Memetic Genetic Algorithms for Time Series Compression by Piecewise Linear Approximation

Tobias Friedrich, Martin S. Krejca(✉), J. A. Gregor Lagodzinski, Manuel Rizzo, and Arthur Zahn

Hasso Plattner Institute, University of Potsdam, Potsdam, Germany
{tobias.friedrich,martin.krejca,gregor.lagodzinski}@hpi.de, arthur.zahn@student.hpi.de

Abstract. *Time series* are sequences of data indexed by time. Such data are collected in various domains, often in massive amounts, such that storing them proves challenging. Thus, time series are commonly stored in a compressed format. An important compression approach is *piecewise linear approximation* (PLA), which only keeps a small set of time points and interpolates the remainder linearly. Picking a subset of time points such that the PLA minimizes the mean squared error to the original time series is a challenging task, naturally lending itself to heuristics. We propose the *piecewise linear approximation genetic algorithm* (PLA-GA) for compressing time series by PLA. The PLA-GA is a memetic $(\mu + \lambda)$ GA that makes use of two distinct operators tailored to time series compression. First, we add special individuals to the initial population that are derived using established PLA heuristics. Second, we propose a novel local search operator that greedily improves a compressed time series. We compare the PLA-GA empirically with existing evolutionary approaches and with a deterministic PLA algorithm, known as Bellman's algorithm, that is optimal for the restricted setting of sampling. In both cases, the PLA-GA approximates the original time series better and quicker. Further, it drastically outperforms Bellman's algorithm with increasing instance size with respect to run time until finding a solution of equal or better quality – we observe speed-up factors between 7 and 100 for instances of 90,000 to 100,000 data points.

Keywords: Genetic algorithms · Time series compression · Piecewise linear approximation · Hybridization · Experimental study

1 Introduction

In the modern age of Industry 4.0 and the Internet of Things, sensors are used in abundance in smart devices, cars, or to monitor production facilities. This leads to a huge accumulation of data over time, whose collection speed is only going to increase in the nearby future [9]. A prevalent type of such sensory data are time series – sequences of measurements indexed by time. In order to store these vast amounts of data in the long term and to allow scalable data analysis techniques, it is often necessary to compress time series [17,22]. To this end, various approaches exist that construct a time series of reduced dimensionality as an approximation of the original, known as *time series representation* [15,27].

In this setting, the use of *lossy compression* methods [18,19,37], adopted from the research on multimedia data, has been often advocated under several terminologies. These methods trade precision for a higher compression factor so that the reconstructed time series is only an approximation of the original, frequently omitting noise, outliers, and other information deemed not worthy of their memory cost. While some of these methods represent the time series in a transformed domain (e.g., involving discrete Fourier [1] or Wavelet transforms [31]), in certain application fields, it is important to maintain the original time domain and the related time stamps information, such as in GPS or accelerometer data [20].

In this work, we study *piecewise linear approximation* (PLA [21,24]), which is among the most important time series representation procedures for lossy compression and has been shown to be efficient in terms of memory and transmission cost compared to other similar methods [37]. PLA compresses a time series by representing it via a sequence of linear segments whose quality is assessed by an error measure between the original and the reconstructed series. This leads to a combinatorial problem similar to NP-hard problems like cluster analysis or subset selection in regression [35]. Consequently, heuristics are applied. We analyze to what extent evolutionary algorithms (EAs) can be used for this problem.

Related Work. Using EAs as heuristics for time series analysis is not a novel approach [3]. However, most research is focused on detecting break points in time series, that is, certain points in the series that have interesting or important structural properties [3,5,7]. To the best of our knowledge, the only prior research concerning EAs for compression by PLA was conducted by Duràn-Rosal et al. [10–12]. The authors consider the restricted setting of *sampling,* for which the compressed series is restricted to consist only of points contained in the original time series. Their works introduce a memetic genetic algorithm [12], particle swarm optimization algorithm [10], and coral reef optimization algorithm [11], all augmented with a local search operator to improve intermediate solutions.

Outside the domain of EAs, other heuristics have been considered for compression by PLA [34]. Especially, in the restricted setting of sampling, Bellman's algorithm [4] is optimal and deterministic. However, compression by PLA usually concerns the slightly different setting where one is not given a number of points m to pick but an error bound ε instead [8,14,26]. The goal is to find a compressed series with a mean squared error (MSE) of at most ε to the original time series. Thus, the number of points to pick can be chosen freely by the compression algorithm. Recent advances were made in this area [23,36]. The benefit of this approach is that the result is guaranteed to be within the specified error distance (if a result is returned). On the downside, the user has no control over the memory that the compressed time series occupies. However, note that the approaches of a given error bound ε and a given compression size m can be roughly converted into one another by performing a binary search on the parameter not provided.

