Comparison of optimization algorithms for automated method development of gradient profiles

van Henten, G.B.; Boelrijk, J.; Kattenberg, C.; Bos, T.S.; Ensing, B.; Forré, P.; Pirok, B.W.J.

DOI
10.1016/j.chroma.2024.465626

Publication date
2025

Document Version
Final published version

Published in
Journal of Chromatography A

Journal of Chromatography A 1742 (2025) 465626

Contents lists available at ScienceDirect

Journal of Chromatography A

journal homepage: www.elsevier.com/locate/chroma

Comparison of optimization algorithms for automated method development of gradient profiles

Gerben B. van Henten [a,b,1,*], Jim Boelrijk [c,d,1], Céline Kattenberg [c,1], Tijmen S. Bos [a,b], Bernd Ensing [c,e], Patrick Forré [c,d], Bob W.J. Pirok [a,b,c,*]

[a] Analytical Chemistry Group, Van 't Hoff Institute for Molecular Sciences, Science Park 904, the Netherlands
[b] Centre for Analytical Sciences Amsterdam (CASA), Amsterdam, the Netherlands
[c] AI4Science Lab, Informatics Institute, University of Amsterdam, Science Park 904, the Netherlands
[d] AMLab, Informatics Institute, University of Amsterdam, Science Park 904, the Netherlands
[e] Computational Chemistry Group, Van 't Hoff Institute for Molecular Sciences, University of Amsterdam, Science Park 904, the Netherlands

ARTICLE INFO

Keywords:
Liquid chromatography
Retention modeling
Method optimization
Automated method development
Chromatographic response function (CRF)

ABSTRACT

Optimization algorithms play an important role in method development workflows for gradient elution liquid chromatography. Their effectiveness has not been evaluated for chromatographic method development using standardized comparisons across factors such as sample complexity, chromatographic response functions (CRFs), gradient complexity, and application type. This study compares six optimization algorithms - Bayesian optimization (BO), differential evolution (DE), a genetic algorithm (GA), covariance-matrix adaptation evolution strategy (CMA-ES), random search, and grid search - for the development of gradient elution LC methods. Utilizing a multi-linear retention modeling framework, these algorithms were assessed across diverse samples, CRFs, and gradient segments, considering two observation modes: dry (*in silico,* deconvoluted), and wet (search-based, requiring peak detection). The optimization algorithms were evaluated based on their data (i.e. number of iterations) and time efficiency. Of the algorithms compared in this study, DE proved to be a highly competitive method for dry optimization purposes in terms of both data and time efficiency. BO outperformed all other algorithms in terms of data efficiency and was found to be most effective for search-based optimization, which requires a low number of iterations (<200). However, BO was found to be impractical for dry optimization requiring a large iteration budget due to its unfavorable computational scaling. It was observed that both the CRF and the sample have a strong influence on the efficiency of the algorithms, emphasizing the need for better benchmark samples and highlighting the importance of assessing CRF-induced complexity in the optimization landscape.

1. Introduction

Gradient elution liquid chromatography (LC) is an indispensable tool for resolving complex chemical samples. However, separating samples containing many compounds requires costly method development, typically involving significant trial-and-error experimental work in order to tune many adjustable parameters such as the type of mobile- and stationary phase, the gradient program, and the pH [1]. To facilitate this method development, automated strategies have been proposed which can broadly be classified into two classes: retention modeling-based [2–5] or search-based methods [6–12].

Retention modeling describes the retention behavior of analytes through retention coefficients which are determined by interpolation of retention models obtained from scanning experiments [13]. Given these retention coefficients, the retention model can be used to simulate chromatographic separations under a broad range of chromatographic conditions (i.e., methods) [14]. Typically, a retention model can be established with as few as two or three measurements. These models can be used to optimize one or multi-step gradient methods (e.g. [5,15–18]) making it a promising tool for automated method development strategies. Some examples of software utilizing retention modeling are Drylab [19,20], ACD-Labs [21], PEWS [22], PIOTR [23], and MOREPEAKS

* Corresponding authors.
E-mail addresses: G.B.vanhenten@uva.nl (G.B. van Henten), B.W.J.Pirok@uva.nl (B.W.J. Pirok).
[1] Equal contribution.

https://doi.org/10.1016/j.chroma.2024.465626
Received 6 September 2024; Received in revised form 20 December 2024; Accepted 23 December 2024
Available online 25 December 2024

[24]. Search-based methods, on the other hand, rely on laboratory-conducted experimental evaluation in order to optimize separation. When strictly simulated chromatograms are used, e.g. known pure compounds signals, we refer to this as "dry" optimization, whereas consecutive real or simulated lab-based experiments containing single signal data can be referred to as "wet".

Although these methods are intrinsically different, they share similar components. For instance, both strategies rely on a numerical criterion that describes the overall separation quality, generally referred to as the chromatographic response function (CRF). The CRF typically incorporates scalars related to the number of detected peaks, the quality of separation, and the total method time [25]. Over the past decades, many different CRFs have been proposed and studies comparing the performance of different CRFs have been performed [25,26].

Besides the definition of the CRF, both optimization strategies require an algorithm that efficiently optimizes the CRF. In this case, efficiency refers to time and/or data efficiency [27]. Data efficiency refers to the number of evaluations (i.e., simulated chromatograms, or lab-based experiments) that need to be performed in order to reach a satisfactory state. Whereas time efficiency refers to the computation time required to find a satisfactory state. In search-based methods, one is typically more interested in data efficiency, as experiments need to be performed in real-time and use up valuable chemicals. However, in the setting of retention modeling, evaluations are fast and *in silico* (i.e., dry), and hence one is more concerned with the time efficiency of the algorithm.

In the past decades, a range of different works has focused on automated method development using different optimization algorithms [9,27–29]. For instance, in the setting of search-based approaches, Berridge et al. utilized simplex methods to optimize experimental parameters using a CRF that incorporated the quality of the separation, the number of detected peaks, and the experiment time [10]. Later works focused on evolutionary algorithms [6,7,30] and Bayesian optimization [8,31]. In the setting of retention modeling, works have focused on grid search methods [2], evolutionary algorithms [4,32], gradient-based methods [5] and Bayesian optimization [33].

Although most works were successful in finding satisfactory method parameters, the effectiveness of these approaches is hard to compare, as they were applied to vastly different samples (both in nature and complexity), used different CRFs, and optimized different parameters. Hence, it is not well-known what optimization algorithm works best in the setting of wet and dry optimization. Especially, the comparison between Bayesian optimization, which has shown to be typically sample efficient, and evolutionary algorithms, which are most commonly used in automated method development workflows, has not yet been made [29].

In this work, we report on a comparison of a range of optimization algorithms that have been used in method optimization and assess their comparative efficiency for method development workflows. Specifically, we compare a genetic algorithm (GA), differential evolution (DE), covariance matrix adaptation evolution strategy (CMA-ES), Bayesian optimization (BO), random search, and grid search. We base our study on a range of different samples, CRFs, gradient segments, and modes of observation. We include all the code used in this study, including the retention modeling framework, CRFs, sample generation methods, and all the implemented optimization algorithms in an open-source Python package that enables users to reproduce these results as well as optimize their own samples.

2. Computational methodology

To make the comparison as comprehensive as possible, a framework was created using multi-linear gradient retention modeling as a simulator (see Section 2.1), as this allowed for the prediction of a range of samples with a wide variation of retention behavior, but also under different gradient complexities (see Section 2.4). Here we opted for realistic chromatographic conditions (i.e. plate number, dead volume, dwell volume, etc.) to produce the chromatograms and corrected for gradient compression in the computation of the peak widths. Besides a variety of samples, discussed in Section 2.2, we also compared the performance of the optimization algorithms on a range of different CRFs discussed in Section 2.5 and using different modes of observation discussed in Section 2.3. The optimization algorithms and their implementation are discussed in Section 2.6.

2.1. Simulation of chromatographic separations

To predict the retention properties of sample components we utilize the linear solvent strength (LSS) model [34] which is defined as follows:

$$\ln k = \ln k_0 - S\varphi \tag{1}$$

where k is the retention factor, φ is the mobile phase modifier fraction, k_0 is the retention factor at $\varphi = 0$, and S represents the solvent strength parameter. The latter two parameters are generally fitted on a range of scanning gradients performed on a specific column and mobile phase setup [10]. To study the performance of the optimization algorithms under different complexities of the gradient program, we used the LSS model in the framework of multi-linear retention modeling which is derived and described in detail elsewhere [30]. The formulas used to calculate the retention times in the multi-linear retention modeling framework are included in Supplementary materials S-1. The model was implemented in-house in Python. This framework allows for the modeling of complex multi-gradient methods, with a growing number of optimizable parameters as the complexity of the gradient program increases. While the LSS model is employed in this study to simulate chromatographic behavior, it's important to note that various alternative modeling strategies exist for retention prediction in LC. Each has its own strengths and weaknesses, and implications for the method development process. A detailed description of alternative modeling strategies is beyond the scope of this work. For a more comprehensive overview of retention modeling techniques and tools, we refer the reader to the following reviews [35,36]. To describe the peak width (W) at 4σ we use the following expression in the case of an isocratic run [37]:

$$W = 4N^{-\frac{1}{2}} t_0 (1 + k_e) \tag{2}$$

where N is the column plate number, t_0 is the column dead time and k_e is the retention coefficient at the retention time t_R. In the case of gradient elution, we correct the gradient compression factor G so that:

$$W = 4GN^{-\frac{1}{2}} t_0 (1 + k_e) \tag{3}$$

We compute G according to the general expression for peak compression derived by Hao et al. [38]:

$$G^2 = \frac{k_e^2}{t_0 (1 + k_e)^2} \left(\frac{(t_D + t_{init})(1 + k_0)^2}{k_0^3} + \int_0^{t_R - t_0 - t_D - t_{init}} \frac{(1 + k_{\varphi(t)})^2}{k_{\varphi(t)}^3} dt \right) \tag{4}$$

where t_D is the dwell time, t_{init} is the isocratic initial time, and $k_{\varphi(t)}$ is the retention factor at $\varphi(t)$. Hao et al. analytically derived these equations for the LSS model under multilinear gradient elution [4], which we implemented in in-house Python code. The chosen values for the dwell time, dead time, and plate number are shown in Table 1 and are kept fixed for all experiments in this work.

Table 1
Fixed parameters used for the chromatographic simulator.

