Computer Physics Communications 142 (2001) 248–254

Computer Physics Communications

www.elsevier.com/locate/cpc

An advanced evolutionary algorithm for parameter estimation of the discrete Kalman filter

Zeke S.H. Chan *, H.W. Ngan, Y.F. Fung, A.B. Rad

Electrical Engineering Department, The Hong Kong Polytechnic University, Hong Kong

Abstract

In this work we design an advanced Evolutionary Algorithm (EA) for optimizing the discrete Kalman filter (KF) model. The EA employs parallel architecture and an advanced mutation operator called the "Selection Follower". Its performance is benchmarked with that of the Expectation-Maximization algorithm (EM) in minimizing the mean-square-error of the KF prediction. Experimental results show that the EA consistently outperforms the EM and runs significantly faster under the same number of function evaluations.

Keywords: Evolutionary algorithm; Genetic algorithm; Kalman filter; Adaptive mutation; Load forecasting

1. Introduction

The discrete Kalman filter (KF) [1] is a robust tracking algorithm that has long been applied to load forecasting, as well as many other engineering fields such as radar tracking. It formulates the least-square filtering problem using state-space method, featuring vector modeling of the random processes and recursive processing of the noisy measurement data.

The major difficulty with using the KF is parameter estimation of the state-space model [2–5]. This is particularly true with large systems and the reasons are two-fold: first, the number of parameters grows exponentially with the state and observation dimension (the "curse of dimensionality"), which in turn induces multiple stationary points (multi-modality) and possibly large ratio of eigenvalues and rotated eigenvectors in the search space. Second, the traditional methods for maximum likelihood (ML) identification of state-space system — the Newton–Raphson (NR) and Expectation-Maximization (EM) method — are local optimizers that converge to only one stationary point at a rate proportional to ratio of eigenvalues. These methods are easily trapped in local-optimum when there are multiple stationary points, and are unable to exploit the existing optima when the eigenvalue ratio becomes large.

In recent years Evolutionary Computation (EC) [6–9] has become a popular global optimization technique. It is a heuristic optimization technique that does not require derivatives or analytical approximation of the objective function, but directly searches for the global optimum using a population of search points. Basing on the principle of Darwinian "Natural Selection" that fit parents have greater survival chance and are more likely to produce fit offspring through the passing on of parental information and the random mutation, EC evolves the population towards the global optimum through iteratively filtering off the less-fit search points and producing the next population with the crossover and mutation genetic operators

* Corresponding author.
E-mail address: eeshchan@inet.polyu.edu.hk (Z.S.H. Chan).

0010-4655/01/$ – see front matter
PII: S0010-4655(01)00332-0

that respectively mix and perturb components of the remaining fit search points. The Schema theorem [9] proves that this iterative procedure exponentially increases the proportion of fit individuals and the average fitness in the population. EC has been successfully applied to high-dimensional test functions [10] and to real-life applications like ARMA load forecasting models, Fuzzy Logic and Neural network models.

However, EC is not without drawbacks. EC often suffers premature convergence, that is when population gets trapped in a local optima and is unable to progress. Common remedies include niching, increasing the mutation rate and periodically re-initializing the population. Premature convergence is particularly severe when encountering high-eigenvalue ratio and eigenvector rotation that commonly exists in high-dimensional problems [11,12].

In this work we apply EC to optimizing the discrete KF model. We design an advanced Evolutionary Algorithm (EA) that uses a parallel architecture and an advanced mutation operator called the "Selection Follower" [13] that specializes in exploiting high-eigenvalue ratio and rotated-eigenvector optima. The EA's structure, operation and settings are explained, and it is used to estimate a set of discrete KF models using both synthetic and real-life observation series. The KF models have 3–10 state dimensions, giving a total of 19–166 components to be estimated. Its performance is benchmarked with that of the Expectation-Maximization algorithm (EM) [2,3]. Experimental results show that EA consistently outperforms the EM in minimizing the mean-square-error of the prediction error and in addition, it runs faster under the same number of function evaluations because it does not require the Kalman Smoothing algorithm as the EM does. We have therefore concluded that the EA is a feasible alternative to optimizing the discrete KF model.

