Giải thuật đàn kiến tự thích ứng cho bài toán điều hướng thu thập

Lê Thế Việt - 20520093, Huỳnh Hoàng Vũ - 20520864

Vietnam National University, Ho Chi Minh City - University of Information Technology

Giảng viên hướng dẫn: TS. Lương Ngọc Hoàng

January, 2024

Our Publication

Vu Hoang Huynh, The Viet Le and Ngoc Hoang Luong. "Self-Adaptive Ant System with Hierarchical Clustering for the Thief Orienteering Problem". In: Proceedings of the 12th International Symposium on Information and Communication Technology. SOICT 2023. ACM, Dec. 2023.

Table of Contents

Problem description

Thief Orienteering Problem (ThOP)

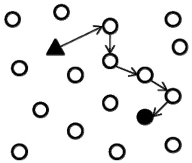
ThOP[1] is a **multi-component optimization problem**, it combines the **Orienteering Problem (OP)** and **Knapsack Problem (KP)**.

[1] André G. Santos et al. "The Thief Orienteering Problem: Formulation and Heuristic Approaches". In: *2018 IEEE Congress on Evolutionary Computation (CEC)*. 2018, pp. 1–9

Problem description

Orienteering problem (OP)

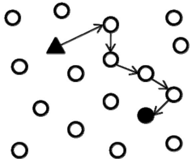
OP is a **routing problem** in which the goal is to determine a path through a given set of points of interest that **maximizes a total score** while **satisfying a given time budget**.



Problem description

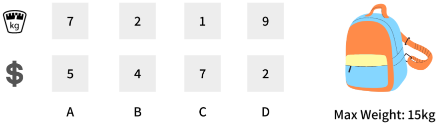
Orienteering problem (OP)

OP is a **routing problem** in which the goal is to determine a path through a given set of points of interest that **maximizes a total score** while **satisfying a given time budget**.

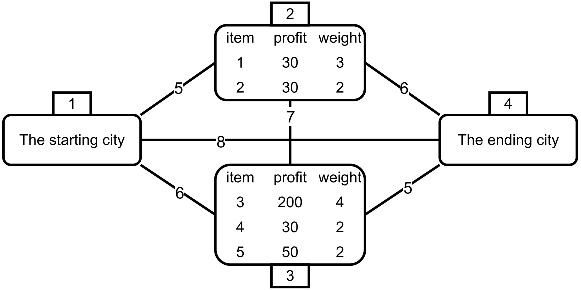


Knapsack problem (KP)

KP is an **optimization problem** in which the goal is to **select a subset of items** from a given set such that **the total value** of the selected items **is maximized**, while the **total weight** of the selected items does **not exceed a given capacity**.



