Научно-практический электронный журнал
Оригинальные исследования (ОРИС), *ISSN 2222-0402*
www.ores.su

УДК 681.5.015.23

АЛГОРИТМ СТАИ СЕРЫХ ВОЛКОВ (GWO) ДЛЯ ЗАДАЧ ОПТИМИЗАЦИИ

Лагунова А.Д.
ФГБУН Институт автоматики и процессов управления Дальневосточного отделения Российской академии наук
Владивосток, Россия (690041, Владивосток, ул. Радио, 5)
e-mail: schimka_06@mail.ru

Аннотация

В данной работе проведено исследование алгоритма стаи серых волков (Gray Wolf Optimizer, GWO) на примере оптимизационной задачи. Было изучено влияние на эффективность алгоритма таких параметров, как: размер популяции, количество итераций, величина коэффициентов атаки и выслеживания. Для исследования эффективности и сравнительного анализа алгоритма был разработан программный модуль, в котором также реализованы такие методы, как: полный перебор, метод сеток, а также их параллельные реализации. В качестве критериев оценки были выбраны: скорость работы алгоритма, наиболее оптимальное найденное значение, среднее отклонение найденных решений. На основе проведенных исследований можно сделать вывод, что GWO является достаточно эффективным. В случае овражной или мультимодальной целевой функции рекомендуется использовать параллельные вычисления для повышения эффективности за счет расчета нескольких стай волков одновременно, либо использовать повторный запуск на лучших решениях.

Ключевые слова: алгоритм стаи волков, оптимизация, эвристический алгоритм, роевая оптимизация, стохастическая оптимизация.

ALGORITHM OF A GRAY WOLF OPTIMIZER IN THE PROBLEM OF OPTIMAL PARAMETRIC SYNTHESIS

Lagunova A.D.
Institute of Automation and Control Processes Far Eastern Branch of Russian Academy of Sciences
Vladivostok, Russia (690041, Vladivostok, 5 Radio Street)
e-mail: schimka_06@mail.ru

ABSTRACT

In this paper, a study of the algorithm of a Gray Wolf Optimizer (GWO) was conducted using the optimization problem as an example. The influence on the efficiency of the algorithm of such parameters as the population size, the number of iterations, the magnitude of the attack and

tracking factors was studied. To study the effectiveness and comparative analysis of the algorithm, a software module was developed, which also implemented such methods as: direct search method, grid method, and their parallel implementations. As evaluation criteria were chosen: the speed of the algorithm, the most optimal value found, the average deviation of the solutions found. On the basis of the conducted research, it can be concluded that the algorithm GWO is quite effective. In the case of a ravine or multimodal objective function, it is recommended to use parallel computing to increase efficiency by calculating several packs of wolves at the same time, or to use the restart on the best solutions.

Key words: gray wolf optimizer (GWO), optimization, heuristic algorithm, swarm optimization, stochastic optimization.

Введение

Оптимизационная задача – это задача нахождения экстремума (минимума или максимума) целевой функции в некоторой области конечномерного векторного пространства, ограниченной набором линейных и/или нелинейных равенств и/или неравенств. Методы оптимизации находят широкое применение в области экономики, техники и управления. Функции, описывающие задачу управления или проектируемую систему, как правило, имеют сложный нелинейный характер, что не позволяет получать оптимальное решение в аналитической форме с помощью классических методов вариационного и дифференциального исчисления. Высокая вычислительная трудоемкость решения оптимизационных задач со стохастическим критерием заставляет искать способы достаточно быстрого получения желаемых результатов [1, 4].

В последние годы активно развиваются эвристические и метаэвристические алгоритмы оптимизации - алгоритмы, включающие практический метод, не являющийся гарантированно точным или оптимальным, но достаточный для решения задачи [2, 3, 5, 9, 15]. Правильность данных алгоритмов для всех возможных случаев не доказана, однако известно, что такие алгоритмы дают достаточно хорошее решение в большинстве случаев. Данные алгоритмы позволяют ускорить решение задачи в 100-1000 раз, что особенно важно в задачах с большим числом переменных. Кроме того, эвристические алгоритмы позволяют найти решение даже в тех случаях, когда точное решение не может быть найдено или его поиск имеет большую вычислительную сложность. В настоящее время наиболее известным представителем эвристических методов является роевой интеллект, описывающий коллективное поведение децентрализованной самоорганизующийся системы. На данный момент существует большое количество роевых алгоритмов, например, метод роя частиц, муравьиный алгоритм, алгоритм кукушки и пр. [5, 8, 11] Одним из новейших алгоритмов этого типа является алгоритм поиска стаей серых волков.

