

Substitution matrix based color schemes for sequence alignment visualization

Kunzmann, Patrick; Mayer, Benjamin E.; Hamacher, Kay
(2020)

DOI (TUprints): <https://doi.org/10.25534/tuprints-00013521>

Lizenz:



CC-BY 4.0 International - Creative Commons, Namensnennung

Publikationstyp: Artikel

Fachbereich: LOEWE

Quelle des Originals: <https://tuprints.ulb.tu-darmstadt.de/13521>

SOFTWARE

Open Access

Substitution matrix based color schemes for sequence alignment visualization



Patrick Kunzmann* , Benjamin E. Mayer and Kay Hamacher

*Correspondence:
kunzmann@bio.tu-darmstadt.de
Department of Computational
Biology and Simulation, TU
Darmstadt, Schnittpahnstraße 2,
64287 Darmstadt, Germany

Abstract

Background: Visualization of multiple sequence alignments often includes colored symbols, usually characters encoding amino acids, according to some (physical) properties, such as hydrophobicity or charge. Typically, color schemes are created manually, so that equal or similar colors are assigned to amino acids that share similar properties. However, this assessment is subjective and may not represent the similarity of symbols very well.

Results: In this article we propose a different approach for color scheme creation: We leverage the similarity information of a substitution matrix to derive an appropriate color scheme. Similar colors are assigned to high scoring pairs of symbols, distant colors are assigned to low scoring pairs. In order to find these optimal points in color space a simulated annealing algorithm is employed.

Conclusions: Using the substitution matrix as basis for a color scheme is consistent with the alignment, which itself is based on the very substitution matrix. This approach allows fully automatic generation of new color schemes, even for special purposes which have not been covered, yet, including schemes for structural alphabets or schemes that are adapted for people with color vision deficiency.

Keywords: Open source, Python, Color space, Optimization, Sequence alignment

Background

Typically, visualization of multiple protein sequence alignments colors the amino acid symbols according to some kind of (chemical) property. Examples for software using this visualization technique are *MSAViewer* [1], *JalView* [2] or *ClustalX* [3]. Figure 1 shows an alignment using the default *ClustalX* color scheme depicting the chemical characteristics of the amino acids. Typically, such a color scheme is created manually by a professional using their intuition and knowledge about any characteristics to be emphasized.

However, the intuition might not reflect the biological similarity of two amino acids in terms of evolutionary distance, as defined by amino acid substitution scores (and thus probabilities).



© The Author(s). 2020 **Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>. The Creative Commons Public Domain Dedication waiver (<http://creativecommons.org/publicdomain/zero/1.0/>) applies to the data made available in this article, unless otherwise stated in a credit line to the data.

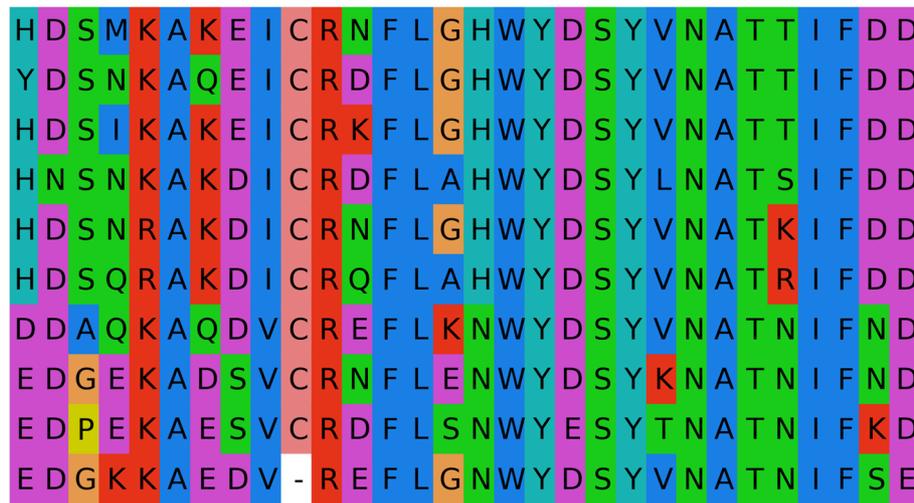


Fig. 1 Visualization of a multiple sequence alignment. The alignment uses the default ClustalX color scheme [3]. The alignment is an excerpt of a multiple protein sequence alignment of bacterial luciferases (*luxA* gene product). The figure was created with Biotite [13]

If the color in an alignment visualization is used to depict general physical amino acid characteristics, in contrast to a specific measurable or computable quantity, e.g. hydrophathy or secondary structure propensity, we reckon substitution matrices would be a more meaningful and reproducible basis for color scheme creation.

