DOI 10.1007/s10115-004-0157-6
Springer-Verlag London Ltd. © 2004
Knowledge and Information Systems (2005) 7: 267–288

Knowledge and Information Systems

Using Markov chain Monte Carlo and dynamic programming for event sequence data

Marko Salmenkivi, Heikki Mannila

Helsinki Institute for Information Technology, Basic Research Unit, Department of Computer Science, University of Helsinki, Finland

Abstract. Sequences of events are a common type of data in various scientific and business applications, e.g. telecommunication network management, study of web access logs, biostatistics and epidemiology. A natural approach to modelling event sequences is using time-dependent intensity functions, indicating the expected number of events per time unit. In Bayesian modelling, piecewise constant functions can be utilized to model continuous intensities, if the number of segments is a model parameter. The reversible jump Markov chain Monte Carlo (RJMCMC) methods can be exploited in the data analysis. With very large quantities, these approaches may be too slow. We study dynamic programming algorithms for finding the best fitting piecewise constant intensity function, given a number of pieces. We introduce simple heuristics for pruning the number of the potential change points of the functions. Empirical evidence from trials on real and artificial data sets is provided, showing that the developed methods yield high performance and they can be applied to very large data sets. We also compare the RJMCMC and dynamic programming approaches and show that the results correspond closely. The methods are applied to fault-alarm sequences produced by large telecommunication networks.

Keywords: Data mining; Dynamic programming; Event sequence; MCMC

1. Introduction

Sequences of events are a common type of data arising in various scientific as well as business applications, e.g., telecommunication network management, web access logs, biostatistics and epidemiology. Intuitively, such a sequence consists of pairs (t, e), where t is the occurrence time and e is the type of event. The occurrence times are not predefined; this property separates a sequence of events from a time series, which typically contains observations at determined time points. An event sequence is a collection of events ordered according to the occurrence times. The

Received 24 October 2002
Revised 29 November 2003
Accepted 18 December 2003
Published online 13 May 2004

event sequences arising in applications can be quite long, containing hundreds of thousands of events. In many cases, it is of interest to detect changes in frequency or density of event occurrences as a function of time and to find explanations for the changes—such as occurrences of other kinds of events or values of covariates.

Stochastically, a sequence consisting of (conditionally) independent events can be seen as being produced by a *Poisson process*, which has some underlying *intensity* describing the frequency of events. Time-dependent intensity functions can be used in modelling the frequencies of event sequences.

In this paper, we show that piecewise constant functions provide flexible tools for intensity modelling from the point of view of data mining research as well as that of more traditional statistical modelling. First, we consider the Bayesian modelling approach and reversible jump Markov chain Monte Carlo (MCMC, RJMCMC) methods, which provide a framework for expressing the available knowledge of the values of the model parameters by assigning probability distributions to the parameters.

MCMC-based approaches are fairly slow and hence we study faster approaches based on the maximum-likelihood approach. We show that, if the best fitting piecewise constant function—instead of the distribution of functions—is to be computed, the optimal solution can be found by using dynamic programming in time $\mathcal{O}(n^2k)$, where n is the number of events and k is the number of pieces. This may be too slow as well. We introduce simple heuristics that can be used to prune the number of the potential change points of the functions. Speedups of several orders of magnitude in the performance of the dynamic programming algorithm are reached with minor loss in accuracy of results. We provide empirical results showing that using samples of approximately 0.5–5% of the total number of potential change points is sufficient for nearly optimal results.

Although the Bayesian posterior average and the piecewise constant intensity function giving the maximum likelihood do not describe the same quantity, they do reflect the same phenomenon; that is, highly probable intensity values of the process that generated the data sequence. Hence, it is interesting to compare the results of the two approaches. We show that the resulting piecewise constant intensity functions correspond closely to the posterior averages produced by the MCMC methods.

The rest of this paper is organized as follows. In Sect. 2, we introduce the concepts of event sequence and intensity function. In Sect. 3, we describe the Bayesian modelling approach and Markov chain Monte Carlo simulation methods. The methods are applied to a telecommunication network alarm data in Sect. 4. In Sects. 5 and 6, we describe the dynamic programming approach and some simple techniques for speeding it. We also give empirical results from trials on real data sets; the appendix contains results on simulated data sets. Section 7 is a short conclusion. An earlier version of this paper appeared in Mannila and Salmenkivi (2001).

2. Intensity models for event sequence data

We next define some notation, following Arjas (1989). Let T be the random variable indicating the waiting time for the next event from a fixed starting time of a process containing events of a single type and let $G(t) = Prob(T \leq t)$ be the distribution function of T. Then $G(t)$ is the probability that (after the starting time, some time t_s) the waiting time before the next event is not longer than t. Assume that the density function $g(t)$ of G exists. Then $\overline{G}(t) = 1 - G(t)$ is the *survival function*. The *intensity* *function* $\lambda(t)$ is defined as follows:

$$\lambda(t) = \frac{g(t)}{\overline{G}(t)}.$$

Thus, $\lambda(t)$ expresses the instantaneous probability of occurrence of an event at time t, given that no event occurred before t.

On the other hand, $\lambda(t)$ can also be interpreted as the expected number of occurrences per time unit. Assume that events of an event type occur at a constant intensity λ_c. Then the survival function $\overline{G}(t) = \exp\left(-\int_0^t \lambda(s)ds\right) = e^{-\lambda_c t}$ and the distribution function of the time intervals between the occurrences (T) is $G(t) = 1 - e^{-\lambda_c t}$. This is the distribution function of the exponential distribution; thus, the time intervals $T \sim \mathrm{Exp}(\lambda_c)$.

The intensity function uniquely determines the distribution function $G(t)$ as

$$\lambda(t) = \frac{g(t)}{\overline{G}(t)} = \frac{dG(t)}{dt} \cdot \frac{1}{\overline{G}(t)} = -\frac{d\overline{G}(t)}{dt} \cdot \frac{1}{\overline{G}(t)} = -\frac{d}{dt} \ln \overline{G}(t)$$

and therefore

$$\int_0^t \lambda(s)ds = -\ln \overline{G}(t)$$

and

$$\overline{G}(t) = \exp\left(-\int_0^t \lambda(s)ds\right).$$

When the data contains several event types, we have to define an intensity function λ_A for each event type A.

Piecewise constant functions A piecewise constant intensity function λ has the form

$$\lambda(t) = \begin{cases} \lambda_1 & \text{if } T_s \le t < c_1, \\ \lambda_2 & \text{if } c_1 \le t < c_2, \\ \vdots & \vdots \\ \lambda_i & \text{if } c_{i-1} \le t \le T_e, \\ 0 & \text{elsewhere.} \end{cases}$$

The values $\{T_s, T_e\} \in \mathbf{R}$ are the start and end times of the observation period, the values $\{\lambda_1, ..., \lambda_i\} \in \mathbf{R}^+$ are the intensity values in i pieces and $\{c_1, ..., c_{i-1}\} \in [T_s, T_e]$ are the *change points* of the function.

There are several reasons for using piecewise constant functions. Piecewise constant functions are relatively simple. The arithmetic operations between such functions result in piecewise constant functions. These operations, as well as integration, can be done easily and efficiently. Such functions can, in a flexible way, be used in subtle data analysis and complex statistical modelling, and they still are easy to interpret.

