

Swarm Intelligence

۱

برای استفاده از یک الگوریتم هوش جمعی مانند الگوریتم PSO (Particle Swarm Optimization) در یک مسئله‌ی جدید مثل مسائل مالی، لازم است برخی از اجزای الگوریتم با توجه به ویژگی‌های مسئله جدید تغییر داده شوند. در حالی که ساختار کلی الگوریتم حفظ می‌شود، موارد خاصی باید متناسب با کاربرد جدید، تنظیم یا بازطراحی شوند. مهم‌ترین بخش، **تابع هدف** است که باید بر اساس هدف اصلی مسئله‌ی جدید تعریف شود. مثلاً در مسئله تخصیص منابع، ممکن است هدف کمینه‌کردن هزینه باشد، در حالی که در مسائل مالی هدف بیشینه‌کردن سود یا کمینه‌کردن ریسک است. بنابراین تابع هدف باید بازنویسی شود تا بتواند کیفیت هر جواب (یا هر ذره در PSO) را در زمینه جدید به درستی ارزیابی کند. از طرف دیگر، **نحوه نمایش ذرات** نیز باید تغییر کند. هر ذره نماینده یک جواب ممکن است، و بسته به نوع مسئله، باید مشخص شود که این جواب شامل چه متغیرهایی است. مثلاً در یک مسئله مالی، یک ذره ممکن است یک بردار شامل درصد سرمایه‌گذاری در هر دارایی باشد. این نمایش باید با نوع داده‌ها و محدودیت‌های مسئله هماهنگ باشد. همچنین، **محدودیت‌های مسئله** باید در الگوریتم اعمال شوند. در مسائل مالی ممکن است محدودیت‌هایی مثل حداقل یا حداکثر سرمایه‌گذاری، یا محدودیت ریسک وجود داشته باشد. این محدودیت‌ها معمولاً از طریق توابع جریمه یا اعمال شرایط اصلاح روی موقعیت ذرات پیاده‌سازی می‌شوند. در نهایت، ممکن است نیاز باشد **پارامترهای الگوریتم** مانند اندازه جمعیت، ضرایب یادگیری، یا شرط توقف با توجه به پیچیدگی و دقت مورد نیاز در مسئله جدید تنظیم شوند. این تنظیمات به بهبود عملکرد الگوریتم در زمینه‌ی خاص کمک می‌کنند. در مجموع، استفاده از الگوریتم PSO در یک حوزه جدید نیاز به تغییراتی در تعریف تابع هدف، نمایش ذرات، اعمال محدودیت‌ها، و تنظیم پارامترها دارد تا الگوریتم بتواند با ویژگی‌های خاص مسئله جدید هماهنگ شود و نتایج مطلوبی ارائه دهد.

۲

بله، می‌توان از الگوریتم‌های هوش جمعی برای بهبود عملکرد مدل‌های یادگیری ماشین استفاده کرد. الگوریتم‌هایی مانند GA، PSO و ACO می‌توانند در بخش‌هایی از فرایند یادگیری ماشین نقش مهمی ایفا کنند، به‌ویژه در **تنظیم بهینه‌ی هایپرپارامترها**، **انتخاب ویژگی‌ها (Feature Selection)** و **آموزش مدل‌ها**. یکی از کاربردهای رایج هوش جمعی در یادگیری ماشین، **تنظیم هایپرپارامترها (Hyperparameter Tuning)** است. مدل‌هایی مانند SVM، Random Forest یا شبکه‌های عصبی دارای ابرپارامترهایی هستند که تأثیر زیادی بر دقت و عملکرد دارند. الگوریتم‌های هوش جمعی می‌توانند فضای جستجوی این پارامترها را به شکل هوشمند و کارا پیمایش کنند و بهترین ترکیب را پیدا کنند، به‌طوری که نیاز به آزمون و خطای دستی یا جستجوی تصادفی کاهش یابد. کاربرد دیگر، **انتخاب ویژگی‌های مؤثر** است. در مجموعه داده‌های بزرگ، همه‌ی ویژگی‌ها برای یادگیری مناسب نیستند و وجود ویژگی‌های بی‌ربط می‌تواند منجر به بیش‌برازش یا کاهش دقت شود. الگوریتم‌های هوش جمعی می‌توانند مجموعه‌ای از ویژگی‌های مؤثر را به عنوان جواب‌های ممکن در نظر بگیرند و بهترین ترکیب را با توجه به عملکرد مدل (مثلاً دقت یا F1-score) انتخاب کنند. همچنین در برخی موارد، **آموزش مستقیم مدل** هم با استفاده از الگوریتم‌های هوش جمعی انجام می‌شود، به‌ویژه در شرایطی که گرادین قابل محاسبه نیست یا تابع هدف پیچیده و غیرخطی است. در نتیجه، الگوریتم‌های هوش جمعی می‌توانند به شکل مؤثری در بهینه‌سازی اجزای مختلف مدل‌های یادگیری ماشین به کار گرفته شوند و موجب بهبود دقت، سرعت و عملکرد کلی آن‌ها شوند.