Цель данной работы – исследование эффективности и сравнительный анализ алгоритма поиска стаи серых волков, выявление недостатков алгоритма для функций разного типа и предложение решений для их устранения.

Алгоритм стаи серых волков

Алгоритм серых волков — метаэвристический стохастический алгоритм роевого интеллекта, разработанный в 2014 году, идея которого построена на основе модели охоты стаи серых волков [13]. Выделяют четыре типа волков: альфа, бета, дельта и омега. Альфа имеет наибольший «вес» в принятии решений и управлении стаей. Далее идут бета и затем дельта, которые подчиняются альфе и имеют власть остальными волками. Волк-омега всегда подчиняется остальным доминирующим волкам.

В математической модели иерархии волков альфа считается наиболее благоприятным решением. Второе и третье лучшее решение, соответственно бета и

дельта. Остальные решения считаются омегами. Полагают, что к самым приспособленным волкам (альфа, бета и дельта), то есть находящимся ближе всего к добыче, будут приближаться остальные волки. После каждого приближения определяется кто на данном этапе альфа, бета и дельта, а затем волки опять перестраиваются. Перестроение происходит до тех пор, пока волки не собираются в стаю, что будет являться оптимальным направлением для атаки с минимальным расстоянием

В ходе алгоритма выполняются 3 основных этапа, в которых волки ищут добычу, окружают и атакуют ее.

В процессе поиска обнаруживаются альфа, бета и дельта — волки, которые находятся ближе к добыче. Остальные волки, подчиняясь доминирующим, могут начать окружать добычу или продолжить произвольное перемещение в поисках лучшего варианта.

Модель окружения волками добычи (Рис. 1) описывается следующими формулами [13]:

$$\vec{D} = \left|\vec{C} \times \vec{X}_p(t) - \vec{X}(t)\right|; \quad (1)$$

$$\vec{X}(t+1) = \vec{X}_p(t) - \vec{A} \times \vec{D}; \quad (2)$$

$$\vec{A} = 2\vec{a} \times \vec{r}_1 - \vec{a}; \quad (3)$$

$$\vec{C} = 2 \times \vec{r_2}, \quad (4)$$

где *t* обозначает текущую итерацию, $\vec{A}$ и $\vec{C}$ – векторы-коэффициенты, $\vec{X}_p$ – вектор положения жертвы, $\vec{X}$ – вектор положения волка, $\vec{D}$ – вектор направления (от волка к жертве). Параметр $\vec{a}$ линейно уменьшается от 2 до 0 в каждой итерации, а $\vec{r}_1$ и $\vec{r_2}$ – случайные вектора из интервала [0,1], которые позволяют моделировать перемещение волков. При этом, пространство может быть *N*-мерное, что означает, что волки будут перемещаться вокруг жертвы в гиперпараллелепипиде.

Самой же охотой, как правило, занимается альфа (иногда при участии беты и дельты). Однако в абстрактном пространстве поиска мы не можем даже предположить, где находится оптимум (жертва). Для построения математической модели охоты предположим, что альфа, бета и дельта имеют наиболее четкие представления о местоположении добычи. Следовательно, необходимо сохранять первые три лучших решения и направлять омега-волков ближе к этим решениям.

Волки заканчивают охоту атакой на добычу, когда добыча перестает убегать. Чтобы математически смоделировать это явление, необходимо уменьшать значение $\vec{a}$. Диапазон колебаний коэффициента атаки $\vec{A}$ также снизится на $\vec{a}$. То есть, $\vec{A}$ — это случайное число в интервале [-2*a*, 2*a*], где *a* — число, последовательно уменьшающееся от 2 до 0. Если $\vec{A}$ находится от -1 до 1, то следующая позиция данного волка будет находиться где-то между текущей позицией и потенциальным положением жертвы.