Poisson likelihood Consider an intensity function $\lambda_e(t)$ defined in time range $[T_s, T_e]$ and consider an event sequence $S_e = \{(e, t_1), \ldots, (e, t_n)\}$, where $t_i \in [T_s, T_e]$ for all $i = 1, \ldots, n$. The Poisson likelihood of the data S_e is given by (see, e.g. Guttorp 1995)

$$L(S_e \mid \lambda_e) = \prod_{j=1}^{n} \left\{ \lambda_e(t_j) \cdot \exp\left(- \int_{t_{j-1}}^{t_j} \lambda_e(t)\, dt \right) \right\} \cdot \exp\left(- \int_{t_n}^{T_e} \lambda_e(t)\, dt \right), \quad (1)$$

where $t_0 = T_s$. Because integration is done over the whole time range $[T_s, T_e]$, the likelihood can also be written in the form

$$L(S_e \mid \lambda_e) = \exp\left(- \int_{T_s}^{T_e} \lambda_e(t)\, dt \right) \cdot \prod_{j=1}^{n} \lambda_e(t_j). \quad (2)$$

3. Bayesian approach and MCMC methods for finding piecewise constant intensity models

Next, we briefly describe the Bayesian modelling approach for finding piecewise constant intensity models. For more details, see, e.g. Bernardo and Smith (1994) and Gelman et al. (1995).

Bayesian modelling Assume that a full probability model M_θ is constructed. Let us denote the vector of the model parameters $\theta = \{\theta_1, \ldots, \theta_n\}$, a particular value of θ by $x = \{x_1, \ldots, x_n\}$ and the data Y. The joint probability distribution determined by M_θ is given by

$$P(\theta, Y) = P(\theta) P(Y|\theta). \quad (3)$$

Here $P(\theta)$ is the *prior distribution* of θ and $P(Y|\theta)$ the *likelihood* of the data.

Finding out the *posterior distribution* $P(\theta|Y)$ is the core of Bayesian data analysis. According to Bayes' rule,

$$P(\theta|Y) = \frac{P(\theta) P(Y|\theta)}{P(Y)} = \frac{P(\theta) P(Y|\theta)}{\int_\theta P(\theta) P(Y|\theta) d\theta}. \quad (4)$$

As the probability of the data $P(Y)$ does not depend on θ, the posterior distribution is proportional to the product of the prior distribution and the likelihood.

The desired quantities are often presented as expected values of a function on the posterior distribution. To know the posterior expectation of a function g, one needs integrate over posterior density function f,

$$E(g(\theta|Y)) = \int_\theta g(x)\, f(x)\, dx. \quad (5)$$

In practical situations integration cannot be done analytically or even with classical numerical methods due to the complexity of the density functions.

Markov chain Monte Carlo methods (see, e.g. Gamerman 1997) approximate the target distribution f by constructing a Markov chain that has f as the equilibrium distribution. The sample set $x_1, \ldots, x_N$ drawn from f can then be used for implicit Monte Carlo integration,

$$E_f g(X) = \int g(s) f(s)\, ds \approx \frac{1}{N} \sum_{i=1}^{N} g(x_i). \quad (6)$$

Metropolis–Hastings algorithm Denote by $T(x, y)$ the probability of moving from state x to state y in the chain. The *reversibility condition* states that, for all states x and y, we have

$$f(x)\,T(x, y) = f(y)\,T(y, x). \tag{7}$$

Intuitively, this means that the flow from x to y will be the same as the flow from y to x. Assuming further that the chain is irreducible (all states communicate) and aperiodic (Tierney 1994), then the reversibility condition implies that the equilibrium distribution of the chain is f; see, e.g. Guttorp (1995). Thus, to obtain a suitable Markov chain, we should find a function T satisfying Eq. (7).

Assume that, in state x, a candidate value x' is drawn from a distribution with density $q(x, x')$. If, for instance, for some x, x', we have $f(x)q(x, x') > f(x')q(x', x)$, then the simulation process moves too often from x to x'. To correct the unbalance, the Metropolis–Hastings algorithm rejects a part of the proposals to move from x to x'.

Thus, to meet the reversibility condition, as the probability of transition from x' to x is defined to be 1, we can write $f(x)q(x, x')\alpha(x, x') = f(x')q(x', x)$, where $\alpha(x, x')$ is the probability of accepting a move from x to x' (Chib and Greenberg 1995). Thus, $\alpha(x, x')$ is given by

$$\alpha(x, x') = \min\left(1, \frac{f(x')\,q(x', x)}{f(x)\,q(x, x')}\right).$$

The Metropolis–Hastings algorithm starts from a random state θ_0, repeatedly draws candidate states θ' and accepts them with the above probability. It is straightforward to show that the method gives a Markov chain the equilibrium distribution of which is f. Note that, in order to run the simulation, we only need to be able to determine the quotient $f(x')/f(x)$, i.e. the distribution must be known up to a constant. In case of the Bayesian posterior distribution, the posterior density function is proportional to the product of the prior and likelihood densities. Thus, the Metropolis–Hastings algorithm can be applied provided these densities can be calculated.

Reversible jump MCMC For the case of piecewise constant functions, we need to consider functions having different numbers of pieces. Thus, the number of parameters of a model is not fixed. Consider a situation where the dimension of the current state x is m and a candidate state x' of a higher dimension n is generated. Let u be a vector of random numbers used when proposing the candidate state x' in state x. To ensure that both sides of the reversibility equation (Eq. 7) have densities on spaces of equal dimension, it must hold that $m + dim(u) = n$.

Let us denote by h the function that determines the candidate state in state x, given u; i.e. $x' = h(x, u)$. Then the acceptance rate of the Metropolis–Hastings algorithm generalized by Green (1995) to state spaces of variable dimensions yields

$$\alpha(x, x') = \min\left(1, \frac{f(x')\,q(x', x)}{f(x)\,q(x, x')}\left|\frac{\partial h(x, u)}{\partial x\,\partial u)}\right|\right). \tag{8}$$

Here, the last term is the Jacobian of the transformation of the random variables x and u. The generalization is known as the reversible jump Markov chain Monte Carlo (RJMCMC). For a detailed description of the RJMCMC, see Green (1995).

Bayesian intensity modelling with piecewise constant representations In many cases, it is natural to assume the real intensities to be continuous, not step functions. Even then, the intensity can be represented by using piecewise constant functions, if the number of pieces is one of the model parameters. RJMCMC methods can be used to approximate the posterior distribution of the intensity.

In the following, we describe a simple model where the intensity $\lambda(t)$ of all occurrences of an event is modelled using a piecewise constant function with a variable number of pieces. The levels of the function λ_j, the change times c_j and the number of pieces k are random variables with the following prior specifications. Here α, β and γ are fixed constants and T_s and T_e are the start and end times of the observation period, respectively:

$$\begin{array}{ll} \text{number of pieces } k & \sim \mathrm{Geom}(\gamma) \\ \text{intensity levels } \lambda_j & \sim \mathrm{Gamma}(\alpha, \beta) \\ \text{change points } c_j & \sim \mathrm{Unif}(T_s, T_e). \end{array} \tag{9}$$

An approximation for the posterior expectation of intensity can be computed from the simulation results in the following way. Each simulation round generates a single piecewise constant realization from the posterior distribution $f(\lambda, t)$. Let us denote the intensity value at time instant t_i in the mth realization by $\lambda^m(t_i)$. The Monte Carlo approximation of the posterior expectation of the intensity at t_i is the average of $\lambda^m(t_i)$, i.e.

$$E_f(\lambda(t_i)) = \int_0^\infty \lambda(t_i) f(\lambda, t_i)\, d\lambda \approx \frac{1}{N} \sum_{m=1}^{N} \lambda^m(t_i), \tag{10}$$

where N is the number of iterations of the MCMC simulation.