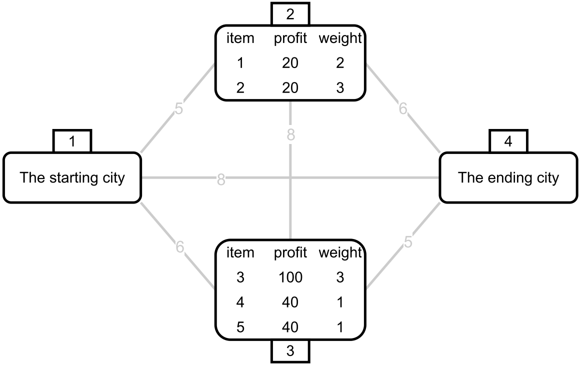
Example



Constraints

- $n = 4$, $m = 5$
- $v_{min} = 0.1$, $v_{max} = 1.0$, $W = 3$, $T = 75$

Example



Constraints

- $n = 4, m = 5$
- $v_{min} = 0.1, v_{max} = 1.0, W = 3, T = 75$

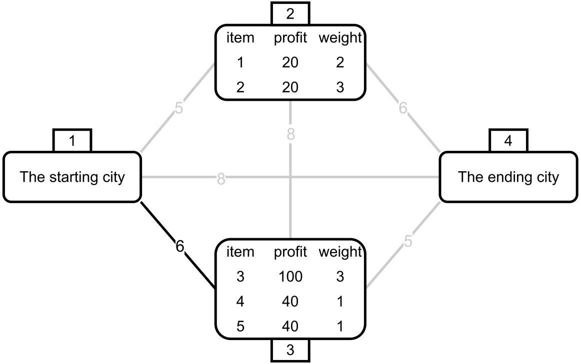
Solution

- $\pi = \langle 1 \rangle$
- $p = \langle 0, 0, 0, 0, 0 \rangle$

Properties

- $p = 0$
- $w = 0$
- $v = v_{max} = 1.0$
- $t = 0$

Example



Constraints

- $n = 4, m = 5$
- $v_{min} = 0.1, v_{max} = 1.0, W = 3, T = 75$

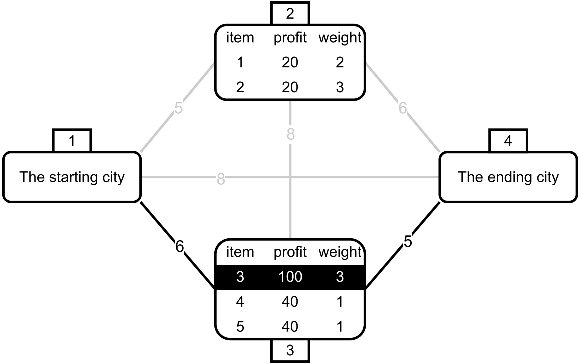
Solution

- $\pi = \langle 1, 3 \rangle$
- $p = \langle 0, 0, 0, 0, 0 \rangle$

Properties

- $p = 0$
- $w = 0$
- $v = v_{max} = 1.0$
- $t = d_{1,3}/v = 6/1.0 = 6$

Example



Constrains

- $n = 4, m = 5$
- $v_{min} = 0.1, v_{max} = 1.0, W = 3, T = 75$

Solution

- $\pi = \langle 1, 3, 4 \rangle$
- $p = \langle 0, 0, 1, 0, 0 \rangle$

Properties

- $p = 100$
- $w = 0 + w_3 = 3$
- $v = v_{max} - w(v_{max} - v_{min})/W = 0.1$
- $t = t + d_{3,4}/v = 6 + 5/0.1 = 56$

ThOP benchmark

ThOP Benchmark Overview

The ThOP benchmark is a collection of 432 instances. Each instance has unique characteristics, including:

- Number of cities: 51, 107, 280, or 1000.
- Number of items per city: 01, 03, 05, or 10.
- Knapsack types: uncorrelated (unc), uncorrelated with similar weights (usw), or bounded and strongly correlated (bsc).
- Knapsack size: 01, 05, or 10 times the size of the smallest knapsack.
- Maximum travel time: 50%, 75%, or 100%.

Related Works

Prior works have proposed various algorithms for ThOP

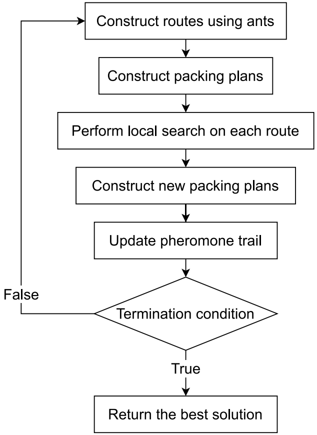
- Iterated local search algorithm (ILS)[1]
- Biased random-key genetic algorithm (BRKGA)[1]
- Genetic Algorithm (GA)[2]
- Ant Colony Optimization algorithm (ACO)[3]
- Max-Min Ant System algorithm (ACO++)[4]

[2] Leonardo M. Faêda et al. "A Genetic Algorithm for the Thief Orienteering Problem". In: *2020 IEEE Congress on Evolutionary Computation (CEC)*. 2020, pp. 1–8

[3] Jonatas B.C. Chagas et al. "Ants can orienteer a thief in their robbery". In: *Operations Research Letters* 48.6 (2020), pp. 708–714. ISSN: 0167-6377

[4] Jonatas B. C. Chagas et al. "Efficiently solving the thief orienteering problem with a max–min ant colony optimization approach". In: *Optimization Letters* 16.8 (Nov. 2021), pp. 2313–2331

Max-min ant colony optimization for ThOP



ACO++

ACO++ overview

- ACO++, or Max-min Ant Colony Optimization for ThOP, emerged as the state-of-the-art algorithm upon its introduction by its authors.
- ACO++ algorithm is a combination of a heuristic approach based on MAX-MIN Ant Colony Optimization with a randomized packing heuristic and local searches.
- ACO++ outperformed all other previous algorithms (ACO, BRKGA, ILS, GA) by more than 96% of the total of test cases.