PSO

۱

۱. نقش هر یک از پارامترهای w, c1, c2 در PSO چیست؟

در الگوریتم PSO، حرکت هر ذره (particle) در فضای جستجو بر اساس معادله‌ای انجام می‌شود که شامل سه مؤلفه است:

- مقدار w (ضریب اینرسی):** این پارامتر نشان‌دهنده تمایل ذره به حفظ سرعت قبلی خود است. مقدار بزرگ‌تر w باعث می‌شود ذره‌ها به جستجوی گسترده‌تر (exploration) ادامه دهند و مقدار کوچک‌تر w باعث تمرکز بیشتر روی نواحی اطراف جواب‌های خوب (exploitation) می‌شود.
- مقدار c1 (ضریب شناختی یا یادگیری فردی):** این پارامتر میزان تأثیر تجربه شخصی هر ذره را در تعیین مسیر جدیدش مشخص می‌کند. هر ذره سعی می‌کند به سمت بهترین موقعیتی که تاکنون خودش تجربه کرده حرکت کند (personal best).
- مقدار c2 (ضریب اجتماعی یا یادگیری جمعی):** این پارامتر تأثیر بهترین تجربه کل جمعیت را مشخص می‌کند. ذره‌ها تمایل دارند از دیگران یاد بگیرند و به سمت بهترین جواب کل همسایه‌ها (neighbourhood best) حرکت کنند.

۲. مفاهیم Exploration و Exploitation در PSO به چه معنا هستند؟

- اکتشاف یا Exploration :** به توانایی الگوریتم در جستجوی نواحی مختلف فضای جواب اشاره دارد. هدف آن جلوگیری از گیر افتادن در نقاط بهینه محلی است. معمولاً w بالا یا مقدارهای تصادفی قوی به exploration کمک می‌کنند.

- **بهره برداری یا Exploitation** : به توانایی الگوریتم در تمرکز روی نواحی امیدوارکننده برای رسیدن به جواب بهتر اشاره دارد. این ویژگی باعث می‌شود که الگوریتم سریع‌تر به بهینه نزدیک شود. پارامترهای c1 و c2 بیشتر در جهت افزایش exploitation عمل می‌کنند.

الگوریتم PSO باید بین این دو ویژگی تعادل ایجاد کند تا هم فضای جستجو را خوب جست و جو کند (explore) و هم از نتایج خوب بهره‌برد (exploit).

۳. اگر $c1 = 0$ و $c2 = 0$ باشد چه اتفاقی می‌افتد؟

در این حالت، معادله بهروزرسانی سرعت فقط وابسته به مؤلفه اینرسی w خواهد بود و هیچ تأثیری از تجربیات شخصی یا جمعی وجود نخواهد داشت. در نتیجه:

- ذره‌ها فقط با سرعت اولیه‌ی خود و جهت اولیه حرکت می‌کنند.
- الگوریتم هیچ یادگیری‌ای ندارد.
- حرکت‌ها تصادفی نیستند ولی بدون هدف‌اند.
- الگوریتم PSO عملاً عملکرد خود را از دست می‌دهد و بهینه‌سازی انجام نمی‌شود.

۴. الگوریتم PSO چگونه می‌تواند از بهینه‌های محلی فرار کند؟ آیا همیشه موفق است؟

الگوریتم PSO می‌تواند از بهینه‌های محلی فرار کند از طریق:

- وجود مؤلفه تصادفی در معادله سرعت (rand در c1 و c2)
- وجود تنوع در جمعیت اولیه
- استفاده از مقدار مناسب w برای حفظ جستجوی گسترده‌تر

با این حال، **موفقیت همیشه تضمین‌شده نیست**. اگر تنوع جمعیت کم باشد یا الگوریتم زود همگرا شود (خصوصاً با مقدار w پایین یا c2 بالا)، ممکن است در یک بهینه محلی گیر بیفتد. نسخه‌های پیشرفته PSO (مثل adaptive PSO یا multi-swarm PSO) برای رفع این مشکل طراحی شده‌اند.