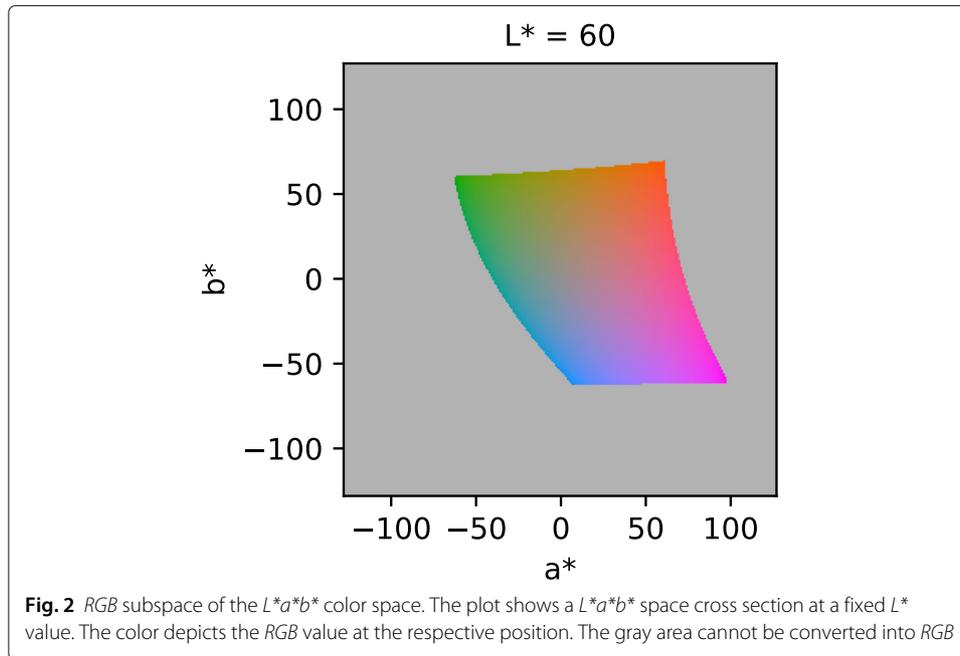
In this article we describe a method and corresponding software that is able to directly generate color schemes from a substitution matrix in an automatic manner: The aim is to find those colors for each symbol, for which the pairwise perceptual color differences correspond to pairwise symbol distances. The more similar two symbols are, the more similar the assigned colors should be. We formulated this criterion as score function and used a *simulated annealing* [4] optimizer, that searches for the optimal colors for which the score function is minimal.

Color space

In order to find the optimal colors that minimize the score function, a *color space* to search in is required. A color space is a 3-dimensional vector space where a point represents a corresponding color. The arguably most common color space is *RGB* or, more exactly, *standard RGB (sRGB)* [5]. It defines a color as an addition of red, green and blue light. Although this color space is very useful in the context of display devices, it lacks perceptual uniformity [6]: changes of an RGB color value do not result in a proportional perceptual change.

In contrast, the *CIE $L^*a^*b^*$* color space [7] shows perceptually approximately uniform characteristics, i.e. the perceptual difference of two colors is approximately proportional to the Euclidean distance of the color components. The *$L^*a^*b^*$* color space consists of these three components:

- **L^*** - The lightness of the color. 0 is completely black and 100 is completely white.
- **a^*** - The green-red component. Green is in the negative direction, red is in the positive direction.
- **b^*** - The blue-yellow component. Blue is in the negative direction, yellow is in the positive direction.



While a^* and b^* values are theoretically unlimited in either direction, only a limited $L^*a^*b^*$ subspace is displayable by devices and thus can be converted into *sRGB* (Fig. 2).

Implementation

Score function

Our method should find similar colors for similar symbols and distant colors for dissimilar symbols. In order to quantify this objective, differences are computed via a score function. The score function takes a combination of *CIE L*a*b** colors (*color conformation*), represented as a $(N \times 3)$ -matrix, and returns a scalar score value. N is the number of symbols in the alphabet (20 for amino acids).

In the following $\langle X \rangle$ denotes the arithmetic mean of all elements of a matrix: $\langle X \rangle := \frac{1}{n} \sum_{ij} X_{ij}$. For a triangular matrix only the non-zero diagonals are taken into account; n is the number of respective matrix elements.

Construction of distance matrices

First, the input substitution matrix M is converted into a distance matrix D' . D' is triangular to remove redundant entries, since the distance is commutative.

$$D'_{ij} = \begin{cases} ((M_{ii} - M_{ij}) + (M_{jj} - M_{ji})) / 2, & \text{for } j \leq i \\ 0, & \text{for } j > i \end{cases}$$

For any substitution matrix M , M_{ii} should be the maximum value in the row/column i , otherwise a symbol would be more similar to another symbol than to itself. Consequently, the main diagonal of D' contains only zeros.

To be agnostic about the magnitude of substitution scores in M , D' is scaled, so that the average distance is 1.

$$D = \frac{D'}{\langle D' \rangle}$$

Calculation of the perceptual difference matrix

On the other hand, the triangular matrix C denotes the pairwise perceptual differences of the $L^*a^*b^*$ colors for each pair of symbols. Our implementation uses the *CIEDE2000* formula [8], which is omitted for brevity. However, the formula can be approximated as the Euclidean distance [7]:

$$C_{ij} \approx \sqrt{(L_i^* - L_j^*)^2 + (a_i^* - a_j^*)^2 + (b_i^* - b_j^*)^2}$$

Note, while D is constant, C is dependent on the (current) color conformation.