ACO++'s Randomized Packing Heuristic

Randomized Packing Heuristic

1. Consider the next item having the highest score.
2. If picking the item does not violate any constraints, add it to the packing plan.
3. While items are not all considered, go to step 1.

ACO++'s Randomized Packing Heuristic

Randomized Packing Heuristic

1. Consider the next item having the highest score.
2. If picking the item does not violate any constraints, add it to the packing plan.
3. While items are not all considered, go to step 1.

Item score s_i is determined using the formula

$$s_i = \frac{p_i^{\theta}}{w_i^{\delta} * d_i^{\gamma}}$$

ACO++'s Randomized Packing Heuristic

Randomized Packing Heuristic

1. Consider the next item having the highest score.
2. If picking the item does not violate any constraints, add it to the packing plan.
3. While items are not all considered, go to step 1.

Item score s_i is determined using the formula

$$s_i = \frac{p_i^\theta}{w_i^\delta * d_i^\gamma}$$

Parameters

- p_i: Profit of item i.
- w_i: Weight of item i.
- d_i: Distance from item i's city to the ending city along the route.
- θ, δ, γ: Randomized values within the range [0,1].

Investigating ACO++ Sensitivity to Parameters

The extensive tuning process

- The ACO++ requires an extensive tuning process to achieve its out-performance results.
- The results were obtained using different sets of parameter values that have been extensively fine-tuned for each specific instance group.

Experiment on ACO++ parameter sensitivity

Our experiments were carried out on two instance groups from the ThOP benchmark:

- `a280_01_unc`: 280 cities, 1 item per city, profits uncorrelated with weights.
- `dsj1000_01_unc`: 1000 cities, 1 item per city, profits uncorrelated with weights.

Investigating ACO++ Sensitivity to Parameters

Experiment 1

Used fine-tuned parameter sets for each instance group.

Experiment 2

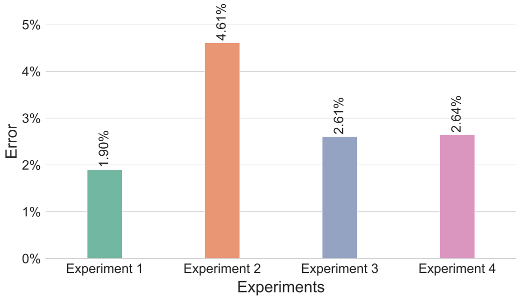
Swapped parameter sets between instance groups.

Experiment 3

Increased α, β, and ρ by 3% and packing tries by 1.

Experiment 4

Used average parameter values from 48 configurations.



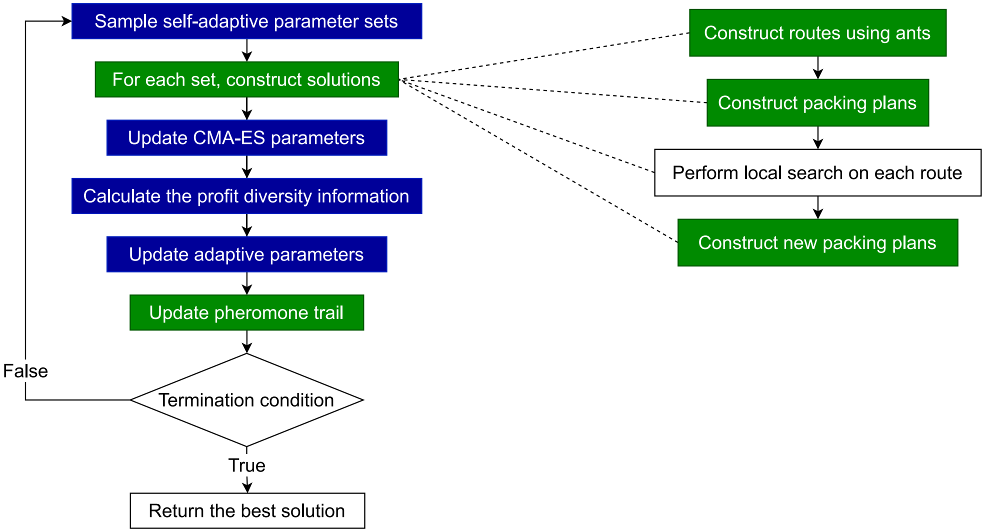
Hình: Mean errors of results of 4 ACO++ experiments with different parameter sets on 18 instances belonging to 2 ThOP instance groups.