This work is organized as follows. In the next section, we define the problem of the discrete KF model estimation. In Section 3, the structure and operation of the advanced EA are discussed, which include the parallel architecture and the SF mutation operator. In Section 4, the EM is briefly described. Experiments that compare the performance of the EA and the EM are described in Section 5. The findings are concluded in Section 6.

2. Problem definition

The discrete KF operates on the following state-space representation [1]. Let y_k denote the observation vector $(m \times 1)$ and x_k the (possibly unobserved) state vector $(n \times 1)$ at time k, the state-space model is described by

$$x_{k+1} = Fx_k + w_k, \tag{1}$$

$$y_k = Hx_k + v_k, \tag{2}$$

given

$$\begin{aligned} E\{w_k w_l^{\mathrm{T}}\} &= Q\delta_{kl}, \\ E\{v_k v_l^{\mathrm{T}}\} &= R\delta_{kl}, \end{aligned} \tag{3}$$

where F, Q, H, R are $(n \times n)$, $(n \times m)$, $(n \times n)$ and $(m \times m)$ matrices, respectively, w_k and v_k uncorrelated, zero-mean Gaussian noise. Eq. (1) is known as the state equation and Eq. (2) the observation equation. The sequence of observation and state vectors are denoted $Y = [y_1, y_2, \ldots, y_N]$ and $X = [x_1, x_2, \ldots, x_N]$. The KF algorithm and the estimation of the initial state vector and state error are as described in [1]. Thus the system may be described by $\theta = [F, H, Q, R]$, and the total number of variables to be estimated is given by

$$\begin{aligned} &\text{no. of components in } \theta \\ &\quad = n \times n + m \times n + (1 + 2 + \cdots + n) \\ &\qquad + (1 + 2 + \cdots + m). \end{aligned} \tag{4}$$

The first two terms in Eq. (4) refer to the number of variables in the F and H matrices, and the last two to the number of asymmetric variables in the Q and R covariance matrices. In this work, our objective is to estimate the parameters θ of the state-space model such that the mean-square-error (MSE) of the KF observation predictions are minimized.

3. Advanced evolutionary algorithm for optimizing the Kalman filter model

We now present the advanced EA for optimizing the discrete KF model. The EA employs an parallel architecture similar to [10], which comprises λ separate "fields" evolving concurrently and periodically interchanging best-fit individuals. Each field has its own

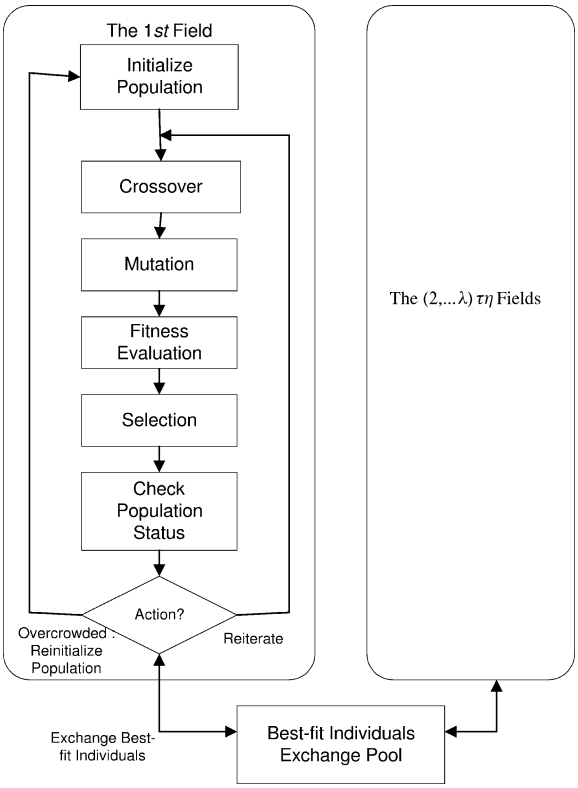

Fig. 1. Schematic diagram of the parallel EA.

population but uses the same genetic operators and respective settings. A schematic diagram of the EA is presented in Fig. 1.