Our Results. We propose the *piecewise linear approximation genetic algorithm* (PLA-GA, Alg. 1), a memetic $(\mu + \lambda)$ genetic algorithm (GA) variant for general time series compression via PLA. It features a seeding and a local search operator, both specific to this domain. Seeding adds special individuals to the initial population, which are computed via established PLA heuristics. The local search operator is a novel contribution that greedily improves a compressed time series.

We empirically analyze the impact of these two memetic operators on the PLA-GA's performance (see 4.2). We find that both operators are favorable, as they help the PLA-GA improve its MSE as well as its run time (see Fig. 1). We then evaluate the performance of the PLA-GA against competing approaches (see Sect. 4.3 and 4.4), which all operate under the sampling restriction. First, we compare the PLA-GA to the two latest and best-performing EAs [10,11] (see Sect. 4.3), which are the only EAs for compression by PLA, to the best of our knowledge. We observe that the PLA-GA outperforms them both with respect to the MSE as well as run time (see Table 1). Second, we compare the PLA-GA to Bellman's algorithm [4] (Sect. 4.4), a deterministic optimal algorithm for the sampling restriction, which the PLA-GA is not restricted to. First, we provide the PLA-GA with a run time budget equivalent to that of Bellman's algorithm. We observe a large variance of the MSE in the results, with the PLA-GA usually outperforming Bellman's – reducing the error of Bellman's up to 55 % (Table 2). Then, we analyze how long the PLA-GA takes to find a solution of Bellman's quality or better. Our results show that the PLA-GA drastically outperforms Bellman's algorithm, expressing speed-up factors between 7 and 100 for instances of 90,000 to 100,000 data points (see Fig. 2). Overall, our results suggest that the PLA-GA is particularly well suited to compression by PLA for long time series.

2 Preliminaries

For $m, n \in \mathbb{N}$, let $[m..n]$ denote the set of all natural numbers in the interval $[m, n]$. A *time series* is a function $S\colon \mathbb{R}_{\geq 0} \to \mathbb{R}$ with a finite domain of $n \in \mathbb{N}^+$ elements, where n is the *length* of S. We call the domain of S, denoted by $\operatorname{dom}(S)$, the *time points of* S. Furthermore, for an index $i \in [1..n]$, let t_i^S denote the i-th smallest time point of S, and let v_i^S denote its function value $S(t_i^S)$. Given a time series S of length n, we call a time series C of length m a *compression of* S if and only if $\operatorname{dom}(C) \subseteq \operatorname{dom}(S)$, $m \leq n$, and if $t_1^S = t_1^C$ and $t_n^S = t_m^C$. Note that while we demand that the time points of C are a subset of those of S, we make no such claims for the function values of C and S. We view C as a piecewise linear approximation (PLA) of S by interpolating values for the time points in $\operatorname{dom}(S) \setminus \operatorname{dom}(C)$ using linear functions. More formally, we define the *PLA of* S *via* C, denoted as C^*, to be a time series of length n with $\operatorname{dom}(C^*) = \operatorname{dom}(S)$

such that, for all $i \in [1..m-1]$ and all $t \in \mathrm{dom}(S) \cap [t_i^C, t_{i+1}^C]$, it holds that $C^*(t) = v_i^C + \big(t - t_i^C\big)\,(v_{i+1}^C - v_i^C)/(t_{i+1}^C - t_i^C)$.

Approximation Error by PLA. Given a time series S and a compression C, we quantify the approximation error via the *mean squared error* (MSE), which is a common measures [10,14,23,25]. We define $\mathrm{MSE}(C,S) = \frac{1}{n}\sum_{i=1}^{n}\big(v_i^S - C^*(t_i^S)\big)^2$.

