УДК 517.8

DOI: http://doi.org/10.25728/pu.2020.2.3

ИССЛЕДОВАНИЕ НАКОПИТЕЛЬНО-СОРТИРОВОЧНОГО МЕТОДА РЕШЕНИЯ ЗАДАЧ ПАРАМЕТРИЧЕСКОЙ ОПТИМИЗАЦИИ

В.А. Ковешников, А.Я. Мехтиев

Аннотация. Отмечено, что при проектировании сложных систем возрастает актуальность решения задач оптимизации. Однако на практике оптимизация затруднена ввиду отсутствия надежных методов, дающих эффективные решения независимо от особенностей математической модели. Разработка методов, позволяющих решать произвольные задачи параметрической оптимизации, представляет собой сложную задачу. Рассмотрена сущность нового подхода, основанного на эвристиках, эксперименте и предусматривающего применение специальных процедур отсечения и сортировки, Парето-анализа и методов теории случайных процессов. Разработано программное обеспечение и несколько модификаций соответствующего метода, проведена их апробация на ряде тестовых функций повышенной сложности с учетом всего спектра задач параметрической оптимизации. Экспериментально доказана высокая эффективность рассматриваемого подхода. Метод может быть применен для решения сложных научно-исследовательских задач, а его программное обеспечение входить в состав больших интегрированных систем, таких как системы автоматизированного проектирования, интеллектуальные системы, везде, где есть многовариантный анализ как механизм принятия решений.

Ключевые слова: случайный поиск, многоэкстремальность, дискретная оптимизация, непрерывная оптимизация, целочисленность, неопределенность.

ВВЕДЕНИЕ

Большинство оптимизационных методов основано на классических понятиях математического анализа (монотонность, непрерывность, градиент, симплекс и пр.). В данной работе акцент делается на эвристику, случайность, эксперимент, логику, специальные процедуры сортировки решений. При этом нет необходимости в каких-либо абстракциях теории оптимизации.

Накопительно-сортировочный метод предназначен для решения оптимизационных задач следующего вида:

целевая функция $f(\mathbf{x}) \to \min$;

ограничения-неравенства

$$g_m(\mathbf{x}) \geq 0, \quad m = \overline{1, m_0}; \qquad (1)$$

оптимизируемые переменные $\mathbf{x} = (x_1, x_2, ..., x_n)$;

ограничения на переменные

$$a_{01} \leq x_{i_1} \leq b_{01}, \quad i_1 = \overline{1, i_{10}}, \qquad (2)$$

$$a_{02} \leq x_{i_2} \leq b_{02}, \quad i_2 = \overline{1, i_{20}}, \qquad (3)$$

причем

$$G = G(\mathbf{x}) = \sum_{m=1}^{m_0} -g_m(\mathbf{x}), \text{ для } g_m(\mathbf{x}) < 0.$$

Ограничения-равенства $h_k(\mathbf{x}) = 0$, $k = \overline{1, k_0}$, посредством замены $h_k^2(\mathbf{x}) \geq 0$, $k = \overline{1, k_0}$, сводятся к ограничениям-неравенствам типа (1).

Ограничения на переменные (2) и (3) «коробчатого» типа. Ограничения (2) распространяются на непрерывные переменные, а ограничения (3) — на целочисленные переменные. Посредством замены переменных и ввиду их взаимно однозначного соответствия дискретные и логические переменные заменяются на целочисленные переменные типа (3).

Какие-либо требования к структуре целевой функции $f(\mathbf{x})$ и ограничений $g(\mathbf{x})$ отсутствуют. Модель задачи, подлежащей оптимизации, может быть произвольного типа вплоть до наличия областей неопределенности («черных дыр»), единственное требование — возможность расчета и количественной оценки компонентов модели. Под реше-

нием имеется в виду любой вектор $\mathbf{x} = (x_1, x_2, ..., x_n)$ как допустимый, так и не допустимый, оптимальный либо не оптимальный.

В зависимости от ситуации поиск может выполняться:

— в строго допустимой области, то есть когда $G \geq 0$;

— в квазидопустимой области, когда $G + \varepsilon \geq 0$; при этом $\varepsilon > 0$ определяет меру возможного нарушения ограничений;

— когда ограничения на область поиска отсутствуют, т. е. в процессе оптимизации участвуют как допустимые ($G(\mathbf{x}) \geq 0$), так и не допустимые ($G(\mathbf{x}) < 0$) решения $\mathbf{x}$.

1. ОСНОВНЫЕ ПОНЯТИЯ. СУЩНОСТЬ МЕТОДА

Массив решений (накопитель) $p_1, p_2, ..., p_N$ можно представлять как множество кортежей вида

$$\begin{gathered} p_1\colon x_{11}, x_{12}, ..., x_{1n}, f_1, G_1, \\ p_2\colon x_{21}, x_{22}, ..., x_{2n}, f_2, G_2, \\ \cdots \\ p_N\colon x_{N1}, x_{N2}, ..., x_{Nn}, f_N, G_N, \end{gathered} \tag{4}$$

где $x_{i1}, x_{i2}, ..., x_{in}$ — собственно координаты решения $\mathbf{x}_i$, f_i — значение целевой функции и G_i — значение показателя нарушения ограничений при $\mathbf{x} = \mathbf{x}_i$.

