

الگوریتم بهینه‌سازی اجتماع مورچگان بر مبنای گرادیان برای فضاهای پیوسته

مهدی افتخاری^{*}، بیژن داعی^{**} و سراج الدین کاتبی^{***}

بخش مهندسی و علوم کامپیوتر، دانشکده مهندسی دانشگاه شیراز

(دریافت مقاله: ۸۳/۹/۱ - دریافت نسخه نهایی: ۸۴/۱۲/۲۸)

چکیده - در این تحقیق یک نسخه جدید از الگوریتم بهینه‌سازی اجتماع مورچه‌ها^۱ که توانایی جستجو در فضای پیوسته^۲ را دارد، ارائه می‌شود. ساختار و مفاهیم اصلی الگوریتم اولیه بهینه‌سازی اجتماع مورچه حفظ شده و تعمیم و توسعه آن به فضای پیوسته انجام و پیاده‌سازی شده است. خاصیت ارتباط غیرمستقیم از طریق محیط (استیگرژی^۳) با تعدادی بردار گرادیان نرمال شده شبیه‌سازی شد. برای اینکه همه مورچه‌ها بتوانند محیط را حس کنند، این بردارها توسط یک حافظه مشترک نگهداری می‌شوند. الگوریتم بهینه‌سازی پیشنهادی، بر روی توابع خاصی که به عنوان محک^۴ در مسائل بهینه‌سازی فضای پیوسته به کار می‌روند، امتحان شده است. نتایج به دست آمده از این الگوریتم با نتایج الگوریتمهای تکاملی مانند الگوریتم ژنتیکی^۵، استراتژی تکاملی^۶ و برنامه‌نویسی تکاملی^۷ مقایسه شده و از لحاظ دقت و حجم محاسبات مورد نیاز نتایج حاصل از الگوریتم پیشنهادی به خوبی با الگوریتمهای دیگر قابل رقابت و در بعضی موارد بهتر است.

واژگان کلیدی: اجتماع مورچه، تکاملی، الگوریتمها، فوق اکتشافی، اتفاقی، محدودیت

Gradient-based Ant Colony Optimization for Continuous Spaces

M. Eftekhari, B. Daei, and S. D. Katebi

Department of Computer Science and Engineering, School of Engineering, Shiraz University

Abstract: A novel version of Ant Colony Optimization (ACO) algorithms for solving continuous space problems is presented in this paper. The basic structure and concepts of the originally reported ACO are preserved and adaptation of the algorithm to the case of continuous space is implemented within the general framework. The stigmergic communication is simulated through considering certain direction vectors which are memorized. These vectors are normalized gradient vectors that are calculated using the values of the evaluation function and the corresponding values of object variables.

The proposed Gradient-based Continuous Ant Colony Optimization (GCACO) method is applied to several benchmark problems

*** - استاد

** - دانشجوی کارشناسی ارشد

* - دانشجوی دکترا

and the results are compared and contrasted with other population-based algorithms such as Evolutionary Strategies (ES), Evolutionary Programming (EP), and Genetic Algorithms (GA). The results obtained from GCACO compare satisfactorily with those of other algorithms and in some cases are superior in terms of accuracy and computational demand.

Keywords: Ant Colony, Evolutionary, Algorithms, Meta-heuristic, Stochastic, Constraint.

۱- مقدمه

را کامل کردند (هر کدام جواب خود را به دست آوردند)، هر مورچه مقداری فرومون توسط متغیرهای مربوط بر روی یالهای ملاقات شده قرار می‌دهد تا این یالها توسط مورچه‌های بعدی بیشتر مورد توجه قرار گیرند، سپس مورچه‌ها می‌میرند. در کنار اضافه شدن فرومونها، مقداری از فرومونها تبخیر می‌شود که نقش اساسی در جلوگیری از رکود^{۱۸} دارد. (رکود یا درجا زدن زمانی اتفاق می‌افتد که همه مورچه‌ها یک جواب به دست بیاورند).

اما حافظه یا حالت داخلی^{۱۹} مورچه شامل نقاط ملاقات شده تا کنون است و لیست ممنوع^{۲۰} نامیده می‌شود. این حافظه برای این به کار می‌رود که مورچه که k ام با استفاده از آن بتواند بفهمد که از نقطه فعلی که در آن قرار دارد باید به کجا حرکت کند و چه نقطه‌هایی را مجبور است برای تکمیل تور خود طی کند.

در این الگوریتم یک جدول تصمیم^{۲۱} یا جدول مسیریابی مورچه^{۲۲} وجود دارد که مقادیر موجود در آن مبنای محاسبه احتمالات برای انتخاب جهت‌های گوناگون است. این جدول با نماد A نمایش داده می‌شود.

$$A = [a_{ij}(t)] \quad a_{ij}(t) = \frac{[\tau_{ij}(t)]^\alpha [\eta_{ij}]^\beta}{\sum_{l \in N_i} [\tau_{il}(t)]^\alpha [\eta_{il}]^\beta} \quad \forall j \in N_i \quad (1)$$

در معادله (۱) پارامتر اکتشاف^{۲۳} است (در مورد مسئله فروشنده دوره گرد، $\eta_{ij} = 1/d_{ij}$ ، به معنی این است که مسافت از نقطه i به j چقدر خوب است، جایی که d_{ij} فاصله بین نقاط i و j بوده و N_i مجموعه همسایه‌های گره i ام است.) و τ_{ij} مقدار فرومون بین دو گره i و j را نشان می‌دهد. پارامترهای $0 \leq \alpha \leq 1$, $0 \leq \beta \leq 1$ توانهایی هستند که توسط آنها می‌توان میزان تأکید بر فرومون و پارامتر اکتشاف را تنظیم

الگوریتمهای بهینه‌سازی اجتماع مورچگان جزو دسته الگوریتمهای جستجوی فوق اکتشافی^۱ اند که از رفتارهای اجتماعی مورچه‌ها و موریه‌های واقعی الهام گرفته شده‌اند. کاربرد متداول آنها در مسائل بهینه‌سازی با فضای گسسته (مانند مسئله معروف فروشنده دوره گرد^۲) است. اجتماع واقعی مورچه‌ها را می‌توان به صورت یک سیستم توزیع شده بزرگ دید که می‌تواند مسائل را (از قبیل پیدا کردن منبع غذا) با استفاده از خاصیتی به نام استیگمرجی حل کند. تحریر کارگرها به کار بیشتر به علت کارایی که به دست می‌آورند، اولین بار توسط گراسه^{۱۰} به عنوان استیگمرجی تعریف شد [۱]. این تعریف نوعی ارتباط غیرمستقیم بواسطه تغییرات محیط را بیان می‌کند [۲]. گراسه این ارتباط را در دو نوع خاص از موریه‌ها به نامهای بلیکزیترمز ناتالانسیس^{۱۱} و کویترمز^{۱۲} مشاهده کرد. خاصیت نشست فرومون^{۱۳} قسمتی از فرایند سربازگیری^{۱۴} است. فرایند سربازگیری توسط زیست‌شناسان نوعی سازماندهی خاص تعریف می‌شود، که در آن اعضای یک جامعه به سمت یک نقطه از فضای کاری جهت‌گیری می‌کنند [۳].

