Наука и Образование
МГТУ им. Н.Э. Баумана
Сетевое научное издание
ISSN 1994-0408

Наука и Образование. МГТУ им. Н.Э. Баумана. Электрон. журн. 2016. № 09. С. 92–110.

DOI: **10.7463/0916.0844601**

Представлена в редакцию: 08.08.2016
Исправлена: 22.08.2016

УДК 519.6

Мета-модельный метод перманентной настройки параметров алгоритмов оптимизации

Агасиев Т. А.[1,*], **Карпенко А. П.**[1]

*taaalex@mail.ru

[1]МГТУ им. Н.Э. Баумана, Москва, Россия

Поставлена задача настройки параметров с использованием мета-модели. Предложен метод перманентной настройки параметров алгоритмов оптимизации, основной особенностью которого является использование суррогатной модели показателя эффективности настраиваемого алгоритма. Представлена программная система META_OPT, реализующая предложенный метод. Выполнен вычислительный эксперимент по исследованию эффективности разработанного алгоритмического и программного обеспечения. Результаты эксперимента показывают, что параметры алгоритма, полученные в результате его настройки, позволяют до 50% повысить его эффективность.

Ключевые слова: глобальная оптимизация, мета-оптимизация, перманентная настройка параметров, суррогатная модель, мета-модель

Введение

Большинство алгоритмов оптимизации имеют свободные параметры, значения которых в значительной мере определяют эффективность этих алгоритмов [1]. Разработчики алгоритма предоставляют пользователю рекомендуемые значения параметров из расчета на стабильное поведение алгоритма при решении широкого спектра оптимизационных задач. В действительности, эти значения не универсальны и, вообще говоря, должны подбираться с учетом особенностей решаемых задач. Автоматизация процесса поиска оптимальных значений этих параметров позволяет пользователю раскрыть потенциал алгоритма оптимизации без помощи специалистов в данной области прикладной математики.

Используем далее следующие основные понятия: исходная задача оптимизации – *базовая задача*; рассматриваемый алгоритм ее решения – *базовый алгоритм*; вектор варьируемых свободных параметров этого алгоритма – его *стратегия*; задача поиска оптимальной стратегии – *задача мета-оптимизации*; алгоритм решения задачи мета-оптимизации – *мета-алгоритм*; программная система, реализующая тот или иной метод мета-оптимизации – *мета-оптимизатор* (*настройщик*).

Существуют различные способы классификации методов мета-оптимизации. За основу берем классификацию, предложенную Эйбеном (*A.E. Eiben*) в работе [1]. В соответствии с этой классификацией на верхнем уровне иерархии выделяют две группы таких методов – методы *настройки* параметров и методы *управления* параметрами. Методы настройки параметров предполагают предварительное отыскание оптимальной стратегии для решения задач некоторого *класса*. В этом случае *стратегия* базового алгоритма не меняется в ходе решения им базовой задачи. Методы управления параметрами предполагают варьирование стратегии в процессе решения базовой задачи с целью отыскания стратегии, которая является оптимальной для данной задачи.

Различаем методы *однократной* и *перманентной* настройки. Методы однократной настройки параметров представляют собой классические методы мета-оптимизации. В данном случае из числа задач рассматриваемого класса формируют тестовый набор задач, для решения которых отыскивают оптимальные стратегии. Найденные стратегии используют в дальнейшем для решения всех задач рассматриваемого класса. В случае перманентной настройки мета-оптимизатор накапливает информацию об эффективности стратегий в процессе нормальной эксплуатации базового алгоритма.

Можно выделить две основные цели мета-оптимизации [2]. Первой является нахождение оптимальной стратегии алгоритма, с которой его производительность максимальна при решении задач рассматриваемого класса. Эта цель предполагает, что приоритетом пользователя является *интенсификация* поиска в процессе мета-оптимизации. Вторая цель – получение информации о свойствах базового алгоритма при различных значениях его свободных параметров для более глубокого понимания этого алгоритма и изучения устойчивости его стратегий к вариациям решаемых оптимизационных задач. Другими словами, в данном случае приоритетом пользователя является *диверсификация* мета-поиска. Большинство современных методов настройки параметров тем или иным образом комбинируют принципы, обеспечивающие диверсификацию и интенсификацию поиска оптимальных стратегий.

Наиболее известны четыре следующие классы методов настройки параметров [2].

Стохастические методы «заточены» под быстрое отыскание эффективных стратегий. Диверсификационная составляющая у данных методов выражена слабо.

Методы выборки позволяют провести обширное исследование пространства стратегий базового алгоритма. Методы этого класса просты и не требуют значительных вычислительных затрат, что оправдывает относительно низкую точность настройки, которые они обеспечивают.

Скрининговые методы основаны на уменьшении числа обращений к базовому алгоритму путем выявления наиболее перспективных стратегий.

Мета-модельные методы используют суррогатную модель показателя эффективности стратегий (ПЭС). Эти методы позволяют уменьшить общее число запусков базового алгоритма в процессе настройки путем использования оценок ПЭС, полученных при помощи суррогатной модели ПЭС. Суррогатная модель уточняется после каждой итерации

мета-оптимизации и позволяет после достаточно большого числа итераций получить адекватное представление о пространстве стратегий.

Работа в значительной мере вдохновлена работой [3, 4], в которой рассмотрен способ организации процесса настройки параметров, предполагающий централизованный сбор информации о пространстве стратегий базового алгоритма на основе трехзвенной клиент-серверной архитектуры. При этом на клиентской стороне работает пользовательская программа, которая решает базовые задачи (с помощью базового алгоритма). Перед началом процесса решения задачи клиентская программа отправляет серверу вектор характерных признаков (ВХП) решаемой задачи (размерность задачи, константа Липшица, число локальных экстремумов и т.д.). На этой основе сервер определяет класс задачи, которому принадлежит данная задача, и выбирает оптимальную стратегию для решения задачи этого класса (базовым алгоритмом). Для поиска оптимальной стратегии на сервере используется генетический алгоритм. Кластеризация задач на сервере производится при помощи самоорганизующейся карты Кохонена на основе ВХП задач. Недостатки указанного подхода заключаются в следующем:

- от пользователя необходима информация о ВХП базовой задачи;
- использование клиент-серверной архитектуры требует стабильного подключения к серверу приложений и увеличивает время отклика мета-оптимизатора;
- вся вычислительная нагрузка мета-оптимизации ложится на сервер приложений, что предполагает его высокую надежность, повышает стоимость оборудования и затраты на поддержку работоспособности.

Целями работы являются:

- разработать мета-модельный метод перманентной настройки параметров, свободный от большей части недостатков метода, предложенного в работе [3, 4];
- реализовать предложенный метод в рамках программной системы автоматизированной перманентной настройки параметров META_OPT;
- исследовать эффективность разработанного алгоритмического и программного обеспечения.