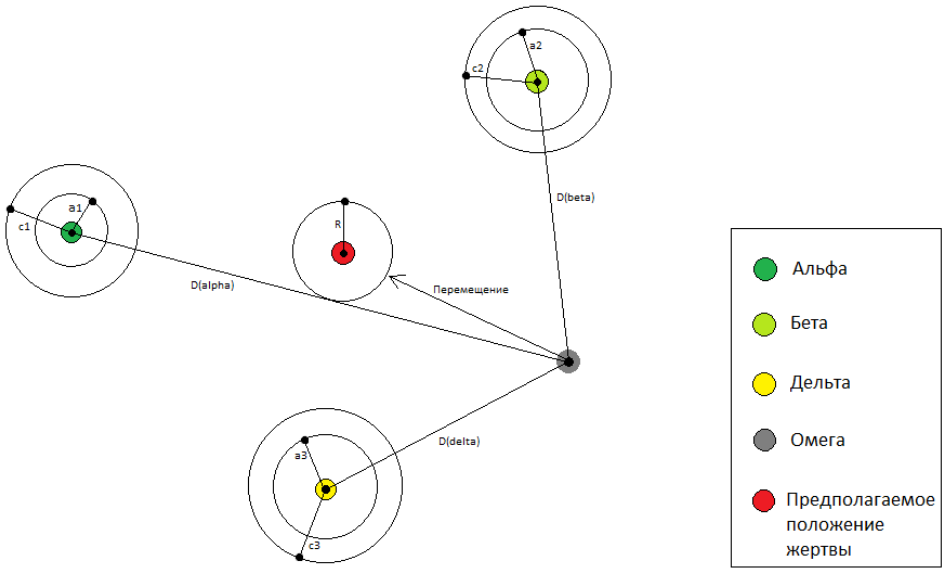


Рис.1. *Окружение жертвы стаей волков*

Возвращаясь к процессу поиска добычи, необходимо обратить внимание на периодическое удаление омег от альфы, беты и дельты ради поиска более благоприятного решения. Чтобы математически сформулировать это расхождение можно изменить интервал для вектора $\vec{A}$. Используя значения больше 1 или меньше -1, мы принуждаем омегу отдаляться от потенциальной добычи, что позволяет алгоритму проводить поиск оптимума на большей территории. Другим важным параметром поиска является коэффициент выслеживания $\vec{C}$, являющийся случайным на интервале [0, 2]. Данный параметр обеспечивает случайные веса для добычи, увеличивая (C> 1) или уменьшая (C <1) влияние (запах) добычи при определении расстояния для перемещения. Это позволяет алгоритму серых волков демонстрировать более случайное поведение в процессе оптимизации, способствуя поиску и обходу локальных оптимумов. Также, стоит отметить, что $\vec{C}$ не уменьшается последовательно в отличие от $\vec{A}$. $\vec{C}$ всегда является случайным как на начальных итерациях, так и на конечных. Это свойство очень полезно в случае локальной стагнации оптимума, особенно на последних итерациях.

В общем виде реализуется следующий алгоритм:

1) Установка размера популяции n и максимального количества итераций $iter$
2) Инициализировать позиции серых волков X_i $(i=1..n)$
3) Инициализировать $\vec{a}$, $\vec{A}$ и $\vec{C}$
4) Вычислить значение целевой функции для каждого волка
5) Определить X_a, X_b, X_d - альфу, бету и дельту
6) Пока t < $iter$:
   * Пересчитать позиции для каждого волка
   * Обновить $\vec{a}$, $\vec{A}$ и $\vec{C}$
   * Вычислить значение целевой функции для каждого волка
   * Обновить X_a, X_b, X_d
   * $t = t + 1$
7) Вывод X_a

Исследование эффективности алгоритма

В ходе исследований, алгоритм был реализован в классическом виде для решения задачи оптимизации. Также была реализована возможность задать такие параметры

работы алгоритма, как: максимальное количество итераций, размер популяции и количество варьируемых параметров для нахождения оптимального значения целевой функции.

Для теста были выбраны три функции (Рис.2) [10,12,14]:

- Функция Де-Джонга (2- и 3-мерная):

$f(x) = \sum_{i=1}^{n} x_i^2 \ , \ x_i \in [-5.12; 5.12]$. Глобальный минимум равен 0 при $x_i = 0$

- Функция Розенброка (2- и 3-мерная):

$$f(x) = \sum_{i=1}^{n-1} \left[100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right] \ , \ x_i \in [-2{,}048; 2{,}048].$$