سیستم مورچه^{۱۵} اولین الگوریتم بهینه‌سازی اجتماع مورچه بود که در سال ۱۹۹۱ [۵و۴] توسط دوریگو^{۱۶} پیشنهاد شد. اولین بار سیستم مورچه توسط مسئله معروف فروشنده دوره گرد که مسئله بهینه‌سازی ترکیبی است، مورد آزمایش قرار گرفت. این الگوریتم حداکثر به تعداد T تکرار اجرا می‌شود. در هر تکرار m عدد مورچه یک تور^{۱۷} که شامل n مرحله است را با استفاده از یک قانون تصمیم‌گیری مبتنی بر احتمال به دست می‌آورند. در الگوریتم سیستم مورچه بعد از اینکه مورچه‌ها تور خود

سیستم مورچه را برطرف کرده و کارایی مطلوب را داشته باشد. از این مرحله به بعد هر جا عبارت الگوریتم بهینه‌سازی اجتماع مورچه استفاده شده است، منظور ساختار کلی الگوریتم سیستم اجتماع مورچه است.

در زمینه تعمیم الگوریتمهای بهینه‌سازی اجتماع مورچه‌ها به فضای پیوسته تحقیقات اندکی صورت گرفته است که اولین آنها توسط بیلچف^{۳۰} [۱۳] در سال ۱۹۹۵ انجام شد. بقیه روشهایی که پیشنهاد شده است، به ترتیب در [۱۶-۱۴] آورده شده است. در اکثر این روشها جستجو در دو مرحله سراسری و محلی انجام می‌شود، که مرحله سراسری آن بیشتر به الگوریتمهای ژنتیک شبیه است. در این مقاله، الگوریتم پیشنهادی بر مبنای چهارچوب کلی الگوریتم سیستم اجتماع مورچه است که به فضای پیوسته تعمیم داده شده است.

۲- روش پیشنهادی، تعمیم الگوریتم بهینه‌سازی اجتماع مورچه به فضای پیوسته

در این بخش ما به توضیح الگوریتم پیشنهادی می‌پردازیم. در شکل (۱) جریانی الگوریتم و شبه کد بهینه‌سازی اجتماع مورچه پیشنهادی که به اختصار GCACO نامیده می‌شود، آورده شده است. سپس موارد تعمیم و اصلاح الگوریتم برای فضای پیوسته شرح داده می‌شود.

اولین تفاوت این است که در الگوریتم پیشنهادی برای هر اجتماع یک شعاع حرکت $0 \leq R_c \leq 1$ در نظر گرفته شده است (برای اولین اجتماع مقدار شعاع ۱ در نظر گرفته می‌شود). سپس مقدار شعاع توسط یک تابع نمایی به شکل زیر کاهش پیدا می‌کند.

$$R_c = R_c \cdot e^{\left(-\frac{t}{T_{\max}}\right)} \quad (4)$$

T_{\max} تعداد تکرارها، t تکرار فعلی است.

تفاوت دیگر GCACO با الگوریتم بهینه‌سازی اجتماع مورچه معمولی این است که بهترین نقطه به دست آمده توسط اجتماع فعلی به عنوان لانه^{۳۱} بعدی در نظر گرفته می‌شود.

کرد. حال در تکرار t احتمال این که مورچه k ام نقطه j ام را برای رفتن انتخاب کند، در حالی که در نقطه i ام است، به صورت زیر است:

$$P_{ij}^k(t) = \frac{a_{ij}(t)}{\sum_{l \in N_i^k} a_{il}(t)} \quad (2)$$

N_i^k در معادله (۲) زیر مجموعه ای از N_i است و شامل نقاطی است که توسط مورچه k ام هنوز ملاقات نشده‌اند. (این نقاط با استفاده از فهرست ممنوع قابل تشخیص‌اند).

اضافه شدن مقدار فرومون و همچنین تبخیر فرومون با معادله زیر در هر تکرار صورت می‌پذیرد.

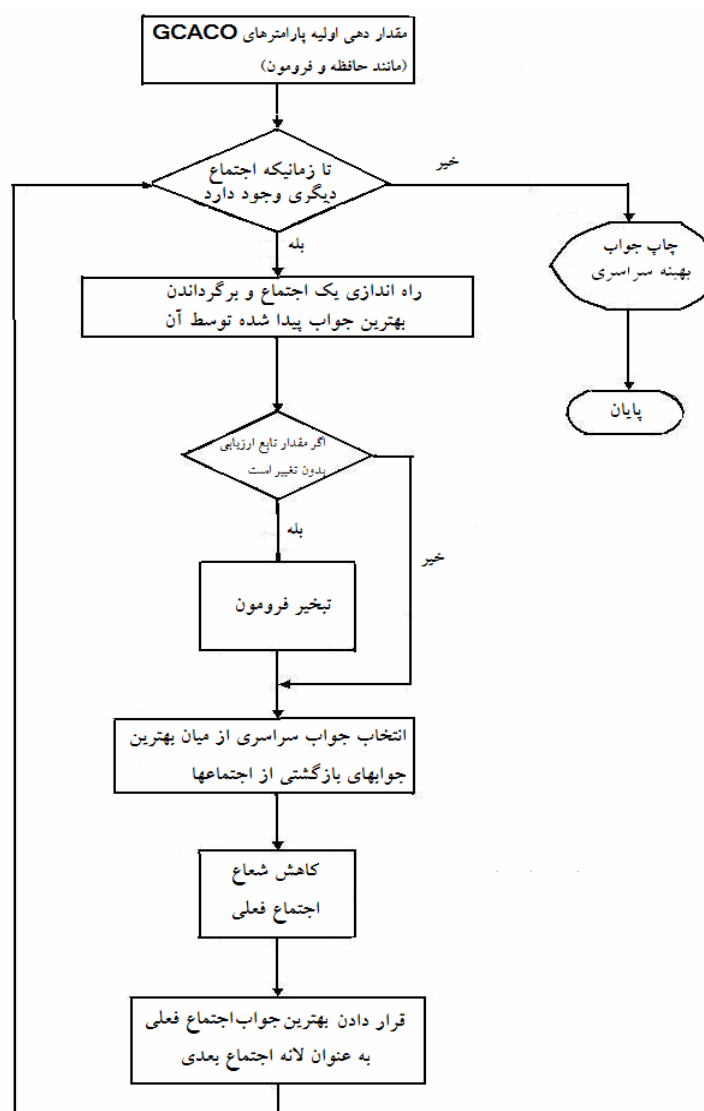
$$\begin{cases} \Delta\tau_{ij}(t) = \sum_{k=1}^m \Delta\tau_{ij}^k(t) \\ \tau_{ij}(t) = (1-\rho)\tau_{ij}(t) + \Delta\tau_{ij}(t) \end{cases} \quad (3)$$