1. Постановка задачи настройки параметров

Рассматриваем класс Φ задач глобальной безусловной оптимизации целевой функции $G(X)$ в n-мерном евклидовом пространстве R^n:

$$\min_{X \in R^n} G(X) = G(X^*).$$

Здесь X^*– искомое оптимальное решение задачи. Полагаем, что класс задач Φ включает в себя N_ϕ задач, то есть

$$\Phi = \{\phi(C_i) = \phi_i, C_i \in D_C;\ i \in [1{:}N_\phi]\},$$

где ϕ_i – i-ая задача класса; C_i – вектор значений характерных признаков задачи ϕ_i; D_C – множество допустимых значений компонентов ВХП [3, 4].

Обозначаем $a(B)$ базовый алгоритм оптимизации, используемый для решения задач класса Φ, где $B = (b_1, b_2, \ldots, b_N)$ – стратегия этого алгоритма. Набор допустимых стратегий B образует множество

$$D_B = \{b_i \mid b_i^- \le b_i \le b_i^+, i \in [1{:}N]\},$$

где b_i^-, b_i^+ – интервалы допустимых значений компоненты b_i вектора B. Множество D_B определяет совокупность алгоритмов оптимизации

$$A = \{a(B) | B \in D_B\}.$$

Показатель эффективности алгоритма $a(B)$ (показатель эффективности стратегии B) при решении задач класса Φ обозначаем $\mu(\Phi, a(B))$.

Задача настройки параметров ставится следующим образом. Найти такую стратегию B^* алгоритма оптимизации $a(B) \in A$, которая обеспечит минимальное значение ПЭС $\mu(\cdot)$ при решении задач класса Φ:

$$min_{B \in D_B}\, \mu(\Phi, a(B)) = \mu(\Phi, a(B^*)). \quad (1)$$

Для построения ПЭС могут быть использованы различные подходы [2]. Функция $\mu(\cdot)$ может, например, формализовать качество решений задач класса Φ, которые найдены базовым алгоритмом со стратегией B^*. Эта функция может также определяться вычислительными затратами базового алгоритма с этой стратегией при решении задач того же класса задач. Таким образом, вообще говоря, ПЭС представляет собой векторную функцию, а задача мета-оптимизации – задачу многокритериальной оптимизации.

Предлагаемый метод перманентной настройки использует суррогатную модель ПЭС $\hat{\mu}(\Phi, a(B))$. В этих обозначениях задача настройки параметров формулируется как задача отыскания глобального минимума суррогатной функции $\hat{\mu}(\Phi, a(B))$:

$$min_{B \in D_B}\, \hat{\mu}(\Phi, a(B)) = \hat{\mu}(\Phi, a(B^*)). \quad (2)$$

2. Предлагаемый метод перманентной настройки

Схема предлагаемого мета-модельного метода перманентной настройки имеет следующий вид (рисунок 1).

1) Базовый алгоритм отправляет мета-оптимизатору исходные значения параметров B алгоритма a, вектор характерных признаков класса Φ_i, которому принадлежит базовая задача, и другую информацию (см. ниже).

2) На основе полученной информации мета-оптимизатор определяет оптимальную стратегию B^* для задач класса Φ_i и предает эту стратегию базовому алгоритму.

3) Используя стратегию B^*, базовый алгоритм решает базовую задачу, вычисляет значение ПЭС $\mu(\Phi_i, a(B^*))$ и передает это значение мета-оптимизатору.

4) Мета-оптимизатор принимает значение величины $\mu(\Phi_i, a(B^*))$ и сохраняет его в своей базе данных.

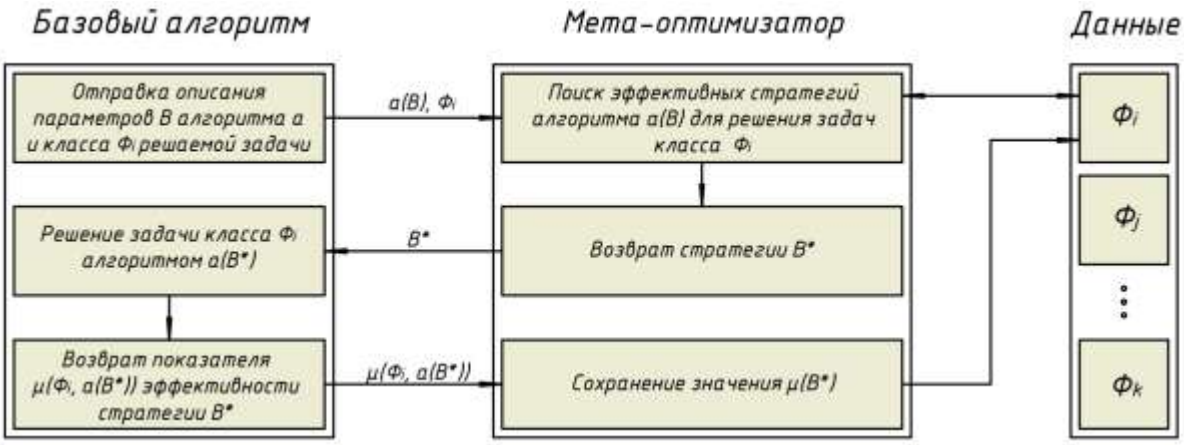


Рисунок 1 – Схема предложенного метода перманентной настройки

«Разгон» метода. Если Φ является новым классом задач (для которого не проводилась настройка параметров базового алгоритма), производится предварительное исследование пространства стратегий (разведка) с целью формирования стартовой обучающей выборки $(B_i, \mu_i), i \in [1{:}L]$. На основе этой выборки строим первую суррогатную модель $\hat{\mu}(\Phi, a(B))$, с помощью которой отыскиваем перспективную стратегию B^* алгоритма a для данного класса задач. После получения от базового алгоритма значения функции $\mu(\Phi, a(B^*))$, уточняем модель $\hat{\mu}(\Phi, a(B))$ и так далее.

В текущей программной реализации мета-оптимизатора (п.3) на этапе «разгона» используем алгоритм локальной унимодальной выборки (*LUS*), в котором исследование пространства поиска происходит по всем его измерениям одновременно [5]. Переход в новое состояние B' из текущего B_r осуществляем по формуле

$$B' = B_r + V,$$

где V – N-мерный случайный вектор, компоненты которого равномерно распределены в заданном интервале:

$$V = U(-\Delta, \Delta).$$

Здесь Δ – шаг дискретизации, в начале поиска равный $\Delta = B^+ - B^-$, где $B^+ = (b_1^+, b_2^+, \ldots, b_N^+)$, $B^- = (b_1^-, b_2^-, \ldots, b_N^-)$.

Если значение ПЭС не удается уменьшить при текущей величине шага дискретизации Δ, то он уменьшается для всех измерений одновременно по формуле

$$\Delta = q\,\Delta, q = \sqrt[\frac{N}{\alpha}]{\frac{1}{2}} = 2^{-\frac{\alpha}{N}}.$$

Здесь коэффициент $\alpha \in (0; 1)$ позволяет контролировать скорость изменения шага дискретизации.

Заметим, что по окончании процесса «разгона» получается как эффективная стратегия для решения базовых задач рассматриваемого класса, так и общее представление о пространстве стратегий базового алгоритма.