С другой стороны, накопитель (обозначим его V) есть ни что иное, как хранилище необходимой информации для организации многошаговой процедуры поиска. Он содержит определенную информацию (V_i) для генерации нового набора решений (V_{i+1}), в котором в конечном итоге (на итерации k) содержится оптимум $\mathbf{x}^* \in V^*$, т. е.

$$V_1 \overset{\otimes}{\rightarrow} V_2 \overset{\otimes}{\rightarrow} V_3 \ldots \overset{\otimes}{\rightarrow} V_{k-1} \overset{\otimes}{\rightarrow} V_k(V^*),$$

где символ $\otimes$ следует понимать как набор правил, операторов, алгоритмов и прочих формальных процедур, приводящих к построению нового, более эффективного набора решений. Данное обстоятельство определяет сущность предлагаемого подхода и рассматривается далее.

Имея несколько решений, можно по-разному получать новые решения. При всем их многообразии принципиально различных подходов два: направленный поиск и случайный поиск [3], к которому примыкают и генетические алгоритмы. Рассматриваемый подход и соответствующий метод — разновидность случайного поиска.

Если основу подхода определяет идея анализа набора фиксированных решений, а переход в следующее состояние (очередная итерация) выполняется посредством многокритериальных многоуровневых сортировок множества решений, причем генерация новых решений выполняется случайно, то мы имеем дело с методом анализа накопителя, накопительным сортировочным методом (НС-методом). Схематично сущность подхода поясняет цепочка операций, представленная на схеме взаимодействия:

$$V_0 \overset{\Delta V_0}{\rightarrow} (V_1 \overset{\Pi\rightarrow\Phi\rightarrow R}{\rightarrow} V_1' \overset{W}{\rightarrow} V_1^*) \overset{\Delta V_1}{\rightarrow} (V_2 \overset{\Pi\rightarrow\Phi\rightarrow R}{\rightarrow} V_2' \overset{W}{\rightarrow} \rightarrow V_2^*) \ldots \overset{\Delta V_{k-1}}{\rightarrow} (V_k \overset{\Pi\rightarrow\Phi\rightarrow R}{\rightarrow} V_k' \overset{W}{\rightarrow} V_k^*) \overset{\Delta V_k}{\rightarrow} \ldots; \tag{5}$$

$$V_0 \overset{\Delta V_0}{\rightarrow} (V_1 \overset{\Phi\rightarrow R}{\rightarrow} V_1' \overset{W}{\rightarrow} V_1^*) \overset{\Delta V_1}{\rightarrow} (V_2 \overset{\Phi\rightarrow R}{\rightarrow} V_2' \overset{W}{\rightarrow} V_2^*) \ldots \ldots \overset{\Delta V_{k-1}}{\rightarrow} (V_k \overset{\Phi\rightarrow R}{\rightarrow} V_k' \overset{W}{\rightarrow} V_k^*) \overset{\Delta V_k}{\rightarrow} \ldots \tag{6}$$

Зависимости (5) символизируют о включении Парето-анализа (на схеме — символ П) как одной из операций обработки накопителя решений, согласно зависимостям (6) операция Парето-анализа не применяется. Получаем модификации метода оптимизации, эффективность которых можно оценить разве что экспериментально.

Смысл символики в записях (5) и (6):

V_0 — начальное состояние накопителя, пустой накопитель; ΔV_0 — набор начальных решений, полученных определенным способом (например, случайно в n-мерном пространстве поиска) — необходимая информация для запуска процедуры поиска; ΔV_{k-1} — набор решений, полученных на $(k-1)$-й итерации процесса поиска путем учета уже имеющихся решений благодаря синтезу одноименных координат и их варьирования для последующего добавления и анализа в накопителе; V_k — набор всех решений, уже имеющихся на $(k-1)$-й итерации V_{k-1}^* и добавленных ΔV_{k-1}, в совокупности формирующих исходное состояние накопителя на следующей k-й итерации, т. е. $V_k = V_{k-1}^* \cup \Delta V_{k-1}$; V_k' — набор решений накопителя, полученных в результате Парето-анализа с участием критериев $f(\mathbf{x})$ и $G(\mathbf{x})$, дискретного времени k, символизирующего глубину процесса оптимизации $\rho^{(k)}$, а также последующего ранжирования решений на основе обобщенной целевой функции $F = F(f(\mathbf{x}), G(\mathbf{x}), \rho^{(k)})$; V_k^* — набор базовых значений на k-й итерации, полученных после выявления и потери бесперспективных решений, содержащий текущие результаты (лучшее допустимое решение, если оно имеется, Парето-оптимальное недопустимое решение, лучшее по целевой функции, текущее решение в ходе итерационного процесса, основное оптимальное решение — оптимальное по цели при допустимых ограничениях либо лучшее по ограничениям недопустимое решение); Ф — символ отображения решений посредством обобщенной целевой функции; R — символ ран-

жирования, фиксирующий необходимость отыскания и последовательной записи решений в метрике обобщенной целевой функции, когда все решения располагаются в порядке понижения значимости; W — символ операции определения и фиксации лучших решений, содержащихся в накопителе.