We now proceed to introducing an application from the field of telecommunications.

4. Application: telecommunication alarm sequences

Telecommunication network software collects lots of information about the performance of the network. One specific type of data is so-called alarms, which are indications of smaller or larger problems in the network (Hätönen et al. 1996).

This data contains hundreds of different event types. It is collected automatically and, accordingly, the amount of data can be several orders of magnitude larger than in the case of traditional data sets. The characteristics of the data change often, as changes in the network structure. Thus, automatic methods are needed in the analysis of the data.

In this section, we apply the RJMCMC methods to modelling fault alarm sequences. In Sects. 5 and 6, dynamic programming methods are used to find piecewise constant intensity functions yielding maximum likelihood, given the number of pieces.

Our methods can, e.g. be used to detect change points of the intensities between stable periods. The change points can be used to create condensed representations of very long sequences. Original or condensed sequences can be used to find interaction patterns of different types of fault alarms. For instance, an interesting question is whether there are event types the increasing intensity of occurrences of which is an indicator of a large problem in the near future. The simple Bayesian piecewise

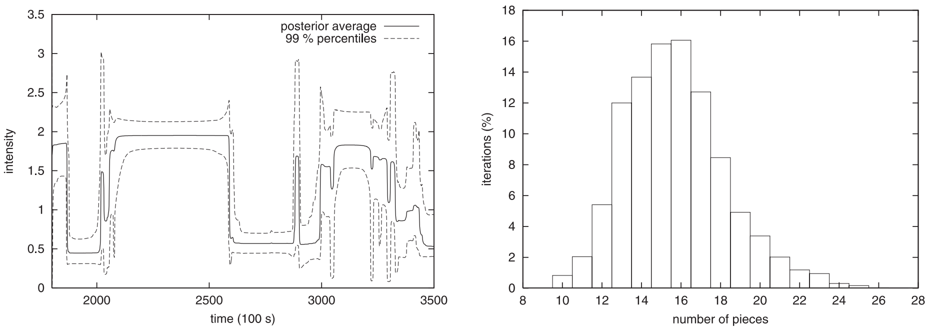

Fig. 1. Posterior average and 99% interval (left) of the telecommunications data set with 2,371 events. Marginal posterior distribution of the number of pieces (right)

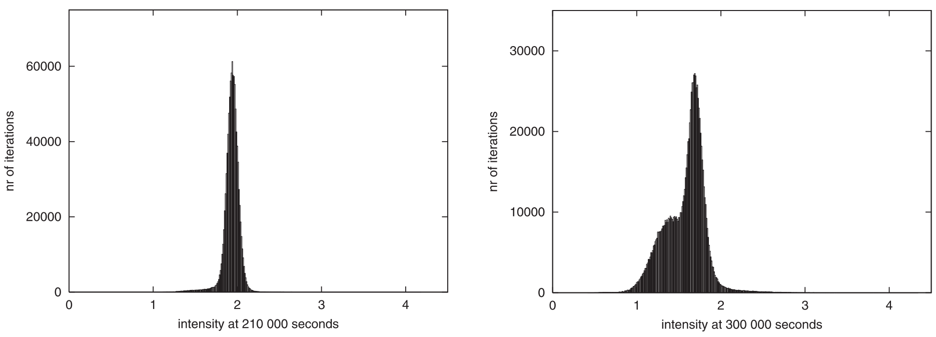

Fig. 2. Conditional posterior distribution of intensity given a time instant t_0; $t_0 = 210{,}000$ seconds (left); $t_0 = 300{,}000$ seconds (right)

constant model can easily be extended to more subtle analysis of interaction between several event types or covariate effects (Eerola et al. 1998).

We illustrate using MCMC methods on a small data set containing 2,371 events. The prior specifications were Geom(0.9995) for the number of pieces and Uniform(0.00001, 1,500) for the intensity-level parameters. We investigated the time period between 180,000 and 380,000 seconds from the start point of the observation. The prior used for the change points was Uniform(180,000, 380,000). From an initial state with 20 pieces, 5,000,000 initial iterations were run before collecting the parameter values. Then 1,000,000 iterations were run, during which the parameter values were collected from every round. The total run time was 55 minutes (Pentium 700 MHz).

The posterior average of the intensity and the 99% percentiles computed at 2,000 time instants are shown on the left in Fig. 1. The marginal posterior distribution of the parameter k, indicating the number of pieces, is presented in the right panel.

The conditional posterior distribution of the intensity given a time instant t_0 is shown in Fig. 2, for the time points $t_0 = 210{,}000$ and $t_0 = 300{,}000$ seconds. Note that both of the conditional distributions depicted here are summarised by three statistics in the left panel of Fig. 1; the percentiles at $2{,}100 \cdot 10^2$ and $3{,}000 \cdot 10^2$ seconds indicate the interval, where 99% of the values of the conditional intensity reside and the posterior average approximates the expectation of the distribution. The conditional distribution at 210,000 seconds provides only a little more information than the average and percentiles; the value is clearly close to 2. The conditional distribution at 300,000 seconds, instead, reveals more detailed information about the distribution of the intensity at the time instant.

5. Intensity modelling using dynamic programming

The Bayesian approach and the MCMC methods provide clear conceptual as well as practical tools that can be used to produce probability distributions expressing the available knowledge of the parameter values. There are, however, problems commonly encountered when applying the methods in practice. First, the convergence of the chain, i.e. how to be convinced that the distribution of the states of the constructed chain really follows the target distribution. And second, the *mixing* of the chain, that is, how to cover the distribution exhaustively in feasible time. For treatment of the problems, see, e.g. Brooks and Giudice (1999) and Brooks and Roberts (1998).

For these reasons, the MCMC methods are not very suitable in circumstances where results have to be produced in a short time. Furthermore, only a small fraction of the information comprised in the posterior distribution is usually utilized in practice. Thus, we are also interested in developing alternative methods, especially for mining very large event sequences.

Instead of trying to find out the posterior distribution of the intensity, we now look at methods for finding the best fitting piecewise constant intensity function. This function can then, for instance, be used as a condensed representation of the original sequence of events.

Once the change points of the optimal piecewise constant intensity function are known, computing the intensity values giving the maximum-likelihood solution is easy: in each piece, the optimal intensity is the number of events divided by the length of the time period. This follows from the fact that the Poisson intensity expresses the average number of occurrences per time unit.

The task of finding the optimal change points is not trivial, however. An important note when trying to detect the change points is the following: the change points of the optimal piecewise constant function are always at occurrence times of the data sequence. (A change point between two events could be moved exactly to the occurrence time of the event with higher intensity value, thus yielding a better likelihood, Eq. (2)).

Hierarchical algorithm We now describe a greedy algorithm for finding change points, here called the hierarchical algorithm. The algorithm was introduced by Hawkins (1976); the presentation below is based on the version of Guralnik and Srivastava (1999).

The basic idea of the method is to split the time interval into two parts at the occurrence time giving the best gain in likelihood (see Fig. 3). A candidate set of potential change points is maintained; in each iteration round, the candidate giving the best gain is selected to be a new change point, the corresponding time interval is split and the candidate is removed from the candidate set. Two new candidates are added to the set by searching the best split points of the left- and right-side intervals of the new change point. Splitting is continued until the stopping criterion is met.