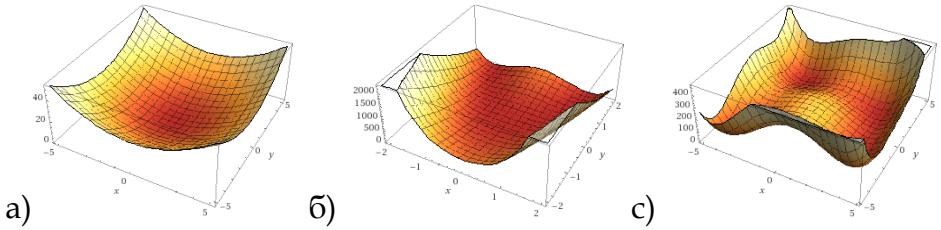
Глобальный минимум равен 0 при $x_i = 0$

- Функция Химмельблау (2-мерная):

$$f(x) = (x^2 + y - 11)^2 + (x + y^2 - 7)^2 \ , \ x_i \in [-5; 5].$$

Глобальный минимум равен 0 при

$f(3;\ 2),\ f(-2.805118;\ 3.131312),\ f(-3{,}779310;\ -3{,}283186),$
$f(3.584428;\ -1.848126).$



а) б) с)

Рис.2. *а) функция Де-Джонга, б) функция Розенброка, с) функция Химмельблау*

Алгоритм серых волков запускался 20 раз для каждого набора входных данных, чтобы оценить результаты в среднем. Для оценки эффективности алгоритма использовались такие параметры, как: наименьшее время работы алгоритма в секундах (*time*), наиболее оптимальное найденное значение во всей серии (*min*), среднее отклонение от реального оптимума (*S*). Результаты сравнивались с методом полного перебора, методом сеток и их параллельной реализацией (*N* от 2 до 8 процессов) [7]. В таблицах приведены только лучшие по скорости выполнения параллельные алгоритмы для каждого набора данных. Для большей информативности даны также сведения о размере пространства поиска (количество точек на входе) и о количестве точек, для которых реально производились вычисления (*point*).

На основании приведенных ниже данных (табл. 1-3), что алгоритм GWO достаточно эффективен в задачах как с одноэкстремальной, так и с многоэкстремальной целевой функцией.

Например, для выпуклой функции Дэ-Джонга алгоритм оказался крайне эффективным (особенно в случае двух переменных). И действительно, вся стая «чувствует запах» одной жертвы и приближается к ней, не имея других вариантов. Причем приближение это приводит к глобальному оптимуму с высокой долей вероятности (погрешность <0.0001). Кроме того, даже не смотря на довольно большое пространство поиска (более 1 млрд. точек), время работы алгоритма занимает всего долю секунды, что почти в 200 тыс. раз быстрее, чем метод полного перебора. Метод сеток также проигрывает волчьей стае, как по скорости, так и по точности, в виду сильной зависимости от правильного выбора сетки. В данном случае обе сетки попали на глобальный оптимум, однако, так происходит не всегда. Даже при наложении более плотной сетки (например,

0.1) высока вероятность получить значение целевой функции довольно далекое от глобального оптимума.

Таблица 1

Функция Де-Джонга

	2 переменные							3 переменные						
	Кол-во точек на входе	iter	N	point	time	min	S	Кол-во точек на входе	iter	N	point	time	min	S
Полный перебор 0,01	1 048 576	-	1	1 048 576	11,916	**0**	-	1 073 741 824	-	1	1 073 741 824	15 755	**0**	-
		-	8		**2,814**	**0**	-		-	8		**3 867**	**0**	-
Сетка 0,16	4 096	-	1	4 096	0,111	**0**	-	262 144	-	1	262 144	3,913	**0**	-
		-	2		**0,106**	**0**	-		-	6		**1,009**	**0**	-
Сетка 0,32	1024	-	1	1024	**0,104**	**0**	-	32 768	-	1	32 768	0,605	**0**	-
		-	2		0,106	**0**	-		-	3		**0,205**	**0**	-
Стая 20 волков	1 048 576	10	1	200	**0,004**	0,0001	0,0261	1 073 741 824	10	1	200	**0,006**	0,0094	0,245
		20		400	0,008	**0**	0,0003		20		400	0,011	0,0002	0,0115
		30		600	0,011	**0**	**0**		30		600	0,018	**0**	0,0031
		40		800	0,015	**0**	**0**		40		800	0,022	**0**	**0,0001**
Стая 30 волков	1 048 576	10	1	300	**0,006**	0,0004	0,0029	1 073 741 824	10	1	300	**0,008**	0,0017	0,0809
		20		600	0,013	**0**	0,0001		20		600	0,017	0,0001	0,0044
		30		900	0,018	**0**	**0**		30		900	0,026	**0**	0,0004
		40		1200	0,022	**0**	**0**		40		1200	0,032	**0**	**0,00003**