In order to relate the $L^*a^*b^*$ color differences in C to the distances in D , a scale factor f_s is introduced. f_s is the proportion of the average distance in D to the average difference in C :

$$f_s = \frac{\langle D \rangle}{\langle C \rangle} = \frac{1}{\frac{1}{n} \sum_{ij} C} = \frac{n}{\sum_{ij} C_{ij}}$$

As C is variable, f_s also dynamically changes during our optimization runs.

Score function

The score function S_T is a sum of two terms: a sum of harmonic potentials between each pair of symbols S_H and a linear *contrast score* S_C :

$$S_T = S_H + S_C$$

The harmonic potentials are used to adjust the relative color differences in accordance with the substitution matrix. The equilibrium distance of each potential corresponds to the distance in the distance matrix D :

$$S_H = \sum_{ij} (f_s C_{ij} - D_{ij})^2$$

However, this term is not sufficient to create an appealing color scheme: due to the scale factor f_s , a scheme with a small average color difference would get the same score as a scheme with a high average color difference. In consequence low contrast color schemes could arise. In order to favor high contrast color schemes the contrast score S_C is introduced. A reciprocal function based on the average color difference is used here. The *contrast factor* f_c is a user-supplied parameter for weighting this term:

$$S_C = \frac{f_c}{\langle C \rangle}$$

Optimization

The question, which color conformation minimizes the score function, is a $(N \times 3)$ -dimensional, continuous optimization problem. As the optimization landscape can be restricted via user input, we face the problem of a non trivially bounded and, depending on the constraint, possibly non-convex optimization problem. In general, obtaining an exact solution in a non-convex, continuous problem setting is computationally hard, as shown for the example of pair potentials in atomic clusters [9]. Therefore, we

resort to heuristic optimization, namely *simulated annealing* (SA) [4], as described in Algorithm 1.

The SA algorithm samples the search space, which is a vector space consisting of color space vectors $\vec{x} = (L_1^*, a_1^*, b_1^*, \dots, L_N^*, a_N^*, b_N^*)$.

Sampling is realized based on a neighborhood definition in color space. For the space of color space vectors this neighborhood is defined by all valid colors reachable by adding

Algorithm 1: Simulated annealing with annealing in energy and step size. *BSF*: *best-so-far*.

Data:

Inverse temperature at start β , annealing rate τ ,
 start step size Δx_0 , end step size Δx_1 ,
 number of iterations T

Result:

Optimal solution \vec{x} and score $S_T(\vec{x})$

We define

$$\alpha \leftarrow \log(\Delta x_1 / \Delta x_0) / T$$

$$\Delta x(t) \leftarrow \Delta x_0 \cdot \exp(\alpha \cdot t)$$

$$\beta(t) \leftarrow \beta_0 \cdot \exp(\tau \cdot t)$$

Initialization

$\vec{x} \leftarrow$ draw random solution from search space

$S \leftarrow S_T(\vec{x})$

$S_{BSF} \leftarrow S$

$\vec{x}_{BSF} \leftarrow \vec{x}$

for t in $(0, \dots, T)$ **do**

$\vec{x}_{new} \leftarrow$ drawFromNeighborhood($\vec{x}, \Delta x(t)$)

$S_{new} \leftarrow S_T(\vec{x}_{new})$

if $S_{new} < S$ **then**

$\vec{x} \leftarrow \vec{x}_{new}$

$S \leftarrow S_{new}$

else

$p \leftarrow$ draw random number from interval $[0, 1]$

$p_{accept} \leftarrow \exp(-\beta(t) \cdot (S_{new} - S))$

if $p \leq p_{accept}$ **then**

$\vec{x} \leftarrow \vec{x}_{new}$

$S \leftarrow S_{new}$

end

end

if $S < S_{BSF}$ **then**

$S_{BSF} \leftarrow S$

$\vec{x}_{BSF} \leftarrow \vec{x}$

end

end

perturbations of the color values drawn from a uniform distribution $U(-\Delta\vec{x}(t), \Delta\vec{x}(t))$, excluding the user specified region.

Initially the search space is sampled with a high temperature, so the algorithm has the ability to escape local minima or even jump over large sections of the search space. By gradually limiting this behavior, which is specified by the annealing schedule $\beta(t)$ (quantified by τ), the algorithm converges to a suitable optimal solution – guarantees about the found optima's quality, however, cannot be given due to the heuristic approach of SA. Yet, the convergence towards the global optimum in infinite time is a proven quality of this algorithm [10, 11]. Therefore, after a sufficiently long run time a non-optimal, but nonetheless *good* solution is found.

Typically SA is used for combinatorial optimization problems, i.e. problems defined on a discrete search space, e.g., the traveling salesman problem [4]. Since its inception SA has also been adapted to various continuous optimization problems.

As it turns out a robust adaption of SA for the continuous problem discussed here, is realized by simply doing an annealing in both the temperature and step size.

We run an ensemble of SA instances, meaning multiple independent instances with different random number generator seeds. The best found solution and score seen during a single SA run are captured in the variables \vec{x}_{BSF} and S_{BSF} .