در معادله بالا ρ پارامتر کاهش فرومون (تبخیر) است

و $\Delta\tau_{ij}^k(t)$ مقدار فرومونی است که توسط مورچه k ام روی یال (i,j) نشست کرده است و $\Delta\tau_{ij}(t)$ مقدار کل فرومون نشست روی یال (i,j) ام است.

الگوریتمهای سیستم مورچه با مسایل فروشنده دوره گرد با اندازه کوچک (در حدود ۳۰ تا ۷۵ شهر)، با سایر روشهای اکتشافی مقایسه شده است [۶]. طبق گزارش دوریگو [۶-۸] نتایج به دست آمده از طرفی خیلی جالب و از طرف دیگر خیلی ناامید کننده است. الگوریتم سیستم مورچه قادر بود که بهترین نتایج به دست آمده توسط الگوریتمهای ژنتیک برای مسئله اولیور^{۲۴}-۳۰ را بهبود بخشد و کارایی شبیه یا بهتر از الگوریتمهای اکتشافی داشته باشد. از طرفی برای مسایل با ابعاد بالا سیستم مورچه هرگز به بهترین جوابها نرسید [۸].

الگوریتم سیستم اجتماع مورچه^{۲۵} توسط دوریگو و گامباردلا^{۲۶} [۹-۱۲] برای بهبود کارایی سیستم مورچه پیشنهاد شد. الگوریتم سیستم اجتماع مورچه با سیستم مورچه سه تفاوت اساسی دارند که در [۶] به تفصیل توضیح داده می‌شود. این الگوریتم، بهترین الگوریتم بهینه‌سازی اجتماع مورچه تاکنون است، که توانسته است نقایص مربوط به الگوریتمهای



شکل ۱- نمودار جریان‌ی مربوط به الگوریتم پیشنهادی

این خاصیت الگوریتم آن را به روشهای حریصانه^{۳۲} جستجو مانند تپه نوردی^{۳۳} نزدیک ساخته است. تبخیر فرومون که در خط ۶ از شکل (۲) آمده است، توسط معادله زیر انجام می‌شود:

$$\tau^{\text{new}} = \text{evap_factor} \times \tau^{\text{old}} \quad (5)$$

عمل تبخیر فرومون زمانی اتفاق می‌افتد که تابع ارزیابی^{۳۴} تغییری پیدا نمی‌کند. (این به معنی آن است که الگوریتم احتمالاً دچار رکود شده است و فرومونهای قبلی باید به فراموشی سپرده شوند.) با توجه به شکل (۲)، هر مورچه توسط تابع

مقدار دمی اولیه پارامترهای GCACO (مانند حافظه و فرومون) تا زمانیکه اجتماع دیگری وجود دارد

بله

راه اندازی یک اجتماع و برگرداندن بهترین جواب پیدا شده توسط آن

اگر مقدار تابع ارزیابی بدون تغییر است

بله

تبخیر فرومون

خیر

انتخاب جواب سراسری از میان بهترین جوابهای بازگشتی از اجتماعها

کاهش شعاع اجتماع فعلی

قرار دادن بهترین جواب اجتماع فعلی به عنوان لانه اجتماع بعدی

چاپ جواب سراسری

پایان

```

1 Procedure GCACO_Meta_heuristic()
2 Initialize_the_parameters_of_GCACO ();
3 While ( t < Tmax )
4 [Best_Eval_Function, Best_solution] = ants_generation_and_activity ();
5 if No_change_in_Eval_Function
6 Evaporate_Pheromone();
7 endif
8 select_global_solution_among_the_Best_ones();
9 Reduce_the_Radius_of_colony_Movement();
10 Set_the_best_solution_of_this_colony_as_the_Nest_of_Next_colony;
11 t=t+1;
12 endwhile
13 endprocedure

14 Procedure ants_generation_and_activity()
15 Initialize_the_Radius_of_colony_by_Rc;
16 While Ant_count < Total_Ants
17 [best_fitness, best_location, routing_table, memory, τ]=new_active_ant();
18 select_global_location_among_the_best_ones;
19 set_returned_τ_memory_routing-table_for_the_next_ant_movement();
20 Ant_count= Ant_count+1;
21 endwhile
22 endProcedure

23 procedure new_active_ant();
24 Initialize Ra = Rc, Nest = Nest_of_colony, τ0 = 2 , φ = 0.05 ;
25 While (Active_ant )
26 do
27 q = Uniform_rand(0,1);
28 if q ≤ q0
29 xnew = xold + Ra × U (-1,1)
30 else
31 best_index = compute_best_direction();
32 xnew = xold + Ra × memory (best_index )
33 endif-else
34 while( not_feasible )
35 if ( memory_is_used )
36 τbest_indexnew = (1-φ) × τbest_indexold + φ × τ0
37 if ( ηbest_index < ηcurrent and fcurrent < fprevious )
38 ηbest_index = |fcurrent - fprevious |
39 endif
40 else
41 ηcurrent = |fcurrent - fprevious |
42 if ( ∇i ηi < ηcurrent and fcurrent < fprevious )
43 NGV = calculate_the_gradient_and_normalize_it;
44 Bad_index = compute_worst_direction();
45 Memory(Bad_index) = NGV;
46 τBad_indexnew = (1-φ) × τBad_indexold + φ × τ0 ;