3.1. Mechanism of isolated field evolution

We now describe the evolutionary operations of each field. Unlike Genetic Algorithm (GA), the EA uses real number representation of individuals. Various researches show that real number representation has advantages over binary representation that it avoids discontinuities in the search space [14–17] caused by binary crossover. The EA operates as follows:

(1) *Initialization*. The EA begins with randomly initializing a population of p parents within the given boundaries of each component.
(2) *Reproduction*. Two parents are randomly picked, crossed-over and mutated to produce an offspring. This procedure is repeated until u offspring are produced.
(3) *Fitness evaluation*. The fitness of offspring are evaluated according to their return from the objective function.
(4) *Selection*. Select the next-generation-parents from this-generation-parents and offspring. High-fitness individuals are given higher chance to survive to the next generation.
(5) *Evaluate the population status*. Depending on the density of population (of the newly selected individuals) and the number of generations, the next action could be either to (a) go back to the reproduction phrase and continue evolving, (b) reinitialize the population if the population density is too high (overcrowded) or (c) exchange the best-fit-individuals with other fields if one of every fixed period is reached. The density of the population is calculated as the population's mean displacement from the best-fit individual.

The EA continues until the global optimum is reached to a given precision. If the value of the optimum is not known, the EA continues until a given number of function evaluations is exhausted. The following describes each operation in detail:

- *Real number representation of the solution vectors*. In order to perform the crossover operation, each set of input variables is arranged into a vector called *string*. The order of arrangement is arbitrary but consistent. In the case of the KF model, the sets of input variables are the matrices $\theta = [F, H, Q, R]$. For state dimension $n = 3$ and observation dimension $m = 1$, the total number of variables in the set θ and therefore the length of the string is 19 according to Eq. (4).
- *Crossover*. The crossover is the first phase of the reproduction phase. Two strings are randomly picked from the parent population and the components of the same locii (position of the components on the string) are swapped with probability P_{cross} to produce the child string.
- *Mutation*. The SF mutation is an original operator introduced in [13] that specializes in optimizing functions with high eigenvalue-ratio and rotated eigenvectors. It is also a general purpose mutation operator that can exploit optima in a gradient-like speed. The SF creates the mutant with the formula

$$z_i = x_{li} + k(x_{mi} - x_{ni}) \quad \forall i \in [1, 2, \ldots, D],\ l, m, n \in [1, 2, \ldots, N] \qquad (5)$$

k is a random number uniformly distributed (or any feasible probabilistic distribution) in $[0, \gamma]$ ($\gamma < 1$). It is for scaling the distribution size and introducing variances. The same k value is applied to each of the i components of the vector sized D. The SF uses three parent vectors x_l, x_m and x_n that are randomly taken from the selected population of N strings for creating the mutant: x_l is the central parent, which is mutated by adding the noise vector $k(x_m - x_n)$. With tradition mutation operators, the noise vector has a fixed mean and variance; in the SF case, it is defined by the distribution of the selected-population.

- *Selection*. The standard selection method for Genetic Algorithm is the Roulette Wheel selection, which assigns each string the survival probability proportional to its fitness value [7]. Variations of this scheme are also found [10]. This scheme has the disadvantage that if the magnitude of fitness variation is unknown, the distribution of low- and high-fitness strings becomes unpredictable. In order to remove this uncertainty, we employ the $(\rho + u)$ selection scheme. $(\rho + u)$ selection puts both the parents and offspring in competition (that totals $(\rho + u)$ competitors) and keeps only the best u strings. In this way, the survival of the strings does not depend on the strings' relative fitness values but on their ranks. $(\rho + u)$ selection is also an elitist selection scheme that the best strings always stay in the population.

3.2. Mechanism of exchanging best-fit individuals

The purpose of using multi-field evolutionary technique is two-fold. First, it makes distributed programming possible for gaining speed. Second, it encourages a more diversified search by breaking one large population into many separate niches that may be drawn to different optima. Although these fields evolve independently, they exchange their best-fit individuals periodically so that through genetic operations a field can accumulate the good "building-blocks" [9] from other fields and increase chance to finding the global optimum.