В случае овражной функции Розенброка от двух переменных GWO оказался не столь эффективным (табл. 2). В серии из 20 запусков минимум равный нулю так и не был обнаружен. Несмотря на то, что были найдены значения близкие к нему (0,002-0,004), средняя погрешность не является удовлетворительной. Но даже при таких результатах GWO оказывается более эффективным, чем параллельный метод сеток, работающий в 30-100 раз медленнее. Для функции трех переменных результаты, ожидаемо, оказались еще хуже: минимальное найденное значение – 0,17 при достаточно высокой величине погрешности – 1,5. Однако очевидно, что точность вычислений повышается при увеличении популяции стаи и количества итераций. Например, при 100 итерациях для популяции из 500 волков, алгоритм находит минимум 0,006 за 1,22 секунды с погрешностью 0,093. Тем не менее, такие результаты также не являются удовлетворительными, так как значительно возрастает время расчетов и количество вычислений целевой функции, что может оказаться очень трудоемким процессом для некоторых задач [7].

Таблица 2

Функция Розенброка

	2 переменные							3 переменные						
	Кол-во точек на входе	iter	N	point	time	min	S	Кол-во точек на входе	iter	N	point	time	min	S
Полный перебор 0,001	16 777 216	-	1	16 777 216	255,8	0	-	68 719 476 736	-	1	68 719 476 736	~ 12 дней	0	-
		-	8		**59,12**	**0**	-		-	8		**~ 35 часов**	**0**	-
Сетка 0,01	167 281	-	1	167 281	2,7	0,0004	-	68 417 929	-		68 417 929	1682	0,0008	-
		-	8		**0,71**	**0,0004**	-		-			**405,5**	**0,0008**	-
Сетка	1 600	-	1	1 600	0,11	0,0797	-	64 000	-	1	64 000	1,807	0,169	-

0,1		-	4		**0,10**	**0,0797**	-		-	6		**0,409**	**0,169**	-
Стая 20 волков	16 777 216	10	1	200	**0,003**	0,5700	16,91	68 719 476 736	10	1	200	**0,005**	0,771	5,196
		20		400	0,008	0,0300	0,106		20	1	400	0,010	0,405	1,911
		30		600	0,013	0,0081	0,071		30	1	600	0,017	0,415	1,860
		40		800	0,017	**0,0025**	0,027		40	1	800	0,021	**0,390**	**1,353**
Стая 30 волков	16 777 216	10	1	300	**0,006**	0,0413	0,158	68 719 476 736	10	1	300	**0,007**	0,488	3,585
		20		600	0,011	0,0210	0,103		20	1	600	0,016	**0,170**	1,505
		30		900	0,018	0,0041	0,096		30	1	900	0,024	0,620	1,283
		40		1200	0,025	**0,0034**	0,030		40	1	1200	0,032	0,371	**1,160**

Несмотря на то, что для мультимодальной функции Химмельблау (табл. 3) GWO работает эффективнее, чем для функции Розенброка, средняя погрешность оказалась не удовлетворительной. Тем не менее, для данной целевой функции, в отличие от той же функции Розенброка, увеличение количества итераций и популяции стаи заметно увеличивает точность расчетов. Например, стая из 20 волков уже при 100 итерациях находит минимум равный 0,0003 за 0,465 секунд. При этом погрешность уменьшается в 10 раз (до 0,029). А стая из 500 волков при 200 итерациях находит минимум 0,00009 с погрешностью 0,0004 за 2,49 секунды. Однако в таком случае мы снова сталкиваемся с проблемой большого количества вычислений целевой функции и значительного увеличения времени.

Таблица 3

Функция Химмельблау

	2 переменные						
	Кол-во точек на входе	iter	N	point	time	min	S
Полный перебор 0,01	1 000 000	-	1	1 000 000	17,423	**0**	-
		-	8		**4,197**	**0**	-
Сетка 0,1	10 000	-	1	10 000	0,203	**0**	-
		-	4		**0,105**	**0**	-
Сетка 0,5	400	-	1	400	**0,104**	**0**	-
		-	2		0,105	**0**	-
Стая 20 волков	16 777 216	10	1	200	**0,004**	0,096	2,476
		20		400	0,008	0,150	1,305
		30		600	0,011	0,092	1,046
		40		800	0,018	**0,013**	**0,301**
Стая 30 волков	16 777 216	10	1	300	**0,007**	0,068	1,022
		20		600	0,011	0,013	0,816
		30		900	0,020	0,013	0,458
		40		1200	0,025	**0,006**	**0,395**