```

```

47  $\eta_{Bad\_index} = |f_{current} - f_{previous}|;$ 
48 endif
49 endelseif
50 Update_ant_routing_Table();
51  $f_{previous} = f_{current}$ 
52  $location_{previous} = location_{current}$ 
53 if ( Ra < 0 )
54 Active_ant = false;
55 endif
56 Ra = Ra * DF;
57 endwhile
58 if (online_delayed_pheromone_update)
59 Bh = Find_direction_with_best_heuristic_value;
60  $\tau_{Bh}^{new} = (1 - \varphi) \times \tau_{Bh}^{old} + \varphi \times 4;$ 
61 Update_ant_routing_Table();
62 end if
63 endProcedure

```

شکل ۲- شبیه کد الگوریتم بهینه‌سازی اجتماع مورچگان برای فضای پیوسته GCACO

حریصانه یا انتخاب چرخشی مانند وسیله قمارچرخان^{۳۵} برای انتخاب بهترین جهت می‌تواند استفاده شود.

$$a_i = \frac{\tau_i \times \eta_i^\beta}{\sum_{j=1}^{\text{Directions}} \tau_j \times \eta_j^\beta} \quad \forall i \in \{\text{Directions}\} \text{ where } \beta=0.4 \quad (6)$$

$$P_i = \frac{a_i}{\sum_{j=1}^{\text{Directions}} a_j} \quad (7)$$

حرکت بر مبنای حافظه توسط معادله (۸) مدل می‌شود. جهت‌های موجود در حافظه بردارهای نرمال شده گرادیان‌اند که در مراحل قبلی الگوریتم باعث بهبود در تابع ارزیابی شده‌اند. Best_index اندیس بهترین جهت از جهت‌های حافظه است که با توجه به احتمال جهت‌ها و یک روش حریصانه یا چرخشی انتخاب می‌شود.

$$\overline{x_{new}} = \overline{x_{old}} + Ra \cdot \text{memory}(\text{best_index}) \quad (8)$$

بعد از حرکت هر مورچه به مکان جدید، ممکن بودن^{۳۶} آن مکان برای اینکه از قیود تعریف شده تجاوز نکند بررسی می‌شود.

در الگوریتم new_active_ant() تا زمانی که یک مورچه فعال است، حرکت را بر اساس یک قانون تصمیم‌گیری تصادفی که با الگوریتم بهینه‌سازی اجتماع مورچه معمولی فرق می‌کند، ادامه می‌دهد. حرکت مورچه‌ها در فضای جستجو کاملاً تصادفی یا بر اساس بردارهای گرادیان ذخیره شده در حافظه است. ابتدا یک عدد تصادفی بین [۰، ۱] تولید می‌شود. اگر این عدد از q کوچکتر بود، مقدار مکان جدید به صورت $\overline{x_{new}} = \overline{x_{old}} + Ra \cdot U(-1,1)$ به دست می‌آید. این رابطه حرکت کاملاً تصادفی مورچه را شبیه‌سازی می‌کند. به این ترتیب که مورچه حرکت خود را با شعاع Ra و در امتداد یک بردار تصادفی که عناصر آن با استفاده از یک توزیع تصادفی یکنواخت بین [-۱، ۱] به دست می‌آیند، تغییر می‌دهد. در غیر این صورت حرکت مورچه بر اساس حافظه است. حافظه از تعداد محدودی جهت تشکیل شده است و در صورت لزوم، بهترین جهت بر اساس احتمال منسوب شده به هر جهت انتخاب می‌شود و حرکت بعدی مورچه بر اساس این جهت است. احتمال جهت i ام از معادله (۷) و بر اساس مقادیر موجود در جدول مسیریابی، معادله (۶)، به دست می‌آید. یک مکانیزم

جدول ۱ - جزئیات توابع محک

شناسه	نام تابع	فرمول ریاضی	بعد	محدوده جستجو	بهبود سراسری
T1	Schaffer F6	$\min f(x,y) = 0.5 + \frac{(\sin^2(\sqrt{x^2+y^2}) - 0.5)}{((1+0.001 \times (x^2+y^2))^2)}$	2**	[-100,100]	0
T2	De Jong's F2	$\min f(x,y) = 100(x^2 - y)^2 + (1-x)^2$	2	[-2.084,2.084]	0
T3	Ackey F1	$\min f(x) = 20 + e - 20 \exp(-0.2 \sqrt{0.2 \sum_{i=1}^5 x_i^2}) - \exp(0.2 \sum_{i=1}^5 \cos(2\pi x_i))$	5	[-8,8]	0
T4	Rastrigin F1	$\min f(x) = 100 + \sum_{i=1}^{10} (x_i^2 - 10 \cos(2\pi x_i))$	10	[-8,3]	0
T5	Rosenbrock	$\min R_n(X) = \sum_{i=1}^{n-1} [100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2]$	5,10	[-5,10]	0

جدول مسیریابی مورچه بر اساس مقادیر جدید فرومون و پارامتر مکاشفه‌ای به روزرسانی می‌شوند.

در صورتی که شعاع حرکت مورچه که هر بار کاهش پیدا می‌کند، از یک حد کمتر شد، مورچه از حرکت باز می‌ایستد یا به اصطلاح می‌میرد. در انتهای کار به روزرسانی برخط و با تاخیر فرومون^{۳۷} انجام می‌گیرد، به این طریق که مقدار فرومون مربوط به جهتی را که مقدار پارامتر مکاشفه‌ای (η) مربوط به آن ماکزیمم است، به مقدار بیشتری افزایش می‌دهیم.

