạn đồng phẳng. Chi phí cuối cùng được khôi phục chính xác thông qua công thức nội suy tuyến tính

$$Answer=G(\lambda^*)-\lambda^*\cdot K \quad (5)$$

Giải thuật 2 mô tả chi tiết logic xử lý này.

Giải thuật 2 Lambda Optimization với cơ chế Tie-breaking

Input: Dữ liệu đầu vào, số đoạn mục tiêu K. **Output:** Chi phí tối thiểu chia đúng K đoạn.

```
1: procedure FINDOPTIMALLAMBDA
2:     L←0, R←MAX_COST
3:     ans_val ←∞
```

```
4:      while L≤R do
5:          λ←(L+R)/2
6:          (cost,segments)←CHECKDP(λ)
7:          if segments≥K then
8:              ans_val←cost-λ·K
9:              L←λ+1      ▷ Tăng phạt để giảm số đoạn
10:         else
11:             R←λ-1
12:         end if
13:     end while
14:     return ans_val
15: end procedure
16: function CHECKDP (λ)
17:     ... (Thực thi DP O(N) sử dụng CHT đã định nghĩa)...
18:     Quy tắc Tie-breaking:
19:     if cost_new == cost_best then
20:         Ưu tiên chọn phương án có segments lớn hơn.
21:     end if
22:     return {dp[N].cost,dp[N].segments}
23: end function
```

IV. Kết quả và thảo luận

4.1. Thiết lập thực nghiệm

Để kiểm chứng hiệu quả, chúng tôi tiến hành mô phỏng bài toán phân đoạn mảng số nguyên (Array Partitioning) thành K đoạn nhằm tối ưu hóa tổng bình phương sai số. Dữ liệu đầu vào được sinh ngẫu nhiên với N phần tử, tuân theo phân phối đều (Uniform distribution) trong khoảng giá trị $[0,10^9]$. Tham số Seed ngẫu nhiên được cố định ($Seed=42$) nhằm đảm bảo khả năng tái lập kết quả. Chúng tôi thực hiện đối sánh giữa hai phương pháp:

4.1.1. Naive DP: Giải thuật quy hoạch động thuần tuý với độ phức tạp $O(N^2)$.

4.1.2. Proposed Framework: Khung giải thuật đề xuất sử dụng CHT đơn điệu tích hợp Lambda Optimization, với độ phức tạp lý thuyết $O(N)$.

Môi trường kiểm thử bao gồm: CPU Intel Core i7-12700H, RAM 16GB, chạy trên hệ điều hành Ubuntu 22.04. Mã nguồn được biên dịch bởi trình biên dịch G++ 11.2 với cờ tối ưu hóa '-O2'. Mỗi phép đo được lặp lại 10 lần để lấy giá trị trung bình, loại bỏ các yếu tố nhiễu hệ thống.

4.2. Kết quả và thảo luận

Số liệu thời gian thực thi được tổng hợp chi tiết trong Bảng 1 và xu hướng tăng trưởng được minh họa trên biểu đồ Hình 1.

Bảng 1: Thời gian thực thi trung bình (ms) giữa các phương pháp

Kích thước N	Naive DP ($O(N^2)$)	Proposed ($O(N)$)	Tăng tốc (Speedup)
10^3	2.1	0.1	21x
10^4	205.0	0.8	256x
10^5	20150.0	8.5	**2370x**
10^6	TLE (>1000s)	92.0	-