Действительно, в самом начале оптимизации случайным образом генерируется набор решений (обычно 1000 решений, т. е. $\Delta V_0 = 1000$), и они дописываются в «хвост» текущего состояния накопителя V_0 (вначале он пустой). На их основе формируется следующее содержимое накопителя V_1 в результате Парето-оптимальной сортировки (П), построения обобщенной функции (Ф) и ранжирования (R). В результате получаем промежуточный массив V_1'.

Операция W предполагает отбор необходимых решений из расширенного накопителя (из имеющихся и добавленных решений в результате смешивания и смещения координат) и определение лучших среди них на текущей итерации. Остается некоторая часть решений (вначале обычно из 1000 исходных решений оставляем 100, а далее из 200 сохраняем 100 лучших). В результате завершается переход к следующей итерации, т. е. $V_0 \to V_1$, о чем свидетельствует символ V_1^*. Таким образом представлен участок $V_0 \overset{\Delta V_0}{\to} (V_1 \overset{\text{П}\to\Phi\to R}{\to} V_1' \overset{W}{\to} V_1^*)$. Аналогично следует понимать последующую символику записи согласно зависимостям (5).

2. МЕХАНИЗМ ГЕНЕРАЦИИ НОВЫХ РЕШЕНИЙ

Новые решения могут быть образованы посредством смешивания координат различных решений и последующей коррекцией одной либо нескольких координат приблизительно так, как это достигается в генетических алгоритмах.

2.1. Стратегия построения решений

Для построения новых решений $\mathbf{x}_н$ можно воспользоваться различными подходами. Наиболее естественные, простые и эффективные среди них:

— подход, когда новое решение $\mathbf{x}_н$ определяется на основе двух других решений накопителя $\mathbf{x}$ и $\mathbf{y}$, т. е. $\mathbf{x}_н$: $\theta(\mathbf{y}, \mathbf{x})$, где θ — некоторое правило (алгоритм) взаимодействия решений;

— подход, когда новое решение $\mathbf{x}_н$ определяется на основе нескольких решений накопителя $\mathbf{x}_1$, $\mathbf{x}_2$, ..., $\mathbf{x}_\alpha$, т. е. $\mathbf{x}_н$: $\theta(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_\alpha)$.

Рассмотрим несколько подробнее первый из них. В этом случае опять же имеется несколько вариантов выбора решений $\mathbf{x}$ и $\mathbf{y}$.

- Выбор базового (первого) решения ($\mathbf{y}$):

— таковым служит лучшее решение накопителя, т. е. *первое* решение *первого* кластера, полученного в результате Парето-ранжирования и последующей сортировки в пределах этого кластера по обобщенной целевой функции, т. е. $\mathbf{y} = \mathbf{x}_1$, или в координатной форме $y(i) = x_1(i)$, $i = \overline{1, n}$;

— базовое решение определяется как случайное решение *первого* кластера, т. е. $\mathbf{y} = \mathbf{x}_\xi$;

— базовое решение определяется как случайное решение *всего* накопителя $\mathbf{y} = \mathbf{x}_{\xi\xi}$;

— совместное применение перечисленных стратегий по вероятностной схеме.

- Выбор дополнительного (второго) решения ($\mathbf{x}$):

— последовательно участвует *каждое* решение накопителя;

— решения назначаются случайно только из представителей *первого* (лучшего) кластера;

— решения назначаются случайно, в отборе участвуют все решения накопителя;

— решения назначаются случайно на основе решений *первого* кластера, но в определенных случаях подключаются решения последующих кластеров.

Дополнительно, многообразие состояний модели задачи формируется путем варьирования значений координат полученного решения $\mathbf{x}_н$ на основе вероятностной схемы по правилу:

$$x_i = x_i^{\min} + (x_i^{\max} - x_i^{\min})\xi, \qquad (7)$$

либо по правилу:

$$x_i = x_i^{(н)}(k) + (2\xi - 1)(x_i^{\max}(k) - x_i^{\min}(k)), \qquad (8)$$

где ξ — равномерно распределенное число в диапазоне $[0 \div 1]$, $x_i^{\min}$ и $x_i^{\max}$ — минимальное и максимальное значения переменной x_i по условию задачи, $x_i^{\min}(k)$ и $x_i^{\max}(k)$ — минимальное и максимальное значения переменной x_i, определяемое на основе анализа значений накопителя решений на k-й итерации, $x_i^{(н)}(k)$ — значение i-й координаты нового решения на k-й итерации.

Зависимость (7) определяет глобализацию и в большей мере используется в начале поиска, зависимость (8) соответствует локализации процесса оптимизации. Характерно, что данные стратегии применяются постоянно в процессе оптимизации, но с разной частотой в зависимости от номера итерации k, т. е. удаленности от начала процесса.

Полученные таким образом решения дописываются в накопитель для последующего анализа и исключения бесперспективных решений, причем объем накопителя сохраняет свое первоначальное значение N.

Рассмотрим более подробно сущность динамического процесса сортировки решений накопителя.