۳- نتایج شبیه سازی

نتایج حاصل از الگوریتم پیشنهادی با نتایج حاصل از برخی الگوریتمهای تکاملی [۱۷] که مبتنی بر جمعیت جواب‌اند، مانند الگوریتم استراتژی تکاملی و برنامه‌نویسی تکاملی مقایسه شده است. در این مقایسه از پنج تابع معروف محک استفاده شده و خصوصیات آنها در جدول (۱) آورده شده است. این توابع محک معمولاً برای آزمایش کارایی الگوریتمهای تکاملی استفاده می‌شوند [۱۷ و ۱۸]. همچنین معیار توقف الگوریتمهای تکاملی ذکر شده، کوچکتر شدن مقدار تابع از یک مقدار بسیار کوچک از قبل تعریف شده و در صورت ارضا نشدن این معیار، اتمام آن

اگر از حافظه استفاده شده باشد، مقدار فرومون جهت استفاده شده در حافظه بر اساس معادله (۹) به روزرسانی می‌شود، سپس پارامتر اکتشاف مربوط به آن جهت (در صورتی که مقدار تابع ارزیابی کاهش پیدا کرده باشد یعنی بهتر شده باشد)، بر اساس معادله (۱۰) به روزرسانی می‌شود.

$$\tau_i^{\text{new}} = (1 - \phi) \cdot \tau_i^{\text{old}} + \phi \cdot \tau_0 \quad (9)$$

$i \in$ direction vector indices

$$\eta_i = \left| f_{\text{current}} - f_{\text{previous}} \right| \quad (10)$$

$i \in$ direction vector indices

اگر از حافظه استفاده نشده باشد، آن گاه در صورتی که مقدار پارامتر اکتشاف فعلی از همه پارامترهای اکتشاف استفاده شده در ساختن جدول مسیریابی بزرگتر باشد (یعنی جهش بهتری در تابع ارزیابی داشته باشیم)، و همین طور مقدار تابع ارزیابی کاهش پیدا کرده باشد (در مسائل کمینه‌سازی)، آن گاه بردار گرادیان محاسبه شده، نرمال می‌شود (تقسیم بر طول بردار می‌شود). سپس این بردار نرمال شده در حافظه به جای بدترین عنصر آن قرار می‌گیرد. همین طور مقدار فرومون و پارامتر مکاشفه‌ای مربوط به برداری است که در حافظه ذخیره شده است، به روزرسانی می‌شود. در انتهای این مرحله مقادیر

جدول ۲- مقادیر پارامترها برای الگوریتمهای مختلف

GA	ES ($\mu + \lambda$)	برنامه نویسی تکاملی	GCACO	الگوریتم
۲۵	25	۲۵	۲۵	تعداد اجرا
۳۰۰	150	۱۵۰	T1-T4: ۵۰ T5: ۶۰	تعداد نسل ^{۳۸} ها یا اجتماع
۴۰۰	T1-T3 : $\mu = ۵۰, \lambda = ۱۰۰$ T4, T5 : $\mu = ۲۰۰, \lambda = ۳۰۰$	T1-T3 : $\mu = ۵۰$ T4, T5 : $\mu = ۲۰۰$	۵۰	اندازه جمعیت ^{۳۹} یا تعداد مورچه‌ها یا (μ, λ)
۰٫۷	۰	--	--	نرخ ترکیب ^{۴۰}
۰٫۲	۱	--	--	نرخ جهش ^{۴۱}

جدول ۳- مقادیر بهینه متغیرها و مقدار بهینه توابع T1, T2, T3 به ترتیب داده شده است

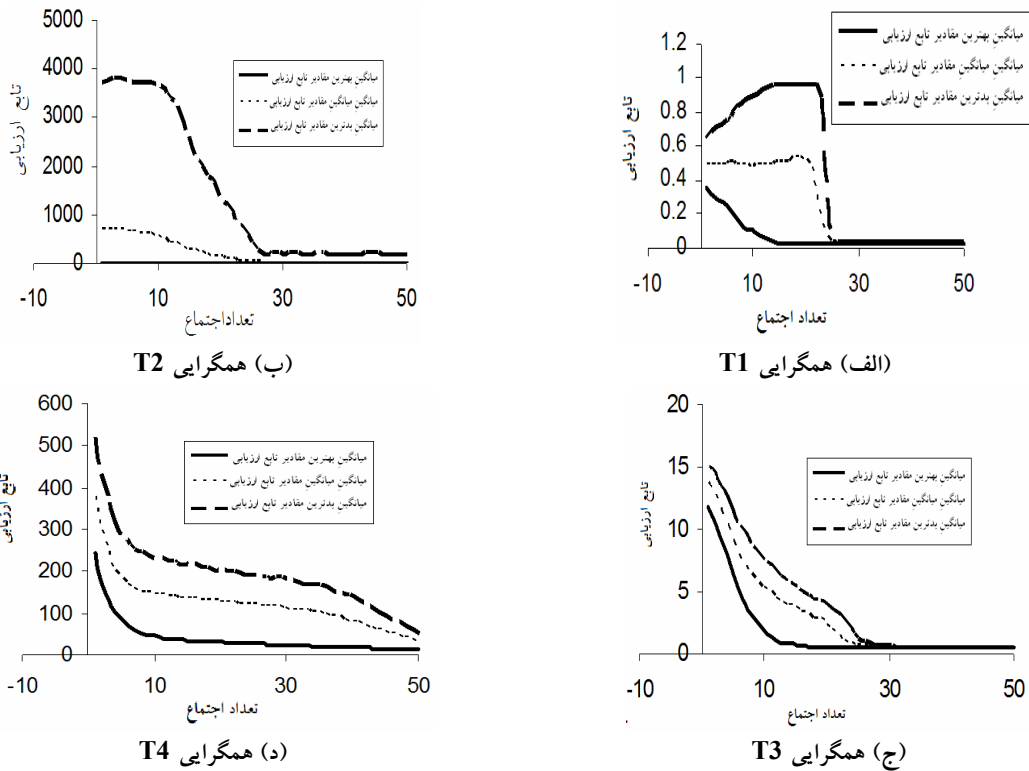
(بهترین جواب، [مقدار تاریخ])			الگوریتم
T3	T2	T1	
([-۰/۰۹۹E-۰, -۱/۷۵۱E-۴, -۷/۸۳۲E-۰, -۱/۵۰۵E-۴, -۳/۷۱۸E-۶], ۴/۴۶E-۴)	([۰/۹۲۹, ۰/۸۶۲], ۵E-۳)	([-۰/۶۶۹, -۳/۰۷], ۰/۰۱)	ژنتیک
([۰/۰۲۲, ۰/۰۰۸, ۰/۰۱۳, ۰/۰۱۴, -۰/۰۰۴], ۰/۰۶۶۱)	([۱/۰۰۱, ۱/۰۰۲], ۱/۳E-۵)	([-۱/۴۶, -۲/۷۸], ۹/۷۲E-۳)	برنامه نویسی تکاملی
([۱/۲۵۶E-۶, ۴/۵۰۸E-۷, ۶/۰۴E-۹, -۲/۹۶۹E-۷, -۷/۸۸۷E-۷], ۱/۹۱E-۶)	([۱/۰۰, ۱/۰۰], ۰)	([-۳/۱۱۹, -۰/۳۴۲], ۹/۷۲E-۳)	استراتژی تکاملی
([۶/۳۹E-۶, ۵/۸۸E-۶, ۴/۰۱E-۶, -۱/۸۸E-۶, ۱/۲۴۷E-۵], ۲/۸۳۲E-۵)	([۱/۰۰۱, ۱/۰۰۲], ۹/۱۵۸E-۷)	(۱/۰۱E-۳ × [-۰/۲۲۴۳, -۰/۱۴۷۳], ۷/۲۱E-۸)	GCACO