Этап «разгона» предполагает использование базовым алгоритмом не самых лучших стратегий. Поэтому желательно уменьшить число этих стратегий. Мы предлагаем с этой целью использовать суррогатную модель ПЭС класса базовых задач, который уже имеется в базе данных мета-оптимизатора и который в некотором смысле близок рассматриваемому новому классу. Другими словами, если новым классом является класс $\Phi' = \Phi(C')$, то мета-оптимизатор отыскивает в своей базе данных наиболее «близкий» класс $\Phi'' = \Phi(C'')$ и использует в качестве суррогатной модели $\hat{\mu}(\Phi', a(B))$ функцию $\hat{\mu}(\Phi'', a(B))$.

В общей постановке оценка близости классов Φ', Φ'' может быть выполнена на основе двух следующих подходов:

- путем оценки близости ВХП указанных классов, когда в качестве класса Φ'' используется тот класс, для которого некоторая векторная норма $\|C' - C''\|$ минимальна;

- путем оценки близости предсказанных моделью $\hat{\mu}(\Phi'', a(B))$ значений ПЭС и значений ПЭС $\mu(\Phi', a(B))$, вычисленных для задач нового класса Φ'.

В предлагаемом методе перманентной настройки используется комбинация этих подходов: на основе оценки близости ВХП классов задач выделяем путем полного перебора некоторое число классов-кандидатов; для каждого из этих классов вычисляем оценку точности предсказания и на этой основе выбираем лучший класс.

Построение суррогатной модели ПЭС. Основными являются следующие требования к суррогатной модели ПЭС.

а) Поскольку функция ПЭС может быть сильно зашумленной, для ее аппроксимации необходима модель с хорошими сглаживающими свойствами.

б) Большая часть свободных параметров базового алгоритма оптимизации являются непрерывными или дискретными, но эти параметры могут быть и категориальными [6] (например, топология соседства частиц в методе роя частиц [7]). Поэтому суррогатная модель должна позволять использование категориальных параметров [8].

в) Для того чтобы накладные расходы на мета-оптимизацию были приемлемы, вычислительная сложность построения суррогатной модели должна быть относительно невысокой.

Отметим, что при выборе суррогатной модели следует учитывать также то обстоятельство, что тип этой модели в значительной мере определят метод оптимизации, который может использоваться в процессе поиска глобально оптимальной стратегии B^* и обеспечивать при этом достаточно высокую эффективность поиска.

Перечисленным выше требованиям к суррогатной модели в наибольшей мере удовлетворяют методы регрессии на основе деревьев принятия решений. К недостаткам этих методов можно отнести ступенчатый вид получающейся модели. Наибольшее распространение получили два метода указанного типа – градиентно-усиленные деревья (*Gradient-Boosted Trees*, *GBT*) [9] и случайный лес (*Random Forest, RF*) [10]. В мета-оптимизаторе для построения суррогатной модели ПЭС используем метод *RF*, который является одним из наиболее широко применяемых методов машинного обучения для решения задач классификации и регрессии. Важно, что метод *RF* может обрабатывать кате-

гориальные параметры и не требует масштабирования переменных аппроксимируемой функции.

Поиск оптимальной стратегии базового алгоритма. После построения модели ПЭС $\hat{\mu}(\Phi, a(B))$ поиск оптимальной стратегии B^* сводится к решению задачи глобальной оптимизации (2). Вообще говоря, для решения этой задачи может быть использован любой из алгоритмов глобальной оптимизации. Отметим, что применение суррогатной модели позволяет сгладить зашумленную функцию $\mu(\Phi, a(B))$ и уменьшить вычислительные затраты на расчет значения ПЭС. Тем самым процесс решение задачи (2) оказывается значительно проще процесса решения исходной задачи (1).

Мета-оптимизатор использует для решения задачи (2) генетический алгоритм с вещественным кодированием хромосом, что повышает скорость вычислений (за счет отказа от перекодирования) и усиливает интенсификационную составляющую поиска. В качестве оператора мутации использован оператор неравномерной мутации Михалевича [11]. Отбор особей в новую популяцию осуществляем с помощью метода на основе *элитизма* [12], который позволяет избежать потери лучших особей текущей популяции, что опять же интенсифицирует поиск. С целью повышения вероятности локализации глобального минимума функции $\hat{\mu}(\Phi, a(B))$, часть новой популяции выбираем случайным образом. Для выбора скрещиваемой родительской пары используем селекцию на основе генотипа – *инбридинг* [13]. Для оценки близости генотипов особей применяем евклидову норму. В качестве оператора скрещивания (кроссинговера) используем *SBX*-кроссинговер [14].

3. Программная реализация метода

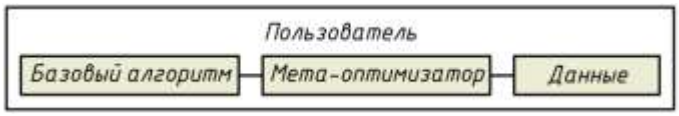
Для хранения данных программная система META_OPT использует встроенную файловую базу данных на основе реляционной СУБД *SQLite*, для взаимодействия с которой имеются библиотеки, поддерживающие различные языки программирования [15]. Библиотеку взаимодействия компонуем вместе с программой, что уменьшает накладные расходы на обмен данными и время отклика системы.

Для разработки META_OPT выбран язык программирования *Java*. Взаимодействие с базой данных происходит при помощи *JDBC* драйвера, предоставляемого в виде программной библиотеки. Разработка программной системы выполнена с использованием операционной системы *Windows* 8. В качестве интегрированной среды разработки использована среда *Intellij Idea* 15.

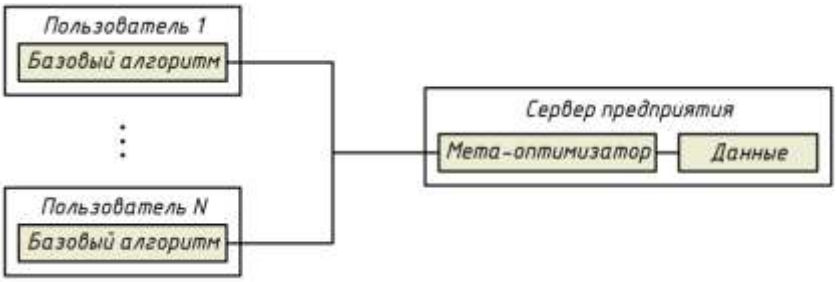
Для обеспечения совместимости системы META_OPT с другими СУБД имеется возможность замены модулей системы, которые обеспечивают ее работу с базой данных. Аналогично имеется возможность замены модулей, которые решают задачу глобальной оптимизации (2), то есть реализуют поиск оптимальной стратегии.