These quantities are referred to as the *best-so-far* solution and score. After the last iteration the optimal solution is given as the minimum of the *best-so-far* solutions within the ensemble of optimizers. In our implementation we also store the ensemble maximum which is given by the maximum of the worst seen solutions over the algorithm run, which we neglected to include in Algorithm 1 in favor of simplicity, as in principle a broad variety of run features could be stored in the same way as the *best-so-far* solution. Furthermore an ensemble average and standard deviation are stored. These two properties are used for further quality analysis of the SA run.

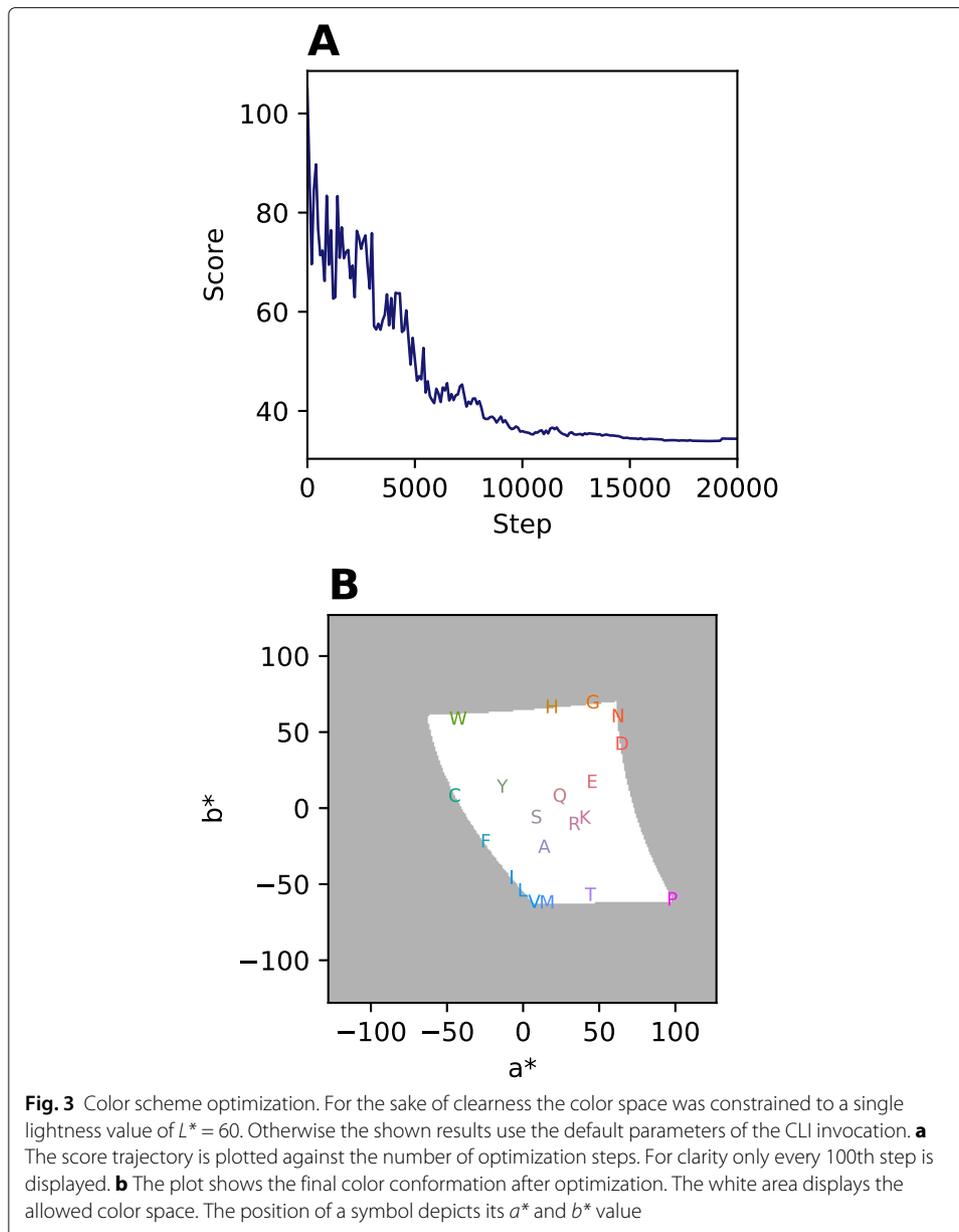
Software

We have implemented the method for color scheme generation in the Python package *Gecos*. In addition to the more flexible Python API, the package offers a command line interface (CLI) for simple color scheme generation. Either way, the alphabet, substitution matrix and color subspace can be customized for the purpose of the user. By default the software creates a color scheme for the *BLOSUM62* matrix [12]. The CLI saves the generated color scheme in JSON format, containing the RGB color code for each symbol. The JSON format can be directly used for alignment visualization in *Biotite* [13]. For usage in other visualization software the color codes must be inserted into the input format of the respective program.