The overall algorithm

Overview

- Our proposed method SAAS, stands for **S**elf-**A**daptive **A**nt **S**ystem, is an extension of ACO++.
- It can adapt its parameters based on the ThOP instance and the search process.
- It also has a lower time complexity in the route-finding and pheromone evaporation phases.

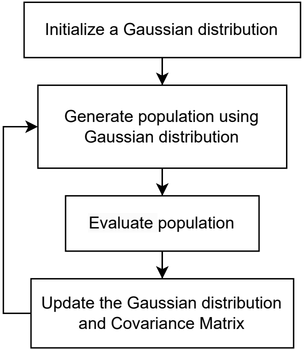
The overall algorithm



CMA-ES

Introduction

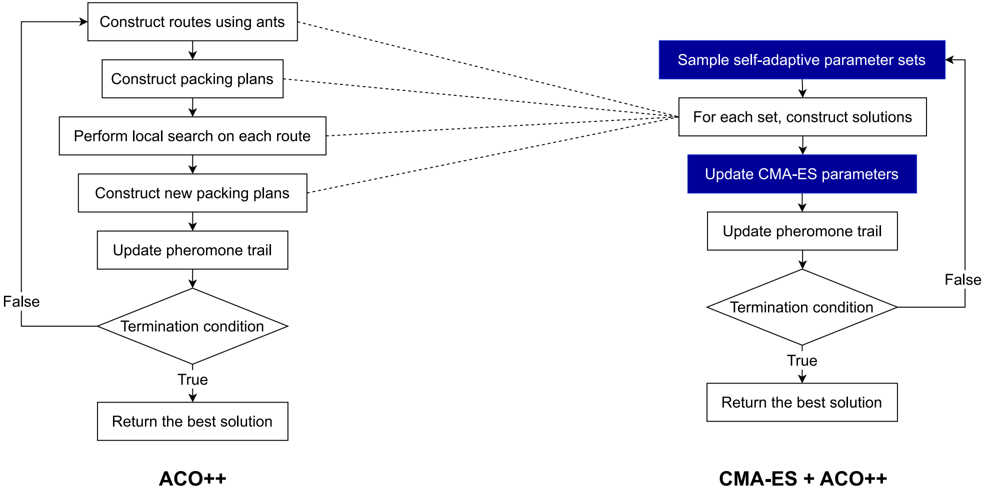
CMA-ES[5] stands for **Covariance Matrix Adaptation Evolution Strategy**, which is a **stochastic**, **derivative-free** method for numerical optimization of **non-linear** or **non-convex continuous** optimization problems.



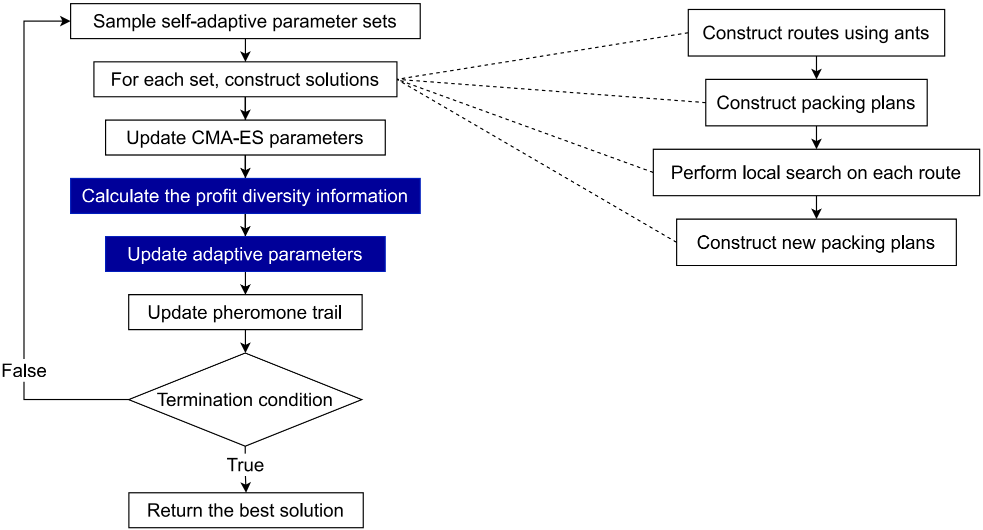
Hình: Simplified CMA-ES.