Система META_OPT предоставляет различные варианты применения (рисунок 2). Одиночные пользователи имеют возможность использовать систему в виде программной библиотеки с локальной базой данных (рисунок 2, а). Для корпоративных пользователей

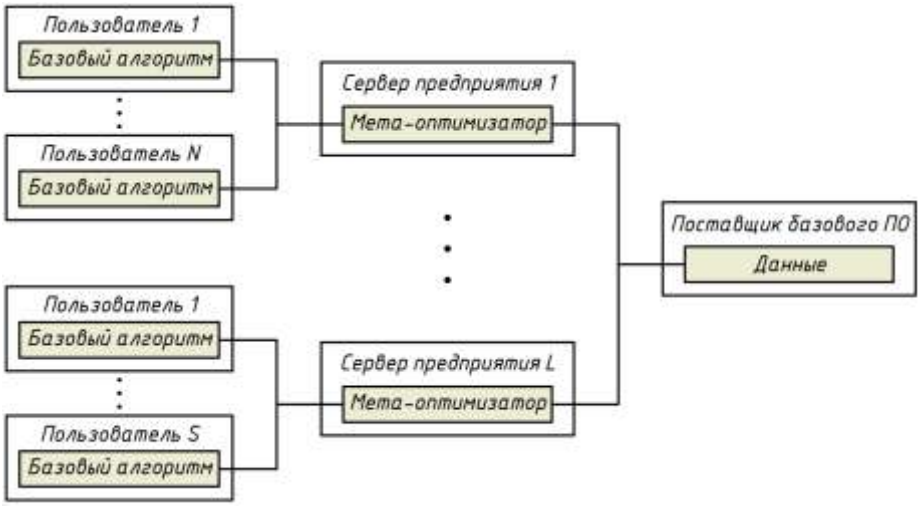
(рисунок 2, б) алгоритм мета-оптимизации можно запускать как на сервере, так и распределенно (на клиентских машинах). Вариант, в котором мета-оптимизатор предоставляет пользователям компания-поставщик базового программного обеспечения, иллюстрирует рисунок 2, в.



а) одиночный пользователь



б) корпоративный пользователь



в) плоставщкик базового программного обеспечения

Рисунок 2 – Возможные схемы организации процесса настройки

Графический интерфейс пользователя системы META_OPT представлен на рисунке 3. С его помощью пользователь указывает уникальное в системе название базового алгоритма, уникальное имя класса задач и особенности базовой задачи. Для формирования ВХП решаемой базовой задачи, пользователь выбирает из числа доступных признаков известные ему и указывает значения этих признаков. Также пользователь задает описания свободных параметров базового алгоритма: имя, текущее (рекомендуемое) значение и область допустимых значений. Для числовых параметров следует указать минимальное и максимальное значения, для категориальных параметров - допустимые значения.

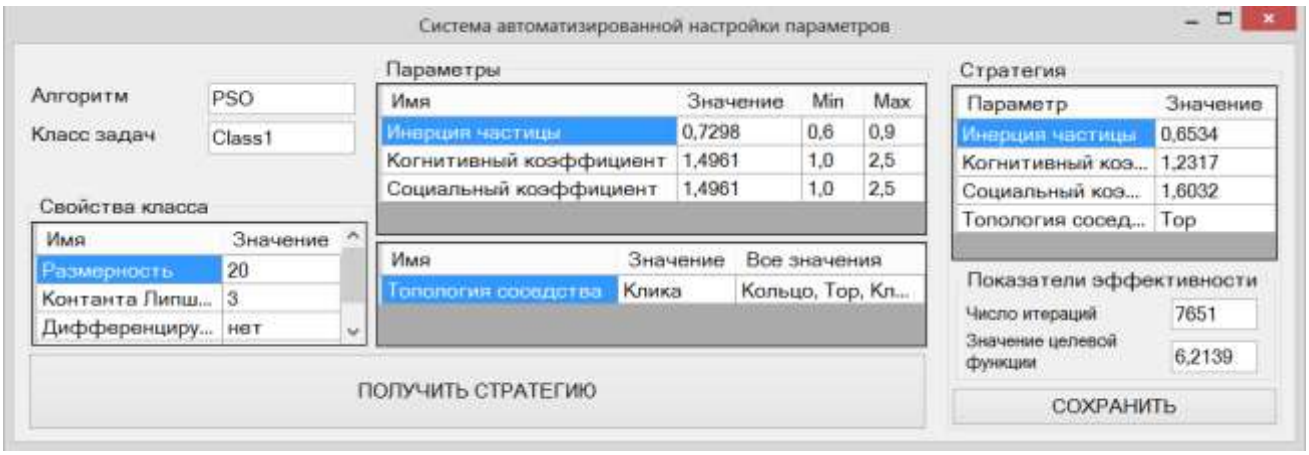


Рисунок 3 – Графический интерфейс пользователя системы META_OPT: базовый алгоритм – алгоритм роя частиц

4. Вычислительный эксперимент

Базовый алгоритм. В качестве базового алгоритма использован алгоритм роя частиц *PSO* [16]. Рой частиц обозначаем $P = \{P_i, i\epsilon[1{:}N_P]\}$, где N_P – число частиц в рое. Положение и «скорость» частицы P_i в момент времени $t = 0, 1, 2 \ldots$ определяют векторы $X_{i,t} = (x_{i,t,1}, \ldots, x_{i,t,n})$, $V_{i,t} = (v_{i,t,1}, \ldots, v_{i,t,n})$ соответственно. В начальный момент времени для каждой частицы P_i заданы ее координаты $X_{i,0} = X_i^0$ и скорость $V_{i,0} = V_i^0$.

Итерации в каноническом алгоритме PSO выполняются по правилам

$$X_{i,t+1} = X_{i,t} + V_{i,t+1},$$

$$V_{i,t+1} = \alpha V_{i,t} + U[0,\beta] \times (X_{i,t}^b - X_{i,t}) + U[0,\gamma] \times (X_{g,t} - X_{i,t}),$$

где α, β, γ – свободные параметры алгоритма; $U[a,b]$ – n-мерный вектор случайных чисел, распределенных по равномерному закону в интервале $[a,b]$; $\times$ – символ покомпонентного умножения векторов; $X_{i,t}^b$ – координаты частицы P_i с лучшим значением целевой функции $G(X)$; $X_{g,t}$ – координаты соседа частицы P_i с лучшим значением той же функции. Рекомендуемые значения свободных параметров равны $\alpha = 0{,}72984, \beta, \gamma = 1{,}49618$.

Хорошо известно, что эффективность алгоритма PSO в значительной степени зависит от используемой роем топологии соседства частиц. Эту топологию определяет неориентированный граф, вершины которого соответствуют частицам, а ребра связывают состоящие в соседстве частицы. Соответствующий свободный параметр алгоритма PSO обозначаем T. Наиболее часто используют следующие значения этого параметра: топология "клика" (полный граф); топология "кольцо"; топология "двумерный тор" (фон Неймана); кластерная топология [7].

Тестовые классы задач. В вычислительном эксперименте для формирования классов базовых задач использованы следующие тестовые функции.