Results and discussion

Optimization

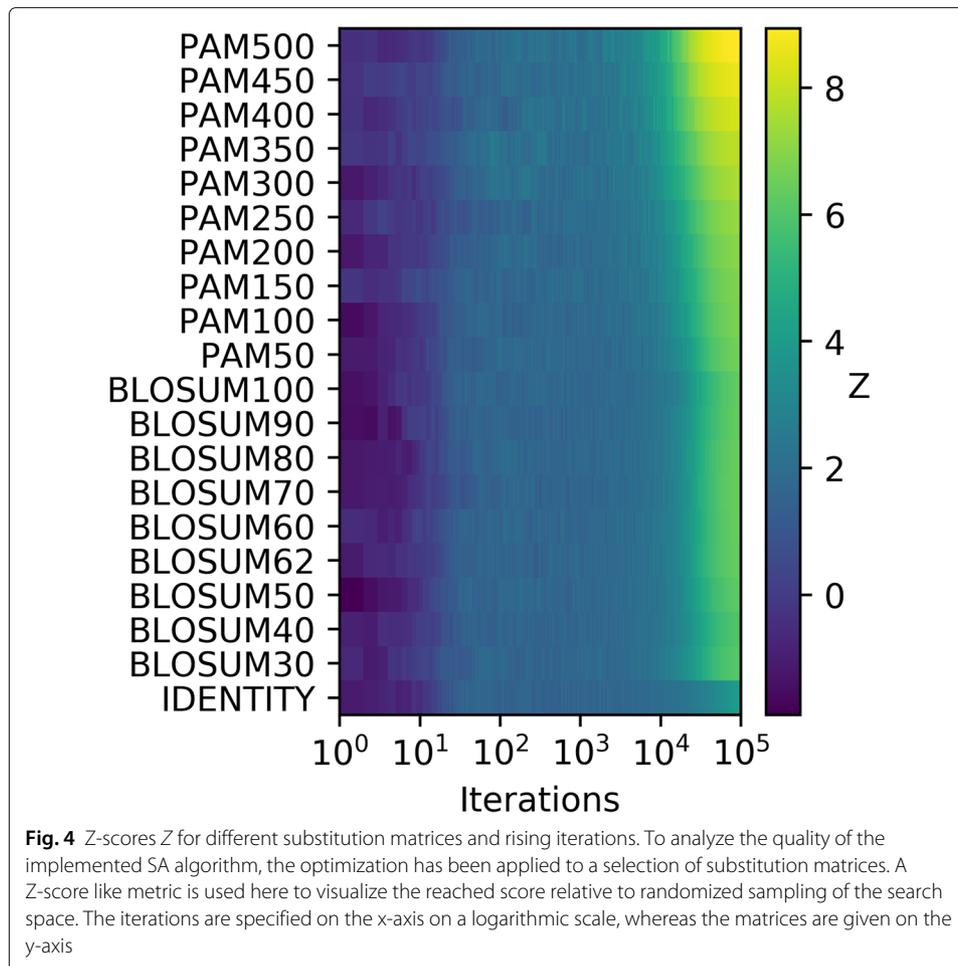
Figure 3 shows the improvement of the score during the optimization process and the resulting color conformation. Generally, the evaluation whether the score of the final color conformation is near the global optimum is not directly possible, since the calculation of the score at the global optimum requires knowledge of the optimal color conformation.



In our approach we compared the outcome of a reference optimization run with random sampling in the color conformation space. To show the robustness of the implemented optimization technique with regards to user input we benchmarked on a selection of substitution matrices. This is visualized in Fig. 4, where a Z-score-like empirical measure

$$Z(t) := -\frac{\langle S_T \rangle_{SA}(t) - \mu(t)}{\sigma(t)}$$

of the optimizer quality relative to the random sampling is plotted over increasing iterations. At a given iteration the difference between the reached average scores of the optimizer and random sampling is normalized to the standard deviation $\sigma(t)$ seen by random sampling so far. The negative sign of $Z(t)$ is explained by the fact that our score S_T gets smaller for a better solution, so the scores reached by the optimizer should be smaller than the ones reached by sampling, in which case $Z(t)$ is positive.



While for the optimizer the averaging is done over the ensemble of optimizers, the mean $\mu(t)$ and standard deviation $\sigma(t)$ for the random sampling is calculated using the so far sampled data. Here the rationale is that a randomized sampling in color space should eventually stabilize, by thus producing a distribution of solutions that can be used as ground truth to separate from steered processes. The SA algorithm is exactly such a steered process, as it essentially does random sampling, yet from a local neighborhood instead of the whole search space, and is biased towards lower scores.

If the optimizer was an unsteered process one would expect low values for Z as this would coincide with the optimizer not finding better values than a randomized sampling. This behavior is seen in Fig. 4 for the early iterations.

However, with increasing iterations Z is getting successively higher until finally reaching a region of $Z \approx 6$ meaning that on average SA reaches a score that is six standard deviations *better* than purely randomized sampling. While it seems that for some of the PAM substitution matrices the algorithm performs better than for the others, the fact stands that overall a superior solution is reached, independently of the user selected input. The only matrix for which we see significantly different behavior in Fig. 4 is the *IDENTITY* matrix. This is expected, since the optimal distance between symbols is equidistant for all pairs of symbols here. Therefore, finding an optimal solution should be more difficult,

which accounts for the overall lower Z-score. Yet, even here the SA optimizer outperforms the random sampling and eventually reaches a region with $Z \approx 4$.