Parameter	Value	Units
Dwell time, t_D	0.355	min
Dead time, t_0	0.479	min
Plate number, N	5000	–

Given the predicted retention times and peak widths using the framework described above, we simulate peak profiles as Gaussian peaks and assume compounds have equal concentration.

2.2. Generating retention parameters

In an attempt to make general conclusions about the relative performance of the optimization algorithms, we studied 11 different samples. For samples 1-8, we uniformly sample the retention parameters of 35 components from 8 specific intervals (see Table 2). For uniformity with prior conducted research, we chose to use the same retention parameter sets as those used in the CRF comparison study by Tyteca et al. [14] for the initial 8 samples. While these intervals cover a broad range of retention behavior, these samples comprise a single distribution, which generally does not benefit that much from multiple gradient segments. Therefore, we have also introduced samples 9 and 10, which are both generated from sample 8. For sample 9, we have taken the retention parameters from sample 8 and multiplied the k_0 values by e^7 to make these compounds elute at higher modifier concentrations. From this combined list of retention parameters, we have then sampled 60 compounds, effectively leading to a sample containing two distinct distributions. For sample 10, we have created an additional set of retention parameters from sample 8 by multiplying the k_0 values by e^{14}. We then randomly sampled 90 compounds from the three sets of retention parameters, effectively leading to three distinct distributions of compounds. Sample 11 is a set of retention parameters that were experimentally determined in [5] and comprises 96 compounds. Note that sample 11 contains several weakly retained compounds capped at $S = 50$. This was the boundary used in the retention modeling study. All samples are displayed in Fig. 1, using a scatter plot of the $\ln k_0$ and S. All the samples contain a fixed number of compounds: 35, 60, 90, and 96 for samples 1–8, 9, 10, and 11, respectively.

2.3. Chromatogram processing

The chromatograms that were generated using the framework described in Section 2.1 were read out using in-house Python code. An example chromatogram is shown in Fig. 2, and examples for all samples are provided in Supplementary materials S-2. We consider two ways of reading out the generated chromatograms. In the first scenario, we directly use the retention times and peak widths as predicted by the retention modeling framework. This corresponds to the dry setting of retention modeling or the setting where we have idealized spectral detection, i.e., the retention time and peak width of all compounds in the mixture are always known. In the second scenario, we treat the predicted chromatogram as 1st-order chromatographic data (e.g. single-trace UV-VIS data) on which we directly perform peak detection. This scenario is often the case for search-based method development approaches [5,27]. In this setting, referred to as wet, compounds that fully overlap are not separately detectable. This will change the number of observed peaks between iterations, which in turn will affect the CRF and the complexity of the optimization process. A simplistic visualization of the difference between wet and dry is included in Supplementary materials S-3. In the wet scenario, peak detection is performed by finding the start, end, and maximum of each peak in the convoluted signal, see Fig. 3. In the case that peaks are not fully separated and thus overlap, the peak valley is used as the end (or start) of the peak. For these cases, this results in an underestimation of the peak width when there is significant overlap. The peak width used in the computation of the CRFs is the peak width at 13.5% of the peak height maximum, which corresponds to 4σ (See Fig. 3). This provides two extreme corner cases against which the CRFs and optimization methods used in this work are tested.

Table 2
Ranges of k_0 and S values for the generation of different samples.

Sample	k_0	S
1	[200, 1000]	[8, 15]
2	[500, 1000]	[10, 20]
3	[1000, 2000]	[5, 15]
4	[200, 1000]	[15, 20]
5	[10, 2000]	[5, 10]
6	[500, 2500]	[10, 20]
7	[1000, 3000]	[5, 15]
8	[200, 700]	[15, 25]

2.4. Parameters of multi-linear gradient program

Fig. 4 illustrates an example of a multilinear gradient profile and the parameters that are optimizable in our study. Here the φ values represent the modifier concentrations at each turning point in the gradient profile. The Δt values represent the duration of each gradient segment, and t_{init} represents the duration of the initial isocratic segment. For a gradient program with N segments, this leads to an optimizable vector containing $N + 1$ φ values, $N\Delta t$ values, and one t_{init} value, this is shown for $N = 3$ in Fig. 4. The ranges these parameters can take are bounded to be between 0 and 1 for φ, between 0.1 and 20 for Δt, and between 0 and 5 for t_{init}. Further, it is assumed that after the last gradient segment, there is a hypothetically endless isocratic segment at a modifier concentration of the last specified φ value. As it is not straightforward to implement inequality constraints in each studied optimization method, we allow for gradient segments with a negative slope, although it is generally known that this does not lead to robust method parameters.

2.5. Description of chromatographic response functions

As the performance of optimization algorithms can be dependent on the function to be optimized, we study a range of different CRFs to assess this dependency. An overview of all the studied CRFs can be viewed in Table 3 [8,10,14,39–43]. We primarily chose these CRFs to create a diverse range of what their functional landscapes might look like but also made sure that these CRFs indeed have been used in literature. As in Tyteca et al. [14], we distinguish between CRFs that do or do not contain a time optimization component (category I and II, respectively). In addition, we also consider CRFs that monotonically increase with the number of observed compounds, or do not (category B and A). Having CRFs that increase with the number of compounds is especially interesting for the wet scenario (see Section 2.3), where it is unknown how many compounds there are in the sample. As in this case, measurements that have more separated peaks will have higher scores. Without this property of the CRF, these measurements will have highly similar scores, irrespective of the number of separated compounds. All the CRFs used in this study use the elemental separation criteria of resolution or Kaiser's peak-to-valley ratio. We define the resolution between neighboring peak pairs $i$and $i+1$ as follows:

$$R^*_{s,i} = \frac{2\left(t_{R,i+1} - t_{R,i}\right)}{W_i + W_{i+1}} \tag{5}$$

Where $t_{R,i}$ is the retention time of peak i and W_i is the width of peak i at 13.5% of its peak height, see Fig. 3. We further assume that resolution values that are higher than a certain $R_{s,req}$ do not benefit the separation, and therefore, we always normalize the resolution so that:

$$R_{s,i} = \left\{ \begin{array}{c} R_{s,i} \text{ if } R_{s,i} < R_{s,req} \\ R_{s,req} \text{ if } R_{s,i} \geq R_{s,req} \end{array} \right\} \tag{6}$$

where $R_{s,req} = 1.5$, is used throughout this study. We define Kaiser's peak-to-valley ratio (f_i/g_i) as follows:

$$f_i \Big/ g_i = 1 - \frac{h_v}{1/2 \cdot (h_{min} + h_{max})} \tag{7}$$

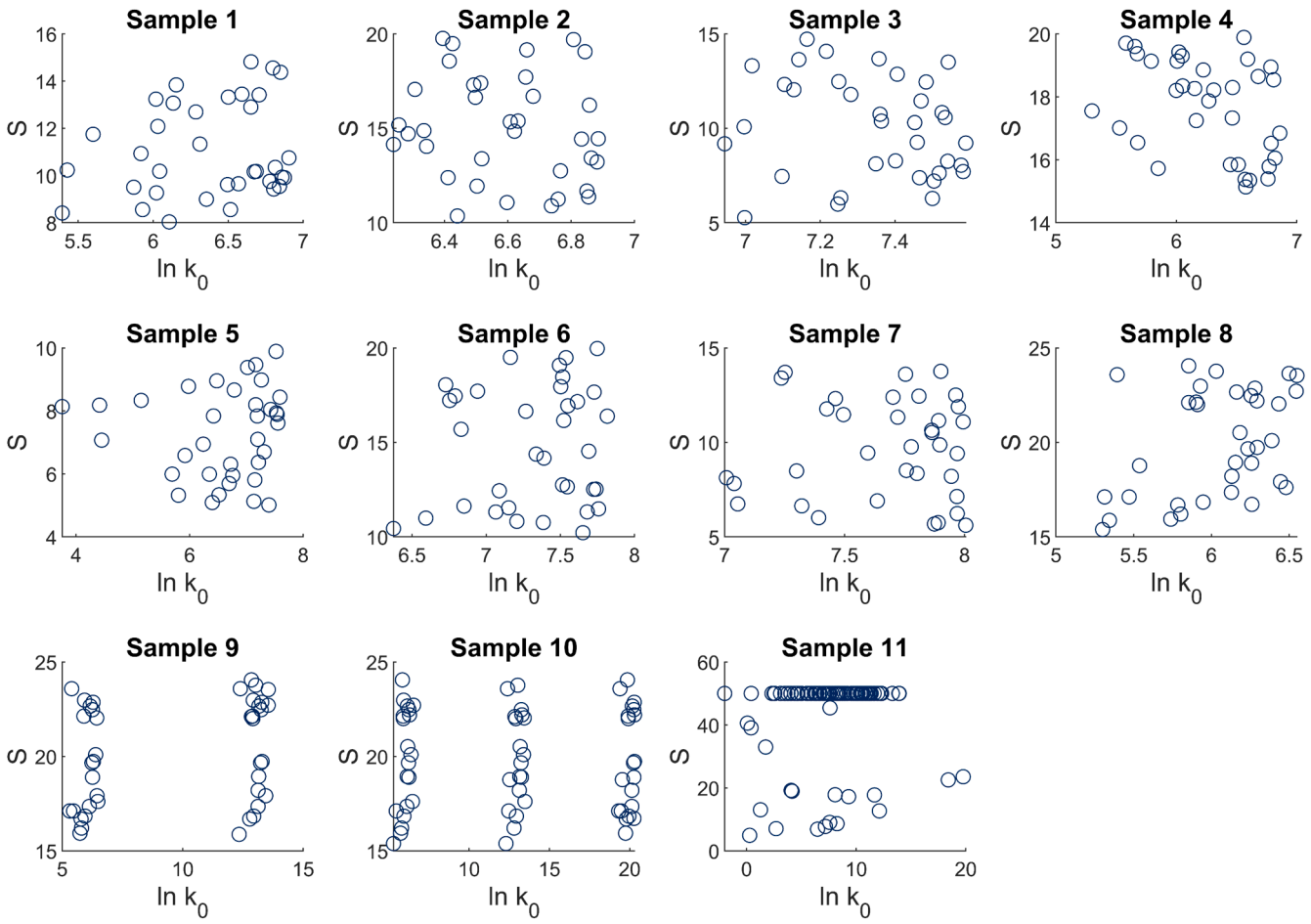

Fig. 1. Scatter plot of the ln k_0 and S for all the created samples used in the algorithm comparison.

