УДК 681.3

ИСПОЛЬЗОВАНИЕ МЕТОДА РОЯ ЧАСТИЦ ДЛЯ ФОРМИРОВАНИЯ СОСТАВА МУЛЬТИВЕРСИОННОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Соловьев Е.В.
научный руководитель док. техн. наук Ковалев И.В.
Сибирский Федеральный Университет Институт Космических и Информационных Технологий

В данной статье рассматривается метод роя частиц (МРЧ) и возможность его применения для задачи формирования мультиверсионного программного обеспечения.

Метод роя частиц (particle swarm optimization) — метод численной оптимизации, для использования которого не требуется знать точного градиента оптимизируемой функции. Данный метод относится к группе алгоритмов роевого интеллекта (swarm intelligence), которые описывают коллективное поведение децентрализованной самоорганизующейся системы. Системы роевого интеллекта, как правило, состоят из множества агентов локально взаимодействующих между собой и с окружающей средой. Сами агенты обычно довольно просты, но все вместе, локально взаимодействуя, создают так называемый роевой интеллект.

Впервые метод был создан для симуляции социального поведения стаи птиц и косяков рыб. В результате развития метода его начали успешно применять к задачам нахождения экстремумов функции.

Рассмотрим работу алгоритма. Для примера возьмем область двухмерного пространства. В допустимой области случайным образом располагаются частицы, а также задаются векторы и скорости их движения. На каждой итерации алгоритма мы рассчитываем направление и скорость движения частицы по всем возможным пространствам. Соответственно в двухмерном пространстве мы рассчитываем направление и скорость движения частицы по оси X и по оси Y. Затем для каждой частицы мы выполняем следующие действия:

1. Вычисляем значение оптимизируемой функции в текущей точке. Если значение функции $f(x, y)$ больше персонального лучшего значения (ПЛЗ) частицы, то мы заменяем его ПЛЗ = $f(x, y)$. Затем мы проверяем глобальное лучшее значение (ГЛЗ) функции, если оно меньше чем $f(x, y)$, то мы заменяем и его ГЛЗ = $f(x, y)$. Мы также храним и координаты ГЛЗ, а каждая частица хранит координаты свое ПЛЗ;
2. Меняем скорость и направление движения частиц. Нужно сказать, что каждая частица стремится к двум точкам: ПЛЗ и ГЛЗ. В следующем уравнении это наглядно показано:

$$v_n^{i+1} = v_n^i + c_1 \cdot R \cdot (p_n - x_n) + c_2 \cdot R \cdot (g_n - x_n),$$

как видно из этой функции v_n^{i+1} — скорость частицы в плоскости n на итерации $i+1$ складывается из следующих величин:
v_n^i — скорость частицы в n-той плоскости на шаге i;
c_1 и c_2 — две положительные константы которые отображают силу влияния ПЛЗ и ГЛЗ функции;
R — случайное значение в интервале $[0, 1]$ по информации в статье [2] и $[-1, 1]$ по информации в статье [3]. Если мы берем значение во втором интервале, то частица способна двигаться абсолютно хаотично, также нужно отметить что берется два отдельных случайных значения для ПЛЗ и ГЛЗ;

p_n — координата в плоскости n персонального лучшего значения;
g_n — координата в плоскости n глобального лучшего значения;
x_n — текущая координата частицы в плоскости n.

Надо отметить что существует несколько методов для решения ситуации, когда частица достигает границы области. Например, можно отражать частицу обратно в область возможных решений с сохранение скорости, либо гасить скорость частицы при соприкосновении с границей области, не давая частице покинуть её. Также можно просто не учитывать данные частицы покинувшей область возможных решений и дожидаться её возвращения в эту область.

Методология создания мультиверсионного программного обеспечения (ПО) строится на двух основных принципах:

1. ПО состоит из модулей;
2. В каждом модуле есть несколько независимых версий, которые мы можем комбинировать.

Основной проблемой создания подобного ПО является подбор оптимального состава версий каждого модуля, чтобы удовлетворить заранее заданные ограничения и минимизировать или максимизировать определенные параметры. Для этой задачи можно использовать метод роя частиц.

Количество измерений будет равняться количеству модулей данного ПО, а область допустимых значений на конкретном измерении будет равна $[0, N_m]$, где N_m — количество версий в модуле m. Значения оптимизируемых функций будут равняться конкретным параметрам (надежность, стоимость, время ответа и т.д.), которые мы получаем в результате их вычисления, основываясь на положении конкретной частицы.

Основной проблемой является то, что при создании мультиверсионного ПО, мы должны оптимизировать его состав по нескольким параметрам, таким как надежность, стоимость, время ответа, время работы без сбоев и т.п. Так как же мы можем определить какое значение лучше, а какое хуже при сравнении данных различных частиц? Перед нами стоит задача многокритериальной оптимизации. Решить ее можно несколькими способами. Самым простым из них является, использование критерия оптимальности по Парето.

Оптимальность по Парето — такое состояние системы, при котором значение каждого частного критерия, описывающего состояние системы, не может быть улучшено без ухудшения положения других элементов.

Таким образом, по словам самого Парето: «Всякое изменение, которое никому не приносит убытков, а некоторым людям приносит пользу (по их собственной оценке), является улучшением». Значит, признаётся право на все изменения, которые не приносят никому дополнительного вреда.

Множество состояний системы, оптимальных по Парето, называют «множеством Парето» или «множеством альтернатив, оптимальных в смысле Парето».

Ситуация, когда достигнута эффективность по Парето — это ситуация, когда все выгоды от обмена исчерпаны.

То есть мы будем считать, что значение в точке x_1 лучше чем в точке x_2 только в том случае если все параметры, по крайней мере не хуже, а один из них превосходит аналогичный параметр в точке x_2.

Каждый может выбрать свой критерий для остановки работы метода роя частиц для данной задачи. Например, когда в течении нескольких итераций не происходит изменения ГЛЗ, мы можем остановить работу алгоритма и принять текущее значение ГЛЗ, как оптимальное решение данной задачи.

Метод роя частиц является довольно молодым методом в семействе биоинспиративных алгоритмов. По сравнению с генетическим алгоритмом, операторы которого могут быть реализованы различным образом, метод роя пчел имеет лишь один оператор — вычисление скорости, что делает его более быстрым, а также в методе роя пчёл можно легко определить достижение точки глобального минимума.

Список литературы

1. Kennedy, J., Eberhart, R. C. Particle Swarm Optimization. — Proceedings of IEEE International Conference on Neural Networks IV. – 1995. - С. 1942-1948.
2. Eberhart, R. C., Shi, Y. A modified particle swarm optimizer. — Proceedings of IEEE International Conference on Evolutionary Computation. – 1998 - С. 69-73.
3. Естественные алгоритмы. Алгоритм поведения роя пчёл. — http://habrahabr.ru/post/104055/ - 2010.
4. Р.Ю. Царев Мультиверсионный подход к повышению отказоустойчивости программного обеспечения систем управления и обработки информации. — В мире научных открытий. - 2010. - № 4 (10). - Ч. 10. - С. 82-84.