3 The Piecewise Linear Approximation Genetic Algorithm

We introduce the $(\mu+\lambda)$ *piecewise linear approximation genetic algorithm* (PLA-GA; Algorithm 1), a memetic genetic algorithm following a $(\mu+\lambda)$ GA outline and using PLA to compress a time series of length n down to length $m \leq n$, minimizing the MSE. The PLA-GA makes use of a *seeding* and a *local search* operator, both of which are tailored to improve the quality of compressed time series. Please find the source code as well as a detailed description of the PLA-GA on GitHub.[1]

Each individual represents a compressed time series C of length m for a given time series S of length n. Since we minimize the MSE of C^* and S, we only store the time points $\mathrm{dom}(C)$; we compute the values v_i^C optimally with respect to the MSE only if needed. That is, we represent a compressed time series C by a bit string x of length n such that, for all $i \in [1..n]$, $x_i = 1$ if $t_i^S \in \mathrm{dom}(C)$, and $x_i = 0$ otherwise, as is common [11,12]. Note that x has exactly m 1s. Recall that we demand each compressed time series to contain the first and last time point of S. Thus, for each individual x, it holds that $x_1 = x_n = 1$. If a bit string fulfills these conditions, we call it *valid,* otherwise *invalid.*

Algorithm 1: The $(\mu+\lambda)$ PLA-GA with parameters $\alpha \in [0,1]$, $k \in \mathbb{N}$, $\mu \geq k$, $\lambda \in \mathbb{N}^+$, and $m \geq 2$, compressing a time series S of length n down to length $m \leq n$, minimizing the MSE

```
1  $P_1 \leftarrow \mu - k$ individuals, each uniformly at random from $\{x \in \{0,1\}^n \mid x \text{ is valid}\}$;
2  $P_2 \leftarrow k$ individuals, each generated by one of the $k$ seeding operators;
3  $P \leftarrow P_1 \cup P_2$;
4  while termination criterion not met do
5      $L \leftarrow \mathrm{ranking\_selection}_\lambda(P, S)$;
6      $O \leftarrow \mathrm{recombine}(L)$;
7      $\mathrm{bitflip\_repair}(O)$;
8      $\mathrm{bitswap\_mutation}_\alpha(O)$;
9      foreach $x \in O$ do  $\mathrm{local\_search}(x, S)$;
10     $P \leftarrow$ out of $P \cup O$, choose $\mu$ best individuals;
```

[1] https://github.com/arthurz0/ga-for-time-series-compression-by-pla/.

In order to determine the *fitness* of an individual for C, that is, its MSE to S, we first determine the values v_i^C optimally as proposed by Marmarelis [25]. This approach is based on solving a system of m linear equations, where each equation represents the error of a piecewise linear function of C to S. Solving these equations takes time in $\mathcal{O}(n)$. Afterward, we compute the MSE.

4 Experimental Evaluation

We evaluate the performance of the PLA-GA empirically in different settings. In Sect. 4.2, we analyze the impact of the memetic operators on the PLA-GA's performance by comparing different variants of hybridization. We find that both memetic operators improve the quality of the solutions and the time to find them.

Afterward, we compare the PLA-GA against other algorithms that compress a time series S of length n to a time series C of length m. These algorithms take a simplified approach to PLA by only choosing points of the original series, that is, for all $t \in \mathrm{dom}(C)$, it holds $C(t) = S(t)$, called *sampling*.

In Sect. 4.3, we compare the PLA-GA against two memetic EAs of Duràn-Rosal et al. [10,11], which, to the best of our knowledge, are the currently best EAs for compression by PLA. Since the EAs require a predetermined budget N of evaluations, we give the PLA-GA a roughly equivalent computation time budget. We observe that the PLA-GA outperforms the EAs during the entire budget.

In Sect. 4.4, we analyze the quality of the MSE achieved by the PLA-GA when it is not bound to a fixed time budget. To this end, we use Bellman's algorithm [4] as a baseline, which is deterministic and optimal under the restriction of sampling. First, we compare the quality reached when giving both algorithms the same run time budget on data sets of up to 30,000 points. The results depend strongly on the data set, ranging from solutions of similar quality to significant improvements of up to 0.55 times the MSE of Bellman in the best experiment. Then, we analyze how much time the PLA-GA needs to find solutions of the same quality as Bellman on problem instances of increasing size. We observe that the PLA-GA scales far better, reaching speed-up factors between 7 and 100 for problem sizes of 90,000 or 100,000 points.

In all of our experiments, we run the PLA-GA with $\mu = 200$, $\lambda = 200$, and $\alpha = \frac{0.8}{m-2}$ (and $k = 3$). These parameters were determined in pilot experiments. The experiments were conducted on an Intel(R) Core(TM) i5-6200U CPU @ 2.30 GHz with 8 GB RAM, unless noted otherwise.