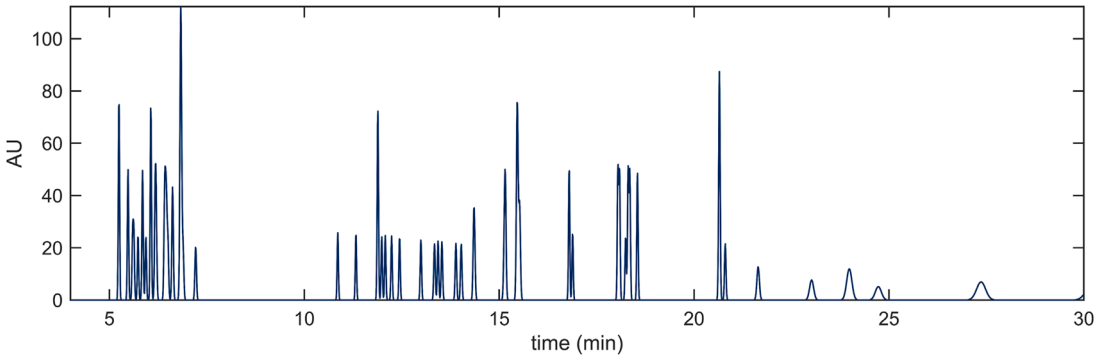

Fig. 2. Example of a simulated chromatogram of sample 10 for a single-segment linear gradient of 0–100% strong modifier and a gradient time of 30 min.

where h_v is the valley between two peaks i and $i+1$, h_{max} is the most intense peak, and h_{min} is the least intense peak between two peaks i and $i+1$, as illustrated in Fig. 3. Other criteria that are used by several CRFs are the number of observed peaks (N_{obs}), the retention time of the first and last eluted compounds ($t_{R,first}$ and $t_{R,last}$), a specified minimum and maximum time ($t_{min} = 2$ min and $t_{max} = 60$ min).

For (sub-)CRFs that do not contain a time component, it is possible to obtain experiments with a high CRF score that have an unrealistically long experiment time. Therefore, for these CRFs, we incorporate a soft time penalty by ignoring peaks that elute after t_{max} by multiplying with the indicator function $1(t_{R,i} < t_{max})$ which is 1 if $t_{R,i} < t_{max}$ and 0 otherwise.

2.6. Optimization methods

We focus on a select group of optimization algorithms: genetic algorithm (GA), differential evolution (DE), covariance matrix adaptation evolution strategy (CMA-ES), Bayesian optimization (BO), random search, and grid search. These algorithms were chosen for several reasons. Firstly, all of these methods can be used for wet and dry method development, as all of these methods can do derivative-free optimization. In addition, all the algorithms have already been applied in LC for method development applications, reinforcing their relevance and utility in this domain. Lastly, some methods, though not necessarily exclusive to chromatography, are recognized as state-of-the-art optimization techniques, making them potentially interesting candidates for

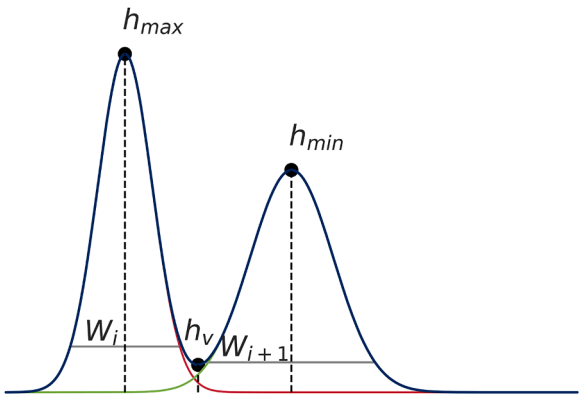

Fig. 3. Illustration of a chromatogram read-out and important measures used to quantify the separation of two peaks. The blue line denotes the chromatogram of a convoluted signal of two peaks (shown in green and red). The peak width is determined at 13.5% of the peak height maximum.

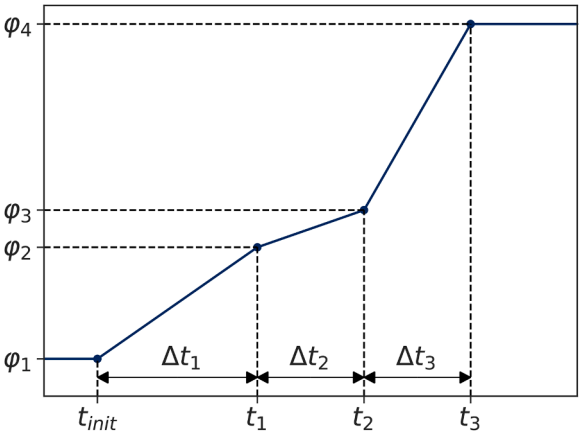

Fig. 4. Example of a multilinear gradient profile with 3 tunable segments.

chromatographic optimization. For GA we used the implementation from the *PyGad library* [44], for DE and grid search we used their respective implementations in *SciPy* [45], for CMA-ES we used the *CMA Python library* [46], for BO we used the implementation in *Scikit-Optimize (Skopt)* [47], and for random search we used an in-house Python implementation.

The full description of these methods is beyond the scope of this work; we provide a brief description of each optimization algorithm and additional implementation details in the Supplementary materials S-4 and references to relevant literature are included there. In our exploration of the various optimization methods, we have made a conscious decision to maintain default settings for each algorithm. This choice mirrors the typical approach of a practitioner who might rely on out-of-the-box configurations. Concurrently, in an effort to ensure fairness and maintain comparability across the algorithms, efforts were made to harmonize certain settings wherever feasible. To make a fair comparison of computational runtime, all computations were performed on an HPC cluster with AMD Rome 7h12 CPUs, where each task could utilize 18 CPU cores with an individual memory allocation of 12GB.

3. Results and discussion

3.1. Rationale for the conducted experiments

A framework was constructed to investigate the efficiency at which different optimization algorithms were capable of optimizing chromatographic gradients. To achieve this, an off-line workflow was designed where 11 different computational samples were employed using simulated retention data. The goal was to test 6 different optimization algorithms against four different gradient program designs (with 1, 2, 3, or 4 gradient segments) and 10 different CRFs from literature. This means that each chromatographic method-optimization experiment was conducted with each combination of optimization algorithm, CRF, gradient-program design, and sample. In addition, as some of the optimization algorithms are dependent on the random initialization and the random seed (GA, DE, CMA-ES, BO, and random search), these were run for 10 trials with different random seeds. We will refer to one combination of CRF, gradient-program design, and sample for the entire iteration budget as a "search-run". A schematic overview of the complete workflow is displayed in Fig. 5.

All experiments were tested in two different scenarios. The first is referred to as dry, where the retention time and peak width of all compounds are known *a priori*. This scenario mirrors the conditions encountered by practitioners in an *in silico* approach, such as with retention modeling studies, where the established retention models allow for a virtual assessment of a broad range of chromatographic conditions based on only simulated measurements. The second scenario is referred to as wet where the algorithm is provided with unprocessed data that features significant co-elution (e.g. initial scouting experiments). This approach is most similar to what practitioners will encounter in the laboratory with real-world LC experiments as well as

Table 3
Overview of all CRFs considered in this study. The CRFs are distinguished into two main categories based on whether they contain a time optimization component or not (Category I and II, respectively). Additionally, within each main category, CRFs are further classified based on whether they monotonically increase with the number of observed compounds (Category B) or do not have this property (Category A). In CRF5, x, a, and b are prefactors which were all set to 1. In CRF6, we choose values $b_0 = 3.93$, b1 = 3.66, b2 = −0.0406, and b3 = −4.646. In CRF8 we use values a = b = 0.5.

Category I-A	**Category II-A**
$CRF_1 = \sum_{i=1}^{n_{obs}-1} \frac{R_{s,i}}{R_{s,req}} \cdot 1(t_{R,i} < t_{max})$ [8]	$CRF_5 = n_{obs}^x + \sum_{i=1}^{n_{obs}-1} R_{s,i} - a\lvert t_{max} - t_{R,last}\rvert + b(t_{min} - t_{R,first})$ [10]
$CRF_2 = \prod_{i=1}^{n_{obs}-1} \frac{R_{s,i}}{R_{s,req}} \cdot 1(t_{R,i} < t_{max})$ [39]	$CRF_6 = \left(\prod_{i=1}^{n_{obs}-1} S_i\right)^{\frac{1}{n_{obs}-1}} \cdot g$, where:
$CRF_3 = \sum_{i=1}^{n_{obs}-1} f_i/g_i \cdot 1(t_{R,i} < t_{max})$	$S_i = \frac{1}{1+e^{-b_0 * R_{s,i}+b_1}}$ & $g = \frac{1}{1+e^{-b_2 * t_{R,last}+b_3}}$ [41]
$CRF_4 = \prod_{i=1}^{n_{obs}-1} f_i/g_i \cdot 1(t_{R,i} < t_{max})$ [40]	
Category I-B	**Category II-B**
$CRF_7 = n_{obs} + \frac{\sum_{i=1}^{n_{obs}} R_{s,i}}{R_{s,req} \cdot (n_{obs}-1)}$ [14]	$CRF_9 = n_{obs} + \sum_{i=1}^{n_{obs}-1} f_i/g_i - \frac{(t_{R,last} - t_0)}{t_{R,last}}$ [42]
$CRF_8 = n_{obs} + a \cdot \sqrt[n_{obs}]{CRF_4}\, b \cdot \frac{(n_{obs}-1)^{CRF_3}}{n_{obs}-1}$ [43]	$CRF_{10} = n_{obs} + \frac{1}{t_{R,last}} \cdot \prod_{i=1}^{n_{obs}-1} f_i/g_i$ [14]