Color schemes

Figure 5a shows a color scheme that was created via the *Gecos* CLI for the *BLOSUM62* substitution matrix. The lightness range was constrained to $60 < L^* < 75$, but otherwise the default CLI parameters were used. The constraint is necessary as the scheme would not look appealing if the whole lightness range - from black to white - would have been included. On the other hand we found that fixing the lightness to a single value results in schemes that have a low contrast. Thus, we recommend using a lightness interval between 15 and 30.

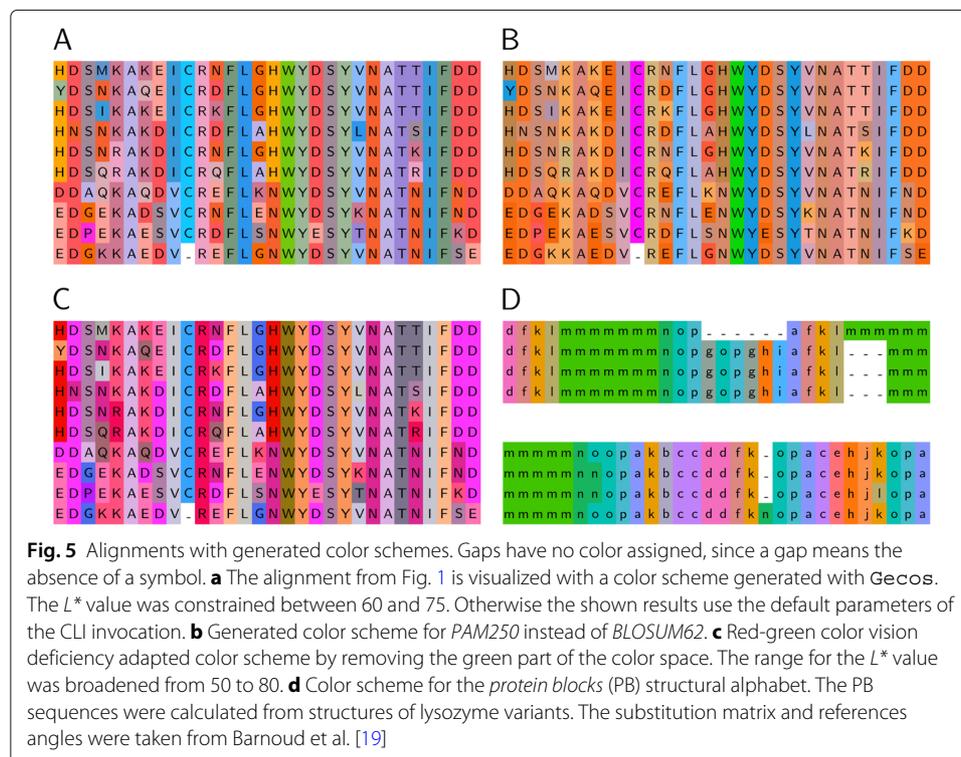
Dependence on substitution matrix

The generated color scheme is strongly dependent on the used substitution matrix. This becomes more clear when comparing the *BLOSUM62* based color scheme (Fig. 5a) with a *PAM250* [14] based scheme (Fig. 5b). Compared to *BLOSUM62*, *PAM250* assigns an extraordinarily low score for substitution of tryptophane and cysteine with any other amino acid. Consequently, the *PAM250* based color scheme especially highlights tryptophane and cysteine, while the contrast between the other symbols is relatively low.

Constraints in a*b* dimensions

In addition to constraining the lightness L^* , it can also be reasonable to limit the color space in the a^* and b^* dimensions. The use cases include

- limiting the saturation of the colors,
- creating a scheme with a specified hue,
- taking color vision deficiency into account.



The most common color vision deficiency is the red-green color blindness, that affects approximately 8 % of the male and approximately 0.5 % of the female population [15–17]. In order to create a color scheme, that is more friendly to red-green color vision deficient people, we ran an optimization on a color subspace where the green part is omitted, i.e. $a^* > 0$ (Fig. 5c). As the reduced size of the color space would inherently cause a loss of contrast, the available lightness interval is increased to $50 < L^* < 80$.

Color schemes for exotic alphabets

Since the described method does not require professional knowledge about the properties of each symbol, but merely needs a substitution matrix, it can be easily transferred to alphabets other than amino acids.

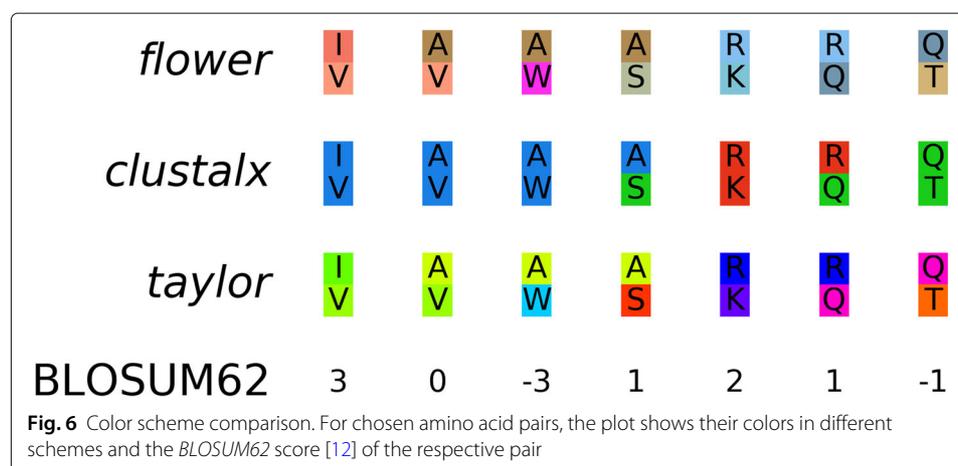
Structural alphabets are such a use case. Structural alphabets encode local protein structure properties, e.g. backbone dihedral angles or pseudo bond angles, into symbols that represent the local conformation. In this way a 3D structure can be converted into a symbol sequence. Creating a color scheme for a structural alphabet can be difficult, because the symbols do not exhibit such tangible properties like charge or hydrophobicity. However, if a substitution matrix is available for the structural alphabet, our approach is able to automatically compile a color scheme.