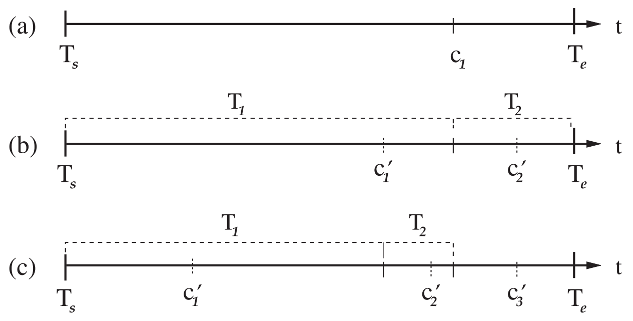

Fig. 3. Idea of the hierarchical algorithm; original period is split into two parts at the best split point (a); best splits (c'_1 and c'_2) of the new intervals are searched. The one giving better likelihood is chosen. It is removed from the candidate set (b); best splits of the new intervals are searched and added to the candidate set (c)

The theoretical time requirement of the hierarchical algorithm is $\mathcal{O}(nk)$ in the worst case and $\mathcal{O}(n\log(k))$ in the average case, where n is the number of events and k is the number of change points. The space needed is $\mathcal{O}(n)$.

A weakness of the algorithm is that the greedy heuristic of the algorithm does not allow removing change points once they have been selected. However, a change point of the optimal solution with m pieces is not necessarily a change point of the optimal solution with n pieces ($m < n$).

Dynamic programming algorithm We now introduce an algorithm based on the dynamic programming idea. The algorithm always finds the optimal solution for the change point detection problem.

Let us denote the data sequence by E and the occurrence times of the events of E by $t_i, 1 \leq i \leq n$. Thus, $T_s < t_1 < \ldots < t_n < T_e$, where T_s, and T_e are the start and end points of the observation period. Assume that the change points of the optimal piecewise constant intensity function with k pieces are known, and denote them by $\hat{c}_1, \ldots, \hat{c}_{k-1}$. Then $\hat{c}_1, ..., \hat{c}_{k-2}$ are the change points of the optimal piecewise constant function with $k-1$ pieces in the subperiod $[T_s, \hat{c}_{k-1}]$.

Let us further denote the last change point of the optimal k piece function with t_i as the end point of the time period by $\mathcal{C}(k, t_i)$. Assume now that change points of the optimal piecewise constant functions with k pieces in $[T_s, t_i]$ are known for all $t_i \in E \cup \{T_e\}$ with $i \leq k$. Now the question is how to find the change points of the optimal function with $k+1$ pieces.

Denote by $\mathcal{L}(k, t_i, t_j)$ the (maximum) likelihood (of the data E) for the optimal piecewise constant intensity function with the number of pieces k and the observation interval $[t_i, t_j]$. Then the maximum likelihood of the function with $k+1$ pieces is given by

$$\mathcal{L}(k+1, T_s, t_j) = \max_{T_s+k \leq i < j} (\mathcal{L}(k, T_s, t_i) + \mathcal{L}(1, t_i, t_j)). \tag{11}$$

The key idea of the dynamic programming algorithm is based on Eq. (11). The algorithm computes the optimal division into k pieces in the time interval $[T_s, t_i]$ for all $t_i \in E \cup \{T_e\}$ and uses these results according to Eq. (11) when computing the best divisions into $k+1$ pieces (see Fig. 4).

The change points of the function can be detected backward by applying the following equations:

$$\hat{c}_k = \mathcal{C}(k+1, T_e), \hat{c}_{k-1} = \mathcal{C}(k, \hat{c}_k), \hat{c}_{k-2} = \mathcal{C}(k-1, \hat{c}_{k-1}), \ldots .$$

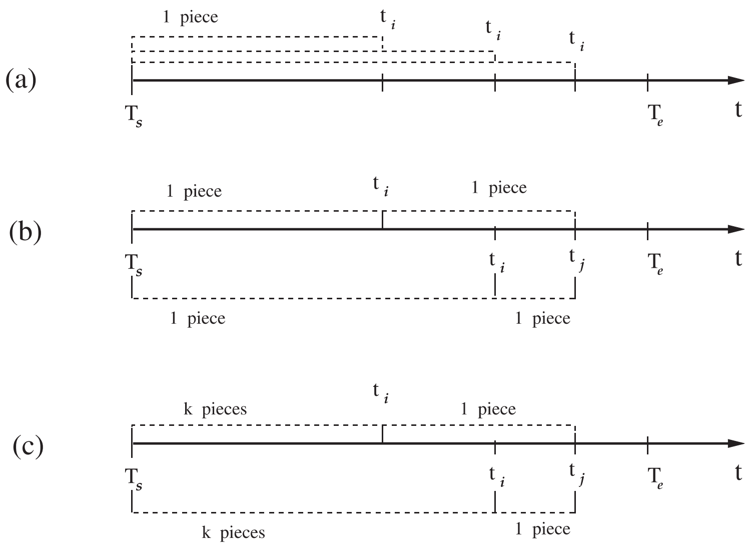

Fig. 4. Idea of the dynamic programming algorithm. Optimal one-piece intensity functions and corresponding likelihoods, $\mathcal{L}(1, T_s, t_i)$, are computed for all ranges $[T_s, t_i]$, $t_i \in E \cup \{T_e\}$ (a). Optimal two-piece functions are computed for all ranges $[T_s, t_j]$,$t_j \in E \cup T_e$: optimal change point maximizes $\mathcal{L}(1, T_s, t_i) + \mathcal{L}(1, t_i, t_j)$, $t_i < t_j$ (b). Similarly, optimal k-piece functions are used when computing optimal $k + 1$-piece functions: optimal change point maximizes $\mathcal{L}(k, T_s, t_i) + \mathcal{L}(1, t_i, t_j)$, $t_i < t_j$ (c)

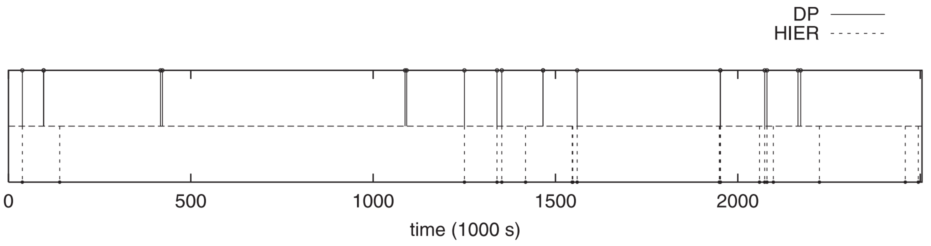

Fig. 5. Change points of the divisions up to 20 pieces generated by the dynamic programming (DP) and hierarchical (HIER) algorithms. The data set contains 46,662 events

Thus, the dynamic programming algorithm not only finds the optimal solution in the time range $[T_s, T_e]$ but also for all the subranges $[T_s, t_i]$, $t_i \in E \cup \{T_e\}$. The while and for loops of the algorithm lead to the time requirement $\mathcal{O}(n^2k)$. The space required is $\mathcal{O}(nk)$ due to storing values for all the subranges and division sizes.

As a preliminary experiment, we generated 20 piece intensity functions using both the hierarchical and dynamic programming algorithms on an alarm dataset of 46,662 events. The results showed considerable differences between the solutions: only 5 out of 19 change points were common to both algorithms and 13 of them were clearly different. The change points are presented in Fig. 5. The corresponding log-likelihood values with different numbers of pieces are given in Fig. 7. The execution times were less than ten seconds for the hierarchical algorithm and nearly 15 hours for the dynamic programming algorithm (Pentium 800 MHz). The results show that the hierarchical method can produce clearly nonoptimal solutions, while the basic dynamic programming algorithm is unacceptably slow.