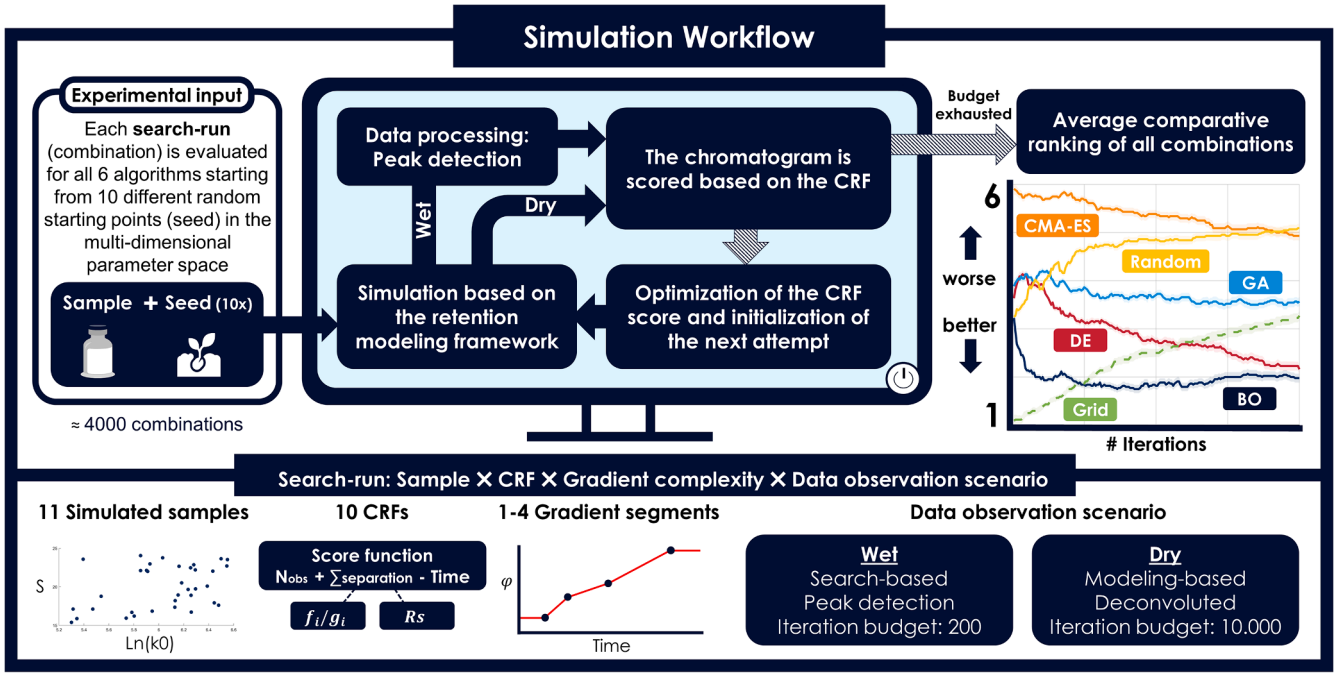

Fig. 5. Schematic overview of the workflow deployed in this study.

what an automated workflow would encounter. For the wet scenario, peak detection is required after each iteration, which means overlapping compounds cannot be resolved. All optimization experiments for both simulated scenarios were evaluated and compared on their ability to efficiently arrive at an optimum.

The definition of efficiency varies depending on the application for which the algorithm is used. For instance, in the dry scenario with fast *in silico* measurements, the workflow benefits more from a time-efficient approach, enabling a higher number of evaluations within an acceptable timeframe. Whereas in the wet scenario, the data efficiency becomes more important as measurements are expensive and time-consuming. Therefore, the algorithms were evaluated based on data efficiency, the number of iterations, and the time efficiency, which refers to the computation time needed.

To assess the data and time efficiency of the algorithms for their general applicability in method development workflows, a comparative analysis of the algorithms based on their average ranking was conducted. This ranking represents how frequently the optimization algorithms succeeded in finding the best solution (the highest CRF value) relative to the other algorithms. Thus, a rank of exactly 1 indicates that the algorithm always found the best scoring solution and a rank of exactly 6 signifies that it found the least favorable solution out of all the studied algorithms. In this manner, the ranking reflects the algorithm's overall capability for general applications. In other words, the effect of the specific factors is minimized. It is important to emphasize that for this comparison study, we excluded the impact of the CRF in relation to the gradient parameters obtained, and only studied the ability of the optimization algorithm to find the highest score regardless of whether the CRFs performance is sensible in terms of the obtained separation. This is due to the complexity of assessing the performance of a CRF, as it is highly dependent on the specific goals and preferences of the analyst.

The following sections discuss the data and time efficiency of the algorithms and are divided into two parts. Section 3.2 focuses on optimization based on the dry scenario, with a high number of iterations available, whereas Sections 3.3 and 3.4 focus on optimization based on the wet scenario similar to the situation encountered with single trace data and real experiments. A visualization of the chromatograms obtained from a single search run is included in Supplementary materials S-5.

3.2. Algorithm performance for experiments conducted in the dry scenario

To study the performance of the algorithms in the dry scenario we allowed each algorithm a total of 10,000 iterations for each search-run. The following two restrictions were considered: I) In the Dry scenario, we only consider the resolution-based CRFs, as CRFs using Kaiser's peak-to-valley ratio are inapplicable to deconvoluted data given that retention times and peak widths are individually modeled for each compound and any peak overlap thus is known *a priori*. II) We observed that CRF 7 displayed practically uniform convergence across all algorithms. This was due to a strong influence of the term related to the number of compounds "*n*" in the CRF, which is constant in the fully deconvoluted setting. Consequently, CRF 7 was omitted from the average ranking for the dry scenario.

3.2.1. Algorithm data-efficiency in the dry scenario

Fig. 6 displays the ranking for each algorithm for the dry scenario. Here, the ranking is averaged over the performance of all computed combinations of the 11 samples, four CRFs, and varying the number of gradient segments (1-4) to study the general performance of the algorithms. As the BO algorithm scales cubically with the number of iterations, it was limited to 200 iterations to keep the computation time manageable given the large number of search-runs that had to be performed. For the ranking after 200 iterations, the BO score was fixed at the final score obtained at 200 iterations (as indicated by the dashed lines in Fig. 6). Similarly, the value of the grid search was set to the best value found on the entire grid and is fixed from the first iteration in Fig. 6. Since there is no notion of "order" in the computation of the grid search, we only utilize grid search as a way of investigating whether the other optimization algorithms can effectively beat the grid search in a lower number of iterations than was performed in the grid search (see Supplementary materials S-4.5 about details regarding the grid search).

From Fig. 6, it is evident that all evolutionary algorithms (CMA-ES, DE, GA) outperformed the grid search after around 500 iterations, which is rather impressive given the, in some cases, 2 orders of magnitude larger number of iterations executed for the grid search. This is consistent with the observation that grid searches are becoming less practical due to factors such as excessively long computation times or insufficient granularity to identify meaningful parameter combinations, especially

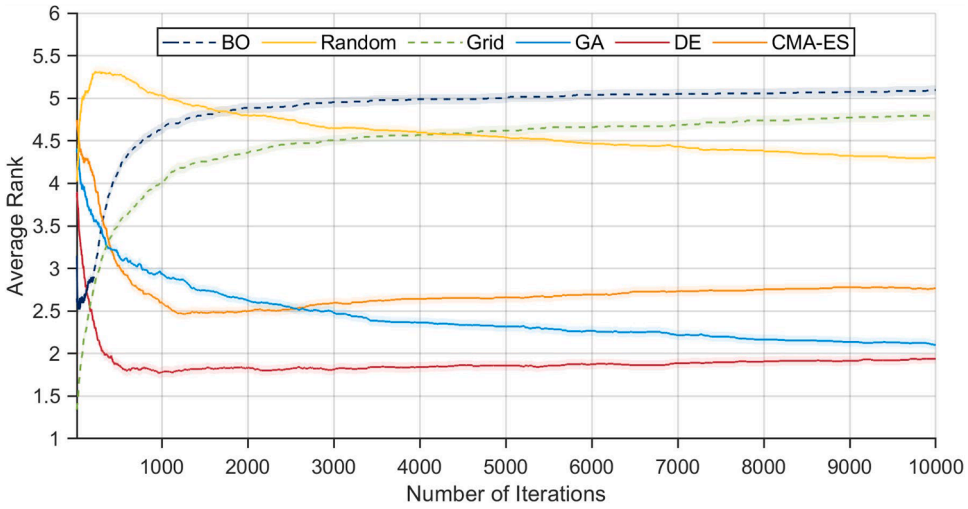

Fig. 6. Comparison of the ranking of the optimization algorithms averaged over all studied search-runs for the dry scenario. All algorithms were run for 10,000 iterations except for grid search and BO. BO was run for 200 iterations and after this iteration, its final score was fixed and used to compute ranking (as shown by the dashed line). Similarly, the grid search was fixed after the first iteration with its best-found value, also indicated by the dashed line. The standard error of the average ranking is shown by the shaded border.

in settings with many adjustable parameters. It was also observed that random search improves over grid search in around 4200 iterations. Random search can outperform grid search when only a limited number of optimization parameters affect the final performance of the separation [48], see also Fig. S2. In such a case, the optimization is said to have a lower intrinsic dimensionality. This was also observed in this study when comparing the average ranking per gradient-segment number, see Fig. 7, where it is shown that random search surpasses grid search faster at high parameter settings than at low parameter settings.

Likewise, if random search is competitive with the other algorithms, it could indicate that the optimization problem is inherently simple, with very large volumes in the parameter space that are optimal and can easily be explored through random selection. This is not the case in this study, as all evolutionary methods improved over the grid search significantly faster and remained better than random search over the entire optimization budget. DE performed the best and obtained the best average ranking after only 1000 iterations and did not get surpassed by the other algorithms in the remaining iterations. This is in line with the established efficacy of DE in addressing these types of problems [32], and the general notion that DE likely outperforms GA in continuous optimization landscapes [49], especially for higher dimensional problems. This was also observed when studying the ranking plot as a function of the number of gradient segments, where the performance of GA in comparison to DE and CMA-ES decreased in data efficiency with each added gradient segment (and thus increased dimensionality).

It is important to note that the number of gradient segments does not necessarily contribute positively to improving the CRF scoring. It is likely that for some samples, a single gradient step may already be effective in separating all components in the sample. In these cases, the non-required segments will add additional parameters that increase the complexity of the optimization process but may not necessarily lead to better solutions, as is showcased by the reduced relative performance of GA when the number of gradient segments increases. Ideally, in an optimization process, one would be able to determine the number of gradient segments required to perform the separation, thus eliminating the additional complexity, and making the optimization algorithm more effective.