2.2. Управление накопителем решений. Правила включения решений в накопитель

Получив новое решение, его необходимо сравнить с имеющимися решениями в текущий момент и обосновать необходимость включения в накопитель для учета в дальнейшей работе. Принципиальной основой здесь являются внешние условия, когда поиск должен выполняться в строго допустимой области либо когда область поиска произвольна, т. е. можно учитывать как допустимые, так и недопустимые решения. В связи с этим целесообразно рассмотреть два блока стратегии управления накопителем.

Стратегии управления при отсутствии ограничений на область поиска

- При *любых* значениях ограничений $g_i(\mathbf{x})$, $i = \overline{1, m}$, и целевой функции $f(\mathbf{x})$ решение поступает в Н.
- Если координаты нового решения *совпадают* с одним из решений Н, то оно теряется. Такая логика позволяет избежать ситуации «информационной пустоты», когда все решения Н совпадают, что наиболее вероятно при решении задач дискретной или целочисленной оптимизации.
- Ограничение *нарушено*, но не более некоторого порога ограничений, тогда оно поступает в Н.
- *Всякое* допустимое решение поступает в Н.
- Решение допустимое, но в Н не поступает, так как *есть более сильные* допустимые решения.
- Если решение *уступает худшему* решению в Н по цели и ограничениям одновременно, то оно в Н не поступает.
- Решение допустимое, поступает в Н, так как там на текущий момент *нет допустимых решений*.
- Новое решение допустимое, оно теряется, так как *уступает худшему* допустимому решению.
- Новое решение допустимое, оно не поступает в Н в связи с тем, что *уступает лучшему* решению без нарушения ограничений (очень жесткий режим).

Стратегии управления при поиске в допустимой области

- В Н поступает *любое допустимое решение*, даже если оно уступает худшему решению Н.
- В Н поступает допустимое решение, если оно *лучше самого худшего* решения Н.
- В Н поступает допустимое решение, если оно *лучше самого лучшего* решения Н в текущий момент (на текущей итерации), это очень жесткий режим.
- В Н решение не поступает, если оно *совпадает* с одним из уже имеющихся решений. Совпадающие решения не дублируются.
- В Н поступает допустимое решение, если оно уступает только *самому лучшему* решению Н.

Стратегия прохода решений в накопитель в значительной мере определяет эффективность всей оптимизационной процедуры.

2.3. Сортировка решений накопителя

Таким образом, располагая набором решений $\mathbf{x}_1(k)$, $\mathbf{x}_2(k)$, ..., $\mathbf{x}_N(k)$ на k-й итерации и образуя на их основе новые решения $\mathbf{x}_1(k+1)$, $\mathbf{x}_2(k+1)$, ..., $\mathbf{x}_L(k+1)$, получаем новый расширенный набор решений $\mathbf{x}_1(k)$, $\mathbf{x}_2(k)$, ..., $\mathbf{x}_N(k)$, $\mathbf{x}_1(k+1)$, $\mathbf{x}_2(k+1)$, ..., $\mathbf{x}_L(k+1)$, или $\mathbf{x}_1$, $\mathbf{x}_2$, ..., $\mathbf{x}_M$, $M \le N + L$, $L \le N$, т. е. $M \le 2N$. Далее для расширенного дискретного набора решений выполняется Парето-оптимальная сортировка [1] — кластеризация по двум показателям (критериям): нарушенным ограничениям $G(\mathbf{x})$ и целевой функции $f(\mathbf{x})$. В результате первого прохода образуется первый кластер — набор Парето-оптимальных решений, тех решений, которые не могут быть улучшены («перекрыты») другими решениями накопителя, причем одновременно по этим двум критериям.

Применяя эту процедуру к оставшимся решениям, можно по аналогии получить второй кластер и так далее вплоть до последнего. В каждом последующем кластере решения будут менее привлекательными в контексте оптимальности, однако данное обстоятельство не является основанием для их забвения, так как на их основе можно получать дополнительные уникальные решения и тем самым повышать качество оптимизации.

В пределах кластера отдать однозначное предпочтение какому-либо из решений не представляется возможным. Поэтому целесообразно применить линейную комбинацию этих критериев и выполнить ранжирование по обобщенному критерию F согласно зависимости

$$F^{(k)}(\mathbf{x}) = f^{(k)}(\mathbf{x}) + \rho^{(k)} G^{(k)}(\mathbf{x}),$$

$$\rho^{(k)} \to \infty \text{ при } k \to \infty, \qquad (9)$$

где k — номер итерации процесса оптимизации.

Проранжировав согласно зависимости (9) все решения накопителя, сохраняем N лучших; оставшиеся решения в числе L штук безвозвратно теряются. На этом цикл завершается, но оптимизация продолжается. Останов можно организовать либо по истечении времени, либо по достижению заданного числа итераций k, либо нажатием клавиши клавиатуры в произвольный момент времени.

Модификации метода предполагают различные стратегии учета кластеров. Особенность заключается в том, что наиболее перспективен первый кластер, а далее их значимость неизбежно снижа-

ется. Тем не менее, важно сохранять многообразие генерируемых решений для повышения эффективности поиска, в том числе и путем учета других кластеров накопителя.