4.1 Data Sets

Most of our time series originate from the publicly accessible UCR Time Series Classification Archive [32], which contains sets of time series for classification purposes. We use the data sets *Rock*, *Ham*, *HandOutlines*, *Mallat*, *Phoneme,* and *StarLightCurves* from different application fields. Since we aim to conduct experiments on time series of up to 100,000 points but the classification data sets consist of instances with a length between 431 (Ham) and 2844 (Rock) points, we concatenate the instances of each data set until the desired length is reached. An exception to this is Mallat (8192 points), where we use the same version as Duràn-Rosal et al. in [10,11].

Further, we use the "PAMAP2 Physical Activity Monitoring Dataset" [28,29] from the UCI Machine Learning Repository [33], containing data from real-world activity tracking. In particular, we use the columns 5 (*Subject5*) and 6 (*Subject6*) from the protocol of *Subject103*, both of which contain 252,311 points of 3D-acceleration data. Since the time series are too long for our purposes, we only take the first n points, where the specific value of n is mentioned in each experiment.

4.2 Evaluation of Hybridization

We examine the benefit added by hybridization and evaluate its cost, demonstrating that both, seeding and local search, improve the PLA-GA.

Setup. We run the PLA-GA in the following four configurations: with no seeding and no local search (PLA-GA-B), with seeding but no local search (PLA-GA-S), with local search but no seeding (PLA-GA-L), and the main algorithm PLA-GA, which uses both seeding and local search. We measure their MSE per iteration and determine the run time cost of seeding and local search. We have 50 independent runs per algorithm and choose $m = 0.01n$.

Results. Fig. 1 shows the MSEs of each PLA-GA variant by iteration. Although it is hardly noticeable in the figure, note that, bar statistical inaccuracies, PLA-GA and PLA-GA-S have the same best MSE in the initial population (iteration 0), as both use seeding, which deterministically introduces high-quality solutions. Similarly, both PLA-GA-B and PLA-GA-L start with roughly the same best MSE, as neither uses seeding. Further, the MSEs of each algorithm are concentrated around the median, as the interquartile ranges are merely visible.

On all data sets, PLA-GA-B performs worst and the PLA-GA best. In between we have PLA-GA-L, which starts worse than PLA-GA-S but overtakes it after some time, on Phoneme even at iteration 1. This substantiates the intuition that seeding helps during the start of the algorithm (since it adds good individuals to the initial population), whereas the gain of local search is incremental and adds up over time. On some data sets, the three better configurations end up with very close MSEs, though it should be considered that small differences gain significance the closer the solutions get to the optimum. When looking at the number of iterations required to reach a specific level of quality, the speed boost granted by the hybridization operators becomes evident.

For the full picture, one must further consider the computational effort of the hybridization operators. The creation of the initial population without seeding costs roughly as much as 1.3 iterations without local search; adding seeding increases this cost by roughly 25 %, so the computational effort of seeding is