Table 1. An example of selecting candidates using regular step size. The data set contains 46,662 events. The step size column indicates how many occurrences are skipped in selection. The log likelihoods (log L) and run times (min, s) refer to the trials described below

q	*step size (m)*	*nr of candidates*	*% of total*	*log L*	*run time*
11	2,048	22	0.047	−230,537	< 0.01
10	1,024	45	0.096	−228,232	< 0.01
9	512	91	0.20	−227,060	< 0.01
8	256	182	0.39	−225,425	0.01
7	128	364	0.78	−223,990	0.03
6	64	729	1.56	−223,293	0.12
5	32	1,458	3.12	−223,051	0.52
4	16	2,916	6.25	−222,892	3.33
3	8	5,832	12.5	−222,747	14.16
2	4	11,665	25.0	−222,643	57.37
1	2	23,331	50.0	−222,613	3 h 43 min
0	1	46,662	100.0	−222,583	14 h 35 min

6. Speeding up the dynamic programming algorithm

Because of infeasible run times of the dynamic programming algorithm on large data sets, we modified the algorithm by allowing only a subset of all the events to be change points.

In the first modified version, the selection process of the potential change points was carried out by picking up every mth occurrence of the event sequence to the candidate set. We first set m to value 2^q, where q is the greatest integer satisfying $2^q < n$, and n is the number of events in the sequence. Then we decreased q by 1, down to 0, in which case all the occurrence times belonged to the candidate set. In Table 1 an example of the candidate selection is shown on the same data set that was used in the preliminary trial above.

A natural way to proceed is to pick the change point candidates not by using predefined regular steps but rather by some heuristic methods for evaluating the value of an occurrence time as a potential change point.

For this purpose, we used a window around each occurrence time. Denote the mth occurrence time of an event by $t(m)$ and the size of the window by w, and let $t_l = t(m) - t(m - w)$ and $t_r = t(m + w) - t(m)$. Then we define

$$h(m) = \frac{t_r - t_l}{t_r + t_l} \tag{12}$$

as a heuristic function for the suitability of the mth point. The more the value of $h(m)$ differs from 0, the better candidate for a change point the mth point is. The observation period was divided into segments such that each segment contained the same number of occurrences. Then the occurrence time with the best value in each segment was selected to the candidate set. The segmentation was done to avoid wide gaps between the candidates. We also used two or three different window sizes in the same run and selected the best value to represent the occurrence time in comparison.

6.1. Experiments

We first ran the preliminary trial described above, now using the modified dynamic programming algorithm that selected candidate values with regular steps. When the candidate set of the potential change points included at least every 128th event of the whole data set, the algorithm resulted in higher likelihood values than the hierarchical algorithm. The execution time in the case was reduced from $14\frac{1}{2}$ hours to 3 seconds, as shown in Table 1. Results very near the optimal solution were obtained when at least every 8th event was included; the time needed was 14 minutes 16 seconds. The log-likelihood values of the trials are indicated by the points in Fig. 6; the values are also given in Table 1.

We then ran the experiment using the modified dynamic programming algorithm that selected candidates using the heuristic function of Eq. (12). The results shown in Fig. 6 indicate that the algorithm utilizing this heuristic produced clearly better results than the hierarchical algorithm even when less than 250 candidates (0.5%) were used as potential change points. With more than 500 candidates, the likelihoods were very near that of the optimal solution. The execution time was only a few seconds. When using the heuristic selection, the algorithm needed a remarkably smaller number of candidates to reach the same quality of solutions as in the case of the selection with regular steps. The results were also quite robust to the window size (Fig. 6).

In Fig. 7, the changes in log-likelihood values are shown when divisions up to 100 pieces were computed on alarm data sets of 46,662 and 15,704 events. The results indicate that the solutions of the hierarchical algorithm are quite far from the optimum, especially with relatively low numbers of splits. Instead, the divisions generated by the heuristic dynamic programming algorithm with different sizes of candidate sets are very close to the optimal solutions.

We also generated four artificial data sets and conducted similar trials on the data. We used a constant intensity in data set 1, piecewise constant nondecreasing intensity in data set 2, piecewise constant intensity in data set 3, and in the case of data set 4, we randomly changed the intensity at every occurrence time. The data sets and results are described in detail in the Appendix. Here we only illustrate the trials on the most complex data set (4). Figure 8 shows in a narrow subrange the actual intensity (dashed line) and the 500 piece function produced by the modified dynamic programming algorithm with 4,676 candidates.

6.2. Increasing the number of candidates

We now investigate more deeply the influence of increasing the number of candidates on improvement in the likelihood values. The trials were run on the generated data sets 2, 3 and 4 (see Appendix).

We started the trials with less than 200 candidates (0.2% of total number of events) and doubled the number of candidates until the maximum number of 25,000 events (25% of the total) was reached.

Figure 9 illustrates the results. We first consider the trials on the third data set. Figures in the middle left show that 780 candidates out of 100,000 (0.78%) are sufficient to yield solutions very close to the optimal. Even using 390 candidates (0.39%) gives nearly as good results. This holds for all the numbers of pieces tested—that is, from 20 to 100 pieces.

The results suggest that our method suits the kind of data in the third set very well; the actual intensity of the process is piecewise constant, and the changes in intensity values between adjacent pieces are relatively large. Thus, the applied heuristic is reliable and selects candidates that really are good change points. Hence, a small number of candidates is sufficient to approximate the optimal solutions from 20 to 100 pieces accurately.

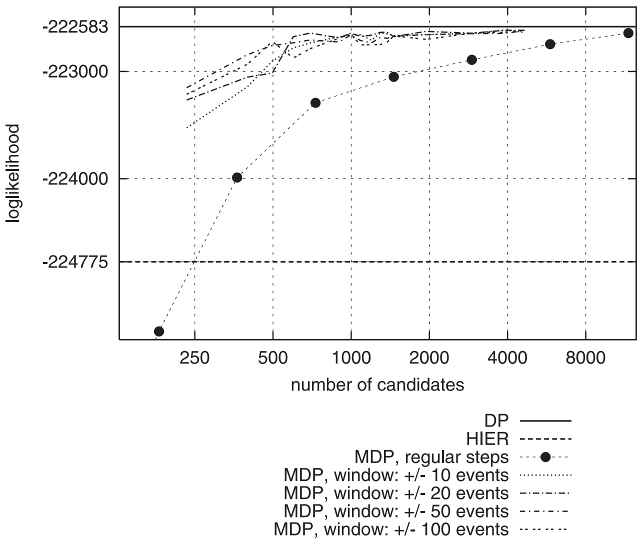

Fig. 6. Log-likelihood values of the 20 piece divisions generated by the modified dynamic programming algorithm (MDP) using regular steps or different window sizes of $+/-$ 10, 20, 50 and 100 events. The optimal solution of the original dynamic programming algorithm (DP) and the solution of the hierarchical algorithm (HIER) are also shown. The data set contains 46,662 alarms