۵. رابطه بین اندازه جمعیت ذرات (swarm size) و دقت نهایی چیست؟

- اندازه بزرگ‌تر جمعیت معمولاً باعث افزایش دقت نهایی می‌شود چون فضای جستجو بهتر پوشش داده می‌شود و احتمال رسیدن به جواب بهینه بیشتر است.
- ولی افزایش بیش از حد تعداد ذره‌ها باعث افزایش زمان اجرا و پیچیدگی محاسباتی می‌شود.
- اندازه خیلی کوچک نیز ممکن است باعث همگرایی زودرس و گیر افتادن در بهینه‌های محلی شود.

در عمل، باید بین دقت و کارایی تعادل برقرار شود و اندازه جمعیت بهینه با آزمون و خطا یا روش‌های خودتنظیم انتخاب شود.

الف) محاسبه سرعت جدید برداری \vec{v}_{new}

برای محاسبه سرعت جدید در الگوریتم بهینه‌سازی ازدحام ذرات (PSO)، از فرمول زیر استفاده می‌کنیم:

$$\vec{v}_{new} = w \cdot \vec{v} + c_1 \cdot r_1 \cdot (\vec{p}_{best} - \vec{x}) + c_2 \cdot r_2 \cdot (\vec{g}_{best} - \vec{x})$$

حساب کردن هر جزء:

1. جزء اول $(w \cdot \vec{v})$:

$$0.6 \cdot [1, 0.5] = [0.6, 0.3]$$

2. جزء دوم $(c_1 \cdot r_1 \cdot (\vec{p}_{best} - \vec{x}))$:

$$\vec{p}_{best} - \vec{x} = [1, -2] - [2, -1] = [-1, -1]$$

$$2 \cdot 0.5 \cdot [-1, -1] = 1 \cdot [-1, -1] = [-1, -1]$$

3. جزء سوم $(c_2 \cdot r_2 \cdot (\vec{g}_{best} - \vec{x}))$:

$$\vec{g}_{best} - \vec{x} = [0, 0] - [2, -1] = [-2, 1]$$

$$2 \cdot 0.5 \cdot [-2, 1] = 1 \cdot [-2, 1] = [-2, 1]$$

جمع کردن تمامی اجزاء:

$$\vec{v}_{new} = [0.6, 0.3] + [-1, -1] + [-2, 1] = [0.6 - 1 - 2, 0.3 - 1 + 1] = [-2.4, 0.3]$$

ب) محاسبه موقعیت جدید \vec{x}_{new}

برای محاسبه موقعیت جدید از فرمول زیر استفاده می‌کنیم:

$$\vec{x}_{new} = \vec{x} + \vec{v}_{new}$$

$-\vec{x} = [2, -1]$

$-\vec{v}_{new} = [-2.4, 0.3]$

محاسبه نهایی:

$\vec{x}_{new} = [2, -1] + [-2.4, 0.3] = [2 - 2.4, -1 + 0.3] = [-0.4, -0.7]$

۳

$f(x) = -x^2 + 5x + 20$

1. پارامترهای مسئله:

- تابع هدف: $f(x) = -x^2 + 5x + 20$ (هدف: بیشینه‌سازی)
- تعداد ذرات: 5
- سرعت اولیه برای همه ذرات: صفر ($v = 0$)
- پارامترها:

- $w = 1$
- $c_1 = c_2 = 1$
- $r_1 = 0.2$
- $r_2 = 0.8$

2. مقداردهی اولیه:

ذره	x اولیه	$f(x)$	p_{best}
1	2	26	2
2	4	24	4
3	6	14	6
4	8	-4	8
5	10	-30	10

- بهترین مقدار سراسری اولیه (**global best**):
 $g_{\text{best}} = 2$ چون $f(2) = 26$ بیشترین مقدار است.