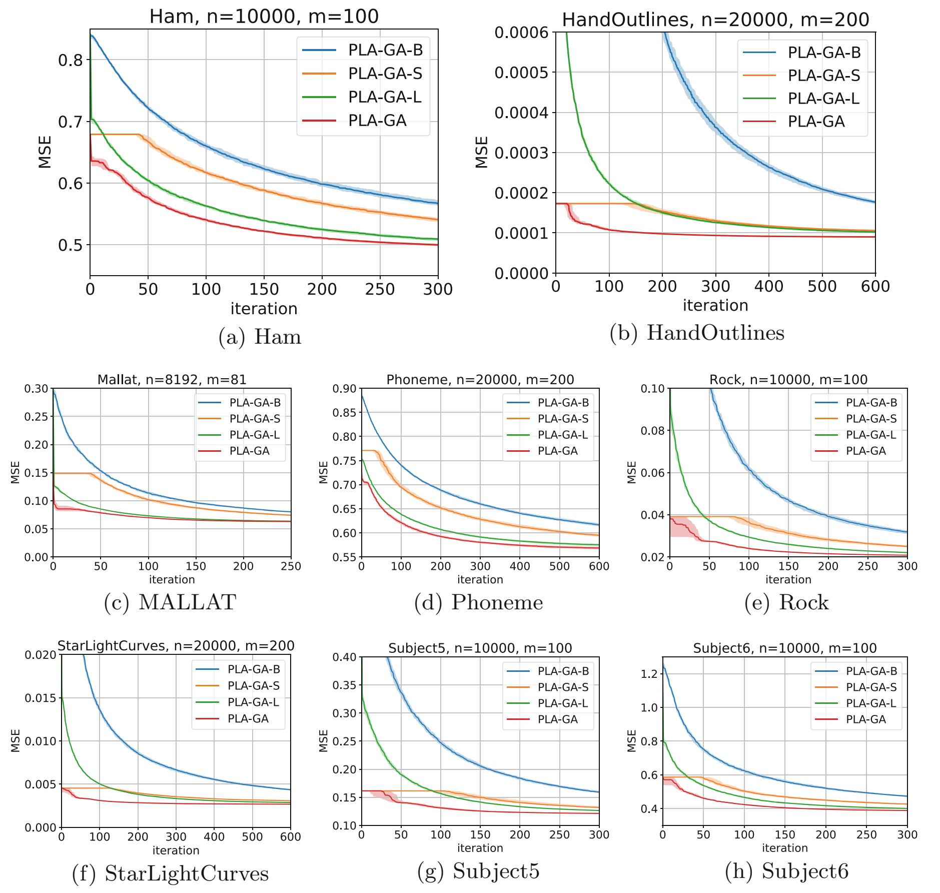

Fig. 1. The MSE of the best solution for four different variants of the PLA-GA (Algorithm 1) plotted against the iteration, compressing a time series of length n down to length m. The solid lines represent the median of 100 runs per algorithm, and the surrounding shaded areas depict the mid 50 %. The legend of each plot denotes the order of the curves in the last iteration. Please also refer to Sect. 4.2.

almost irrelevant for the total run time. The local search, on the other hand, is very expensive: an iteration with the local search takes roughly 2.9 times as long as an iteration without. Nonetheless, the quality benefit of local search makes it worthwhile, leading to an overall better performance despite its high cost.

4.3 Comparison with Other Evolutionary Algorithms

Duràn-Rosal et al. propose several memetic EAs for compressing time series by PLA under the sampling restriction [10–12]. Their best are the particle swarm algorithm *ACROTSS* [10] and the coral reef algorithm *DBBePTOSS* [11]. Both algorithms require a bound N on the number of evaluations in advance, as they

Table 1. Median best MSE of each of the 100 runs for the algorithms DBBePTOSS [11], ACROTSS [10], and the PLA-GA (Algorithm 1) when compressing a time series of length n. The results state the MSE after the first and the last iteration within the given budget. Scale denotes the factor that applies to all MSEs in the table. Please also refer to Sect. 4.3.

Data set	n	Scale	DBBePTOSS		ACROTSS		PLA-GA	
			Start	End	Start	End	Start	End
Mallat	8192	10^{-5}	15889	2207	1427	999	626	410
Ham	10000	10^{-3}	982	474	399	330	246	196
Rock	10000	10^{-5}	15264	1405	1122	918	906	682
Subject5	10000	10^{-4}	5813	1148	920	791	750	598
Subject6	10000	10^{-3}	1251	408	347	308	301	256
HandOutlines	20000	10^{-7}	45619	538	377	335	294	215
Phoneme	20000	10^{-3}	1122	588	534	496	484	393
StarLightCurves	20000	10^{-6}	30833	1115	789	684	319	185

perform local searches after certain iterations, relative to N. When comparing these algorithms to the PLA-GA, we provide it with a similar budget. However, since the PLA-GA's computational effort of a fitness evaluation is much higher due to calculating optimal function values for the compressed time series first, we instead give the PLA-GA a wall-clock-time budget matching the converted median run time of the fastest competing approach.

Setup. For the EAs of Duràn-Rosal et al., we choose the parameters as stated in the respective publications, including the budget of $N = 3.5n$ evaluations and compression length $m = 0.025n+1$. For each algorithm, we start 100 independent runs and log the MSE of the best individual in the population in each iteration. Unfortunately, the EAs of Duràn-Rosal et al. are implemented in Matlab, while the PLA-GA runs in Julia. Thus, we cannot directly take the wall-clock time of the EAs as budget for the PLA-GA. Instead, we divide the times of the EAs by 5, which is roughly the speed-up of our implementation of Bellman's algorithm [4] when run in Julia compared to Matlab. However, we acknowledge that no constant conversion factor exists [2,30].

Results. Table 1 shows the median best MSE of each algorithm for the first and last iteration. The PLA-GA clearly outperforms the competing approaches, as even its starting values are consistently better than the end values of DBBePTOSS and ACROTSS. This is most likely due to the less restrictive approach of the PLA-GA and its use of seeding.