[5] Nikolaus Hansen. "The CMA Evolution Strategy: A Comparing Review". In: *Towards a New Evolutionary Computation: Advances in the Estimation of Distribution Algorithms*. Ed. by Jose A. Lozano et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 75–102. ISBN: 978-3-540-32494-2

Self-adaptive mechanism with CMA-ES



Utilizing the profit diversity information for adaptation



Utilizing the profit diversity information for adaptation

Key ideas

Inspired by AACO-NC[6], we use profit diversity information to dynamically change both the number of ants for each ES individual and the pheromone evaporation rate.

Profit diversity information

$$p_i = \frac{\#\text{occurrences}(P_i)}{n_{\text{ants}}} \tag{1}$$

$$S = \{P \mid P \text{ is a unique profit value found by the current swarm}\},$$

$$H = -\sum_{i=1}^{|S|} p_i \cdot \log_2 p_i \tag{2}$$

[6] Petr Stodola et al. "Adaptive Ant Colony Optimization with node clustering applied to the Travelling Salesman Problem". In: *Swarm and Evolutionary Computation* 70 (2022), p. 101056. ISSN: 2210-6502

Utilizing the profit diversity information for adaptation

Adapting the pheromone evaporation rate

- The evaporation rate **increases** for **high** profit diversity and **decreases** for **low** diversity.

$$\rho = \rho_{\mathbf{min}} + (\rho_{\max} - \rho_{\min}) \cdot \frac{H - H_{\min}}{H_{\max} - H_{\min}}. \tag{3}$$

Adapting the number of ants for each ES individual

- Unlike the evaporation rate, the value of n_{indv} **increases** for **low** profit diversity to encourage exploration and **decreases** for **high** diversity to facilitate exploitation.

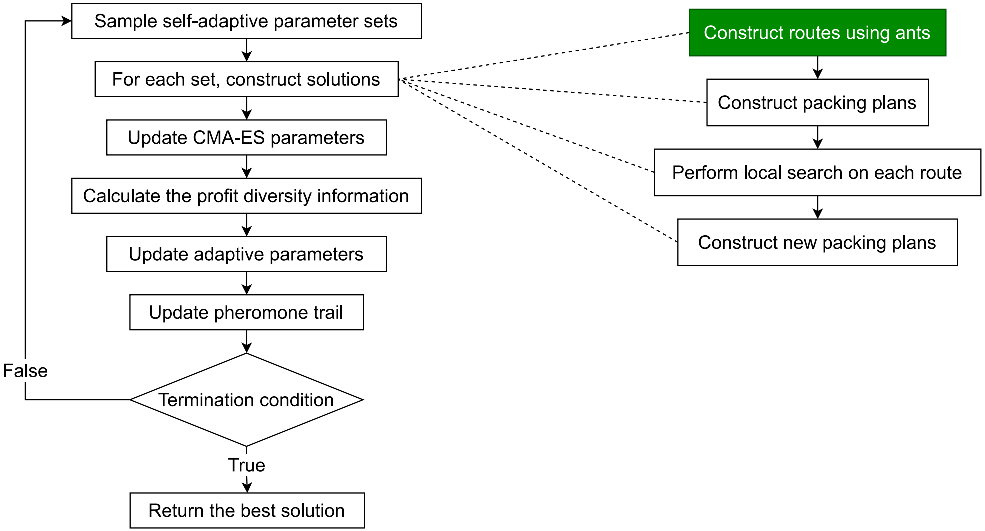
$$n_{\text{indv}} = n_{\text{indv\_}\mathbf{max}} \text{ - } (n_{\text{indv\_max}} - n_{\text{indv\_min}}) \cdot \frac{H - H_{\min}}{H_{\max} - H_{\min}}. \tag{4}$$