Сферическая функция ($G(X^*) = 0, X^* = (1, \ldots, 1)$):

$$G(X) = \sum_{i=1}^{n} x_i^2;\ n = 90.$$

Овражная унимодальная функции Розенброка ($G(X^*) = 0, X^* = (1, \ldots, 1)$):

$$G(X) = \sum_{i=1}^{n-1}((x_i - 1)^2 + 100(x_i^2 - x_{i+1})^2);\ n = 20.$$

Мультимодальная функция Растригина ($G(X^*) = 0, X^* = (0, \dots, 0)$:

$$G(X) = 10\,n\ \sum_{i=1}^{n}(x_i^2 - 10\cos(2\pi x_i));\ n = 50.$$

Мультимодальная функция Экли ($G(X^*) = 0, X^* = (0, \dots, 0)$:

$$G(X) = -20e^{f_1(X)} - e^{f_2(X)} + 20 + e;$$

$$f_1(X) = -0{,}2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\,, f_2(X) = \frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i);\ n = 30.$$

Из указанных тестовых функций было сформировано 10 классов задач (таблица 1), включающих в себя различное число тестовых функций.

Таблица 1 – Классы пользовательских задач

№ класса	1	2	3	4	5	6	7	8	9	10
Задачи класса	1	2	3	4	1, 3	2, 4	3, 4	1, 3, 4	1, 2, 4	1,2,3,4

Организация вычислительного эксперимента. Размер популяции алгоритма PSO принят равным 50. В качестве условия окончания итераций базового алгоритма применено условие достижения заданного числа итераций (равного 5000). Суммарно выполнено около 100 000 запусков алгоритма PSO. Использованы следующие ПЭС:

- μ_1 – расстояние до глобального минимума $|G(X^*) - G(X')|$, где X' – лучшее найденное решение за 1000 итераций алгоритма PSO;
- μ_2 – число итераций алгоритма до локализации глобального минимума целевой функции с точностью $|G(X^*) - G(X')| < 0{,}1$.

Для анализа эффективности системы META_OPT, указанные оптимизационные задачи также решались базовым алгоритмом PSO с рекомендуемыми значениями параметров $\alpha = 0{,}72984, \beta = \gamma = 1{,}49618, T-$ клика (таблица 2).

Для обработки результатов вычислительного эксперимента использован интерпретируемый язык программирования высокого уровня *Python*, для визуализации результатов исследования - библиотека (модуль-пакет для *Python*) *Matplotlib*, которая позволяет быстро создавать высококачественные рисунки различных форматов [17].

Таблица 2 – Настраиваемые параметры базового алгоритма PSO

Имя параметра	**Рекомендуемое значение параметра**	**Область допустимых значений параметра**
α	0,7298	[0,6; 0,9]
β	1,4961	[1,0; 2,5]
γ	1,4961	[1,0; 2,5]
T	клика	клика, кольцо, двумерный тор, кластерная

Результаты эксперимента и их обсуждение. Результаты вычислительного эксперимента представлены в таблицах 3, 4, которые иллюстрируют рисунки 4, 5 соответственно. Приняты следующие обозначения: $\mu_i', i = 1,2$ - значение ПЭС μ_i, полученное при использовании алгоритма PSO с рекомендуемой стратегией; $\bar{\mu}_i, \bar{\mu}_i'$ – соответствующие усредненные значения ПЭС по 10 запускам базового алгоритма с данной стратегией.

Таблица 3 – Результаты настройки: ПЭС μ_1

№ класса	α	β	γ	T	$\bar{\mu}_1$	$\bar{\mu}_1'$
1	0,713	2,086	1,145	клика	6,5	89,3
2	0,744	1,145	1,395	тор	7,5	23,2
3	0,740	2,109	1,446	кольцо	13,4	18,6
4	0,699	1,845	1,698	кольцо	20,4	42,4
5	0,513	2,046	1,721	клика	37,8	57,1
6	0,513	2,097	1,702	клика	31,5	66,3
7	0,671	2,390	1,192	тор	15,3	30,5
8	0,774	1,959	1,094	тор	21,1	54,6
9	0,679	1,714	1,566	клика	24,3	64,2
10	0,634	1,067	1,074	клика	33,0	42,6

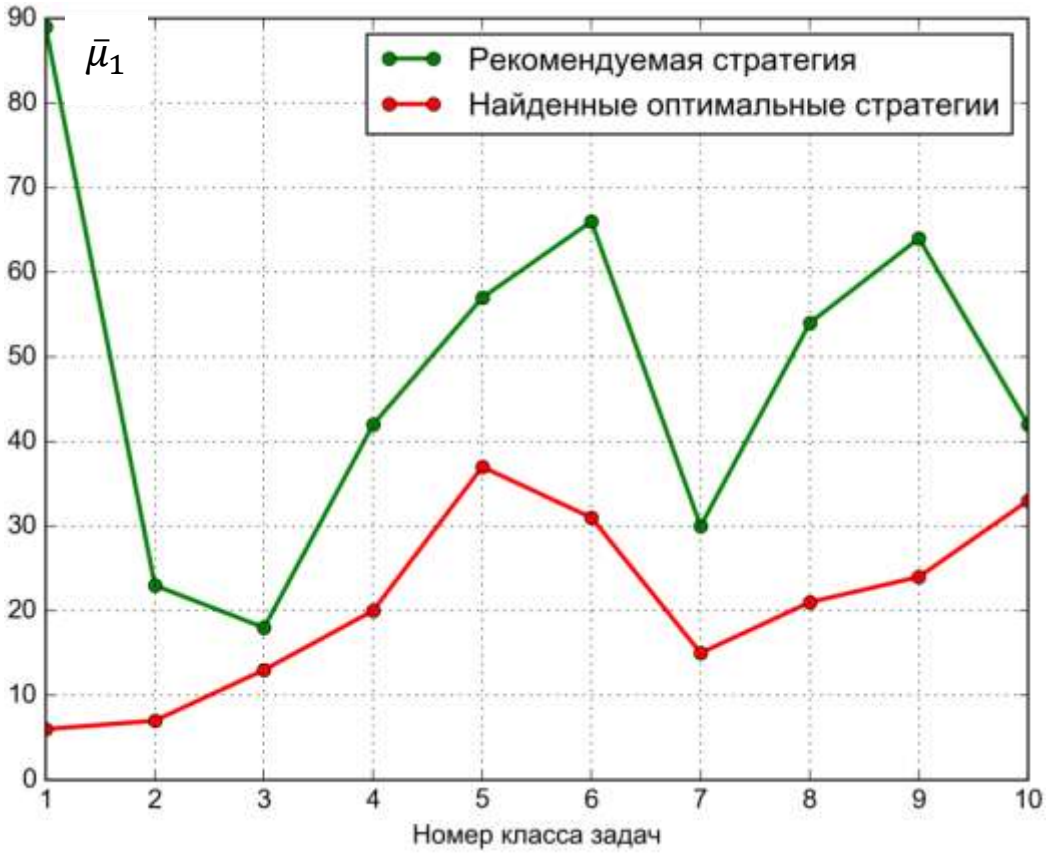


Рисунок 4 – Результаты настройки: ПЭС μ_1

Таблица 4 – Результаты настройки: ПЭС μ_2

№ класса	α	β	γ	T	$\bar{\mu}_2$	$\bar{\mu}_2'$
1	0,522	1,818	1,915	клика	2278,8	3235,9
2	0,655	1,040	1,807	клика	2818,4	3912,3
3	0,670	2,401	1,293	тор	1890,6	3439,9
4	0,788	1,514	1,518	кольцо	2018,9	3748,6
5	0,711	1,687	1,506	клика	2073,8	3125,9
6	0,661	1,512	1,805	кольцо	3115,3	4389,0
7	0,602	2,470	1,362	тор	2636,7	3525,4
8	0,766	1,781	1,166	клика	2910,7	3642,2
9	0,798	1,523	1,384	кольцо	3377,0	4545,4
10	0,720	1,709	1,556	клика	2796,3	4459,5