The mechanism of exchanging best-fit individuals is as follows:

(1) *Share to the pool*. A common "share pool" is established to contain the best-fit individuals of the niches. At every fixed number of generation, a field copies its best-fit individual to the share pool.
(2) *Obtain from the pool*. A field then obtains η strings from the share pool. These new strings, shared by other fields, are mixed with the parents to breed new offspring.

The "building blocks" [9] of the new strings from the share pool are extracted with the crossover and mutation operations into the offspring.

4. The expectation-maximization algorithm

The EM was first applied to the discrete KF model estimation in [3]. It simplifies the problem by assuming the state vectors observable, such that the ML estimates of θ are obtained by minimizing the negative log likelihood $J(X, Y, \theta)$

$$\begin{aligned} J(X, Y, \theta) &= -L(X, Y, \theta) \\ &= \sum_{k=1}^{N} \left\{ \log|Q| + (x_k - Fx_{k-1})^{\mathrm{T}} Q^{-1} (x_k - Fx_{k-1}) \right\} \\ &\quad + \sum_{k=0}^{N} \left\{ \log|R| + (y_k - Hx_k)^{\mathrm{T}} R^{-1} (y_k - Hx_k) \right\} \\ &\quad + \text{constant}. \end{aligned} \tag{6}$$

The EM iteratively minimizes this quantity through alternating between the Kalman filtering and smoothing recursions. Details of the EM can be found in [2,3]. Fig. 2 demonstrates a typical EM run in minimizing the MSE of the KF prediction. Note that the MSE decreases all the time. However, in order to ensure stability of the system, the initial estimates of theta must be reasonably close to an optimal set of theta.

5. Experiments

Our experiments aim to verify the feasibility of EA in optimizing the discrete KF model, using the EM as the performance benchmark. The experiments comprise two parts. In the first part, we generate four state-space models of different state dimension $n = (4, 6, 8, 10)$ and the same observation dimension $m = 1$. The elements of these models are randomly assigned but within constraints such that the models are always asymptotically stable. With each model,

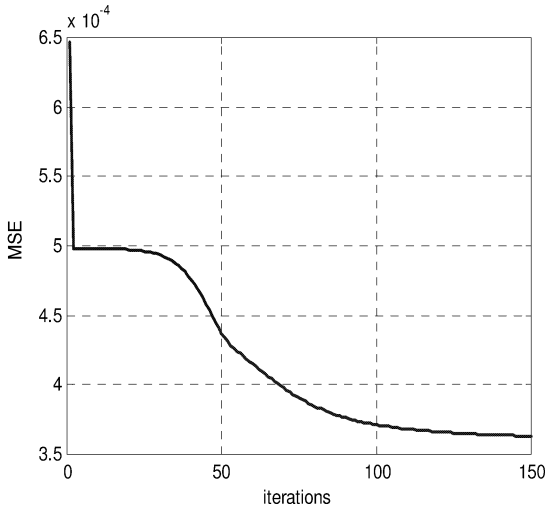

Fig. 2. A typical EM run in identifying the KF model.

we generate a series of 200 observations. We then use the EA and the EM to optimize new models of the same sizes with the corresponding series. Using the MSE as the optimization criteria, we compare the performance of the EA and the EM. In the second part of experiments, we use a real-life de-trended load series to train a KF model, following the same procedures and criteria.

For each observation series, both the EA and EM algorithms are allowed a maximum of 100,000 evaluations to ensure "fair" competition — though they spend their resources in different ways. The EA algorithm spends its 100,000 evaluations over 5 EA runs of 20,000 evaluations each. The EM spends its resources testing different starting points and iterating from them. In order to ensure the KF models are asymptotically stable, we limit each element of F within $1/\sqrt{n}$ such that its eigenvalues is always smaller then 1 [18]. Other limits are determined arbitrarily and are as listed in Table 1.

The settings of the EA and the EM are as listed in Table 2. Fig. 3 shows the detrended load series we use for the second part of the experiments.