Parameters controlled

Bảng: List of Parameters controlled by Parameter Control Mechanisms

Parameter	Parameter control mechanism	Range
α	Self-adaptive	[0, 1]
β	Self-adaptive	[0, 1]
$\rho_{\text{min}}, \rho_{\text{max}}$	Self-adaptive	[0, 1]
θ	Self-adaptive	[0, 1]
δ	Self-adaptive	[0, 1]
γ	Self-adaptive	[0, 1]
n_{indv}	Adaptive	$[n_{\text{indv\_max}}, n_{\text{indv\_min}}]$
ρ	Adaptive	$[\rho_{\text{min}}, \rho_{\text{max}}]$

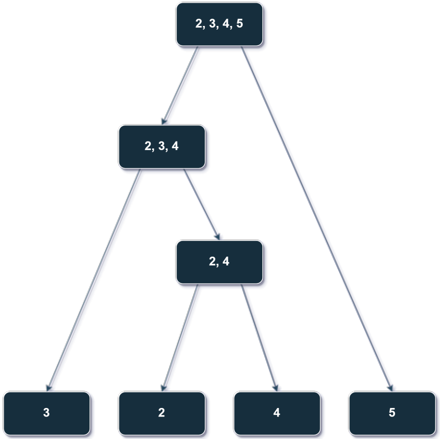
Ant traversal on cluster trees



Ant traversal on cluster trees

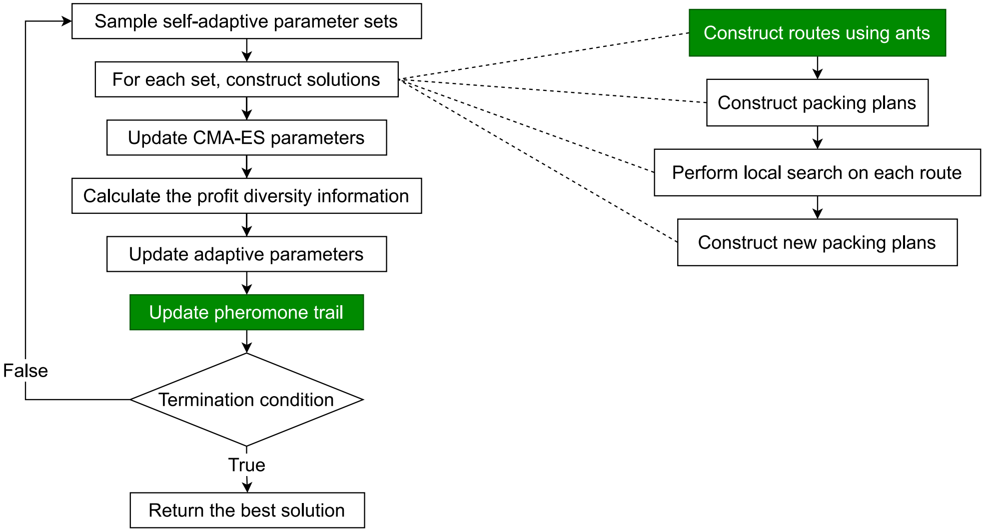
Key ideas

- We use hierarchical clustering to build the tree architecture.
- Each city has its own cluster tree that represents the edges going to n cities.
- Ants will traverse cluster trees instead of moving directly from one city to another.
- This way, we can reduce the time complexity of choosing the next city to $\Theta(\log n)$.



Hình: Cluster tree example.