Разработанное к настоящему времени математическое и программное обеспечение позволяет оперировать накопителями объемом от одного до тысячи решений ($N = 1 \div 1000$). Выбор объема накопителя определяется сложностью, точностью и временем решения. Обычно при $N = 30(50)$ удается решать достаточно сложные оптимизационные задачи.

3. СХОДИМОСТЬ МЕТОДА

Решения $p_1, p_2, ..., p_N$, в составе накопителя $V(k)$ согласно выражению (4) целесообразно применить для оценки сходимости процесса поиска. Для этого для каждой координаты x_i следует сначала определить среднее значение $x_i^{\text{ср}}$, а затем найти отклонения $x_i^{\text{ср}} - x_i$, просуммировать их и найти среднее, т. е.

$$x_i^{\text{ср}}(k) = \frac{1}{N}\sum_{j=1}^{N} x_{ji}(k), \quad \Delta_i(k) = \frac{1}{N}\sum_{j=1}^{N} |x_i^{\text{ср}}(k) - x_{ji}(k)|,$$

$$i = \overline{1, n},$$

где N — число решений, объем накопителя, n — число оптимизируемых переменных, k — номер итерации.

Если диапазон изменения переменной x_i на текущей итерации меньше некоторого значения ε_0, поиск в данном направлении становится малопривлекательным. Следует считать найденное значение оптимальным либо продолжить поиск с другим большим шагом-размахом, считая глобальность не достигнутой. Предпочтение следует отдать второму утверждению, чтобы обеспечить «живучесть алгоритма» и двигаться дальше.

Для этого следует использовать предшествующую аналогичную информацию, обобщая и усредняя ее. Тогда, имея ряд $\Delta_i(1), \Delta_i(2), ..., \Delta_i(k_0)$, получим

$$\nabla_i = \sum_{k=1}^{k_0} \Delta_i(k).$$

Так как массив для хранения средних значений отклонений имеет фиксированную длину k_0, с определенного момента новое значение должно дописываться в «хвост», а первое — теряться, что позволяет поддерживать процесс в динамике с неограниченным ресурсом по времени. Величины $\Delta_i(1), \Delta_i(2), ..., \Delta_i(k_0)$ должны тенденциозно стремиться к нулю, что определяет сходимость процесса оптимизации. Если на очередной итерации $\Delta_i(k) < \varepsilon_0$, целесообразно продолжать поиск в размере ∇_i, т. е. $\tilde{x}_i = x_i + \nabla_i(2\xi - 1)$, где i — индекс переменной, $\tilde{x}_i$ — новое сгенерированное значение на основе текущего значения накопителя переменной x_i, ξ — случайное число в диапазоне [0—1].

Если и эта величина не приводит к результату, т. е. $\nabla_i < \varepsilon_0$, что опять же свидетельствует о завершении процесса поиска, то поиск целесообразно продолжить в исходном масштабе

$$\tilde{x}_i(\xi) = x_{ia} + (x_{ib} - x_{ia})\xi = x_{ia} + \Delta x_i^0 \xi,$$

где x_i, x_{ib} и x_{ia} — текущее, верхнее и нижнее значение i-й переменной.

Такой подход, с одной стороны, обеспечивает сходимость оптимизации, а с другой — всегда сохраняется возможность глобализации. Если существует более сильное решение и оно на текущий момент не найдено, возможность его отыскания всегда сохраняется. По существу реализуется бесконечный процесс. Для его останова целесообразно воспользоваться либо таймером, либо ограничением числа итераций, либо требованием пользователя на основе визуализации текущего процесса оптимизации.

Согласно *методу анализа накопителя* (НС-методу) выходная информация включает в себя допустимое оптимальное решение (если оно существует), лучшее по ограничениям недопустимое решение, Парето-оптимальное решение, лучшее текущее решение на последней итерации, число обращений к модели, среднее время расчета модели, меру и число нарушений каждого ограничения. Данная информация позволяет оценить качество модели и перспективы оптимизационного исследования.

Для проверки правомерности и эффективности подхода проводились компьютерные эксперименты на основе многочисленного набора тестовых функций, охватывающих все многообразие задач параметрической оптимизации — дискретные, непрерывные, целочисленные, булевы, многоэкстремальные, смешанные, задачи класса NP (гарантированное решение определяется только полным перебором вариантов). При этом рассматривались решения с числом оптимизируемых переменных 5, 15, 50, 100, 500, что крайне важно при оценке работоспособности метода.

4. ПРОВЕРКА ДОСТОВЕРНОСТИ НС-МЕТОДА

Для оценки работоспособности таких алгоритмов обычно пользуются специальными тестовыми задачами [2, 5, 6—14]. Рассмотрим некоторые из них по блокам. Подробная информация представ-

лена на web-странице https://optimnc.wixsite.com/nc-optim [13].

Блок 1. Что касается решения таких хорошо известных в теории параметрической оптимизации задач, как задачи Griewangk, Schwefel, Askley, Helix, Леви, Растригина, Rosenbrock, Powel, Wood, Kowalik и др., то все они были получены в стандартном режиме без какой-либо дополнительной настройки алгоритма и модели.