The results are listed in Table 3. The results favor the EA in every category, but we must stress that some parts of the comparisons are miss-leading because the EA and the EM spend their computational resources differently. For the EA, the statistics are obtained from the best solutions of each 5 runs of 20,000 evaluations, whereas for the EM, the statistics are obtained from testing and iterating from different points until 100,000 evaluations are exhausted. The mean and maximum values of the EM would therefore be unfavorably compared, because many "bad" points

Table 1
Constraints imposed on the elements of θ

Absolute values of elements of	Constraints
F	$<1/\sqrt{n}$
Q	$<1/\sqrt{n}$
H	<1
R	$<1/\sqrt{m}=1$

Table 2
Settings of the EA and the EM

EA	
Number of parents ρ	20
Number of offsprint u	80
Communication interval (gen)	5
Mutation type	Selection follower
k value for selection follower	0.45
Crossover probability	0.65
Number of fields	8
Number of immigrants each interval	5
Reinitialization density	0.01
EM	
Max. iterations per EM run	200

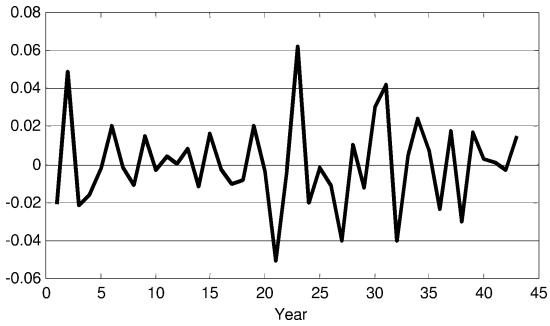

Fig. 3. The normalized and detrended US annual energy series 1949–1996. Data extracted from the Energy Information Administration (EIA) and the Bureau of Economic Analysis.

Table 3
Performance comparison between the EA and the EM over different observation series

	Synthetic data								Real-life data	
n	4		6		8		10		Load series (3)	
Total elements	31		64		109		166		19	
	EA	EM	EA	EM	EA	EM	EA	EM	EA	EM
Min	0.4842	0.5168	1.7614	1.7681	0.9046	0.9221	2.8564	2.8946	3.3297e−4	3.5602e−4
Mean	0.5068	0.5305	1.7659	1.7880	0.9086	0.9609	2.8597	3.0504	3.3598e−4	4.8002e−4
Max	0.5145	0.5952	1.7687	1.9188	0.9146	1.0469	2.8617	3.5404	3.4040e−4	4.9832e−4
Time (s)	12,925	33,953	16,598	44,665	22,515	66,034	31,313	112,724	2643	8166

are included in the statistics. Therefore, we will focus on the comparison of the minimum MSE values, and then the time factor.

Comparing the minimum MSE values, EA performs better in every model. It records lower MSE by roughly 0.03 in every case except for $n = 6$ and the load series (only lower by 0.007 and 0.000027, respectively). Thus for both synthetic series and real-life series, EA outperforms EM.

Now if we inspect only the synthetic series and compare the mean values, EA records lower by $(0.02, 0.02, 0.05, 0.19)$ for $n = (4, 6, 8, 10)$. There is a trend that the higher the model dimension, the better EA performs. A similar trend is observed when we compare the maximum values too. This is expected because EA is a global- and EM is a local-optimization algorithm: the larger and more multi-modal the search space, the better EA performs; and vice-versa.

Regarding the time factor, EA is roughly 2.5–3 times faster than EM is in every case. This time difference is not caused by parallel-processing EA, since we employ only one CPU for running both algorithms. In fact, the EA receives extra overhead for multiplexing one CPU for all 8 fields. The time difference is because EM requires the Kalman Smoothing algorithm to compute statistics for estimating the model parameters. Kalman Smoothing algorithm is an iterative procedure that requires similar amount of computations as the Kalman Filtering algorithm, but in addition it requires memory storage of statistics computed in the Filtering algorithm. Thus comparing with the EM, EA is much less CPU intensive, for it requires only the Kalman filtering algorithm to compute the MSE and its overheads for genetic operations and multiplexing are relatively small. If we incorporate parallel processing, EA will run even faster.