The performance of CMA-ES in the dry optimization process is likely due to the tendency of early convergence that is associated with the selection process as the covariance matrix adapts. Inherently, this tendency of early convergence can be positive, as it may lead to fast convergence towards the optimal solution. However, in this study, it neither proved more data efficient with a lower number of iterations (1-500) nor capable of finding the true optimum, as DA performed better,

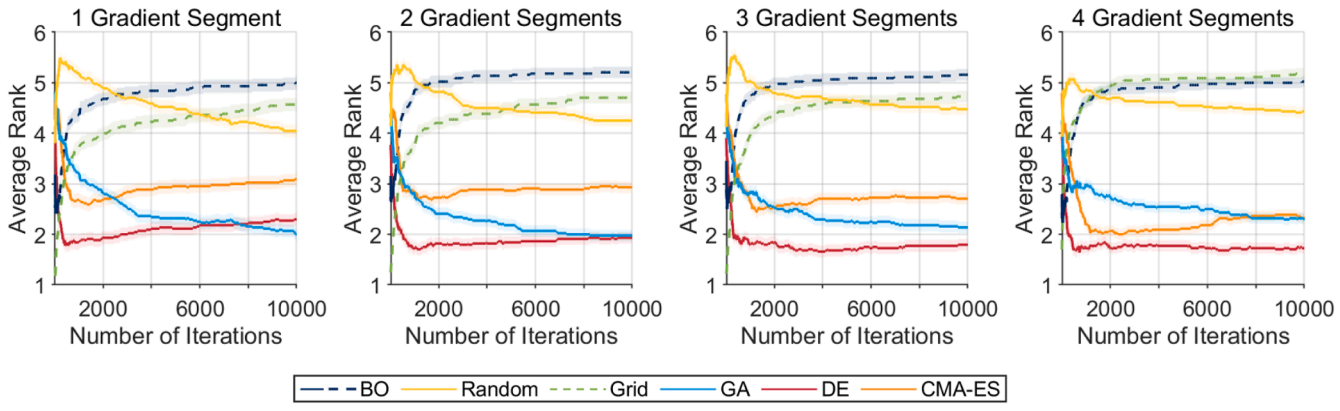

Fig. 7. The ranking for each level of gradient complexity (1–4 gradient segments) averaged over all studied CRFs and samples in the dry scenario. All algorithms were run for 10,000 iterations except for grid search and BO. BO was run for 200 iterations and after this iteration, its final score was fixed and used to compute ranking (as shown by the dashed line). Similarly, the grid search was fixed after the first iteration with its best-found value, also indicated by the dashed line. The standard error of the average ranking is shown by the shaded border.

and GA consistently caught up or improved over CMA-ES given enough iterations.

BO was the most effective at the lowest range of iterations, and was only surpassed by the grid search, but is impractical after further iterations in this data regime due to its cubically scaling computation cost with the number of iterations when using a Gaussian process regression model. This will be discussed further in the next section.

3.2.2. Algorithm time efficiency in the dry scenario

Fig. 8 displays the average cumulative computation time of the optimization algorithm in the dry scenario for the first 200 iterations. As mentioned in Section 2.6, it is important to note that all computations were performed using a similar amount of compute and memory. For grid search the computation time of the entire evaluation is shown as a horizontal dashed line. The average run time is strongly influenced by the number of iterations evaluated for the grid search, which was scaled with the dimensionality of the optimization problem, see Supplementary materials S-4.5. As a result, the average run times for the individual gradient complexities of $n = 1$, 2, 3, and 4 are approximately 6 s, 9 s, 51 s, and 52 s, respectively. As the random search algorithm is simplistic in nature, the runtime of the algorithm gives an indication of the computation overhead of the retention modeling and chromatogram simulation. The additional difference in computation time observed for the other optimization algorithms can be attributed to their respective complexities.

The cubical scaling complexity of Bayesian optimization when using a Gaussian process regression model becomes immediately apparent from Fig. 8A. Where the overhead of the algorithm quickly surpasses that of the simulator. The evolutionary algorithms, CMA-ES, DE, GA, showed an attractive linear scaling with the number of iterations, with runtimes of 12 s, 11 s, and 9 s respectively but each showed slightly different overheads. CMA-ES uses a more complex selection strategy which likely led to the minor increase in computation time, followed by DE and GA. However, this difference is somewhat negligible, as the data efficiency and performance should be favored here over the minor increase in runtime. Indicating that DE is a highly competitive time efficient method for the task of optimizing *in silico* samples.

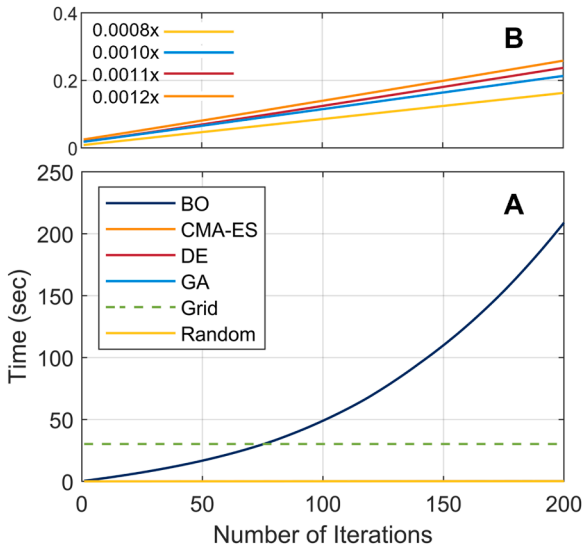

Fig. 8. Cumulative computation time (s) of the optimization algorithms averaged over all studied samples, gradient complexities, and the applied CRFs for the dry scenario for the first 200 iterations (panel A). The grid search was fixed after the first iteration, indicated by the dashed line. Panel B displays a zoom-in of the first 0.4 s and contains the linear formula for the respective algorithms with x being the number of iterations.

3.3. Algorithm performance for experiments conducted in the wet scenario

The convoluted optimization setting is most similar to what a practitioner will encounter in the laboratory with real measurements. For "wet" experiments, one is typically interested in reducing the number of measurements to reduce cost, time, or solvent consumption, and thus requires a data-efficient algorithm. We therefore chose to focus on illustrating the 25 and 200 iteration limits in Fig. 9. 25 iterations correspond to a realistic number of measurements achievable in a method-development process where each measurement takes in the order of hours. Meanwhile, 200 iterations serve as the upper limit of the number of measurements that could realistically be performed when dealing with experiments in the range of minutes. This choice is on the high side, to emphasize the impact on both data efficiency and computational time. Fig. 9, shows the ranking for each algorithm averaged over all the different scenarios for the Wet scenario in the range of 1–200 iterations. It can be seen here that the grid search benchmark performed the best, as was expected given the large number of function iterations performed in the grid search. However, even with the small number of algorithm iterations, grid search was occasionally outperformed by the other algorithms. On average, BO outperformed the other algorithms in terms of data efficiency, followed by DE, GA, and CMA-ES, respectively. The data efficiency of BO is particularly noticeable in the range of 1–25 iterations, where it quickly catches up with the performance of the grid search. DE after 200 iterations comes close to the performance of BO and surpasses the grid search, which is rather impressive for an evolutionary algorithm that is generally expected to underperform without sufficient iterations. CMA-ES showed the same tendency as in the dry optimization where surprisingly the CMA selection strategy does not lead to rapid improvement. All optimization algorithms outperform random search as the number of iterations increases, confirming the suitability of the algorithms for wet method optimization purposes. The relative performance of random search further indicates that the performance landscapes can be considered complex and that it likely contained narrow and sparse optima. Time efficiency-wise we showed in Section 3.2.2, Fig. 8, that BO is considerably slower than the other optimization methods, however, in the case of real experiments that are on the minute-hour time scale, its computation time is negligible. Therefore, in terms of data efficiency BO, proved most competitive for automated method development purposes that rely on real measurement input.

3.4. Factors influencing the optimization performance in the wet scenario

Since the performance of the optimization algorithm depends on the shape and complexity of the response function landscape, it is important to evaluate the dependency of the CRFs, samples, and the influence of the gradient complexity (1–4 gradient segments). In the dry scenario, the influence of the sample and CRF was less pronounced due to the number of components being known. In the wet scenario, due to the required peak detection for convoluted components and the resulting varying number of observed peaks, the individual contributions and discrepancies can become more apparent. The following sections focus on the individual contribution and discrepancies of the CRF, sample, and number of gradient segments, for the wet setting.

3.4.1. Influence of the CRF

While the overall rankings indicate which algorithm most often finds the best solution for general use, it is still possible for one optimization algorithm to significantly outperform another for a particular type of landscape, e.g., one with many local optima. In some cases, this can be linked to the used CRF, where a high-ranking performance of the grid search indicates that the CRF produces a difficult-to-navigate landscape, and where the strong performance of DE/BO indicates that there are many spiky optima which the grid search cannot find with its current

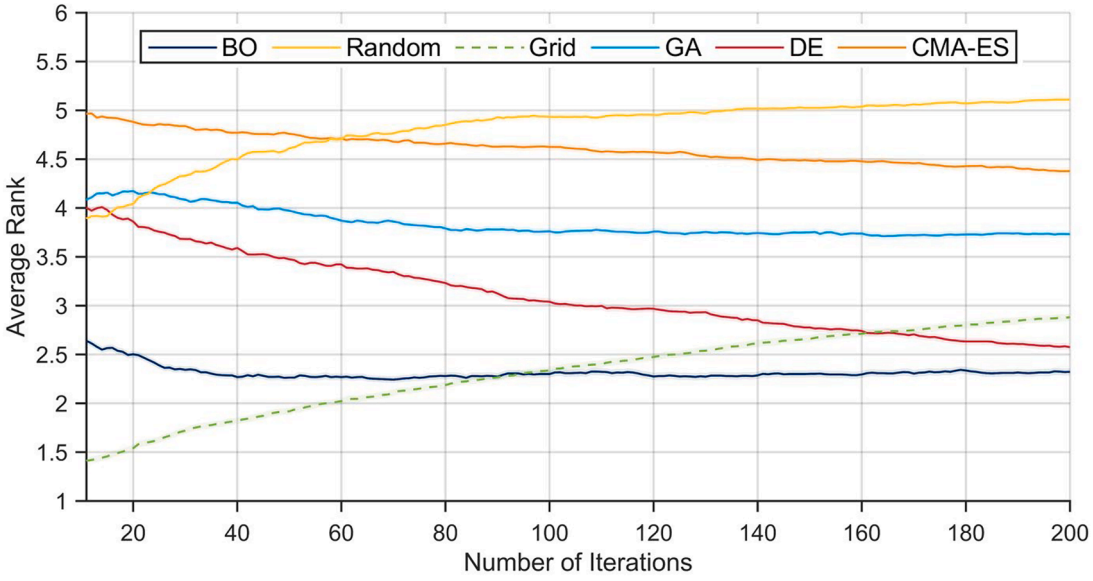

Fig. 9. Comparison of the ranking of the optimization algorithms averaged over all studied search-runs for the wet scenario. All algorithms were run for 200 iterations except for grid search which was fixed after the first iteration with its best-found value, indicated by the dashed line. The standard error of the average ranking is shown by the shaded border.

coarseness. The influence of the CRF is especially noticeable for ill-conditioned problems, where a small change in the input data or parameter leads to a large change in the solution score. Two examples of this are depicted in Fig. 10, with the product of Kaiser's valley-to-peak ratio (CRF 4) and the product of resolution (CRF 2). Due to both CRFs being based on multiplication, a single overlapping component would result in a significantly large change in the score value, or the case of a completely overlapping peak, even a zero value. For CRF 4, CMA-ES performed significantly better than observed in the average ranking and the relatively higher ranking of the random and grid search indicates that the CRF indeed produces a spikey solution landscape in which the CMA-ES and BO perform better. For CRF 2, the evolutionary algorithms likely struggle with local minima as a small change in parameters for a single component significantly reduces the landscape on all sides. Therefore, the grid search produces a higher total score for this CRF. See Supplementary materials S-6 for the ranking plots of all individual CRFs.