Lazy evaporation

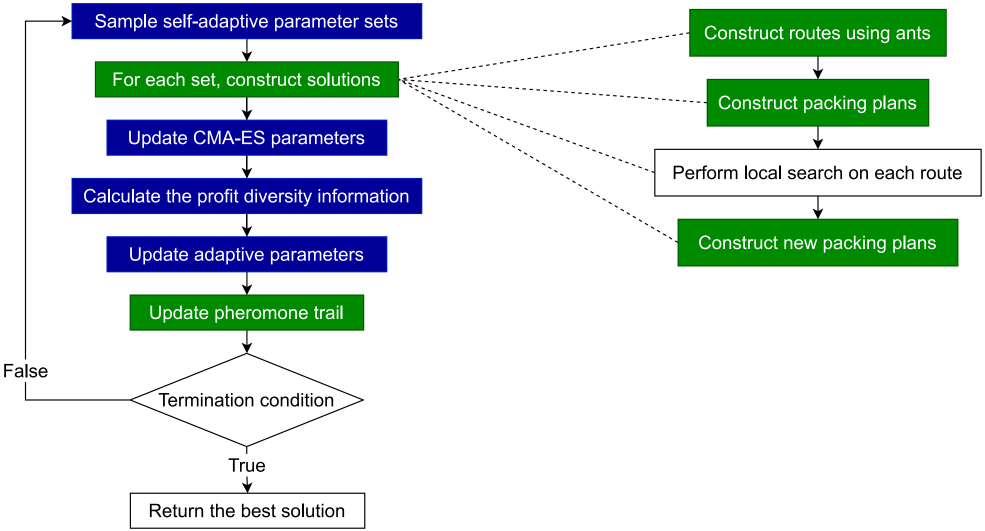


Lazy evaporation

Key ideas

- The key idea of lazy evaporation is to keep track of historical and desired states.
- The desired state consists of the number of times that pheromones need evaporating.
- Each edge has its historical state including the number of times that the pheromone of the edge has been evaporated.
- By comparing historical states and the desired state, we can determine how to calculate the desired pheromones of edges when needed.

The overall algorithm



Experiments

Hardware

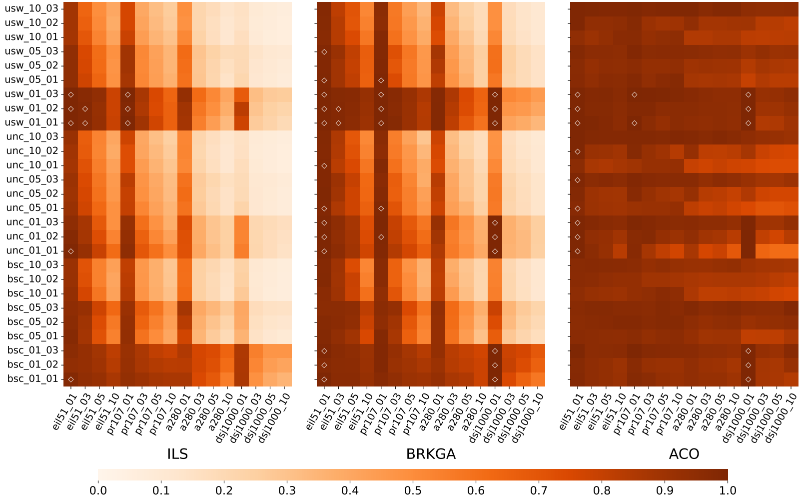
Experiments were run on the same machine with Intel(R) Core(TM) i7-8750H @ 2.20GHz for a fair comparison.

Hyperparameter tuning

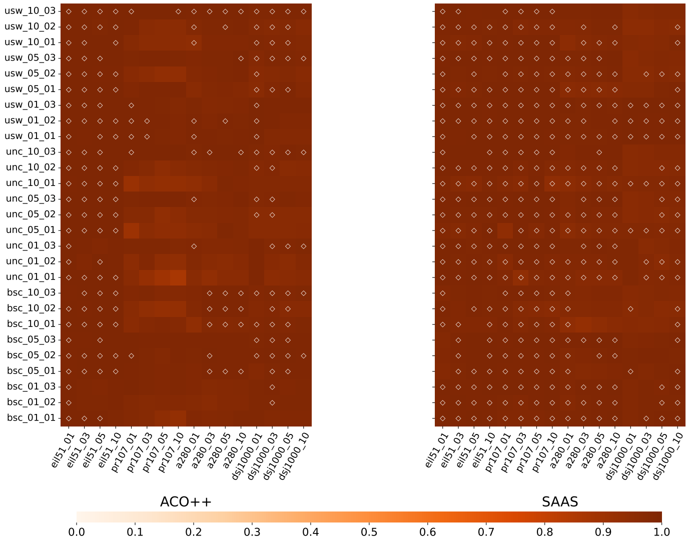
- BRKGA, ACO, and ACO++ used the same parameters fine-tuned in the ACO++ paper.
- 240,000 experiments were performed for tuning each algorithm. ILS had no parameters to fine-tune.
- evosax[7]framework was used to tune SAAS hyperparameters.
- 45,000 experiments were performed to tune SAAS for all benchmark instances.