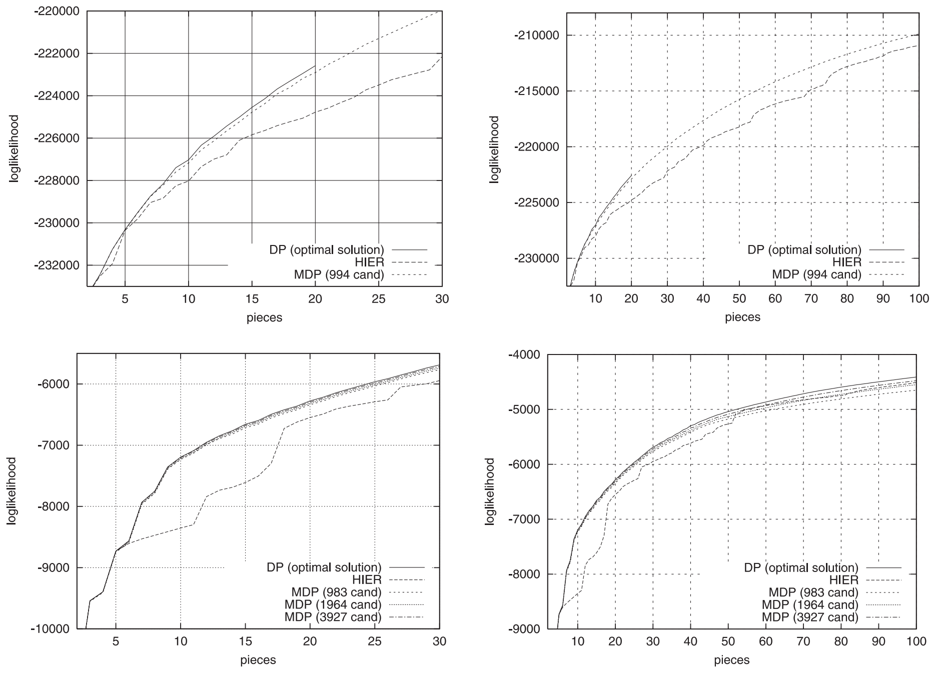

Fig. 7. Log likelihoods of the divisions up to 30 (left) and 100 pieces (right) generated by the original dynamic programming (DP), hierarchical (HIER) and the modified dynamic programming (MDP) algorithms on alarm datasets of 46,662 events (top, 994 candidates, window $+/-$ 10 events) and 15,683 events (bottom, 983, 1964 and 3927 candidates, window $+/-$ 50 events)

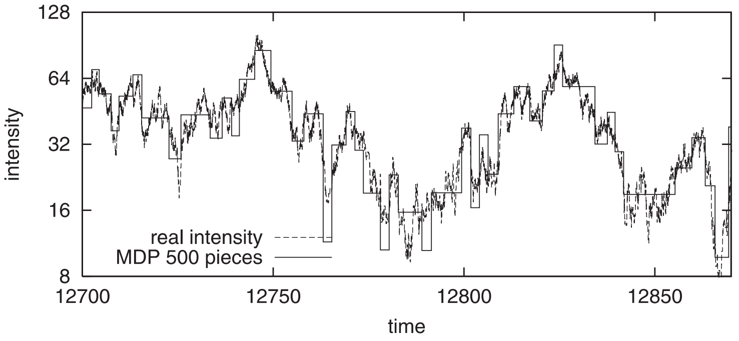

Fig. 8. A trial on generated data set 4. Real intensity (dashed line) and the solution of the modified dynamic programming algorithm (MDP) with 4,676 candidates for $k = 500$ (solid line). The total time range of the data set was [0, 13200]

The results on the second and fourth data sets are somewhat different. In the case of the second set, 1,560 candidates (1.56% of all events) are enough to produce nearly optimal solutions for small numbers of pieces. When it comes to solutions for 70 or more pieces, the results improve rather strongly even if 25,000 candidates (25%) are used instead of 12,500. It took about 40 seconds to compute the solutions with 1,560 candidates and about 2 hours when 12,500 candidates were included in the candidate set.

The fourth data set was generated by changing intensity after every event (for details, see Appendix). Thus, intuitively, one might expect that it would be a difficult task to find good approximations using piecewise constant functions. However, as Fig. 9, on the bottom-left, shows, in the case of a small number of pieces, the results improve only slightly when more than 877 candidates (0.88%) are available. When at least 3,500 candidates (3.5%) are employed, increasing candidates does not lead to strong improvements even if the solutions for 100, 150 or 200 pieces are considered. On the other hand, even 200 pieces is not many when approximating the intensity that changes 100,000 times. The right-side picture indicates that the log likelihoods remain relatively stable after about 50 candidates per piece are available. The 100-piece solution with 3,506 candidates is illustrated in Appendix in Fig. 15.

To reach nearly optimal solutions with a small number of splits, i.e. when 877 candidates were used, the computations took 30 seconds. To yield the level of 3,500 candidates, 10 minutes were needed. It should be noticed that the run times are influenced by the maximum number of pieces used. While the trials on the other data sets were run until 25,000 candidates were included in the candidate set, the trial on the fourth set was already stopped after 14,000 candidates. This was done because of the larger maximum number of pieces, which made the run times longer. The maximum number of pieces adopted in the case of the second set was 100. In the case of the fourth set, we used the maximum of 200 pieces.

Now we proceed to comparing the presented results with those produced by the MCMC methods.

6.3. Dynamic programming vs. MCMC methods

In Sect. 4, we applied the Bayesian modelling approach and MCMC methods to alarm sequence data. The specified model for the overall intensity was piecewise