Table 2. Median MSE of the best solution of 50 runs of the PLA-GA (Algorithm 1) after a computation time equivalent to Bellman's algorithm [4] when compressing a time series of length n. Ratio shows the median MSE of the PLA-GA divided by that of Bellman's. Best ratio is the best MSE of the PLA-GA divided with the MSE of Bellman. Scale denotes the factor that applies to all MSEs in the table. Please also refer to Sect. 4.4.

Data set	n	Scale	Bellman	PLA-GA	Ratio	Best ratio
Mallat	8192	10^{-4}	911	741	0.81	0.79
Ham	20000	10^{-3}	512	496	0.97	0.96
Rock	20000	10^{-3}	270	184	0.68	0.64
Subject5	30000	10^{-3}	131	121	0.92	0.91
Subject6	30000	10^{-3}	300	287	0.96	0.94
HandOutlines	20000	10^{-7}	1663	910	0.55	0.54
Phoneme	20000	10^{-3}	566	573	1.01	1.00
StarLightCurves	20000	10^{-5}	418	270	0.65	0.64

4.4 Comparison with Bellman

We compare the MSE and the wall-clock time of the PLA-GA against Bellman's algorithm [4], which deterministically computes an optimal solution in the restricted setting of sampling. First, we examine the MSE of the PLA-GA when giving it a budget equivalent to Bellman's wall-clock-time. Second, we analyze how the PLA-GA's run time scales with the input size n in comparison to Bellman's algorithm, which has a rather slow run time of $\mathcal{O}(n^2m)$.

Setup. Both sets of experiments have 50 independent runs per algorithm and $m = 0.01n$. In the first set, the PLA-GA has a run time budget equivalent to the wall-clock-time of Bellman's algorithm, and the time series have up to 30,000 points. For the second set, we stop the PLA-GA once it finds a solution equal to Bellman's quality or better for the first time. The second set was conducted on an Intel(R) Core(TM) i5-6500 CPU @ 3.20 GHz and 8 GB RAM.[2]

Results. Table 2 compares the MSE of the solutions found by the PLA-GA and Bellman with the same amount of time; a small ratio is desirable. The results vary drastically between the experiments, ranging from similar MSEs on some data sets to huge improvements over Bellman on others. This variance is to be expected, as the difference between Bellman and the best ratio, which is an upper bound on the global optimum, also differs between each data set.

[2] Bellman's algorithm was modified to calculate the segment errors incrementally on the fly. This reduced the memory complexity from $\mathcal{O}(n^2)$ to $\mathcal{O}(mn)$, allowing us to run larger experiments. This modification does not affect the asymptotic run time.

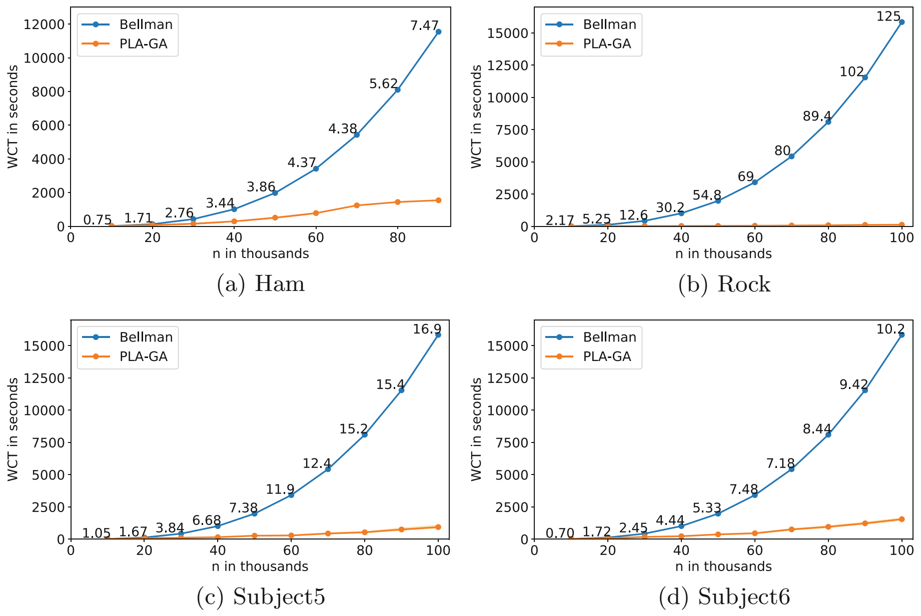

Fig. 2. Median Wall-clock time it takes for the PLA-GA (Algorithm 1) and Bellman's algorithm [4] on increasing instance sizes. The PLA-GA is stopped once it finds a solution of at least Bellman's quality. In each figure, the lower curve shows the run time of the PLA-GA. The numbers over Bellman's curve denote the speed-up factor for that value of n, which is the quotient of Bellman's run time divided by that of the PLA-GA. Please also refer to Sect. 4.4.