Блок 2. Отдельного внимания заслуживают дискретные задачи (в отличие от упомянутых в блоке 1 непрерывных оптимизационных задач), где и для НС-метода наступают определенные сложности. К ним относится и задача коммивояжера, на плоскости и окружности со случайно расположенными на них объектами. Тогда при размерности 100 объектов — точек число вариантов составляет около $9{,}332621544394418 \cdot 10^{157}$ ед., и для окружности радиуса $r = 50$ м длина минимального пути не должна превышать длины окружности $L = 314$ м. Несколько независимых экспериментов свидетельствуют о гарантированной возможности достижения результата с показателем $L = 400$—470 м за время, не превосходящее примерно 2 ч; в трех случаях были зафиксированы результаты, близкие к оптимальным: 279, 316 и 353 м. Заметим, что имеют место маршруты, когда показатель L достигает 7000 м и более.

Блок 3. Рассмотрим результаты решения на примере так называемого теста «шахматы», относящегося к труднорешаемым задачам класса NP. Основное назначение теста — проверить возможность поиска в условиях отсутствия традиционной логики поисковых процедур. Задача заключается в том, что шахматную доску ($n = 8$) необходимо покрыть минимальным числом коней так, чтобы все клетки оказались под боем, задача имеет приблизительно 10^{20} вариантов решений. При $n = 13$ имеем около 10^{50} решений, при $n = 22$ вариантность составляет 10^{145} ед. Оптимальное решение здесь может быть найдено (гарантировано) только полным перебором вариантов. При этом время решения на компьютере ориентировочно составляет соответственно около 10^6 лет, 10^{37} лет и 10^{132} лет, что свидетельствует о сложности, запредельном многообразии!

Формально данный тест в общем виде записывается следующим образом:

найти минимум функции

$$f = \sum_{i=1}^{n} \sum_{j=1}^{n} x_{i,j},$$

при условии, что для всех $i \in 1, \ldots, n$ и $j \in 1, \ldots, n$ имеет место неравенство

$$g_{i,j} + g_{i-2,j-1} + g_{i-1,j-2} + g_{i+1,j-2} + g_{i+2,j-1} + g_{i+2,j+1} + g_{i+1,j+2} + g_{i-1,j+2} + g_{i-2,j+1} \geq 1,$$

где

$$g_{i,j} = \begin{cases} x_{i,j}, \text{ если } (1 \leq i \leq n) \wedge (1 \leq j \leq n), \\ 0, \text{ в противном случае,} \end{cases}$$

где $x_{i,j}$ — псевдобулевы переменные (случайным образом принимающие значения 0 или 1).

Далее представлены имеющиеся выборочные решения для случаев, когда размер шахматной доски n изменяется от 4 до 22, в скобках указаны лучшие значения целевой функции f — рекорды, найденные с помощью рассматриваемого алгоритма. Имеем: n (f) = 4 (4), 5 (5), 6 (8), 7 (10), 8 (12), 9 (14*), 10 (16*), 11 (21*), 12 (25*), 13 (28), 16 (45**), 17 (55**), 20 (71**), 22 (85**). Символ «звездочка» свидетельствует о сложности полученного решения, оптимальность здесь не гарантируется, предполагаются субоптимальные решения [15]. Наиболее интересны случаи $n = 8$ и $n = 13$. Очевидно, при возрастании размерности вероятность поиска оптимума уменьшается.

Блок 4. Другой сложный набор тестов представлен в материалах отчета международной конференции по проблемам оптимизации [4], где в основном рассматриваются непрерывные задачи, а поиск оптимума «закрыт» различными формальными особенностями представленных моделей. Очевидно, для классических оптимизационных методов такие задачи не разрешимы. Результаты решения данных задач G01÷G24 представлены в таблице.

В столбце «модель задачи» символом «звездочка» отмечены тесты, модель которых в виду сложности решения была тождественно преобразована относительно исходной постановки. Для задач G08, G17 и G18 получены более точные решения, чем представленные в отчете. Задача G20 не имеет допустимых решений, и нет оснований для ее объективного контроля. Большинство полученных решений находится в допустимой области, когда все ограничения выполнены, т. е. $G(\mathbf{x}^*) = 0$. Уровень сложности определен на основе анализа структуры модели и результатов работы оптимизатора.

Блок 5. В работе [14] представлены 22 теста из класса оптимизационных задач без ограничений (неограниченные эталоны). В задачах С.1.3, С.1.19 и С.1.21 найдены ранее неизвестные оптимальные решения. В задачах С.2.17 и С.2.18 из блока «ограниченные эталоны» получено несколько близких решений, при этом модель представляется не вполне корректной.