Clearly, the descriptors in the CRF can have a big impact on the final performance of the optimization algorithm. Therefore, it is not unthinkable that although a CRF might best describe the chromatography that the practitioner had in mind, it might lead to suboptimal performance for an optimization algorithm. For example, a CRF that combines many metrics to describe a separation may form a very complex landscape in which the best solution is not found in a reasonable computational time and may be outperformed by a simpler CRF that identifies a good solution more successfully at a potential cost of slightly less effective method parameters.

As a consequence, future work on the quality of (new) CRFs in automated method development processes should also take into account the complexity of the CRF in the evaluation of their performance. Simultaneously, the rapid development and increase in data and time efficiency of the algorithms can also tackle this issue if the optimization algorithms become more efficient at finding the optima in complex landscapes.

3.4.2. *Influence of sample*

In most cases, the effect of the sample showed little individual impact on the trends of algorithm ranking. However, in the case of samples 9–11, which all contained multiple clusters and an increased number of components, there was a break in the trend. Fig. 11 displays the ranking

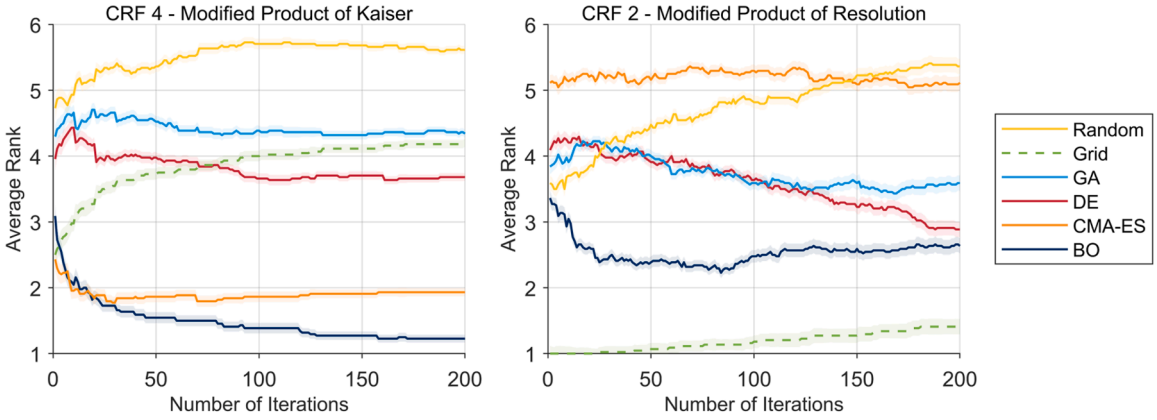

Fig. 10. The ranking of CRF 4, and CRF 2, averaged over all samples and gradient complexities in the wet scenario. All algorithms were run for 200 iterations except for grid search which was fixed after the first iteration with its best-found value, indicated by the dashed line. The standard error of the average ranking is shown by the shaded border.

averaged per sample of samples 10 and 11, the complete overview of all samples is added in Supplementary materials S-7. It is observed in Fig. 11 that CMA-ES on average performed worse than the random search, which is somewhat unexpected. A potential reason for this is the lack of tuning for the CMA-ES. Given that samples 9–11 contain multiple distributions, it would benefit from multi-linear gradients that combine shallow and steep gradient-segment blocks to accommodate all distributions in the sample. Therefore, optimal gradient parameters are likely to be structured (from low to high) with the exact retention behavior of the distributions leading to subtle changes in what is optimal. This structure is not reflected in the default initial conditions of CMA-ES, and it may therefore take a considerable number of iterations for the algorithm to reach more favorable regions of the method parameters. Additionally, this also increases the chance that the algorithm gets stuck in a local optimum. Meanwhile, DE, GA, and BO can make larger jumps in the method-parameter space and are therefore less affected by the optimization landscape for samples 9–11.

Although we studied 11 samples in this work, ranging from single distributions with different retention ranges, to multiple distributions and retention parameters determined from experimental measurements. We still feel that it remains difficult to assess the influence of the sample complexity on the optimization algorithm's performance. Partly because it will remain dependent on the choice of CRF, and the number of gradient segments. But also, because the 11 studied samples do not fully capture the space of samples that one can encounter in the laboratory. In this study, the range in sample size was limited to 30–96 components, but it is conceivable that a steep decrease or increase could reveal greater differences between the algorithm scores depending on the studied CRF. We believe the community would benefit from a better (centralized) benchmark set of retention parameters where optimal gradient conditions, both in the complexity (i.e., number of segments) and their modifier concentrations, and our group will focus on this in the future. In addition, there could also be alternative methods to sample retention parameters, which could create samples that are more complex to solve and might require more intricate gradient programs to resolve.

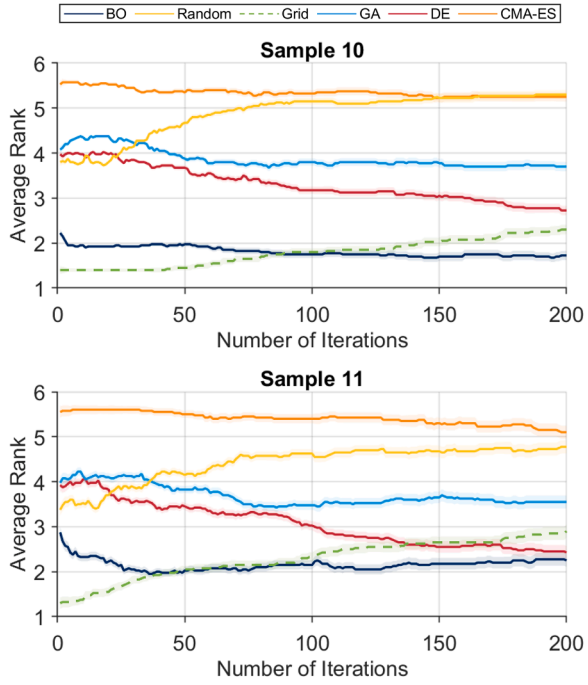

Fig. 11. The ranking of sample 10 (top panel) and sample 11 (bottom panel), averaged over all CRFs and gradient complexities for the wet scenario. All algorithms were run for 200 iterations except for grid search which was fixed after the first iterations with its best-found value, indicated by the dashed line. The standard error of the average ranking is shown by the shaded border.

3.4.3. *Influence of the number of gradient segments*

The ranking averaged per number of gradient segments in the wet scenario is shown in Fig. 12. Here, the addition of a gradient segment did not significantly impact the optimization performance ranking for the wet scenario, with BO performing better than DE in all cases, followed by GA and CMA-ES, and with random search performing the worst. The shortcomings of the algorithms, specifically GA, regarding the additional dimensionality introduced by increasing the number of gradient segments, are not as apparent as was observed in the dry scenario (see Section 3.2.1 and Fig. 7) due to the low number of iterations.

It should also be noted that as the number of gradient segments increases, the mesh of the grid search becomes increasingly coarse due to the number of grid points (iteration budget) being fixed. It would therefore be expected that the grid search performance deteriorates with increasing gradient segments, and this is indeed observed in Fig. 12, where the grid search was surpassed in ranking faster when a higher number of gradient segments was used. Again, highlighting that using population-based methods is highly advised when dealing with many optimizable parameters, and that the use of grid search becomes unreliable and unfeasible.

The results of this study suggest that the number of segments had little effect on the choice of algorithm for optimization purposes in the wet scenario, which could also be related to the studied samples, as discussed in Section 3.4.2. It is possible that the influence would be more significant for a larger number of gradient segments, but this would not be commonly considered in practice for LC gradient optimization.

4. Conclusion

In this work, we have created a framework utilizing multi-linear retention modeling to compare the performance of BO, DE, GA, CMA-ES, random search, and grid search, while varying these factors. From this study, we draw the following conclusions related to the use of optimization algorithms in the automation of gradient elution LC:

- For dry optimization, our results suggest that DE is a highly competitive method in terms of both data and time efficiency. BO outperformed the other algorithms in terms of data efficiency but became impractical due to its computational scaling, making it unsuitable for *in silico* modeling approaches requiring a large iteration budget.
- For wet optimization, BO appeared to be superior for the search-based approaches, which rely on real measurement input, occasionally even outperforming the grid-search benchmark that comprised a much larger iteration budget.
- In principle, all tested evolutionary algorithms were found to be suitable for method development purposes, however, CMA-ES performed worse than expected compared to the other algorithms in both the dry and wet scenarios.
- The study revealed a strong influence of the CRF, and to a lesser extent, the number of compounds in the sample and the gradient complexity, on the efficiency of the optimization algorithms. This complicates the already challenging design of new CRFs for method development purposes, as the CRF not only has to describe the chromatography adequately but also needs to produce an optimization landscape that is efficiently navigable by the optimization algorithms, which we identify as an important topic of research for effective method optimization.