3. مرحله اول اجرای PSO:

فرمول‌ها:

$$v_{\text{new}} = w \cdot v + c_1 \cdot r_1 \cdot (p_{\text{best}} - x) + c_2 \cdot r_2 \cdot (g_{\text{best}} - x)$$
$$x_{\text{new}} = x + v_{\text{new}}$$

محاسبه برای هر ذره:

ذره	x قبلی	v قبلی	v_{new}	x_{new}	$f(x_{\text{new}})$	p_{best} جدید
1	2	0	0	2	26	2
2	4	0	-1.6	2.4	26.24	2.4
3	6	0	-3.2	2.8	26.16	2.8
4	8	0	-4.8	3.2	25.76	3.2
5	10	0	-6.4	3.6	25.04	3.6

- بهترین مقدار سراسری جدید:
 $g_{\text{best}} = 2.4$ چون $f(2.4) = 26.24$ بالاترین مقدار است.

4. مرحله دوم اجرای PSO:

اکنون با $g_{\text{best}} = 2.4$ ادامه می‌دهیم.

ذره	x قبلی	v قبلی	v_{new}	x_{new}	$f(x_{\text{new}})$	p_{best} جدید
1	2	0	0.32	2.32	26.2176	2.32
2	2.4	-1.6	-1.6	0.8	23.36	2.4
3	2.8	-3.2	-3.52	-0.72	16.904	2.8
4	3.2	-4.8	-5.44	-2.24	5.76	3.2
5	3.6	-6.4	-7.36	-3.76	-6.36	3.6

- بهترین مقدار سراسری نهایی:**

$g_{\text{best}} = 2.4$ چون هنوز $f(2.4) = 26.24$ بالاترین مقدار است.

5. نتیجه‌گیری:

- پس از دو مرحله اجرای PSO:

- بهترین موقعیت به‌دست‌آمده: $x = 2.4$

- مقدار تابع: $f(2.4) = 26.24$

- نقطه بهینه واقعی تابع:

$$x^* = \frac{-b}{2a} = \frac{-5}{2 \cdot (-1)} = 2.5$$

$$f(2.5) = -6.25 + 12.5 + 20 = 26.25$$

- بنابر این PSO توانسته به مقدار بهینه بسیار نزدیک شود. الگوریتم با استفاده از اطلاعات جمعی و تصادفی، ذرات را از موقعیت‌های اولیه دور به سمت ناحیه بهینه هدایت کرده است.

ACO

۱

(الف) در چه شرایطی ممکن است الگوریتم ACO به یک جواب غیر بهینه همگرا شود؟

الگوریتم ACO ممکن است در شرایطی به یک جواب غیر بهینه همگرا شود که اکثر مورچه‌ها روی یک مسیر نسبتاً خوب (اما نه بهینه) تمرکز کنند. این معمولاً زمانی رخ می‌دهد که:

- فرومون روی یک مسیر اولیه زیاد شود** و باعث جذب بیشتر مورچه‌ها به آن شود.
 - نرخ تبخیر فرومون (ρ) پایین** باشد و فرومون‌های مسیرهای ابتدایی مدت زیادی باقی بمانند.
 - پارامتر α (تأثیر فرومون) بالا** باشد و در نتیجه مورچه‌ها بیش از حد به فرومون توجه کرده و مسیرهای دیگر را کشف نکنند.
- در چنین حالتی، الگوریتم از کاوش مسیرهای دیگر بازمی‌ماند و روی یک جواب محلی همگرا می‌شود.

(ب) نقش نرخ تبخیر ρ و پارامترهای α و β در جلوگیری از این همگرایی چیست؟

- پارامتر ρ (نرخ تبخیر):** باعث حذف تدریجی فرومون‌ها می‌شود. اگر ρ به‌اندازه کافی بالا باشد، مسیرهای ضعیف فراموش می‌شوند و فضا برای کاوش مسیرهای بهتر باز می‌ماند. این مانع همگرایی زودهنگام می‌شود.
- پارامتر α (تأثیر فرومون):** تعیین می‌کند مورچه‌ها چقدر به فرومون‌ها توجه کنند. اگر α خیلی بالا باشد، الگوریتم بیشتر به مسیرهای قبلی متکی می‌شود (exploit) و تنوع کاهش می‌یابد. مقدار متعادل α باعث حفظ تعادل بین تجربه و کاوش می‌شود.
- مقدار β (تأثیر اطلاعات ابتکاری):** کنترل می‌کند مورچه‌ها چقدر به ویژگی‌های ذاتی مسیر (مثل فاصله یا هزینه کمتر) توجه کنند. مقدار مناسب β کمک می‌کند که حتی مسیرهایی با فرومون کم ولی کیفیت خوب هم انتخاب شوند و از همگرایی زودرس جلوگیری شود.