The *protein block* (PB) alphabet is such a structural alphabet [18]. It contains 16 symbols from a to p, each representing a different local peptide backbone conformation. Figure 5d shows a scheme based on the PB substitution matrix [19], generated with *Gecos*.

Comparison with existing amino acid color schemes

Our method aims to generate color schemes that depict evolutionary similarity of amino acids better than the existing, manually created ones. Based on chosen amino acid pairs, we compared a color scheme generated by *Gecos*, namely *flower* implemented in *Biotite* [13], with two traditional color schemes: the default one from *ClustalX* [3] (denoted as *clustalx*) and the *Taylor* [20] color scheme (denoted as *taylor*) (Fig. 6).

Both *clustalx* and *taylor* use identical or similar colors, respectively, for alanine (A) valine (V) and isoleucine (I). However, valine is much more (evolutionary) similar to isoleucine than to alanine, according to the BLOSUM62 [12] matrix. In fact, alanine is slightly more similar to serine (S) than to valine. Still, *clustalx* and *taylor* assign very



dissimilar colors to alanine and serine, as alanine is classified as hydrophobic in contrast to the more polar serine. *clustalx* also uses the same color for alanine and tryptophan (W), because both amino acids are hydrophobic. However, the similarity score between both residues is exceptionally low.

A similar result can be seen for the positively charged and polar residues: although all three color schemes are able to depict the high similarity between lysine (K) and arginine (R), as both are usually positively charged, *clustalx* and *taylor* use highly different colors for the arginine-glutamine (Q) pair, as glutamine is neutral. Instead, they use identical or similar colors, respectively, for the glutamine-threonine (T) pair. Although both are uncharged polar residues, glutamine is closer to arginine than to threonine from an evolutionary point of view.

In contrast, *flower* does not categorize amino acids into groups based on e.g. hydrophobicity or residue size, but only uses colors based on pairwise amino acid similarities. Consequently, falsely suggested similarities, e.g. the alanine-tryptophan pair in *clustalx*, and falsely suggested dissimilarities, e.g. the alanine-serine pair in *clustalx* and *taylor*, are less likely to occur. Hence, *flower* is more reliable for visual analysis of multiple sequence alignments, when it comes to identification of evolutionary distant or close regions.

Conclusion

Our method enables the user to create a new color scheme for sequence alignments in a fast and easy way - may it be for special purposes like exotic alphabets, where no color scheme does exist yet, or simply because a new scheme appeals more than the existing ones.

Traditional color schemes depict specific characteristics of amino acids, for example hydrophobicity, charge, secondary structure propensities or a combination of these. Taking a substitution matrix as basis for the color scheme incorporates more evolutionary meaning. Although a specific property cannot directly be read from the color, this novel approach depicts the similarity of amino acids better in terms of substitution probability. Therefore, this method is consistent with the alignment, which itself is based on substitution probabilities and thus on molecular evolution.

Availability and requirements

Project name: Gecos

Project home page: <https://gecos.biotite-python.org/>

Operating system(s): Windows, OS X, Linux

Programming language: Python

Other requirements: At least Python 3.6, the packages *biotite*, *numpy* and *scikit-image* must be installed

License: BSD 3-Clause

Any restrictions to use by non-academics: None

Abbreviations

API: Application programming interface; CIE: International commission on illumination; CLI: Command line interface; SA: Simulated annealing; sRGB: Standard RGB

Acknowledgements

We thank Daniel Bauer and Philipp Babel (TU Darmstadt) for helpful feedback on the aesthetics of the generated color schemes. We thank Tobias Schreck (TU Graz) for valuable feedback on color space theory.

Authors' contributions

PK developed the project idea. PK and BM developed the *Gecos* package and wrote its documentation. KH guided the development process. PK, BM and KH wrote the manuscript. The author(s) read and approved the final manuscript.

Funding

This work was supported by the Ministry of Research and the Arts of the Hessen state (LOEWE project iNAPO). Furthermore, this work was supported by the Deutsche Forschungsgemeinschaft (DFG), Germany (GRK1657). The funding bodies had no influence on the study design, data collection, analysis, and interpretation and in manuscript preparation.

Availability of data and materials

The *Gecos* source code is hosted at <https://github.com/biotite-dev/gecos> and its official documentation at <https://gecos.biotite-python.org/>. The version 1.1, that was used in this study, is available as archive [21].