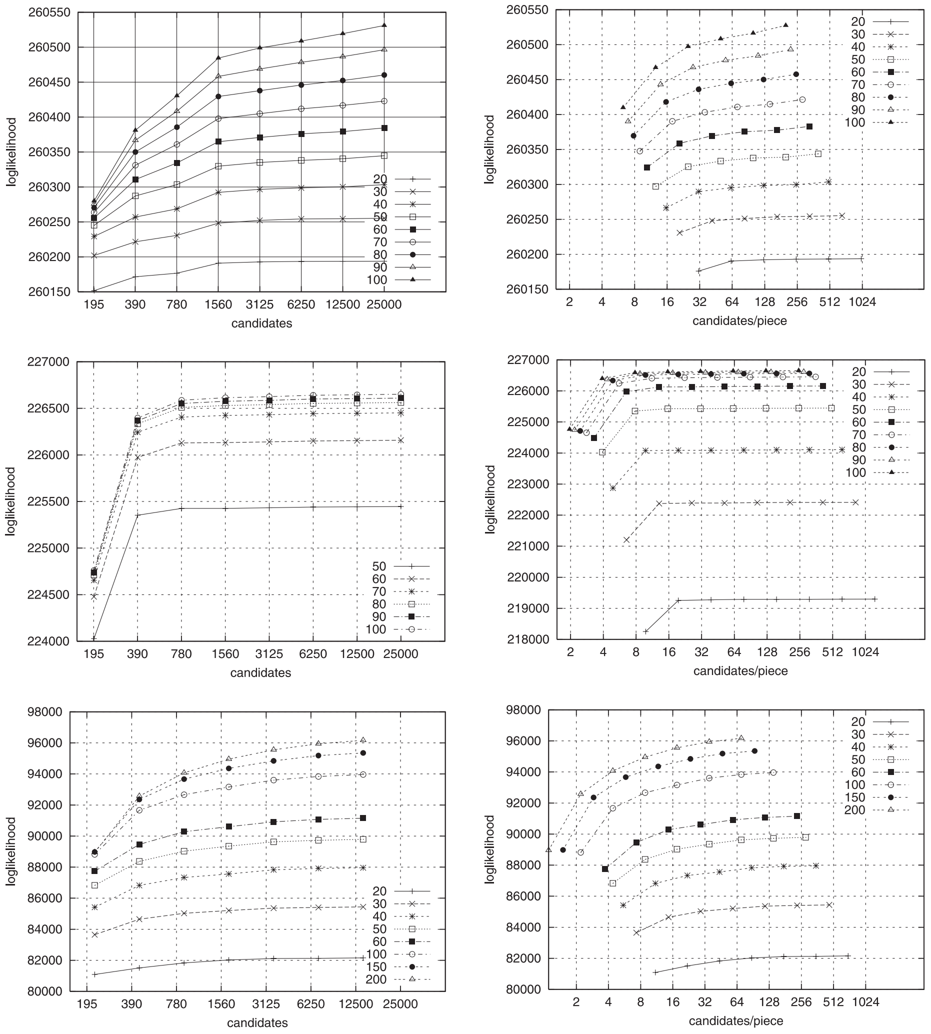

Fig. 9. Log likelihoods of k-piece solutions with candidate sets of increasing sizes; generated data sets 2 (top), 3 (middle), and 4 (bottom), the number of pieces $k \leq 100$ (sets 2 and 3), $k \leq 200$ (set 4). The log-likelihood values corresponding to the numbers of candidates are shown in the left panels. The points refer to solutions with a specific number of pieces: 20-, 30-, etc. up to 100-piece solutions. To better compare the solutions with different numbers of pieces, we also give the same log likelihoods as the function of the number of candidates per piece (right panels). Positive log likelihoods are yielded due to the high intensities used when generating the data sets

constant. Smooth posterior intensity curves could be produced due to the number of pieces being one of the model parameters.

Although not describing the same quantity, identically, the posterior average and the best fitting piecewise constant intensity with a predefined number of pieces reflect the same phenomenon. Thus, it is of interest to compare the results. Figure 10 illustrates the correspondence of the two approaches in trials on the alarm data set with 15,683 events. The time interval in both figures is from $5,000 \cdot 10^2$ to $6,000 \cdot 10^2$

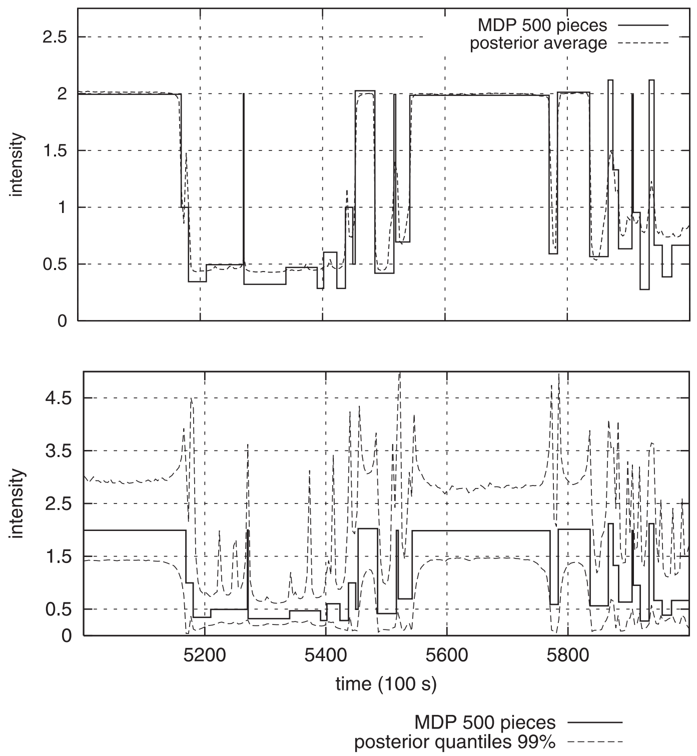

Fig. 10. Posterior average (top) and 99th percentiles (bottom) and 500 function produced by the modified dynamic programming (MDP) algorithm with 1,964 candidates. The data set contains 15,683 events

seconds, the whole range being $[0, 11753.5 \cdot 10^2]$ seconds. The posterior average produced by the MCMC methods is compared with a 500-piece intensity function, which was generated by the modified dynamic programming algorithm. The candidate set contained 1,964 possible change points, which is 12.5% of the total number of occurrences in the first data set. The average number of pieces during the MCMC simulation was 510.

The posterior average intensity and the piecewise constant intensity are close to each other. However, the steps of the function are slightly larger in the solution produced by dynamic programming than the changes of the posterior curve. This is due to the strategy of maximizing likelihood, which tends to exaggerate the changes. To put it more accurately, an optimal change point is always at some occurrence time. To maximize the likelihood, the occurrence time is selected to belong to the segment having higher intensity. Hence, the intensity level can still be raised a bit in the higher segment and, correspondingly, lowered in the other segment.

During the MCMC simulation, however, the distribution is detected instead of the best fitting solution. Thus, in part of the realizations, the occurrence time probably belongs to the lower segment. Naturally, as a smooth representation of intensity is produced from the different realizations, the changes are slightly more gradual.

The 99% posterior interval, shown at the bottom in Fig. 10, indicates the spread of the distributed intensity values. The maximum-likelihood solutions reside between the percentile curves.

7. Conclusion

We have considered finding piecewise constant intensity descriptions from event sequences. With MCMC techniques, piecewise constant functions can be used in a flexible way to represent continuous intensities. However, the MCMC methods needed for approximation of the posterior distribution are time consuming and the results may be sensitive to the choices of the simulation technical details, such as proposal-generating strategies and distributions.

Methods for finding single piecewise constant intensity functions maximizing the likelihood of the data were developed in Sect. 5. We showed that dynamic programming methods can be used to find the best fitting intensity function in $\mathcal{O}(n^2k)$ time, where n is the number of events in the sequence and k is the number of pieces of the intensity function. To speed up the dynamic programming algorithm, we applied heuristic methods. The approaches were based on using subsets of all the occurrence times as potential change points. We presented experimental results that indicated very remarkable speedups with minor loss in accuracy of results.

Appendix

Generated data sets

We generated data sets such that time intervals between events were distributed exponentially with known time-dependent intensities. The sequence of the occurrence times is then a realization of a Poisson process, the intensity of the process being known. The modified dynamic programming algorithm using the heuristic selection of candidates was tested on the artificial data sets to make observations about the performance of the algorithm in different known circumstances. We next describe the four datasets used in the experiments. Each set contained 100,000 events. The intensities of the data sets are presented in Fig. 11.

Set 1 The first set was generated using the constant intensity $\lambda = 2$. Thus, the intervals τ between events were exponentially distributed such that $\tau \sim \text{Exp}(2)$.