۲

در هر تکرار ، الگوریتم مراحل زیر را انجام می‌دهد:

1. ساخت مسیر توسط هر مورچه:

- هر مورچه باید یک مسیر شامل تمام n شهر را طی کند.
- برای انتخاب هر گره بعدی، مورچه باید بین $O(n)$ گزینه بررسی کند.
- پس ساخت یک مسیر برای یک مورچه هزینه‌ای برابر با $O(n^2)$ دارد.
- در کل، برای m مورچه:

2. به‌روزرسانی فرومون:

$$O(m \cdot n^2)$$

- معمولاً بین همه n^2 یال، فرومون‌ها بازنویسی یا بروزرسانی می‌شوند.
- هزینه این مرحله:

$$O(n^2)$$

در مجموع در هر تکرار داریم:

$$O(m \cdot n2 + n2) = O(m \cdot n2) = \text{زمان هر تکرار}$$

(چون $m \cdot n^2$ از n^2 بزرگتر است)

بنابراین، برای t تکرار کل پیچیدگی زمانی می‌شود:

$O(t \cdot m \cdot n^2)$

۳

(الف)

فرمول احتمال انتخاب یال (از i به j):

$$P_{ij} = \frac{(\tau_{ij})^\alpha \cdot (\eta_{ij})^\beta}{\sum_{k \in N_i} (\tau_{ik})^\alpha \cdot (\eta_{ik})^\beta}$$

که:

- متغیر τ_{ij} مقدار فرومون روی یال $j \rightarrow i$ است.
- متغیر $\eta_{ij} = \frac{1}{d_{ij}}$ تابع heuristic است.
- متغیر N_i مجموعه نودهای مجاور از گره i است.
- در اینجا: $\alpha = 1$ و $\beta = 2$

داده‌های مربوط به گره **A**:

- مسیرهای ممکن از **A**:

- یال **A** \rightarrow **B** با فاصله 2 $\implies \eta_{AB} = 1/2$
- یال **A** \rightarrow **C** با فاصله 3 $\implies \eta_{AC} = 1/3$

- $\tau_{AB} = \tau_{AC} = 1$

محاسبه احتمال‌ها:

صورت کسر‌ها:

- برای $A \rightarrow B$: $1^1 \cdot (1/2)^2 = 1 \cdot 1/4 = 0.25$
- برای $A \rightarrow C$: $1^1 \cdot (1/3)^2 = 1 \cdot 1/9 \approx 0.111$

مخرج کسر (مجموع):

$$0.25 + 0.111 \approx 0.361 \quad 0.25 + 0.111 \approx 0.361$$

نهایتاً:

$$P_{AB} = \frac{0.25}{0.361} \approx 0.692$$
$$P_{AC} = \frac{0.111}{0.361} \approx 0.308$$

(ب)

- فرمول افزایش فرومون روی یال‌های طی شده، اگر یال (i, j) در مسیر مورچه k باشد:
- $$\Delta \tau_{ij} = \frac{Q}{L_k}$$

گام 1: طول مسیرها

- مورچه اول: $L_1 = 2 + 2 + 3 = 7 \rightarrow A \rightarrow B \rightarrow C \rightarrow D$
- مورچه دوم: $L_2 = 3 + 3 = 6 \rightarrow A \rightarrow C \rightarrow D$

گام 2: افزایش فرومون

- مورچه اول $(A \rightarrow B, B \rightarrow C, C \rightarrow D)$: $\Delta \tau = \frac{10}{7} \approx 1.429$

- مورچه دوم ($\mathbf{A \rightarrow C}$ ، $\mathbf{C \rightarrow D}$): $\Delta\tau = \frac{10}{6} \approx 1.667$

گام 3: محاسبه فرومون جدید روی یال‌ها

همه فرومون اولیه 1 هستند، پس بعد از تبخیر داریم: $0.8 \cdot 1 = 0.8$

یال	در مسیر؟	تبخیر شده	$\Delta\tau$	τ_{new}
$A \rightarrow B$	مورچه اول	0.8	+1.429	2.229
$B \rightarrow C$	مورچه اول	0.8	+1.429	2.229
$C \rightarrow D$	هر دو مورچه	0.8	$1.429 + 1.667$	3.896
$A \rightarrow C$	مورچه دوم	0.8	+1.667	2.467
$B \rightarrow D$	استفاده نشده	0.8	0	0.8

نکته: فرض کردیم یال‌ها جهت‌دار نیستند.