[7] Robert Tjarko Lange. *evosax: JAX-based Evolution Strategies*. 2022. arXiv: 2212.04180 [cs.NE]

Approximation ratio of the solution approaches



Approximation ratio of the solution approaches

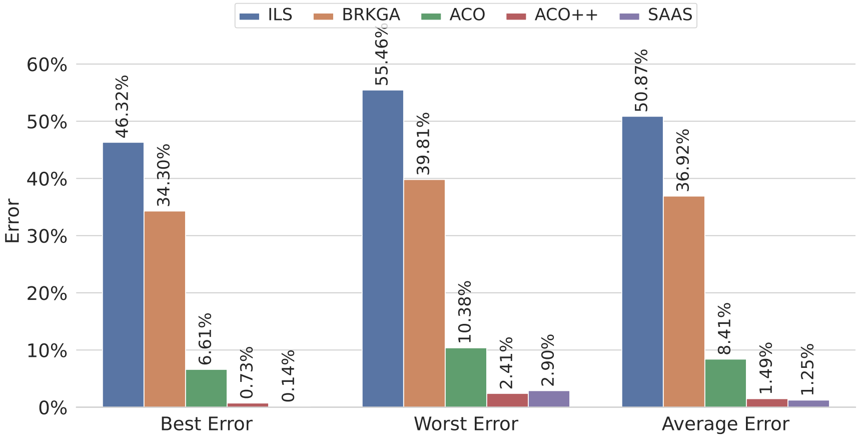


Comparing performance between algorithms

Bảng: Percentage of the number of instances in which algorithm i found better or equal quality solutions than algorithm j

i $\downarrow$ j $\rightarrow$	ILS	BRKGA	ACO	ACO++	SAAS
ILS	-	2.55%	4.40%	2.55%	2.31%
BRKGA	100.00%	-	16.20%	8.80%	7.18%
ACO	97.22%	87.27%	-	5.79%	4.86%
ACO++	99.54%	95.83%	97.69%	-	41.90%
SAAS	99.77%	97.92%	98.61%	78.24%	-

Error rate of SAAS solutions



Hình: Mean errors overall benchmark instances.

Conclusion

Conclusion

- Parameter control mechanisms are incorporated to improve adaptability and flexibility.
- Lazy evaporation technique is used to reduce the time complexity of the evaporation phase.
- Hierarchical clustering is used to improve the time complexity of finding routes.
- SAAS is more efficient than ACO++ and requires only one hyperparameter set to run all 432 benchmark instances.
- The SAAS algorithm showcases remarkable performance when it surpasses all other algorithms for ThOP.

References

- Santos, André G. et al. "The Thief Orienteering Problem: Formulation and Heuristic Approaches". In: *2018 IEEE Congress on Evolutionary Computation (CEC)*. 2018, pp. 1–9.
- Faêda, Leonardo M. et al. "A Genetic Algorithm for the Thief Orienteering Problem". In: *2020 IEEE Congress on Evolutionary Computation (CEC)*. 2020, pp. 1–8.
- Chagas, Jonatas B.C. et al. "Ants can orienteer a thief in their robbery". In: *Operations Research Letters* 48.6 (2020), pp. 708–714. ISSN: 0167-6377.
- — ."Efficiently solving the thief orienteering problem with a max–min ant colony optimization approach". In: *Optimization Letters* 16.8 (Nov. 2021), pp. 2313–2331.
- Hansen, Nikolaus. "The CMA Evolution Strategy: A Comparing Review". In: *Towards a New Evolutionary Computation: Advances in the Estimation of Distribution Algorithms*. Ed. by Jose A. Lozano et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 75–102. ISBN: 978-3-540-32494-2.
- Stodola, Petr et al. "Adaptive Ant Colony Optimization with node clustering applied to the Travelling Salesman Problem". In: *Swarm and Evolutionary Computation* 70 (2022), p. 101056. ISSN: 2210-6502.
- Lange, Robert Tjarko. *evosax: JAX-based Evolution Strategies*. 2022. arXiv: 2212.04180 [cs.NE].