One important trade-off of our study was that algorithms were run using default settings as this reflects a use case for the general practitioner. However, tuning the individual algorithms may significantly improve their performance. We deliberately opted not to tune the individual algorithms to maintain a general comparison and assess their

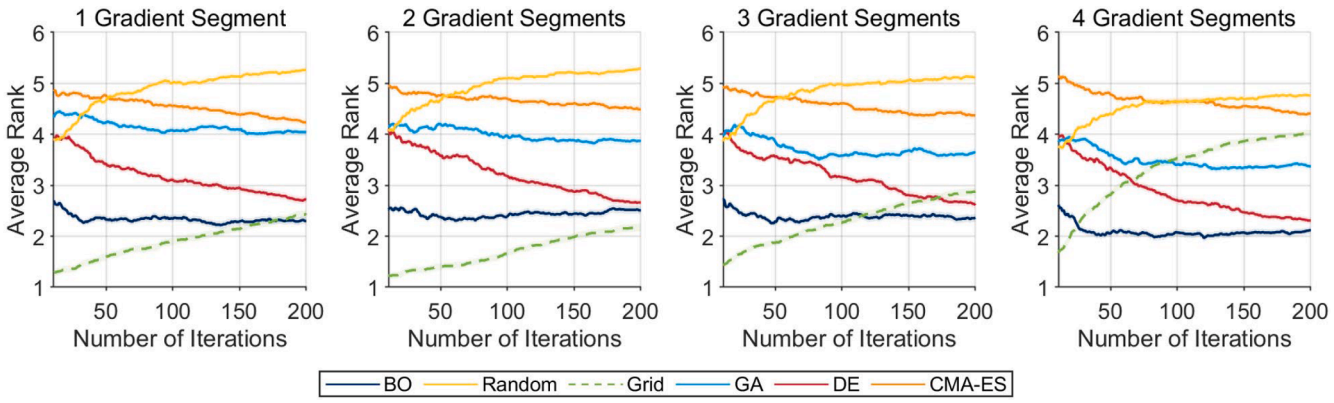

Fig. 12. The ranking for each level of gradient complexity (1–4 gradient segments) averaged over all studied CRFs and samples in the wet scenario. All algorithms were run for 200 iterations except for grid search which was fixed after the first iteration with its best-found value, indicated by the dashed line. The standard error of the average ranking is shown by the shaded border.

broader applicability to LC method development. Tuning each algorithm individually could have led to a more accurate reflection of their highest potential, but it would have made a fair comparison more challenging. In addition, it would make the comparison more specific for the optimization problem. The same workflow and framework can be utilized to test optimization algorithms for specific use-cases, potentially incorporating algorithm-specific tuning.

It must also be noted that different starting points (i.e. random seed) could have affected the performance. For instance, in terms of the required gradient complexity and the experimental designs. A better starting point could greatly improve the performance of the algorithms, which could be achieved through expert or prior knowledge, or more extensive scanning strategies.

We have attempted to make the simulated sample realistic by implementing peak compression and a diverse set of samples. However, there are still opportunities to improve the simulated sample to include all the phenomena and peak shapes that can be expected in LC method development. To this end, we believe that the chromatographic community would benefit from the creation of a widely accepted set of standardized benchmark samples of varying complexity and retention behavior (with known optima) against which future algorithms and/or CRFs could be benchmarked [50,51].

CRediT authorship contribution statement

Gerben B. van Henten: Writing – original draft, Methodology, Investigation, Formal analysis, Conceptualization. **Jim Boelrijk:** Writing – original draft, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Céline Kattenberg:** Writing – review & editing, Methodology, Investigation, Formal analysis, Conceptualization. **Tijmen S. Bos:** Writing – review & editing, Supervision, Formal analysis, Conceptualization. **Bernd Ensing:** Writing – review & editing, Funding acquisition. **Patrick Forré:** Writing – review & editing, Supervision, Funding acquisition. **Bob W.J. Pirok:** Writing – review & editing, Supervision, Funding acquisition, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

TB acknowledges the PARADISE project (ENPPS.TA.019.001) and received funding from the Dutch Research Council (NWO) in the framework of the Science PPP Fund for the top sectors and from the Ministry of Economic Affairs of the Netherlands in the framework of the "PPS Toeslagregeling". BP acknowledges the TTW VENI research program (Project 19173, "Unleashing the Potential of Separation Technology to Achieve Innovation in Research and Society (UPSTAIRS) "), which is financed by the Dutch Research Council (NWO).

This work was performed in the context of the Chemometrics and Advanced Separations Team (CAST) within the Centre for Analytical Sciences Amsterdam (CASA). The valuable contributions of the CAST members are gratefully acknowledged.

Supplementary materials

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.chroma.2024.465626.

Data availability

All code used in this study can be found at https://github.com/cast-amsterdam/LCOpt.git. In addition, a Google Collab tutorial is made available explaining the installation and use of our code base. All generated data can be found at: https://doi.org/10.5281/zenodo.13710886.

References

[1] G.B. van Henten, T.S. Bos, B.W.J. Pirok, Approaches to accelerate liquid chromatography method development in the laboratory using chemometrics and machine learning, LCGC Europe (2023) 202–209, https://doi.org/10.56530/lcgc.eu.rh7676j5.

[2] B.W.J. Pirok, S. Pous-Torres, C. Ortiz-Bolsico, G. Vivó-Truyols, P.J. Schoenmakers, Program for the interpretive optimization of two-dimensional resolution, J. Chromatogr. A 1450 (2016) 29–37, https://doi.org/10.1016/J.CHROMA.2016.04.061.

[3] E. Tyteca, D. Guillarme, G. Desmet, Use of individual retention modeling for gradient optimization in hydrophilic interaction chromatography: separation of nucleobases and nucleosides, J. Chromatogr. A 1368 (2014) 125–131, https://doi.org/10.1016/J.CHROMA.2014.09.065.

[4] W. Hao, B. Li, Y. Deng, Q. Chen, L. Liu, Q. Shen, Computer aided optimization of multilinear gradient elution in liquid chromatography, J. Chromatogr. A 1635 (2021) 461754, https://doi.org/10.1016/J.CHROMA.2020.461754.

[5] T.S. Bos, J. Boelrijk, S.R.A. Molenaar, B. Van 'T Veer, L.E. Niezen, D. Van Herwerden, S. Samanipour, D.R. Stoll, P. Forré, B. Ensing, G.W. Somsen, B.W. J. Pirok, Chemometric strategies for fully automated interpretive method development in liquid chromatography, Anal. Chem. 94 (2022) 16060–16068, https://doi.org/10.1021/ACS.ANALCHEM.2C03160/SUPPL_FILE/AC2C03160_SI_002.ZIP.

[6] J. Bradbury, G. Genta-Jouve, J.W. Allwood, W.B. Dunn, R. Goodacre, J.D. Knowles, S. He, M.R. Viant, MUSCLE: automated multi-objective evolutionary optimization of targeted LC-MS/MS analysis, Bioinformatics. 31 (2015) 975–977, https://doi.org/10.1093/BIOINFORMATICS/BTU740.

[7] S. O'Hagan, W.B. Dunn, M. Brown, J.D. Knowles, D.B. Kell, Closed-loop, multiobjective optimization of analytical instrumentation: gas chromatography/time-of-flight mass spectrometry of the metabolomes of human serum and of yeast