Результаты решения задач G01—G25 [4]

Модель задачи	Сложность	Качество решения	Значение ограничений $G(\overline{x}^*)$	Цель $F(\overline{x}^*)$
G01	3	Достоверно (просто)	0	−15
G02	9	Достоверно	0	−0,803618472632700
G03	9—10	Достоверно (сложно)	0	−0,999997491623985
G04	3—5	Достоверно (просто)	0	−30 665,5388872998
G05	8—9	Достоверно (сложно)	0	5126,80668255243
G06	1—2	Достоверно (просто)	0	−6 961,81087343529
G07	1—2	Достоверно (просто)	0	25,0531296505572
G08	3—5	Достоверно (просто)	0	−0,105458911312976
G09	5	Достоверно (просто)	0	680,633990673840
G10*	10	Достоверно (сложно)	0	7049,24797069537
G11	1—2	Достоверно (просто)	0	0,75000000
G12	5	Достоверно (просто)	0	−1,00000000000000
G13	9—10	Достоверно (сложно)	0	0,05401085049234521
G14	5—7	Достоверно	0	−47,4785194571968
G15	5—7	Достоверно	0	−961,718467443156
G16	3—7	Достоверно	0	−1,90515505505526
G17	5—7	Достоверно	0	8 829,74501927317
G18	5—7	Достоверно	0	−0,866025233067233
G19	9—10	Достоверно (сложно)	0	−32,9022500988993
G20	1—10 (не ясно)	Не определено	0,758575821066371	0,06847236591050188
G21*	10	Достоверно (сложно)	0,000720244352123	193,761578718546
G22*	10	Достоверно (сложно)	0,000000000000011	236,746535851213
G23*	10	Достоверно (сложно)	0	−400,000018998981
G24	1—3	Достоверно (просто)	0	−5,50801327159533

Блок 6. Особый интерес представляет задача с различными типами неупорядоченных переменных такого содержания:

необходимо найти минимальное значение функции

$$F(x, A, B, C) = Ax^2 + Bx + C \to \min,$$

определенной на интервале [0—10 000], исключая пять подынтервалов: от 1 до 1,5, т. е. [1—1,5], далее [3—5], [7—10], [15—40], [60—90].

Параметр A имеет 100 целочисленных значений, расположенных в последовательности: 98, 96, 97, 99, 100, 91, 94, 95, 92, 93, 7, 8, 9, 10, 2, 3, 4, 5, 6, 89, 81, 82, 83, 86, 87, 88, 84, 85, 90, 35, 36, 37, 31 32, 33, 34, 38, 39, 40, 11, 12, 13, 15, 16, 17, 18, 14, 19, 20, 53, 54, 51, 52, 55, 56, 57, 1, 59, 60, 58, 44, 43, 47, 48, 45, 46, 41, 42, 49, 50, 63, 64, 61, 62, 65, 66, 67, 68, 69, 70, 78, 79, 80, 71, 72, 75, 76, 73, 74, 77, 21, 22, 23, 24, 25, 26, 28, 29, 27, 30.

Параметр B определен пятью *дискретными значениями* [−0,344; −0,785; 9,1; −10,9; 55].

Параметр C имеет два значения [1, 0] — *булева* переменная.

Переменная x непрерывна в диапазоне [1—10 000].

Необходимо, чтобы $A + B + C > 0$, а значение $AB > -0{,}8$. Ко всему прочему, значение аргумента функции следует определить предельно точно (это требование фиксирует сходимость в связи с вычислительными погрешностями алгоритма, «глубину» поиска при наличии непрерывных переменных).

По существу имеем смешанную задачу с различными типами неупорядоченных переменных (монотонность, непрерывность, дифференцируемость и прочие условности оптимизации отсутствуют), определенную кусочно, и при наличии двух функциональных ограничений.

Многократные запуски программы-оптимизатора [16] показали 100 %-ю надежность получения достоверного решения за 1—20 с работы компьютера среднего класса (типа Acer Aspire C22-820). При этом погрешность определения непрерывной координаты составила 10^{-9}—10^{-10} или 10—11 знаков в мантиссе, например 2,99999999998250, что свидетельствует о высокой сходимости метода.

Оптимальное решение: целевая функция $f(x_1, x_2, x_3, x_4) = 1{,}07249996069797$; целочисленная переменная $x_1 = 1$, дискретная переменная $x_2 = -0{,}785$; логическая переменная $x_3 = 0$, непрерывная переменная $x_4 = 1{,}50000000001672$.

ЗАКЛЮЧЕНИЕ

В заключение сформулируем основные выводы.

Изложен новый подход к решению задач параметрической оптимизации. Рассмотрены основные процедуры изложенного метода, их взаимосвязь и последовательность применения.

Разработано программное обеспечение и несколько модификаций метода, проведена их ап-

робация на многочисленных тестовых функциях, представляющих все многообразие задач параметрической оптимизации.

Компьютерные эксперименты свидетельствуют о высокой эффективности метода при решении всех типов задач параметрической оптимизации на основе единого алгоритма, не меняя программный код, фиксируя особенности конкретной задачи лишь при вводе информации. Данное обстоятельство служит основанием считать данный подход и соответствующий метод достаточно универсальными при решении задач параметрической оптимизации.