جدول ۴- مقادیر بهینه متغیرها و مقدار بهینه برای تابع T4

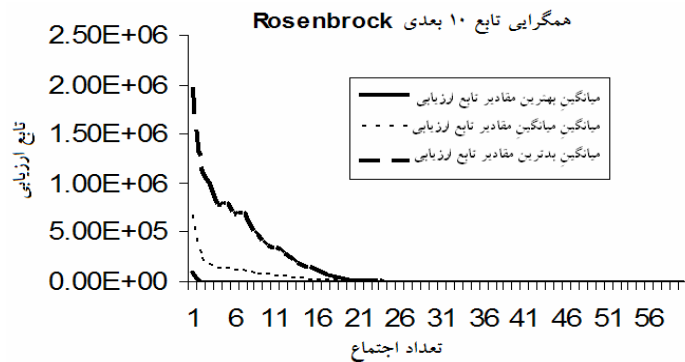
بهترین جواب	مقدار تابع	الگوریتم
[۰/۰۰۱, ۸/۵۵E-۰, -۰/۰۰۱۳, ۱/۰۲۵E-۴, -۲/۴۸۹E-۴, -۲/۵۰۳E-۴, -۲/۹۰۷E-۴, -۱/۸۴۴E-۴, -۱/۹۸۳E-۴, ۲/۳۳۴E-۵]	۶/۹۰۳E-۴	ژنتیک
[۰/۰۷۳۷, -۱/۰۵۴, ۰/۰۱۴۳, ۰/۹۵۱۸, -۰/۹۶۳۹, -۰/۹۶۷۴, -۱/۸۰۱۴, ۰/۱۲۱۸, ۰/۰۲۲۹, ۰/۰۸۵۹]	۲۰/۸۷۷	برنامه نویسی تکاملی
[۰/۹۹۵۶, ۰/۰۰۱۲, ۲/۵۶۱۲E-۴, -۰/۹۸۹۱, ۰/۹۹۵۷, ۱/۹۹۱, ۰/۰۰۲۶۴, ۰/۰۰۵۳۹, -۰/۹۹۱۵, ۱/۹۹۱]	۱۱/۹۵۷	استراتژی تکاملی
[۰/۰۰۸, -۰/۰۰۱, -۰/۰۳۸, -۰/۹۸۳, ۰/۰۲۲, ۰/۹۹۸, ۰, ۱/۰۰۳, ۰/۰۱۱۳, -۰/۰۲۷]	۳/۵۸۲	GCACO

محک T1 تا T4 در جدولهای (۳ و ۴) آورده شده است. همچنین نتایج آزمایشها برای توابع Rosenbrock پنج بعدی و ده بعدی در جدولهای (۵ و ۶) آمده است. همچنین نتایج آزمایشها برای توابع Rosenbrock پنج بعدی

تعداد تکرار از قبل تعیین شده می باشد. لازم به ذکر است که مقدار کمینه سراسری همه توابع محک صفر می باشد. مقادیر پارامترهای مشابه الگوریتمهای تکاملی و الگوریتم پیشنهادی در جدول (۲) داده شده است. نتایج آزمایشها برای



شکل ۳- همگرایی الگوریتم GCACO در ۲۵ اجرا با تعداد ۵۰ اجتماع در هر اجرا، برای توابع T1, T2, T3, T4



شکل ۴- همگرایی الگوریتم GCACO در ۲۵ اجرا با تعداد ۶۰ اجتماع در هر اجرا

جدول ۵ - مقادیر بهینه متغیرها و مقدار بهینه برای تابع Rosenbrock پنج بعدی

الگوریتم	مقدار تابع	بهترین جواب (n=5)
ژنتیک	۰/۰۱۸۶	[۱/۰۰۷, ۱/۰۰۲, ۱/۰۰۳, ۱/۰۰۶, ۱/۰۰۵]
برنامه نویسی تکاملی	۲/۸۹۶E-۴	[۱/۰۰۱, ۱/۰۰۱, ۱/۰۰۰, ۱/۰۰۱, ۱/۰۰۰۲]
استراتژی تکاملی	۲/۴۱۵E-۶	[۱/۰۰۰۱, ۱/۰۰۰۱, ۱, ۱, ۱/۰۰۰۱]
GCACO	۹/۷۹۴E-۴	[۰/۹۹۷۰, ۰/۹۹۴۰, ۰/۹۸۸۰, ۰/۹۷۸۶, ۰/۹۵۵۱]

جدول ۶- مقادیر بهینه متغیرها و مقدار بهینه برای تابع Rosenbrock ده بعدی

الگوریتم	مقدار تابع	(n=10) بهترین جواب
ژنتیک	6-7E54	[0/9999, 0/9992, 0/9999, 0/9999, 0/9999, 0/9999, 0/9999, 0/9999, 0/9999, 0/9999]
برنامه نویسی تکاملی	8371	[0/9952, 0/9929, 0/9858, 0/9718, 0/9443, 0/8932, 0/7992, 0/7993, 0/7740, 0/5554]
استراتژی تکاملی	0/1039	[1/009, 1/003, 1/002, 1/009, 1/002, 1/007, 1/009, 1/007, 1/009, 1/00]
GCACO	0/0947	[0/9950, 0/9929, 0/9989, 0/9995, 0/9983, 0/9977, 0/9877, 0/9888, 0/9983, 0/9835]

پارامترها سریعتر همگرا می شود).

در آزمون بعدی نقش علامت بردار نرمال شده گرادیان به جای خود آن بررسی شده است. به این طریق که به جای بردار نرمال شده گرادیان برداری که شامل علامت آن است در حافظه ذخیره و به کارگیری شود.