fermentations, Anal. Chem. 77 (2004) 290–303, https://doi.org/10.1021/AC049146X.
[8] J. Boelrijk, B. Ensing, P. Forré, B.W.J. Pirok, Closed-loop automatic gradient design for liquid chromatography using Bayesian optimization, Anal. Chim. Acta 1242 (2023) 340789, https://doi.org/10.1016/J.ACA.2023.340789.
[9] M.W. Watson, P.W. Carr, Simplex Algorithm for the Optimization of Gradient Elution High-Performance Liquid Chromatography, Anal. Chem. 51 (1979) 1835–1842, https://doi.org/10.1021/AC50047A052/ASSET/AC50047A052.FP.PNG_V03.
[10] J.C. Berridge, Unattended optimisation of reversed-phase high-performance liquid chromatographic separations using the modified simplex algorithm, J. Chromatogr. A 244 (1982) 1–14, https://doi.org/10.1016/S0021-9673(00)80117-X.
[11] M. Tirapelle, D.N. Chia, F. Duanmu, M.O. Besenhard, L. Mazzei, E. Sorensen, In-silico method development and optimization of on-line comprehensive two-dimensional liquid chromatography via a shortcut model, J. Chromatogr. A (2024) 1721, https://doi.org/10.1016/j.chroma.2024.464818.
[12] A. Kensert, P. Libin, G. Desmet, D. Cabooter, Deep reinforcement learning for the direct optimization of gradient separations in liquid chromatography, J. Chromatogr. A 1720 (2024) 464768, https://doi.org/10.1016/J.CHROMA.2024.464768.
[13] M.J. den Uijl, P.J. Schoenmakers, G.K. Schulte, D.R. Stoll, M.R. van Bommel, B.W. J. Pirok, Measuring and using scanning-gradient data for use in method optimization for liquid chromatography, J. Chromatogr. A 1636 (2021) 461780, https://doi.org/10.1016/J.CHROMA.2020.461780.
[14] E. Tyteca, G. Desmet, A universal comparison study of chromatographic response functions, J. Chromatogr. A 1361 (2014) 178–190, https://doi.org/10.1016/J.CHROMA.2014.08.014.
[15] E. Tyteca, S.H. Park, R.A. Shellie, P.R. Haddad, G. Desmet, Computer-assisted multi-segment gradient optimization in ion chromatography, J. Chromatogr. A 1381 (2015) 101–109, https://doi.org/10.1016/J.CHROMA.2014.12.085.
[16] Y. Shan, W. Zhang, A. Seidel-Morgenstern, R. Zhao, Y. Zhang, Multi-segment linear gradient optimization strategy based on resolution map in HPLC, Sci China B Chem 49 (2006) 315–325, https://doi.org/10.1007/S11426-006-2004-Y/METRICS.
[17] S. López-Ureña, J.R. Torres-Lapasió, R. Donat, M.C. García-Alvarez-Coque, Gradient design for liquid chromatography using multi-scale optimization, J. Chromatogr. A 1534 (2018) 32–42, https://doi.org/10.1016/J.CHROMA.2017.12.040.
[18] M.O. Besenhard, A. Tsatse, L. Mazzei, E. Sorensen, Recent advances in modelling and control of liquid chromatography, Curr. Opin. Chem. Eng. 32 (2021) 100685, https://doi.org/10.1016/J.COCHE.2021.100685.
[19] L.R. Snyder, J.W. Dolan, D.C. Lommen, Drylab® computer simulation for high-performance liquid chromatographic method development : I. Isocratic elution, J. Chromatogr. A 485 (1989) 65–89, https://doi.org/10.1016/S0021-9673(01)89133-0.
[20] J.W. Dolan, D.C. Lommen, L.R. Snyder, Drylab® computer simulation for high-performance liquid chromatographic method development : II. Gradient Elution, J. Chromatogr. A 485 (1989) 91–112, https://doi.org/10.1016/S0021-9673(01)89134-2.
[21] ACD/Labs LC-GC Simulator, (2015).
[22] E. Tyteca, A. Périat, S. Rudaz, G. Desmet, D. Guillarme, Retention modeling and method development in hydrophilic interaction chromatography, J. Chromatogr. A 1337 (2014) 116–127, https://doi.org/10.1016/J.CHROMA.2014.02.032.
[23] B.W.J. Pirok, S. Pous-Torres, C. Ortiz-Bolsico, G. Vivó-Truyols, P.J. Schoenmakers, Program for the interpretive optimization of two-dimensional resolution, J. Chromatogr. A 1450 (2016) 29–37, https://doi.org/10.1016/J.CHROMA.2016.04.061.
[24] S.R.A. Molenaar, P.J. Schoenmakers, B.W.J. Pirok, MOREPEAKS, Zenodo (2021). https://doi.org/10.5281/ZENODO.5710443.
[25] J.T.V. Matos, R.M.B.O. Duarte, A.C. Duarte, Trends in data processing of comprehensive two-dimensional chromatography: state of the art, Journal of Chromatography B 910 (2012) 31–45, https://doi.org/10.1016/J.JCHROMB.2012.06.039.
[26] G. Lan, J.M. Tomczak, D.M. Roijers, A.E. Eiben, Time efficiency in optimization with a bayesian-Evolutionary algorithm, Swarm. Evol. Comput. 69 (2022) 100970, https://doi.org/10.1016/J.SWEVO.2021.100970.
[27] J.C. Berridge, Simplex optimization of high-performance liquid chromatographic separations, J. Chromatogr. A 485 (1989) 3–14, https://doi.org/10.1016/S0021-9673(01)89129-9.
[28] S. López-Ureña, J.R. Torres-Lapasió, R. Donat, M.C. García-Alvarez-Coque, Gradient design for liquid chromatography using multi-scale optimization, J. Chromatogr. A 1534 (2018) 32–42, https://doi.org/10.1016/J.CHROMA.2017.12.040.
[29] E. Bosten, A. Kensert, G. Desmet, D. Cabooter, Automated method development in high-pressure liquid chromatography, J. Chromatogr. A 1714 (2024) 464577, https://doi.org/10.1016/J.CHROMA.2023.464577.
[30] P. Nikitas, A. Pappa-Louisi, P. Agrafiotou, Multilinear gradient elution optimisation in reversed-phase liquid chromatography using genetic algorithms, J. Chromatogr. A 1120 (2006) 299–307, https://doi.org/10.1016/J.CHROMA.2006.01.005.
[31] T.M. Dixon, J. Williams, M. Besenhard, R.M. Howard, J. MacGregor, P. Peach, A. D. Clayton, N.J. Warren, R.A. Bourne, Operator-free HPLC automated method development guided by Bayesian optimization, Digit. Discov. 3 (2024) 1591–1601, https://doi.org/10.1039/D4DD00062E.
[32] B. Huygens, K. Efthymiadis, A. Nowé, G. Desmet, Application of evolutionary algorithms to optimise one- and two-dimensional gradient chromatographic separations, J. Chromatogr. A 1628 (2020) 461435, https://doi.org/10.1016/J.CHROMA.2020.461435.
[33] J. Boelrijk, B. Pirok, B. Ensing, P. Forré, Bayesian optimization of comprehensive two-dimensional liquid chromatography separations, J. Chromatogr. A 1659 (2021) 462628, https://doi.org/10.1016/J.CHROMA.2021.462628.
[34] L.R. Snyder, Linear elution adsorption chromatography : VII. gradient elution theory, J. Chromatogr. A 13 (1964) 415–434, https://doi.org/10.1016/S0021-9673(01)95138-6.
[35] M.J. den Uijl, P.J. Schoenmakers, B.W.J. Pirok, M.R. van Bommel, Recent applications of retention modelling in liquid chromatography, J. Sep. Sci. 44 (2021) 88–114, https://doi.org/10.1002/JSSC.202000905.
[36] F. Gritti, Perspective on the Future Approaches to Predict Retention in Liquid Chromatography, Anal. Chem. 93 (2021) 5653–5664, https://doi.org/10.1021/ACS.ANALCHEM.0C05078/SUPPL_FILE/AC0C05078_SI_001.PDF.
[37] L.R. Snyder, J.W. Dolan, High-Performance Gradient Elution: the Practical Application of the Linear-Solvent-Strength Model, Wiley, 2006, https://doi.org/10.1002/0470055529.
[38] W. Hao, K. Wang, B. Yue, Q. Chen, Y. Huang, J. Yu, D. Li, Influence of the pre-elution of solute in initial mobile phase on retention time and peak compression under linear gradient elution, J. Chromatogr. A 1618 (2020) 460858, https://doi.org/10.1016/J.CHROMA.2020.460858.
[39] B.W.J. Pirok, A.F.G. Gargano, P.J. Schoenmakers, Optimizing separations in online comprehensive two-dimensional liquid chromatography, J. Sep. Sci. 41 (2018) 68–98, https://doi.org/10.1002/JSSC.201700863.
[40] E.V. Dose, Off-line optimization of gas chromatographic temperature programs, Anal. Chem. 59 (1987) 2420–2423, https://doi.org/10.1021/AC00146A021/ASSET/AC00146A021.FP.PNG_V03.
[41] B. Divjak, M. Moder, J. Zupan, Chemometrics approach to the optimization of ion chromatographic analysis of transition metal cations for routine work, Anal. Chim. Acta 358 (1998) 305–315, https://doi.org/10.1016/S0003-2670(97)00644-2.
[42] B. Jancic-Stojanovic, T. Rakić, N. Kostić, A. Vemić, A. Malenović, D. Ivanović, M. Medenica, Advancement in optimization tactic achieved by newly developed chromatographic response function: application to LC separation of raloxifene and its impurities, Talanta 85 (2011) 1453–1460, https://doi.org/10.1016/J.TALANTA.2011.06.029.
[43] E. Tyteca, A. Liekens, D. Clicq, A. Fanigliulo, B. Debrus, S. Rudaz, D. Guillarme, G. Desmet, Predictive elution window stretching and shifting as a generic search strategy for automated method development for liquid chromatography, Anal. Chem. 84 (2012) 7823–7830, https://doi.org/10.1021/AC301331G/SUPPL_FILE/AC301331G_SI_001.PDF.
[44] A.F. Gad, PyGAD: an intuitive genetic algorithm python library, Multimed. Tools. Appl. (2021), https://doi.org/10.1007/s11042-023-17167-y.
[45] P. Virtanen, R. Gommers, T.E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S.J. van der Walt, M. Brett, J. Wilson, K.J. Millman, N. Mayorov, A.R.J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E.W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E.A. Quintero, C.R. Harris, A.M. Archibald, A.H. Ribeiro, F. Pedregosa, P. van Mulbregt, A. Vijaykumar, A.Pietro Bardelli, A. Rothberg, A. Hilboll, A. Kloeckner, A. Scopatz, A. Lee, A. Rokem, C.N. Woods, C. Fulton, C. Masson, C. Häggström, C. Fitzgerald, D.A. Nicholson, D.R. Hagen, D. V. Pasechnik, E. Olivetti, E. Martin, E. Wieser, F. Silva, F. Lenders, F. Wilhelm, G. Young, G.A. Price, G.L. Ingold, G.E. Allen, G.R. Lee, H. Audren, I. Probst, J. P. Dietrich, J. Silterra, J.T. Webber, J. Slavič, J. Nothman, J. Buchner, J. Kulick, J. L. Schönberger, J.V.de Miranda Cardoso, J. Reimer, J. Harrington, J.L. C. Rodríguez, J. Nunez-Iglesias, J. Kuczynski, K. Tritz, M. Thoma, M. Newville, M. Kümmerer, M. Bolingbroke, M. Tartre, M. Pak, N.J. Smith, N. Nowaczyk, N. Shebanov, O. Pavlyk, P.A. Brodtkorb, P. Lee, R.T. McGibbon, R. Feldbauer, S. Lewis, S. Tygier, S. Sievert, S. Vigna, S. Peterson, S. More, T. Pudlik, T. Oshima, T.J. Pingel, T.P. Robitaille, T. Spura, T.R. Jones, T. Cera, T. Leslie, T. Zito, T. Krauss, U. Upadhyay, Y.O. Halchenko, Y. Vázquez-Baeza, SciPy 1.0: fundamental algorithms for scientific computing in Python, Nat. Methods 17 (2020) 261–272, https://doi.org/10.1038/s41592-019-0686-2, 2020 17:3.
[46] N. Hansen, Y. Akimoto, P. Baudis, CMA-ES/pycma, (2019). https://doi.org/10.5281/ZENODO.7573532.
[47] T. Head, M. Kumar, H. Nahrstaedt, G. Louppe, I. Shcherbatyi, scikit-optimize/scikit-optimize, (2021). https://doi.org/10.5281/ZENODO.5565057.
[48] J. Bergstra, Y. Bengio, Random Search for Hyper-Parameter Optimization Yoshua Bengio, Journal of Machine Learning Research 13 (2012) 281–305. http://scikit-learn.sourceforge.net (accessed April 12, 2024).
[49] E.P. Dos Santos Amorim, C.R. Xavier, R.S. Campos, R.W. Dos Santos, Comparison between genetic algorithms and differential evolution for solving the history matching problem. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), LNCS, 2012, pp. 635–648, https://doi.org/10.1007/978-3-642-31125-3_48, 7333.
[50] N.B.L. Milani, A.R. García-Cicourel, J. Blomberg, R. Edam, S. Samanipour, T.S. Bos, B.W.J. Pirok, Generating realistic data through modeling and parametric probability for the numerical evaluation of data processing algorithms in two-dimensional chromatography, Anal. Chim. Acta 1312 (2024) 342724, https://doi.org/10.1016/J.ACA.2024.342724.
[51] B.A. Weggler, L.M. Dubois, N. Gawlitta, T. Gröger, J. Moncur, L. Mondello, S. Reichenbach, P. Tranchida, Z. Zhao, R. Zimmermann, M. Zoccali, J.F. Focant, A unique data analysis framework and open source benchmark data set for the analysis of comprehensive two-dimensional gas chromatography software, J. Chromatogr. A 1635 (2021) 461721, https://doi.org/10.1016/J.CHROMA.2020.461721.