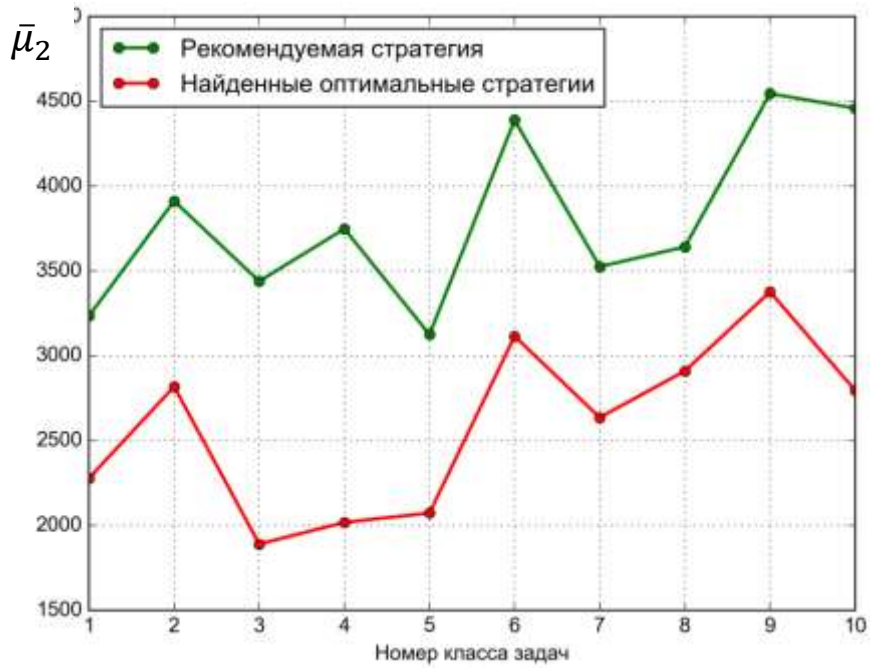


Рисунок 5 – Результаты настройки: ПЭС μ_2

Представленные результаты вычислительного эксперимента показывают, что, в среднем, после 50 итераций настройки, значения ПЭС, соответствующие найденным системой стратегиям, оказываются лучше, чем у рекомендуемой стратегии. После стагнации поиска в пространстве стратегий базового алгоритма ПЭС μ_1 оказался лучше на 30-50%, а показатель μ_2 – на 20-45%.

Обобщающие результаты исследования в пространстве рассматриваемых ПЭС представлены на рисунке 6. Рисунок показывает, что значения ПЭС, соответствующие найденным в результате настройки стратегиям базового алгоритма, расположены, в основном, ближе к началу системы координат $0\mu_1\mu_2$ по сравнению с аналогичными значениями, соответствующими рекомендованной стратегии. Это свидетельствует о том, что предложенный метод мета-оптимизации и его программная реализация обеспечивают повышение

эффективности базового алгоритма по обоим рассматриваемым показателям эффективности.

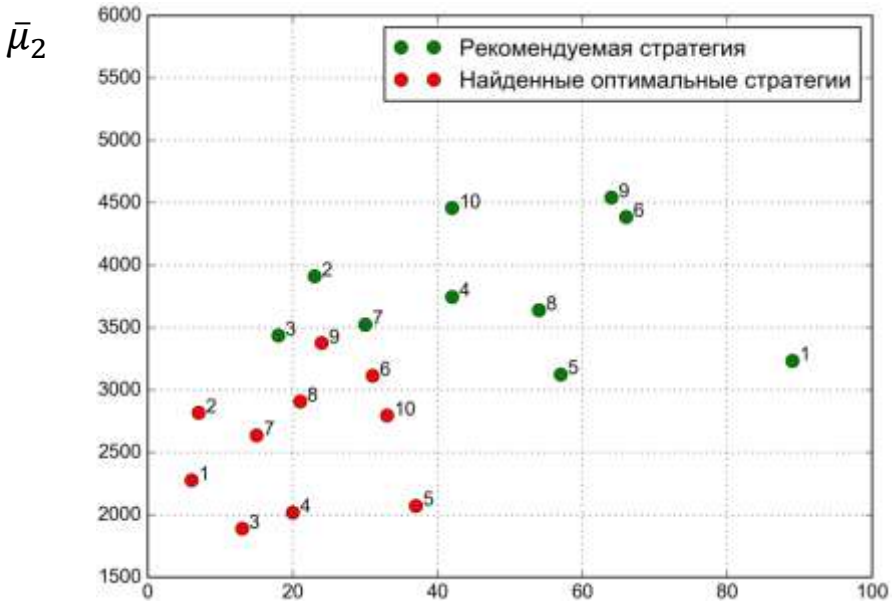


Рисунок 6 – Сводные результаты настройки базового алгоритма PSO: номера точек есть $\bar{\mu}_1$ а классов рассматриваемых задач

Заключение

Работу можно считать продолжением и развитием работы [3, 4], в которой рассмотрен способ организации процесса настройки параметров оптимизируемого алгоритма, программная реализация которого предполагает централизованный сбор информации о пространстве стратегий этого алгоритма на основе трехзвенной клиент-серверной архитектуры. Основной целью данной работы является разработка метода перманентной настройки параметров, свободного от большей части недостатков метода, предложенного в указанной публикации.

Основные особенности предложенного метода заключаются в следующем:

- аппроксимация показателей эффективности стратегий базового алгоритма с помощью суррогатных моделей этих показателей;

- ускорение процесса мета-оптимизации путем использования суррогатной модели, соответствующей классу базовых задач оптимизации, «близкого» к рассматриваемому классу.

В предложенном методе перманентной настройки для построения суррогатной модели ПЭС используем метод регрессии на основе деревьев принятия решений, точнее говоря - метод случайного леса (*Random Forest, RF*). В процессе поиска класса базовых задач, наиболее «близкого» к рассматриваемому классу, используем двухступенчатый метод на основе оценки близости ВХП классов задач и оценки точности предсказания соответствующих суррогатных моделей ПЭС. Собственно задачу мета-оптимизации решаем генетическим алгоритмом с вещественным кодированием хромосом.