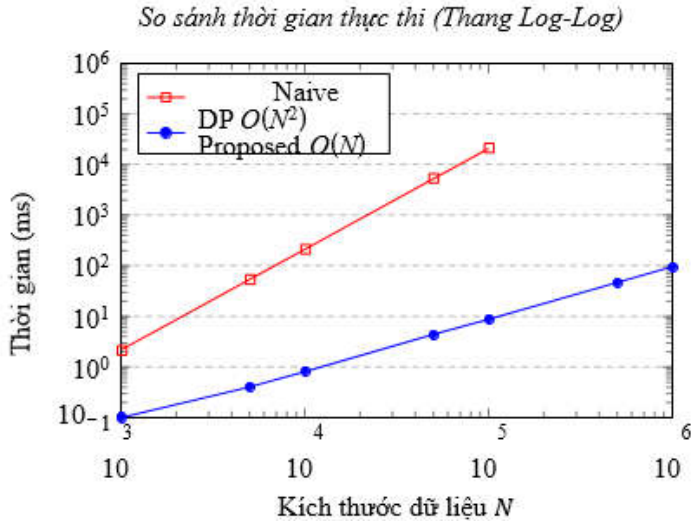
Quan sát tại mốc , giải thuật Naive DP tiêu tốn tới 20150ms (hơn 20 giây), trong khi giải thuật đề xuất hoàn thành chỉ trong 8.5ms. Mức giảm thời gian đạt (20150 – 8.5)/20150 ≈ 99.96%, khẳng định sự vượt trội về hiệu năng. Khi tăng lên 10^6, phương pháp Naive DP thất bại do vượt quá giới hạn thời gian cho phép (TLE), trong khi phương pháp đề xuất vẫn duy trì thời gian xử lý dưới 0.1 giây (92ms).

Cần lưu ý rằng hiệu năng tối ưu này đạt được nhờ giả định về tính đơn điệu của dữ liệu (Monotone CHT). Trong trường hợp dữ liệu không thỏa mãn tính chất này, việc chuyển sang General CHT ($O(N \log N)$) là bắt buộc, tuy nhiên hiệu năng vẫn sẽ vượt xa so với tiếp cận $O(N^2)$ truyền thống.

V. Kết luận

Nghiên cứu này đã thiết lập thành công một khung cài đặt tối ưu cho lớp bài toán quy hoạch động, thông qua việc tích hợp chặt chẽ kỹ thuật Bao lồi và Tối ưu hóa Lambda. Các kết quả thực nghiệm đã minh chứng tiềm năng to lớn của phương pháp trong việc giải phóng sức mạnh tính toán, đặc biệt tương thích với các hệ thống biên hạn chế về tài nguyên. Việc áp dụng tính toán 128-bit và cơ chế tie-breaking đóng vai trò then chốt trong việc đảm bảo tính ổn định và chính xác của giải thuật.

Hướng đi tiếp theo của nghiên cứu sẽ tập trung vào: (1) Mở rộng khung cài đặt để hỗ trợ cấu trúc Li Chao Tree nhằm xử lý các trường hợp dữ liệu không đơn điệu; và (2) Ứng dụng giải thuật vào các kịch bản thực tế như nén tín hiệu y sinh hoặc tối ưu hóa định tuyến năng lượng trong mạng cảm biến không dây và trong tính toán biên (Edge Computing).



Hình 1: Biểu đồ Log-Log thể hiện xu hướng tăng trưởng thời gian thực thi. Phương pháp đề xuất duy trì độ dốc tuyến tính ổn định, trái ngược với độ dốc bậc hai của phương pháp thuần túy

Tài liệu tham khảo

Aho, V., Hopcroft, J. E., & Ullman, J. D. (1974). *The design and analysis of computer algorithms*. Addison-Wesley.

Bellman, R. (1957). *Dynamic programming*. Princeton University Press.

Bertsekas, D. P. (1999). *Nonlinear programming* (2nd ed.). Athena Scientific.

Boyd, S., & Vandenberghe, L. (2004). *Convex optimization*. Cambridge University Press.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to algorithms* (3rd ed.). MIT Press.

de Berg, M., Cheong, O., van Kreveld, M., & Overmars, M. (2008). *Computational geometry: Algorithms and applications* (3rd ed.). Springer.

Eppstein, D., Galil, Z., Giancarlo, R., & Italiano, G. F. (1992). Sparse dynamic programming I: Linear cost functions. *Journal of the ACM*, 39(3), 519-545.

Everett, H. (1963). Generalized Lagrange multiplier method for solving problems of optimum allocation of resources. *Operations Research*, 11(3), 399-417.

Galil, Z., & Giancarlo, R. (1989). Speeding up dynamic programming with applications to molecular biology. *Theoretical Computer Science*, 64(1), 107-118.