پاسخ نهایی:

یال	فرومون جدید τ_{ij}
$A \rightarrow B$	2.229
$B \rightarrow C$	2.229
$C \rightarrow D$	3.896
$A \rightarrow C$	2.467
$B \rightarrow D$	0.8

۴

الف) تعداد تمام مسیرهای ممکن در این مسئله را بدست آورید.

در مسئله TSP با 4 گره (A, B, C, D) و فرض بازگشت به گره اولیه (مثلاً A)، تعداد مسیرهای Hamiltonian که از یک گره شروع شده و به همان گره برمی‌گردند برابر است با:

$$(n - 1)! = (4 - 1)! = 3! = 6(n - 1)! = (4 - 1)! = 3! = 6$$

پس ۶ مسیر ممکن داریم.

ب) کدام مسیر بیشتر احتمال دارد تا توسط مورچه‌ها یافت شود؟

در الگوریتم ACO، مسیرهایی که مجموع طولشان کمتر باشد، معمولاً توسط مورچه‌ها بیشتر انتخاب می‌شوند، زیرا تابع heuristic برای هر یال به صورت $\eta_{ij} = \frac{1}{d_{ij}}$ تعریف شده و فاصله کمتر باعث افزایش احتمال انتخاب یال می‌شود.

براساس داده‌ها، پس از محاسبه طول مسیرها:

1. A-B-C-D-A = 14
2. A-B-D-C-A = **12**
3. A-C-B-D-A = 14
4. A-C-D-B-A = **12**
5. A-D-B-C-A = 14
6. A-D-C-B-A = 14

تنها مسیرهای شماره **2** و **4** طول برابر با 12 دارند که کوتاهترین مسیرها هستند.

حال باید بررسی کنیم که در مراحل اولیه الگوریتم، مورچه‌ها بیشتر به کدام مسیر گرایش دارند. چون در شروع مقدار فرومون روی همه یال‌ها برابر است، عامل تعیین‌کننده فقط مقدار heuristic یا همان $1/d_{ij}$ خواهد بود.

از گره A:

- $d(A, B) = 2 \implies \eta = 0.5$
- $d(A, C) = 3 \implies \eta = 0.333$
- $d(A, D) = 5 \implies \eta = 0.2$

پس بیشترین احتمال انتخاب اولیه مربوط به $\mathbf{A \rightarrow B}$ است.

در نتیجه، مسیر $\mathbf{A \rightarrow C \rightarrow D \rightarrow B \rightarrow A}$ هم از نظر طول بهینه است، هم از نظر احتمال انتخاب در گام اول بیشتر است، چون مورچه‌ها با احتمال بالاتری مسیر $\mathbf{A \rightarrow B}$ را انتخاب می‌کنند.

بنابراین:

مسیر $\mathbf{A \rightarrow C \rightarrow D \rightarrow B \rightarrow A}$ بیشترین احتمال را دارد که توسط مورچه‌ها کشف و تقویت شود.

ج) تفاوت استفاده از یک مورچه برای حل این مسئله با حالتی که از چندین مورچه استفاده شود را به‌طور کامل شرح دهید.

- **با یک مورچه:**

- تنها یک مسیر در هر تکرار ساخته می‌شود.
- اکتشاف (exploration) فضای جستجو بسیار محدود است.
- خطر بالای گیر افتادن در جواب‌های غیر بهینه وجود دارد.
- سرعت همگرایی پایین است چون فرومون فقط روی یک مسیر به‌روزرسانی می‌شود.

- **با چند مورچه:**

- چندین مسیر به‌صورت موازی در هر تکرار ساخته می‌شوند.
- پوشش بهتری از فضای جستجو انجام می‌شود.
- احتمال پیدا کردن مسیر بهینه بیشتر است.
- با ترکیب تبخیر فرومون و همزمانی، مسیرهای ضعیف حذف می‌شوند و مسیرهای خوب تقویت می‌شوند.
- همگرایی سریع‌تر و دقیق‌تر به سمت مسیر بهینه اتفاق می‌افتد.

در نتیجه، استفاده از چندین مورچه باعث افزایش کیفیت، سرعت و پایداری الگوریتم **ACO** می‌شود.