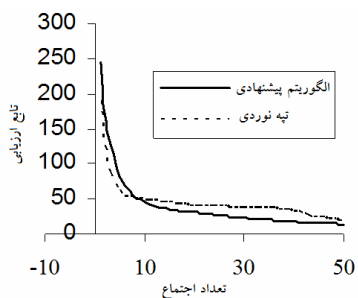
مقایسه شکل‌های (۵) و (۶) نشان می دهد که در نظر گرفتن علامت بردار گرادیان به جای خود بردار گرادیان می تواند در بهترین جواب را بهبود ببخشد.

۴- نتیجه گیری

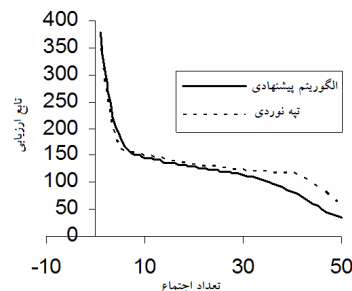
یک روش جدید با نام اختصار GCACO برای بهینه سازی توابع در فضای پیوسته بر اساس الگوریتم بهینه سازی اجتماع مورچگان ارائه شده است. اگر چه الگوریتم اجتماع مورچگان برای بهینه سازی در فضای گسسته و بهینه سازی ترکیبی نسبتاً پیشرفته کاربردهای فراوان پیدا کرده است، اما نوع آوری اصلی این مقاله تعمیم و توسعه این الگوریتم به فضای پیوسته و به خصوص بهینه سازی توابع است. رفتار الگوریتم پیشنهادی نسبت به الگوریتمهای قبلی بیشتر شبیه به مورچه های واقعی است. نتایج نشان می دهند که الگوریتم GCACO از نظر محاسباتی بسیار کارتر از الگوریتمهای دیگر تکاملی مانند استراتژی تکاملی و برنامه نویسی تکاملی است. اگر چه GCACO همانند دیگر الگوریتمهای تکاملی احتیاج به تنظیم پارامترها دارد اما نتایج به دست آمده برای مسائل محک نشان دهنده پیمایش یک مسیر مطلوب تا رسیدن به بهینه سراسری در مقایسه با سایر روشهاست. این همگرایی مطلوب به علت

و ده بعدی در ذیل آمده است. می توان نتیجه گرفت که GA در مسائل با ابعاد بالا از نظر یافتن جواب بهینه سراسری در این مورد خوب عمل می کند. در حالی که سایر الگوریتمهای تکاملی و همین طور الگوریتم GCACO برای توابع با ابعاد بالا (در این مثال $n=10$) مشابه به یکدیگر عمل می کنند. اما به علت حجم محاسباتی کمتر، (تعداد کمتر اجتماعها و مورچه های این الگوریتم در مقایسه با تعداد نسلها و جمعیت سایر الگوریتمها) کارتر بودن الگوریتم GCACO در مسائل مختلف بهینه سازی فضای پیوسته کاملاً مشهود است. همان طوری که از نتایج می توان استنباط کرد، GCACO از لحاظ کیفیت جواب در بیشتر موارد بهتر از ES و EP عمل کرده است. در هر حال بر اساس قضیه معروف و اثبات شده تحت عنوان no free lunch theorem هیچ کدام از روشها به تنهایی برای همه مسائل برتری ندارد [۱۷].

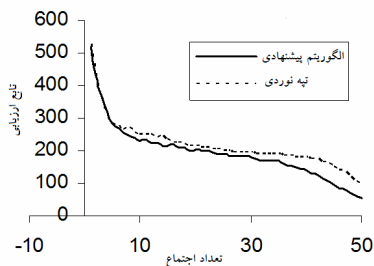
در ادامه مطالعات انجام شده، نقش خاصیت ارتباط مورچه ها توسط محیط در الگوریتم پیشنهادی با استفاده از تابع T4 مورد بررسی قرار گرفت. اگر GCACO با مقدار پارامتر $q = 1$ اجرا شود، فقط حرکت تصادفی ملاحظه می شود و نقش پارامترهای تلاش دسته جمعی (حافظه و فرامون) حذف می شود. بنابراین با حذف نقش فرامون و حافظه، الگوریتم پیشنهادی مانند یک الگوریتم جستجوی تپه نوردی خاص عمل می کند. شکل (۵)، همگرایی GCACO را با در نظر گرفتن حافظه و فرامون و بدون در نظر گرفتن این دو نشان می دهند. همان طوری که در شکلها مشاهده می شود، نقش پارامترهای تلاش دسته جمعی در اجتماعات آخر بیشتر آشکار می شود (الگوریتم با وجود این



ب) میانگین بهترین مقادیر تابع ارزیابی T4

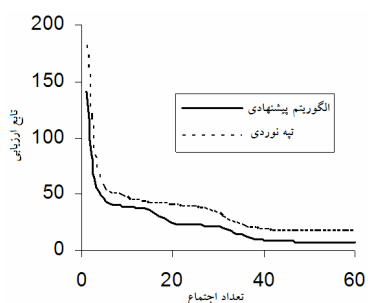


الف) میانگین میانگین مقادیر تابع ارزیابی T4

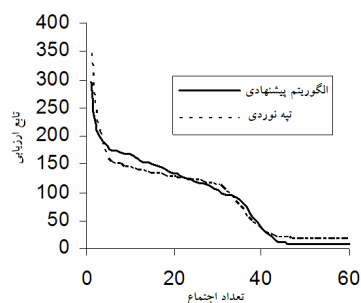


ج) میانگین بدترین مقادیر تابع ارزیابی T4

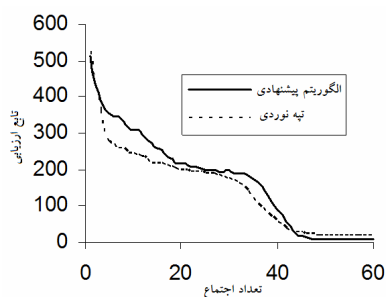
شکل ۵- همگرایی الگوریتم GCACO در ۲۵ اجرا با تعداد ۵۰ اجتماع در هر اجرا (با وجود پارامترهای ارتباط با محیط و بدون آنها)



ب) میانگین بهترین مقادیر تابع ارزیابی T4



الف) میانگین میانگین مقادیر تابع ارزیابی T4



ج) میانگین بدترین مقادیر تابع ارزیابی T4

شکل ۶- همگرایی الگوریتم GCACO در ۲۵ اجرا با تعداد ۵۰ اجتماع در هر اجرا (با وجود پارامترهای ارتباط با محیط و بدون آنها). در این شکلها نحوه همگرایی الگوریتم با در نظر گرفتن علامت بردار گرادیان به جای خود بردار گرادیان نشان داده شده است.