ЛИТЕРАТУРА

1. *Моисеев Н.Н.* Математические задачи системного анализа. — М.: Наука, 1981. [*Moiseev, N.N.* Matematicheskie zadachi sistemnogo analiza. — M.: Nauka, 1981. (In Russian)]
2. *Химмельблау Д.* Прикладное нелинейное программирование. — М.: Мир, 1975. — 534 с. [*Khimmel'blau, D.* Prikladnoe nelineinoe programmirovanie. — M.: Mir, 1975. — 534 s. (In Russian)]
3. *Растригин Л.А.* Методы случайного поиска. — М.: Наука, 1968. [*Rastrigin, L.A.* Metody sluchainogo poiska]. — M.: Nauka, 1968. (In Russian)]
4. *Liang, J.J., Runarsson, T.P., Mezura-Montes, E.,* et al. Problem Definitions and Evaluation Criteria for the CEC 2006 Special Session on Constrained Real-Parameter Optimization. — Technical Report. — Singapore: Nanyang Technological University, 2006. — 19 p.
5. *Батищев Д.И., Исаев С.А.* Оптимизация многоэкстремальных функций с помощью генетических алгоритмов // Высокие технологии в технике, медицине и образовании. — Воронеж: ВГТУ, 1997. — Ч. 3. [*Batishchev, D.I., Isaev, S.A.* Optimizatsiya mnogoehkstremal'nykh funktsii s pomoshch'yu geneticheskikh algoritmov] // Vysokie tekhnologii v tekhnike, meditsine i obrazovanii. — Voronezh: VGTU, 1997. — Ch. 3. (In Russian)]
6. *Sastry, K.* Analysis of Mixing in Genetic Algorithms: A Survey. IlliGAL report No. 2002012. — University of Illinois, Urbana-Champaign, 2002.
7. *More, J.J., Gabow, B.S., and Hillstrom, K.E.* Testing Unconstrained Optimization Software // ACM Trans. Meth. Software. — 1981. — P. 17—41.
8. *Fogel, L.J., Owens, A.J., and Walsh, M.J.* Artificial Intelligence Through Simulated Evolution. — Hoboken: John Wiley & Sons, 1966.
9. *URL:* http://www.faqs.org/faqs/nonlinear-programming-faq/
10. *URL:* http://en.wikipedia.org/wiki/List_of_optimization_software
11. *URL:* http://wiki.mcs.anl.gov/NEOS/index.php/Nonlinear_Programming_FAQ
12. *Ковешников В.А., Фатуев В.А., Троицкий Д.И., Пантелеев И.Ю.* Разработка и исследование универсального алгоритма случайно-генетической оптимизации // Тр. междунар. конф. SICPRO '09. — М., 2009. [*Koveshnikov, V.A., Fatuev, V.A., Troitskii, D.I., Panteleev, I.Yu.* Razrabotka i issledovanie universal'nogo algoritma sluchaino-geneticheskoi optimizatsii] // Tr. mezhdunar. konf. SICPRO '09. — M., 2009. (In Russian)]
13. *URL:* https://optimnc.wixsite.com/nc-optim/
14. *Саймон Д.* Алгоритмы эволюционной оптимизации. — М.: ДНК Пресс, 2020. — 1002 с. [*Saimon, D.* Algoritmy ehvolyutsionnoi optimizatsii. — M.: DNK Press, 2020. — 1002 s. (In Russian)]
15. *Гиг ван, Дж.* Прикладная общая теория систем. — М.: Мир — 1981. — Т. 1. — 336 с. [*Gig van, Dzh.* Prikladnaya obshchaya teoriya sistem. — M.: Mir — 1981. — T. 1. — 336 s. (In Russian)]
16. *Ковешников В.А.* Программа реализации универсального алгоритма решения задач параметрической оптимизации. Сертификат на программное обеспечение No 2018663352, 2018. [*Koveshnikov, V.A.* Program for realization of unified algorithm to solve the parametric optimization problems. State Record Certificate for PC software, no. 2018663352, 2018. (In Russian)]

Статья представлена к публикации членом редколлегии М.В. Хлебниковым.

Поступила в редакцию 4.10.2019, после доработки 20.12.2019. Принята к публикации 20.12.2019.

Ковешников Владимир Алексеевич — канд. техн. наук, ✉ kbkedr@tula.net,

Мехтиев Аббас Ядулла-оглы, ✉ kbkedr@tula.net,

АО «Конструкторское бюро приборостроения им. академика А.Г. Шипунова», г. Тула.

DATA ACCUMULATION AND SORTING METHOD FOR SOLVING PARAMETRIC OPTIMIZATION PROBLEM

V.A. Koveshnikov, A.Ya. Mekhtiev

KPB Named after Academician A. Shipunov, Tula, Russia
✉ kbkedr@tula.net

Abstract. It is noted that optimization problem is highly relevant for complex system development. However, such optimization is difficult since there are no reliable methods that give efficient solutions regardless of the features of a specific mathematical model. Developing a method for solving arbitrary parametric optimization problems is a complicated but highly relevant task. The new approach is considered, which is based on heuristics and experiments. It uses special truncation and sorting procedures, Pareto methods, and random process theory methods. The software implementation and multiple modifications of the method proposed are developed and tested with a number of highly complex test functions covering the entire range of parametric optimization problems. It is shown experimentally that the approach proposed is highly efficient. The method can be applied for solving complex research problems and its software can become a part of large integrated systems such as CADs, smart systems, etc. where multivariate analysis is used for decision making.

Keywords: random search, multi-extremality, discrete optimization, continuous optimization, integer values, uncertainty.