6. Conclusions

In this work we design an advanced EA for estimating the discrete KF model. The EA uses a parallel architecture and an advanced SF mutation operator. Its structure, operation and settings are explained. In order to demonstrate EA's feasibility for optimizing the KF model, EA is benchmarked with the EM and is used to optimize a set of KF models that include both synthetic and real-life observation series. The models' state dimensions n vary from 3 to 10, totaling 19 to 166 components to be estimated. We find that EA consistently outperforms EM in minimizing the MSE of the KF models, and that the larger the dimension of the problem the better EA performs. In addition, because the EA does not require the Kalman Smoothing algorithm as the EM does, EA runs faster by a factor of 2.5–3. We conclude that EA is a robust and feasible alternative for optimizing the discrete KF model.

Acknowledgements

The first author would like to thank Lord Jesus for His ever-extravagant provision, and the financial support given to this project by the Research Committee of the Hong Kong Polytechnic University.

References

[1] J.D. Hamilton, Time Series Analysis, Princeton University Press, Princeton, NJ, 1994.
[2] V. Digalakis, J.R. Rohlicek, M. Ostendorf, ML estimation of a stochastic linear system with the EM algorithm and its application to speech recognition, IEEE Trans. Speech Audio Process. 1 (1993) 431–442.
[3] R.H. Shumway, Applied Statistical Time Series Analysis, Prentice-Hall, Englewood-Cliffs, NJ, 1988.
[4] R.G. Brown, Introduction to Random Signal Analysis and Kalman Filtering, John Wiley & Son, 1983.
[5] Q.C. Lu, W.M. Grady, M.M. Crawford, An adaptive algorithm for short-term multinode load forecasting in power systems, IEEE Trans. Circuits Systems 35 (1988) 1004–1010.
[6] U.H.T. Baeck, H. Schwefel, Evolutionary computation: comments on the history and current state, IEEE Trans. Evolutionary Comput. 1 (1997) 3–17.
[7] T. Baeck, Evolutionary Algorithm in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms, Oxford University Press, New York, 1995.
[8] T. Baeck, U. Hammel, H.-P. Schwefel, Evolutionary computation: comments on the history and current state, IEEE Trans. Evolutionary Comput. 1 (1997) 3–17.
[9] D.E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, Reading, MA, 1989.
[10] H. Muhlenbein, M. Schomisch, J. Born, The parallel genetic algorithm as function optimizer, Parallel Comput. 17 (1991).
[11] R. Salomon, Evolutionary algorithms and gradient search: similarities and differences, IEEE Trans. Evolutionary Comput. 2 (1998) 45–55.
[12] R. Salomon, Accelerating the evolutionary-gradient-search procedure: Individual step sizes, in: Proc. 5th Int. Conf. Parallel problem Solving from Nature, Berlin, Germany, 1998.
[13] Z.S.H. Chan, H.W. Ngan, A.B. Rad, A practical mutation operator and its application to the Kalman filter, in: Proc. PowerCon, Perth, Western Australia, 2000.
[14] L.J. Eshelman, R.A. Caruana, J.D. Schaffer, Biases in the crossover landscape, in: Proc. 3rd ICGA, 1989.
[15] L.J. Eshelman, J.D. Schaffer, Real-coded genetic algorithms and interval-schemata, in: Foundations of Genetic Algorithms, Vol. 2, M. Kaufmann Publishers, San Mateo, CA, 1993, pp. 187–202.
[16] W.M. Spears, Crossover or mutation?, in: Foundations of Genetic Algorithm, Vol. 2, M. Kaufmann Publishers, San Mateo, CA, 1993, pp. 221–237.
[17] L.B. Booker, Recombination distributions for genetic algorithms, in: Foundations of Genetic Algorithms, Vol. 2, M. Kaufmann Publishers, San Mateo, CA, 1993, pp. 29–44.
[18] J.S. Bay, Fundamentals of Linear State Space Systems, WCB/McGraw-Hill, 1999.