Для разработки расширяемой программной системы META_OPT, реализующей предложенный метод, использованы язык программирования *Java* и реляционная СУБД *SQLite*. В качестве интегрированной среды разработки использована среда *Intellij Idea*. Система может функционировать под управлением большинства известных операционных систем, поддерживающих виртуальную *Java*-машину. Разработка велась с использованием операционной системы *Windows* 8.

Системы META_OPT предоставляет различные варианты организации процесса мета-оптимизации: одиночный пользователь - в виде программной библиотеки с локальной базой данных; корпоративный пользователь – мета-оптимизатор функционирует на сервере компании или распределенно на клиентских машинах; компания-поставщик базового программного обеспечения – пользователи компаний-клиентов получают возможность настраивать базовые алгоритмы оптимизации с использованием общей базы данных системы.

Выполнено экспериментальное исследование эффективности разработанного алгоритмического и программного обеспечения. Вычислительный эксперимент выполнен с использованием в качестве базового алгоритма *PSO*, который имеет три вещественных настраиваемых параметра и один категориальный параметр (топология соседства частиц).

В вычислительном эксперименте для формирования классов базовых задач использованы сферическая тестовая функция, овражная унимодальная функции Розенброка, мультимодальная функция Растригина, мультимодальная функция Экли размерностей 90, 20, 50, 30 соответственно. Суммарно выполнено около 100 000 запусков алгоритма PSO. В качестве ПЭС использованы два показателя: расстояние от найденного базовым алгоритмом решения до глобального минимума рассматриваемой тестовой функции за заданное число итераций; число итераций алгоритма до локализации глобального минимума этой функции с заданной точностью.

Результаты вычислительного эксперимента показывают, что после стагнации процесса мета-оптимизации первый ПЭС улучшен на 30-50%, а второй ПЭС – на 20-45% по сравнению со значениями этих показателей, которые обеспечивают рекомендуемые авторами алгоритма *PSO* стратегии. Важно, что при этом обеспечивается повышение эффективности базового алгоритма по обоим рассматриваемым показателям эффективности одновременно.

Остается открытым вопрос, по каким правилам следует выбирать свойства решаемых задач для их оптимального (с точки зрения процесса настройки) объединения в классы. Требуются также дополнительные исследования для обоснования рациональных оценок эффективности базового алгоритма.

Работа выполнена в рамках проекта РФФИ № 16-07-00287 «Многокритериальная оптимизация на основе аппроксимации соответствующего множества (фронта) Парето».

Список литературы

1. Eiben A.E., Michalewicz Z., Schoenauer M., Smith J.E. Parameter Control in Evolutionary Algorithms // Parameter Setting in Evolutionary Algorithms. Springer Verlag. 2007. P. 19-46. (Ser. Studies in Computational Intelligence; vol. 54). DOI: 10.1007/978-3-540-69432-8_2
2. Smit S.K. Parameter tuning and scientific testing in evolutionary algorithms. Vrije Universiteit, Amsterdam, 2012. Available at: http://dspace.ubvu.vu.nl/bitstream/handle/1871/38404/dissertation.pdf , accessed 01.03.2016.
3. Карпенко А.П., Свианадзе З.О. Метод мета-оптимизации поисковых алгоритмов оптимизации // Наука и образование. МГТУ им. Н.Э. Баумана. Электрон. журн. 2011. № 1. DOI: 10.7463/0111.0164546
4. Karpenko A.P., Svianadze Z.O. Meta-optimization based on self-organizing map and genetic algorithm // Optical Memory and Neural Networks. 2011. Vol. 20, iss. 4. P. 279-283. DOI: 10.3103/S1060992X11040059
5. Pedersen M.E.H. Tuning & simplifying heuristical optimization. PhD Thesis. University of Southampton, Computational Engineering and Design Group, School of Engineering Sciences, 2010. Available at: http://eprints.soton.ac.uk/342792/1.hasCoversheetVersion/MEH_Pedersen_PhD_Thesis_2010.pdf , accessed 10.04.2016.
6. Hutter F., Hoos H. H., Leyton-Brown K. Sequential model-based optimization for general algorithm configuration // Learning and Intelligent Optimization. International Conference on Learning and Intelligent Optimization. Springer Berlin Heidelberg, 2011. P. 507–523. DOI: 10.1007/978-3-642-25566-3_40
7. Карпенко А.П., Селиверстов Е.Ю. Обзор методов роя частиц для задачи глобальной оптимизации (particle swarm optimization) // Наука и образование. МГТУ им. Н.Э. Баумана. Электрон. журн. 2009. № 3. DOI: 10.7463/00309.0116072
8. Ugolotti R. Meta-optimization of Bio-inspired Techniques for Object Recognition. Diss. Università di Parma, Dipartimento di Ingegneria dell'Informazione, 2015. Available at: http://dspace-unipr.cineca.it/bitstream/1889/2827/1/TesiDottoratoUgolottiOnline.pdf , accessed 10.04.2016.
9. Friedman J.H. Stochastic gradient boosting // Computational Statistics and Data Analysis. 2002. Vol. 38, iss. 4. P. 367–378. DOI: 10.1016/S0167-9473(01)00065-2
10. Breiman L. Random forests // Machine Learning. 2001. Vol. 45, iss. 1. P. 5-32. DOI: 10.1023/A:1010933404324
11. Michalewicz Z. Genetic algorithms, numerical optimization, and constraints // Proceedings of the sixth international conference on genetic algorithms. Vol. 195. Morgan Kaufmann,

San Mateo. CA, 1995. P. 151-158. Available at: http://cs.adelaide.edu.au/users/zbyszek/Papers/p16.pdf , accessed 12.03.2016.
12. Blickle T., Thiele L. A comparison of selection schemes used in evolutionary algorithms // Evolutionary Computation. 1996. Vol. 4, no. 4. P. 361–394. DOI: 10.1162/evco.1996.4.4.361
13. Мясников А.С. Островной генетический алгоритм с динамическим распределением вероятностей выбора генетических операторов // Наука и образование. МГТУ им. Н.Э. Баумана. Электрон. журн. 2010. № 1. Режим доступа: http://technomag.neicon.ru/doc/136503.html (дата обращения 01.07.2016).
14. Deb K., Beyer H.G. Self-adaptive genetic algorithms with simulated binary crossover // Evolutionary Computation. 2001. Vol. 9, no. 2. P. 197–221. DOI: 10.1162/106365601750190406
15. Owens M. The Definitive Guide to SQLite. Apress, 2006. 464 p.
16. Poli R., Kennedy J., Blackwell T. Particle swarm optimization // Swarm Intelligence. 2007. Vol. 1, iss. 1. P. 33–57. DOI: 10.1007/s11721-007-0002-0
17. Hunter J.D. Matplotlib: A 2D Graphics Environment // Computing in Science and Engineering. 2007. Vol. 9, iss. 3. P. 90–95. DOI: 10.1109/MCSE.2007.55

Science & Education of the Bauman MSTU
Electronic journal
ISSN 1994-0408

Science and Education of the Bauman MSTU, 2016, no. 09, pp. 92–110.