Стоит также обратить внимание на поведение алгоритмов при наличии нескольких минимумов. На данной функции метод сеток находит минимум в точке (3, 2) и не видит других точек, так как их координаты заданы числами с шестью знаками после запятой. Даже заведомо зная координаты этих точек, невозможно подобрать такую сетку, которая находила бы сразу несколько этих точек. Кроме того, если бы все минимумы находились в точках с такой же точностью, алгоритм не нашел бы ни одну из них, т.к. только в случае наложения очень плотной сетки (0,000001) можно было бы гарантировать их нахождение. Однако такая сетка повысит количество вычислений функции и, даже при

распараллеливании на 8 процессов, займет более 100 часов. Стая волков в этом случае оказывается намного эффективнее алгоритма сеток, т.к. она менее чувствительна к размерам пространства поиска и позволяет находить точки с координатами заданной точности.

Кроме того, в случае мультимодальной целевой функции, обнаруживается такое явление, как блуждание волков между жертвами. Если все три доминирующих волка напали на след одной жертвы, то и вся стая нападет на нее. Но если три доминанты приближаются к разным жертвам, то вся стая начинает метаться между ними, что заметно снижает эффективность алгоритма. Как уже было указано выше, увеличение итераций и количества волков в стае для функции Химмельблау значительно повысило точность решения и снизило погрешность. Тем не менее, даже при увеличении популяции до 1000 особей, стая продолжила случайно блуждать между жертвами, а точный минимум так и не был найден.

В качестве еще одной интересной особенности GWO, стоит отметить заметное учащение случаев окружения и нападения на одну жертву при значительном увеличении популяции стаи. При стандартной популяции в 20-30 особей случаи окружения одной или нескольких жертв появлялись с одинаковой частотой.

Методы повышения эффективности

На основе приведенных исследований эффективности алгоритма GWO и его поведения в случае различный целевых функций, можно сделать вывод о высокой эффективности алгоритма для выпуклых функций. В случаях овражных или мультимодальных функций, требуется значительное увеличение времени и количества расчетов целевой функции, тем не менее это не гарантирует нахождение точного минимума.

Для повышения эффективности алгоритма есть смысл использовать не увеличение популяции стаи, а увеличение количества стай. При этом у каждой стаи должна быть своя территория, на которую запрещено заходить посторонним волкам. В этом случае овражные и мультимодальные функции в определенной степени перестают быть проблемными. Каждой стае придется обследовать меньшую территорию, что в овражных функциях не даст всем волкам скапливаться вместе и позволит хотя бы одной стае найти более точные решения. В случае функции с несколькими экстремумами, деление на несколько стай позволит каждой стае более тщательно исследовать свою территорию и не отвлекаться на другие жертвы вне нее границ, что, в свою очередь, увеличит вероятность нахождения нескольких жертв и в определенной степени избавится от бесцельного блуждания между несколькими жертвами.

В природе разные стаи охотятся независимо друг от друга. Чтобы смоделировать такое поведение большого количества волков можно разбить вычисления на несколько процессов, каждый из которых будет моделировать поведение одной стаи на выделенной ей территории. Многопроцессорные вычисления позволят не только с большей вероятностью находить точные экстремумы, но и не увеличивать время расчетов за счет параллельной обработки данных для каждой стаи.

Еще одним методом повышения эффективности может стать вторичной запуск стай на места наиболее вероятного расположения жертвы. Например, в случае нахождения стаями нескольких минимальных значений, находящихся в удаленных друг от друга точках, имеет смысл запустить по одной стае на территории вблизи этих точек. Это позволит стае игнорировать другие решения и особенности поверхности функций.

Заключение

Высокая вычислительная трудоемкость решения оптимизационных задач со стохастическим критерием заставляет искать способы достаточно быстрого получения желаемых результатов. На данный момент разработано не малое количество методов и алгоритмов, в известной мере ускоряющих процедуру поиска оптимальных решений. Наиболее радикальным направлением, сокращающим трудоемкость решения сложных вычислительных задач, стали эвристичекие алгоритмы, одним из которых является алгоритм поиска стаей серых волков (GWO).