Figure 2 compares the run times of Bellman and the PLA-GA with increasing problem size n and shows the resulting speed-up factors. The results differ strongly between the data sets, reaching a speed-up of 7 for Ham on 90,000 and 100 for Rock on 100,000. Thus, the PLA-GA seems to be more sensitive to the type of data than Bellman. Nonetheless, the speed-up factor clearly grows with n across all four data sets, indicating that the PLA-GA scales far better than Bellman.

5 Conclusions

We introduced the PLA-GA, which is a memetic $(\mu + \lambda)$ GA for compressing a time series by PLA. We showed that its two memetic operators – seeding and local search – have a positive impact on the algorithm's solution quality and run time. Further, we ran experiments comparing the PLA-GA to other EAs for time series compression and an optimal deterministic algorithm for a more restrictive setting. We observed that the PLA-GA outperforms all of these approaches with respect to solution quality and run time, leading to speed-up factors between 7 and 100 in comparison to the optimal algorithm. This indicates a clear advantage of the PLA-GA to the existing approaches.

An interesting question for future research is how well the PLA-GA performs on data streams instead of fixed time series. Especially in industrial settings, time series data are produced in streams in rapid succession. Analyzing how well the compression works if the PLA-GA only sees certain chunks of data instead of the entire data set would provide insights into whether the PLA-GA is also useful for on-the-fly compression. Moreover, it would be interesting to observe whether including a statistical modeling step in the procedure is beneficial. In fact, we currently use a mathematical procedure for obtaining optimal predicted values with respect to the mean squared error, but taking advantage of the dependence between time points by statistical modeling (e.g., local linear smoothing [16]) before this step could further improve the quality of the approximation.

Acknowledgments. We thank Durán-Rosal et al. [10–12] very much for immediately providing us with the source code of their evolutionary algorithms upon request.

References

1. Agrawal, R., Faloutsos, C., Swami, A.: Efficient similarity search in sequence databases. In: Proceedings of FODO 1993, pp. 69–84 (1993)
2. Aruoba, S.B., Fernández-Villaverde, J.: A comparison of programming languages in economics. Working Paper 20263, National Bureau of Economic Research (2014)
3. Baragona, R., Battaglia, F., Poli, I.: Evolutionary Statistical Procedures. An evolutionary computation approach to statistical procedures designs and applications. Springer (2011).https://doi.org/10.1007/978-3-6SD42-16218-3
4. Bellman, R., Kotkin, B.: On the approximation of curves by line segments using dynamic programming. RAND Corporation, II. Technical report (1962)
5. Cucina, D., Rizzo, M., Ursu, E.: Multiple changepoint detection for periodic autoregressive models with an application to river flow analysis. Stoch. Environ. Res. Risk Assess. **33**(4–6), 1137–1157 (2019)
6. Doerr, B., Doerr, C., Kötzing, T.: Static and self-adjusting mutation strengths for multi-valued decision variables. Algorithmica **80**(5), 1732–1768 (2018)
7. Doerr, B., Fischer, P., Hilbert, A., Witt, C.: Detecting structural breaks in time series via genetic algorithms. Soft. Comput. **21**(16), 4707–4720 (2017)
8. Douglas, D.H., Peucker, T.K.: Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. Cartograph. Int. J. Geograph. Inf. Geovis. **10**(2), 112–122 (1973)
9. Dunning, T., Friedman, E.: Time Series Databases: New Ways to Store and Access Data. O'Reilly and Associates, Sebastopol (2014)
10. Durán-Rosal, A.M., Gutiérrez, P.A., Poyato, Á.C., Hervás-Martínez, C.: A hybrid dynamic exploitation barebones particle swarm optimisation algorithm for time series segmentation. Neurocomputing **353**, 45–55 (2019)
11. Durán-Rosal, A.M., Gutiérrez, P.A., Salcedo-Sanz, S., Hervás-Martínez, C.: Dynamical memetization in coral reef optimization algorithms for optimal time series approximation. Progress in AI **8**(2), 253–262 (2019)
12. Durán-Rosal, A.M., Peáa, P.A.G., Martínez-Estudillo, F.J., Hervás-Martínez, C.: Time series representation by a novel hybrid segmentation algorithm. In: Proceedings of HAIS 2016, vol. 9648, pp. 163–173 (2016)

13. Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. NCS. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-44874-8
14. Elmeleegy, H., Elmagarmid, A.K., Cecchet, E., Aref, W.G., Zwaenepoel, W.: Online piecewise linear approximation of numerical streams with precision guarantees. PVLDB **2**(1), 145–156 (2009)
15. Esling, P., Agon, C.: Time-series data mining. ACM Comput. Surveys (CSUR) **45**(1), 1–34 (2012)
16. Fan, J., Yao, Q.: Nonlinear Time Series. Nonparametric and parametric methods. Springer (2008). https://doi.org/10.1007/978-0-387-69395-8
17. Fu, T.C.: A review on time series data mining. Eng. Appl. Artif. Intell. **24**(1), 164–181 (2011)
18. Hollmig, G., et al.: An evaluation of combinations of lossy compression and change-detection approaches for time-series data. Inf. Syst. **65**, 65–77 (2017)
19. Hung, N.Q.V., Jeung, H., Aberer, K.: An evaluation of model-based approaches to sensor data compression. IEEE Trans. Knowl. Data Eng. **25**(11), 2434–2447 (2013)
20. Hurvitz, P.M.: GPS and accelerometer time stamps: proper data handling and avoiding pitfalls. In: Proceedings of ACM SIGSPATIAL 2015, pp. 94–100 (2015)
21. Keogh, E., Chu, S., Hart, D., Pazzani, M.: Segmenting time series: a survey and novel approach. In: Data Mining in Time Series Databases, pp. 1–21. World Scientific (2004)
22. Last, M., Horst, B., Abraham, K.: Data mining in time series databases. World Scientific (2004)
23. Lin, J.W., Liao, S.W., Leu, F.Y.: Sensor data compression using bounded error piecewise linear approximation with resolution reduction. Energies **12**(13), 2523 (2019)
24. Lovrić, M., Milanović, M., Stamenković, M.: Algoritmic methods for segmentation of time series: an overview. JCBI **1**(1), 31–53 (2014)
25. Marmarelis, M.G.: Efficient and robust polylinear analysis of noisy time series. arXiv preprint, CoRR abs/1704.02577 (2017)
26. Ramer, U.: An iterative procedure for the polygonal approximation of plane curves. Comput. Graph. Image Process. **1**(3), 244–256 (1972)
27. Ratanamahatana, C.A., Lin, J., Gunopulos, D., Keogh, E., Vlachos, M., Das, G.: Mining time series data. In: Data Mining and Knowledge Discovery Handbook, pp. 1069–1103. Springer, Boston (2005). https://doi.org/10.1007/0-387-25465-X_51
28. Reiss, A., Stricker, D.: Creating and benchmarking a new dataset for physical activity monitoring. In: Proceedings of PETRA 2012, pp. 1–8 (2012)
29. Reiss, A., Stricker, D.: Introducing a new benchmarked dataset for activity monitoring. In: Proceedings of ISWC 2012, pp. 108–109 (2012)
30. Sinaie, S., Nguyen, V.P., Nguyen, C.T., Bordas, S.: Programming the material point method in Julia. Adv. Eng. Softw. **105**, 17–29 (2017)
31. Struzik, Z.R., Siebes, A.: Wavelet transform in similarity paradigm. In: Proceedings of PAKDD, pp. 295–309 (1998)
32. The UCR time series classification archive. http://www.springer.com/lncs. Accessed 27 June 2020
33. UCI machine learning repository. http://archive.ics.uci.edu/ml. Accessed 27 June 2020
34. Visvalingam, M., Whyatt, J.D.: Line generalisation by repeated elimination of points. Cartograph. J. **30**(1), 46–51 (1993)
35. Welch, W.: Algorithmic complexity: three NP-hard problems in computational statistics. J. Stat. Comput. Simul. **15**, 17–25 (1982)

36. Zhao, H., Dong, Z., Li, T., Wang, X., Pang, C.: Segmenting time series with connected lines under maximum error bound. Inf. Sci. **345**, 1–8 (2016)
37. Zordan, D., Martínez, B., Vilajosana, I., Rossi, M.: On the performance of Lossy compression schemes for energy constrained sensor networking. ACM Trans. Sens. Netw. **11**(1), 15:1–15:34 (2014)