DOI: **10.7463/0916.0844601**

Received: 08.08.2016
Revised: 22.08.2016

A Model-Based Method for Permanent Parameter Tuning of Optimization Algorithms

T. A. Agasiev[1,*], **A.P. Karpenko**[1]

*taaalex@mail.ru

[1]Bauman Moscow State Technical University, Moscow, Russia

Keywords: global optimization, meta-optimization, permanent parameter tuning, surrogate model, meta-model

This article can be taken as a sequel and a forward development to the earlier publications [3, 4]. It considers a way for the process structuring to tune parameters of the optimized algorithm. The software implementation of the algorithm involves a centralized collection of information about the space strategies of this algorithm based on a three-tier client/server architecture. The main objective of the work is to develop a permanent parameter tuning method free from most of the drawbacks of the method proposed in abovementioned publication:

- the user has to give information about vector of features (VF) of the basic task;
- the use of a client/server architecture requires a stable connection to the app server and increases the response time of the meta-optimizer;
- the app server has to handle the entire computation load of meta-optimization, which expects its high reliability, increasing cost of equipment and cost of maintainance in good working order.

The main features of the proposed method are as follows:

- approximation of the strategy efficiency indicators (SEI) of the basic algorithm through the surrogate models of these indicators;
- acceleration of meta-optimization by using a surrogate model being in compliance with a class of the basic optimization problems, which is "close" to the class under consideration.

In the proposed permanent tuning method to build a SEI surrogate model we use a regression method based on decision trees, more precisely a random forest (RF) method. When searching the class of basic problems, the most "close" to the class under consideration we use the two-step method based on estimating the proximity of VF classes of problems and on estimating the predicting accuracy of the relevant SEI surrogate models. Actually, we solve the task of the meta-optimization by the genetic algorithm with real coding of chromosomes.

The Java programming language and relational database SQLite are used to develop META_OPT scalable software system to implement the proposed method. The META_OPT

Systems also provide various options for meta-optimization process: for single and corporate users and for companies to supply the basic software.

An experimental study of the efficiency of the developed algorithms and software has been conducted. The simulation experiment has been carried out using the *PSO* as a basic algorithm, which has three real parameters tuned and a categorical parameter (particle neighborhood topology).

The simulation experiment results show that after stagnation of the meta-optimization process the first SEI and the second one have been improved, respectively, by 30-50% and 20-45% as compared to the values of these indicators, which provide strategies recommended by the authors of the *PSO* algorithm.

References

1. Eiben A.E., Michalewicz Z., Schoenauer M., Smith J.E. Parameter Control in Evolutionary Algorithms. In: *Parameter Setting in Evolutionary Algorithms*. Springer Verlag. 2007, pp. 19-46. (Ser. *Studies in Computational Intelligence*; vol. 54). DOI: 10.1007/978-3-540-69432-8_2
2. Smit S.K. *Parameter tuning and scientific testing in evolutionary algorithms*. Vrije Universiteit, Amsterdam, 2012. Available at: http://dspace.ubvu.vu.nl/bitstream/handle/1871/38404/dissertation.pdf , accessed 01.03.2016.
3. Karpenko A.P., Svianadze Z.O. Meta-optimization method of search optimization algorithms. *Nauka i obrazovanie MGTU im. N.E. Baumana = Science and Education of the Bauman MSTU,* 2011, no. 1. DOI: 10.7463/0111.0164546
4. Karpenko A.P., Svianadze Z.O. Meta-optimization based on self-organizing map and genetic algorithm. *Optical Memory and Neural Networks*, 2011, vol. 20, iss. 4, pp. 279-283. DOI: 10.3103/S1060992X11040059
5. Pedersen M.E.H. *Tuning and simplifying heuristical optimization. PhD Thesis*. University of Southampton, Computational Engineering and Design Group, School of Engineering Sciences, 2010. Available at: http://eprints.soton.ac.uk/342792/1.hasCoversheetVersion/MEH_Pedersen_PhD_Thesis_2010.pdf , accessed 10.04.2016.
6. Hutter F., Hoos H. H., Leyton-Brown K. Sequential model-based optimization for general algorithm configuration. In: *Learning and Intelligent Optimization. International Conference on Learning and Intelligent Optimization*. Springer Berlin Heidelberg, 2011, pp. 507–523. DOI: 10.1007/978-3-642-25566-3_40
7. Karpenko A.P., Seliverstov E.Yu. Review of the particle swarm optimization method (PSO) for a global optimization problem. *Nauka i obrazovanie MGTU im. N.E. Baumana = Science and Education of the Bauman MSTU,* 2009, no. 3. DOI: 10.7463/00309.0116072

8. Ugolotti R. *Meta-optimization of Bio-inspired Techniques for Object Recognition. Diss*. Università di Parma, Dipartimento di Ingegneria dell'Informazione, 2015. Available at: http://dspace-unipr.cineca.it/bitstream/1889/2827/1/TesiDottoratoUgolottiOnline.pdf , accessed 10.04.2016.
9. Friedman J.H. Stochastic gradient boosting. *Computational Statistics and Data Analysis*, 2002, vol. 38, iss. 4, pp. 367–378. DOI: 10.1016/S0167-9473(01)00065-2
10. Breiman L. Random Forests. *Machine Learning*, 2001, vol. 45, iss. 1, pp. 5-32. DOI: 10.1023/A:1010933404324
11. Michalewicz Z. Genetic algorithms, numerical optimization, and constraints. *Proceedings of the sixth international conference on genetic algorithms. Vol. 195*. Morgan Kaufmann, San Mateo. CA, 1995, pp. 151-158. Available at: http://cs.adelaide.edu.au/users/zbyszek/Papers/p16.pdf , accessed 12.03.2016.
12. Blickle T., Thiele L. A comparison of selection schemes used in evolutionary algorithms. *Evolutionary Computation*, 1996, vol. 4, no. 4, pp. 361–394. DOI: 10.1162/evco.1996.4.4.361
13. Myasnikov A.S. Island genetic algorithm with dynamic distribution of probabilities of the choise of genetic operators. *Nauka i obrazovanie MGTU im. N.E. Baumana = Science and Education of the Bauman MSTU,* 2010, no. 1. Available at: http://technomag.neicon.ru/doc/136503.html , accessed 01.07.2016.
14. Deb K., Beyer H.G. Self-adaptive genetic algorithms with simulated binary crossover. *Evolutionary Computation*, 2001, vol. 9, no. 2, pp. 197–221. DOI: 10.1162/106365601750190406
15. Owens M. *The Definitive Guide to SQLite*. Apress, 2006. 464 p.
16. Poli R., Kennedy J., Blackwell T. Particle swarm optimization. *Swarm Intelligence*, 2007, vol. 1, iss. 1, pp. 33–57. DOI: 10.1007/s11721-007-0002-0
17. Hunter J.D. Matplotlib: A 2D Graphics Environment. *Computing in Science and Engineering*, 2007, vol. 9, iss. 3, pp. 90–95. DOI: 10.1109/MCSE.2007.55