پیاپیاده‌سازی، کارایی محاسباتی و نرخ همگرایی مطلوب از مزایای دیگر الگوریتم پیشنهادی است. اگر چه آثار و تبعات قضیه معروف no free lunch theorem (که بیانگر این است که هیچ کدام از روشهای حل یک مسئله برای تمامی مسائل مشابه کارا نیست) از آزمایشات انجام شده کاملاً مشهود است. اما کارایی محاسباتی الگوریتم پیشنهادی، حداقل در پنج مسئله بررسی شده به وضوح مشخص است.

برقراری یک تعادل مناسب بین اکتشاف و بهره برداری در فضای جستجو هست که از طریق تنظیم مناسب پارامترها برای هر مسئله حاصل می‌شود. در این راستا شناخت مسئله و تجربه کاربر می‌تواند مفید واقع شود. آزمایشات انجام شده به خوبی نشان می‌دهند که اندازه جمعیت و تعداد نسلها در الگوریتم نسبت به سایر الگوریتمها کمتر است. این بدان معنی است که این الگوریتم از نظر محاسباتی کاراتر است. تسهیل در

واژه نامه

- | | | |
|--------------------------------------|-------------------------------------|---------------------------------------|
| 1. ant colony optimization | 16. Dorigo | 30. Bilchev |
| 2. continuous spaces | 17. tour | 31. nest |
| 3. stigmergic property | 18. stagnation | 32. greedy |
| 4. bench mark | 19. internal state | 33. hill climbing |
| 5. genetic algorithm | 20. Tabu list | 34. evaluation function |
| 6. evolutionary strategy | 21. decision table | 35. roulette wheel |
| 7. evolutionary programming | 22. ant routing table | 36. feasibility |
| 8. meta-heuristic | 23. heuristic | 37. delayed online pheromone updating |
| 9. traveling sales man problem (TSP) | 24. Oliver-30 | 38. generation |
| 10. Grassé | 25. ant colony system (ACS) | 39. population size |
| 11. Bellicositermes Natalensis | 26. Gambardella | 40. combination (x-over) |
| 12. Cubitermes | 27. pseudo random proportional rule | 41. mutation |
| 13. pheromone deposition | 28. exploitation | |
| 14. recruitment | 29. bias exploration | |

مراجع

- Grassé P. P., "La reconstruction du nid et les coordinations interindividuelles chez bellicositermes natalensis et cubitermes sp. La théorie de la stigmergie: essai d'interprétation du comportement des termites constructeurs," *Insectes Sociaux*, Vol. 6, pp. 41–81, 1995.
- Theraulaz, G., and Bonabeau, E., "A Brief History of Stigmergy," *Artificial Life*, Vol. 5, pp. 97–116, 1999.
- Pasteels, J. M., Deneubourg, J. L., and Goss, S., "Self-Organization Mechanisms in Ant Societies (i): Trail Recruitment to Newly Discovered Food Sources," *Experientia Supplementum*, Vol. 54, pp. 155-175, 1987.
- Dorigo, M., "Optimization, Learning and Natural Algorithms (in Italian)," PhD thesis, Dipartimento di Elettronica e Informazione, Politecnico di Milano, IT, 1992.
- Dorigo, M., Maniezzo, V., and Colorni, A., "Positive Feedback as a Search Strategy," *Technical Report 91-016*, Dipartimento di Elettronica, Politecnico di Milano, IT, 1991.
- Dorigo, M., Di Caro, G., and Gambardella, L. M., "Ant Algorithms for Discrete Optimization," *Artificial Life*, Vol. 5, No. 3, pp. 137-172, 1999.
- Dorigo, M., Maniezzo, V., and Colorni, A., "The Ant System: Optimization by a Colony of Cooperating Agents," *IEEE Transactions on Systems, Man, and Cybernetics—Part B*, Vol. 26, No. 1, pp. 29–41, 1996.
- Whitley, D., Starkweather, T., and Fuquay, D., "Scheduling Problems and Travelling Salesman: The Genetic Edge Recombination Operator," *In Proceedings of the Third International Conference on Genetic Algorithms*, pp. 133–140. Palo Alto, CA: Morgan Kaufmann, 1989.
- Dorigo M. and Gambardella, L. M., "Ant Colonies for the Traveling Salesman Problem," *BioSystems*, Vol. 43, pp. 73–81, 1997.
- Dorigo M., and Gambardella, L. M., "Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem," *IEEE Transactions on Evolutionary Computation*, Vol. 1, No. 1, pp. 53–66, 1997.
- Gambardella, L. M. and Dorigo, M., "Solving Symmetric and Asymmetric TSPs by Ant Colonies,"

- In Proceedings of the IEEE Conference on Evolutionary Computation, ICEC96*, pp. 622–627, IEEE Press 1996.
12. Gambardella, L. M., and Dorigo, M, HAS-SOP: “A Hybrid Ant System for the Sequential Ordering Problem,” *Technical Report* PP. 11-97, IDSIA, Lugano, CH, 1997.
 13. BILCHEV, G., and PARMEE, I. C., “The Ant Colony Metaphor for Searching Continuous Design Spaces,” *Proceedings of the AISB Workshop on Evolutionary Computation*, University of Sheffield, UK, April, pp. 3-4, 1995.
 14. Mathur, M., Karale, S. B., Priye, S., Jyaraman, V. K. and Kullkarni, B. D., “Ant Colony Approach to Continuous Function Optimization,” *Ind. Eng. Chem. Res.* Vol. 39, pp. 3814-3822, 2000.
 15. Dréo, J., and Siarry, P., “A New Ant Colony Algorithm Using the Heterarchical Concept Aimed at Optimization of Multim minima Continuous Functions,” *Proceedings of Third International Workshop, ANTS 2002*, Brussels, Belgium, 2002.
 16. Li Yan-jun, Wu Tie-jun, “An Adaptive Ant Colony System Algorithm for Continuous-Space Optimization Problems,” *Journal of Zhejiang University Science*, Vol. 4, No. 1, pp. 40-46, 2003.
 17. Bach, T., *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. New York Oxford: Oxford University Press, 1996.
 18. Fogel, D. B., *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*. 2nd ed, New York: IEEE Press, 2000.