Сравнительный анализ алгоритма GWO показал, что:

- с помощью коэффициентов выслеживания и атаки, возможно более точно смоделировать поведение волков и увеличить вероятность нахождения жертвы
- алгоритм показывает достаточно высокую эффективность для выпуклых функций
- алгоритм является более эффективным, по сравнению с методом сеток, особенно, на большом пространстве поиска (точность $>1*10^{-4}$)
- на овражных функциях GWO имеет высокую погрешность
- в случае мультимодальной функции, стая будет бесцельно метаться, если доминирующие волки приблизятся к разным жертвам
- в случае мультимодальной функции GWO с большой вероятностью находит локализацию нескольких жертв
- увеличение количества итераций или популяции стаи позволяет в определенной степени повысить точность вычислений и снизить погрешность
- повышение точности за счет увеличения популяции и количества итераций ведет к значительному увеличению количества вычислений целевой функции, что подходит не для всех задач
- использование параллельных вычислений для запуска алгоритма с несколькими стаями позволит эффективнее находить экстремума в случае овражных или мультимодальных функций, не увеличивая время работы
- в случае мультимодальной функции целесообразен повторный запуск стай в окрестности найденных точек

Список литературы

1. Абрамов О.В., Катуева Я.В. Технология параллельных вычислений в задачах анализа и оптимизации / О.В. Абрамов, Я.В. Катуева // Проблемы управления. – 2003. №4. – С. 11-15
2. Ахмедова Ш.А. Последовательный и параллельный стайный алгоритм для задач условной и безусловной оптимизации / Ш.А. Ахмедова // Актуальные проблемы авиации и космонавтики. – 2012. – Т.1. №8. – С. 289-290
3. Брестер К.Ю. О решении задач многокритериальной оптимизации самонастраивающимся генетическим алгоритмом / К.Ю. Брестер // Актуальные проблемы авиации и космонавтики. – 2012. – Т.1. №8. – С. 290-291
4. Диго Г.Б., Диго Н.Б., Катуева Я.В. Применение детерминированных критериев в задачах стохастической оптимизации / Г.Б. Диго, Н.Б. Диго, Я.В. Катуева // Многопроцессорные вычислительные системы. – 2006. - №2(12). - С. 82-88
5. Кулиев Э.В., Щеглов С.Н., Пантелюк Е.А., Кулиева Н.В. Адаптивный алгоритм стаи серых волков для решения задач проектирования / Э.В. Кулиев, С.Н.

Щеглов, Е.А. Пантелюк, Н.В. Кулиева // Известия ЮФУ. Технические науки. - 2017 . - №7. - С.28-38.
6. Лагунова А.Д., Назаров Д.А. Параллельный алгоритм решения задачи оптимального параметрического синтеза на основе метода сеток / А.Д. Лагунова, Д.А. Назаров // Труды Международного симпозиума "Надежность и качество". - 2018. - Т.1. - С. 255-258
7. Матренин П.В. Методы стохастической оптимизации [Текст]: учебное пособие / П.В. Матренин, М.Г. Гриф, В.Г. Секаев. – Новосибирск: НГТУ, 2016. - 65 с.
8. Орловская Н.М. Анализ биоинспирированных методов глобальной оптимизации // Труды МАИ. – 2014. - №73. URL: https://mai.ru/upload/iblock/85d/85d945530545d38e6d4cbd591417766d.pdf (дата обращения: 17.04.2019)
9. Сагун А.В., Хайдуров В.В., Кунченко-Харченко В.И. Метод стаи волков и его модификация для решения задачи поиска оптимального пути / А.В. Сагун, В.В. Хайдуров, В.И. Кунченко-Харченко // Фізико-математична освіта : науковий журнал. – 2017. - №2(12). – С. 135-139
10. Сергиенко А. Б. Тестовые функции для глобальной оптимизации / А.Б. Сергиенко. – Красноярск: Изд-во СГАУ им. М.Ф. Решетнева, 2015. – 112 с.
11. Частикова В.А., Дружинина М.А., Кекало А.С. Исследование эффективности алгоритма поиска косяков рыб в задаче глобальной оптимизации // Современные проблемы науки и образования. - 2014. - №4. - URL: http://science-education.ru/ru/article/view?id=14142 (дата обращения: 17.04.2019)
12. Bossek Y. SMOOF: Single- and Multi-Objective Optimization Test Functions / Y. Bossek // The R Journal. - 2017. - Vol. 9 (1). - P. 103-113
13. Mirjalili S., Lewis A. Grey Wolf Optimizer / S. Mirjalili, A. Lewis // Advanced in Engineering Software. - 2014. - Vol 69. – P. 46-61.
14. Molga M., Smutnicki C. Test functions for optimization needs: [Электронный ресурс]. 2005. URL: https://www.vafaeijahan.com/en/wp-content/uploads/2012/02/Test-functions-for-optimization-needs.pdf (Дата обращения: 17.04.2019).
15. Zong W.G., Williams J.C. Ecological Optimization using Harmony Search / W.G. Zong, J.C. Williams // American conference on applied mathematics. – 2008. – P. 148-152