Ethics approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

Competing interests

The authors declare that they have no competing interests.

Received: 22 January 2020 Accepted: 30 April 2020

Published online: 24 May 2020

References

1. Yachdav G, Wilzbach S, Rauscher B, Sheridan R, Sillitoe I, Procter J, Lewis SE, Rost B, Goldberg T. MSAViewer: Interactive JavaScript visualization of multiple sequence alignments. *Bioinformatics*. 2016;32(22):3501–3. <https://doi.org/10.1093/bioinformatics/btw474>.
2. Waterhouse AM, Procter JB, Martin DMA, Clamp M, Barton GJ. Jalview Version 2-A multiple sequence alignment editor and analysis workbench. *Bioinformatics*. 2009;25(9):1189–91. <https://doi.org/10.1093/bioinformatics/btp033>.
3. Larkin MA, Blackshields G, Brown NP, Chenna R, Mcgettigan PA, McWilliam H, Valentin F, Wallace IM, Wilm A, Lopez R, Thompson JD, Gibson TJ, Higgins DG. Clustal W and Clustal X version 2.0. *Bioinformatics*. 2007;23(21):2947–8. <https://doi.org/10.1093/bioinformatics/btm404>.
4. Kirkpatrick S, Gelatt CD, Vecchi MP. Optimization by simulated annealing. *Science*. 1983;220(4598):671–80. <https://doi.org/10.1126/science.220.4598.671>.
5. IEC: IEC 61966-2-1. Multimedia systems and equipment - Colour measurement and management - Part 2-1: Colour management - Default RGB colour space - sRGB. 1999. <https://webstore.iec.ch/publication/6169>.
6. Bernard J, Steiger M, Mittelstädt S, Thum S, Keim D, Kohlhammer J. A survey and task-based quality assessment of static 2D colormaps. In: *Visualization and Data Analysis 2015*, vol. 9397; 2015. p. 247–62. <https://doi.org/10.1117/12.2079841>.
7. CIE: ISO/CIE 11664-4. Colorimetry – Part 4: CIE 1976 L*a*b* colour space. 2019. <https://www.iso.org/standard/74166.html>.
8. CIE: CIE 142-2001. Improvement to industrial colour-difference evaluation. 2001. <http://cie.co.at/publications/improvement-industrial-colour-difference-evaluation>.
9. Wille LT, Vennik J. Computational complexity of the ground-state determination of atomic clusters. *J Phys A Math Gen*. 1985;18(8):419–22. <https://doi.org/10.1088/0305-4470/18/8/003>.
10. Aarts E, Lenstra JK. *Local Search in Combinatorial Optimization*, 1st edn. New York: Wiley; 1997.
11. Laarhooven PJM, Aarts EHL. No Title. Norwell: Kluwer Academic Publishers; 1987. <https://doi.org/10.1007/978-94-015-7744-1>.
12. Henikoff S, Henikoff JG. Amino acid substitution matrices from protein blocks. *Proc Natl Acad Sci U S A*. 1992;89(22):10915–9.
13. Kunzmann P, Hamacher K. Biotite: A unifying open source computational biology framework in Python. *BMC Bioinformatics*. 2018;19(1):. <https://doi.org/10.1186/s12859-018-2367-z>.
14. Dayhoff MO. A Model of Evolutionary Change. In: *Proteins in Atlas of Protein Sequence and Structure*, vol. 5 Suppl; 1978. p. 345–52.
15. NEI. Facts about color blindness. https://nei.nih.gov/health/color_blindness/facts_about. Accessed 28 June 2019.
16. Kovalev VA. Towards image retrieval for eight percent of color-blind men. In: *Proceedings - International Conference on Pattern Recognition*, vol. 2; 2004. <https://doi.org/10.1109/ICPR.2004.1334414>.
17. Al-Aqtum MT, Al-Qawasmeh MH. Prevalence of colour blindness in young Jordanians. *Ophthalmologica*. 2001;215(1):39–42. <https://doi.org/10.1159/000050824>.
18. De Brevern AG, Etchebest C, Hazout S. Bayesian probabilistic approach for predicting backbone structures in terms of protein blocks. *Protein Struct Funct Genet*. 2000;41(3):271–87. [https://doi.org/10.1002/1097-0134\(20001115\)41:3<271::AID-PROT10>3.0.CO;2-Z](https://doi.org/10.1002/1097-0134(20001115)41:3<271::AID-PROT10>3.0.CO;2-Z).
19. Barnoud J, Santuz H, Craveur P, Joseph AP, Jallu V, de Brevern AG, Poulain P. PBxplore: a tool to analyze local protein structure and deformability with Protein Blocks. *PeerJ*. 2017;5:4013. <https://doi.org/10.7717/peerj.4013>.
20. Taylor WR. Residual colours: A proposal for aminochromography. *Protein Eng*. 1997;10(7):743–6. <https://doi.org/10.1093/protein/10.7.743>.
21. Kunzmann P, Mayer B. *Gecos* 1.1.0 repository snapshot. 2019. <https://doi.org/10.5281/zenodo.3490531>.

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.