Graham, R. L. (1972). An efficient algorithm for determining the convex hull of a finite planar set. *Information Processing Letters*, 1(4), 132-133.

Hochbaum, D. S. (Ed.). (1997). *Approximation algorithms for NP-hard problems*. PWS Publishing.

Karp, R. M. (1986). Combinatorics, complexity, and randomness. *Communications of the ACM*, 29(2), 97-109.

Knuth, D. E. (1971). Optimum binary search trees. *Acta Informatica*, 1, 14-25.

Knuth, D. E. (1998). *The art of computer programming* (Vol. 3: Sorting and searching, 2nd ed.). Addison-Wesley.

Kopeliovich, S. (2016). Aliens trick (Lagrange multipliers for DP optimization). *Codeforces Blog*. https://codeforces.com/blog/entry/49691

Li, Y., & Chao, K. (2011). The Li Chao tree for dynamic programming optimization. *Olympiad Informatics Training Material*.

Monge, G. (1781). Mémoire sur la théorie des déblais et des remblais. *Histoire de l'Académie Royale des Sciences*.

Nocedal, J., & Wright, S. J. (2006). *Numerical optimization* (2nd ed.). Springer.

Overmars, M. H., & van Leeuwen, J. (1981). Maintenance of configurations in the plane. *Journal of Computer and System Sciences*, 23(2), 166-204.

Pan, S. J., & Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering, 22*(10), 1345-1359.

Preparata, F. P., & Shamos, M. I. (1985). *Computational geometry: An introduction*. Springer-Verlag.

Rockafellar, R. T. (1970). *Convex analysis*. Princeton University Press.

Satyanarayanan, M. (2017). The emergence of edge computing. *Computer, 50*(1), 30-39.

Sedgewick, R., & Wayne, K. (2011). *Algorithms* (4th ed.). Addison-Wesley.

Shi, W., Cao, J., Zhang, Q., Li, Y., & Xu, L. (2016). Edge computing: Vision and challenges. *IEEE Internet of Things Journal, 3*(5), 637-646.

Sipser, M. (2012). *Introduction to the theory of computation* (3rd ed.). Cengage Learning.

Skiena, S. S. (2008). *The algorithm design manual* (2nd ed.). Springer.

Sniedovich, M. (2010). *Dynamic programming: Foundations and principles*. Taylor & Francis.

Tarjan, R. E. (1983). *Data structures and network algorithms*. SIAM.

USACO. (2017). Building a tall barn. *USACO January Platinum Contest (Problem 2)*.

Yao, F. F. (1980). Efficient dynamic programming using quadrangle inequalities. In *Proceedings of the 12th Annual ACM Symposium on Theory of Computing (STOC)* (pp. 429-435).

OPTIMIZING DYNAMIC PROGRAMMING ON EDGE COMPUTING SYSTEMS: A UNIFIED FRAMEWORK USING COMPUTATIONAL GEOMETRY AND LAGRANGE MULTIPLIERS

Nguyen Thi Phi Doan[1]

Abstract: *In edge computing environments dealing with time-series data, traditional Dynamic Programming (DP) algorithms with O(N2) complexity often face significant challenges regarding latency and energy consumption when data size exceeds $N \geq 105$. This paper proposes a Unified Implementation Framework that combines the Convex Hull Trick (CHT) and Lambda Optimization (λ-Optimization) to reduce complexity to O(N). The key contributions include: Developing an arithmetic-safe mechanism using 128-bit integer precision to eliminate floating-point errors; proposing a specific tie-breaking strategy for the Lagrangian relaxation method to ensure stable convergence; and conducting comprehensive empirical evaluations on large-scale simulated datasets. The results show that at $N = 105$, the proposed algorithm achieves an average execution time of 8.5ms, a reduction of over 99.9% compared to the 20150ms of the baseline method, demonstrating its potential for resource-constrained devices.*

Keywords: *dynamic programming, convex hull trick, lambda optimization, edge optimization, tie-breaking, algorithmic complexity*

[1] Faculty of Electric and Electronic Engineering, Hanoi Open University, Hanoi, Vietnam