References

1. Abramov OV, Katueva Ya.V. Technology of parallel computing in the tasks of analysis and optimization / O.V. Abramov, Ya.V. Katueva // Management problems. - 2003. №4. - P. 11-15 [in Russian].
2. Akhmedova Sh.A. Consecutive and parallel sharing algorithm for problems of conditional and unconditional optimization / Sh.A. Akhmedova // Actual problems of aviation and cosmonautics. - 2012 - T.1. №8. - P. 289-290 [in Russian].
3. Brester K.Yu. On solving problems of multicriteria optimization of a self-adjusting genetic algorithm / K.Yu. Brester // Actual problems of aviation and astronautics. - 2012 - T.1. №8. - P. 290-291 [in Russian].
4. Digo G. B, Digo N. B, Katueva Ya.V. Application of deterministic criteria in problems of stochastic optimization / G. B. Digo, N.B. Digo, Ya.V. Katueva // Multiprocessor Computing Systems. - 2006. - №2 (12). P. 82-88 [in Russian].

5. Kuliev E.V., Shcheglov S.N., Pantelyuk E.A., Kuliyeva N.V. Adaptive algorithm for solving design problems / E.V. Kuliev, S.N. Scheglov, E.A. Pantelyuk, N.V. Kulieva // News of SFU. Technical science. - 2017 - №7. - P.28-38 [in Russian].
6. Lagunova A.D., Nazarov D.A. Parallel algorithm for solving the problem of optimal parametric synthesis based on grid parameters / A.D. Lagunova, D.A. Nazarov // Proceedings of the International Symposium "Reliability and Quality." - 2018. - T.1. - P. 255-258 [in Russian].
7. Matrenin P.V. Methods of stochastic optimization [Text]: study guide / P.V. Matrenin, M.G. Grief, V.G. Sekaev. - Novosibirsk: NSTU, 2016. – 65 p. [in Russian].
8. Orlovskaya N.M. Analysis of bioinspired methods of global optimization // Proceedings of the MAI. - 2014. - №73. URL: https://mai.ru/upload/iblock/85d/85d945530545d38e6d4cbd591417766d.pdf (access date: 04/17/2019) [in Russian].
9. Sagun A.V., Khaidurov V.V., Kunchenko-Kharchenko V.I. The method of searching for optimal paths / A.V. Sagun, V.V. Khaidurov, V.I. Kunchenko-Kharchenko // Physics and Mathematics Journal of Science. - 2017. - №2 (12). - P. 135-139 [in Russian].
10. Sergienko A. B. Test functions for global optimization / A. B. Sergienko. - Krasnoyarsk: Publishing House of SSAU them. Mf Reshetnev, 2015. – 112 p. [in Russian].
11. Chastikova V.A., Druzhinina M.A., Kekalo A.S. Investigation of the effectiveness of the algorithm for searching fish schools // Modern problems of science and education. - 2014. - №4. - URL: http://science-education.ru/ru/article/view?id=14142 (access date: 04.17.2019) [in Russian].
12. Bossek, Y. SMOOF: single- and multi-purpose test optimization functions / Yu. Bossek // R. Magazine - 2017. - Tom. 9 (1). - P. 103-113.
13. Mirdzhalili S., Lewis A. Gray wolf Optimizer / S. Mirdzhali, A. Lewis // Advanced level in engineering software. - 2014. - Volume 69. - From 46-61.
14. Molga M., Smutnitsky S. Test functions for optimization needs: [Electronic resource]. 2005. URL: https://www.vafaeijahan.com/en/wp-content/uploads/2012/02/Test-functions-for-optimization-needs.pdf (Revised: 04/17/2019).
15. Zong W.G., Williams J.C. Environmental Optimization Using Harmony Search / W.G. Zong, J.C. Williams // American Conference on Applied Mathematics. - 2008. – P. 148-152.