Set 2 Second, piecewise constant intensity with 100 pieces was used. The intensity at start was set to $\lambda(T_s = 0) = 1$. Then, 99 occurrence indices out of the total of 100,000 indices were selected randomly; denote the set of the selected indices by I. These indices determined the change points of the intensity function $C_{cp} = c_1, \dots, c_{99}$ such that $t_i \in C_{cp}$ if and only if $i \in I$. At each change point, the intensity was increased by 1. Thus, the intensity function was as follows:

$$\lambda(t) = \begin{cases} 1 & \text{if } 0 \le t < c_1, \\ 2 & \text{if } c_1 \le t < c_2, \\ \vdots & \vdots \\ 100 & \text{if } c_{99} \le t < T_e \\ 0 & \text{elsewhere.} \end{cases}$$

We generated the time intervals τ_i (which, of course, determined the occurrence times t_i) according to λ. Thus, at T_s, the time interval τ_1 was generated from Exp(1) and occurrence time was set to $t_1 = T_s + \tau_1$. Then, at t_1, the interval τ_2 was generated from Exp(1) if $1 \notin I$; otherwise, it was generated from Exp(2), and the value $t_1 + \tau_2$

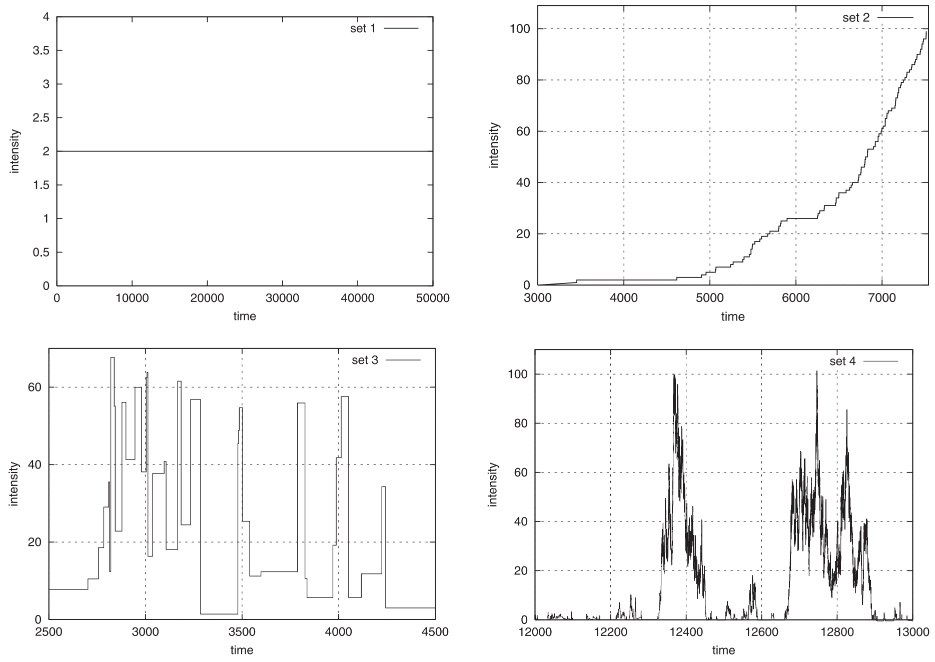

Fig. 11. Intensities of the generated data sets 1–4. In the cases of the third and fourth sets (bottom), subranges of the whole time periods are only shown because of the frequent changes of the intensities

was given to t_2, etc. Finally, $\tau_{100,001}$ was drawn from Exp(100) and T_e was set to value $t_{100,000} + \tau_{100,001}$.

Set 3 Like the second data set, the third set was generated using piecewise constant intensity with 100 pieces. Also, the change points were selected similarly, i.e. 99 out of all the occurrence indices were chosen randomly. Now, however, the intensities in each piece were drawn from the uniform distribution in range $[0.0001, 70]$, i.e., $\lambda_i \sim \mathrm{Unif}(0.0001, 70)$ for each $0 < i \leq 100$.

Set 4 Finally, we generated 100,000 events such that the intensity value was changed at each occurrence time. The intensities $\lambda_1, \ldots, \lambda_{100,001}$ were drawn from normal distribution such that $\lambda_i \sim \mathrm{Norm}(t_i, \sigma^2)$, where $t_i = t_1, \ldots, t_{100,000}$ are the generated occurrence times and σ^2 is a fixed constant.

Trials

In Fig. 12, 10- and 20-piece intensity functions produced by the modified dynamic programming algorithm on the first data set are shown together with the real intensity $\lambda = 2$. A candidate set of 1,494 candidates was used. The results are clear; the variance of the time intervals in the data is reflected by occasional peaks in the piecewise constant intensities. The improvements in results with the increasing numbers of pieces are illustrated in Fig. 13, where the log likelihoods of the different solutions are shown. In the case of set 1 (top left), the increase is linear with respect to the number of pieces but the changes are very small. This corresponds to

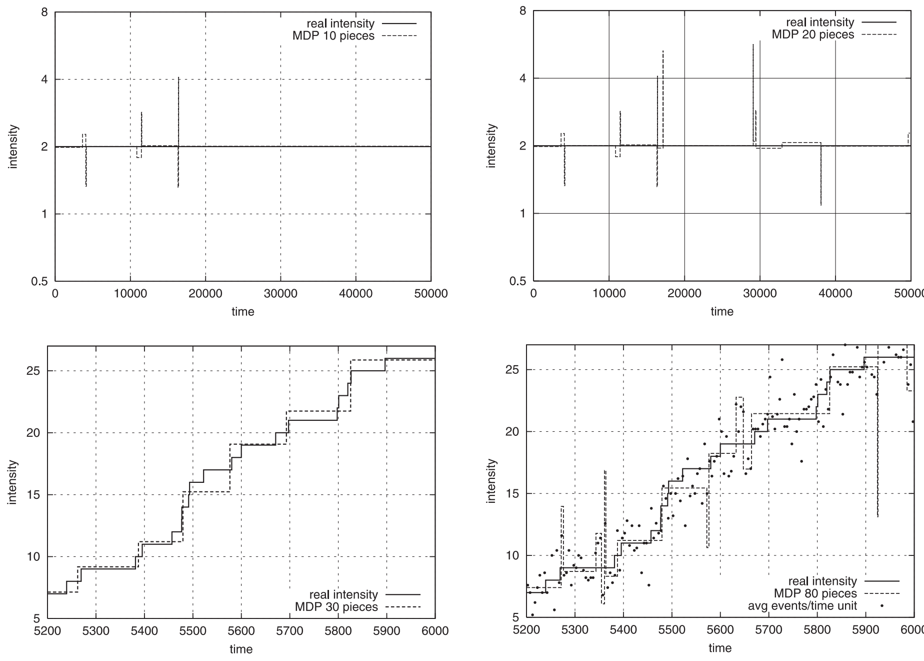

Fig. 12. Top: trials on first data set. Real intensity (solid line), and 10- (left) and 20- (right) piece approximations of the modified dynamic programming algorithm (MDP, dashed lines). Bottom: trials on second data set. Real intensity (solid line), and 30- (left) and 80- (right) piece approximations of the MDP algorithm (dashed lines). The points in the right panel show the average numbers of events in the data per time unit in successive periods of 5 time units

the intuition that approximating the constant intensity with a small number of splits gives accurate solutions.

Trials on the second data set were executed with a candidate set including 1,962 occurrence times. Figure 12 shows the 30- and 80-piece approximations of the dynamic programming algorithm as well as the real intensity. The 30-piece approximation is extremely good, taking several steps of the real intensity at one jump. The 80-piece solution does not make any progress; the clear direction has been replaced by overfitting to the variance of the data. The actual locations of the events are illustrated by the points in the figure. Each point indicates the average number of events in the data per time unit in a period of 5 time units. The successive points correspond to successive time periods. The periods are not allowed to overlap; thus, each event influences one point only.

The good quality of the solution with a relatively small number of pieces observed in Fig. 12 is also supported by Fig. 13; the first splits result in large changes in likelihoods, until slowing down after 10 pieces. Very slight improvements are yielded from then on.

The trials on the third data set are illustrated in Fig. 14. The real intensity is shown with 50- and 80-piece solutions of the modified dynamic programming algorithm using 2,441 candidates. Again, points are used to give an idea about the location of events in the data.

The piecewise constant approximations are very close to the actual intensity. The 50-piece function does not have enough parameters to detect all the substan-

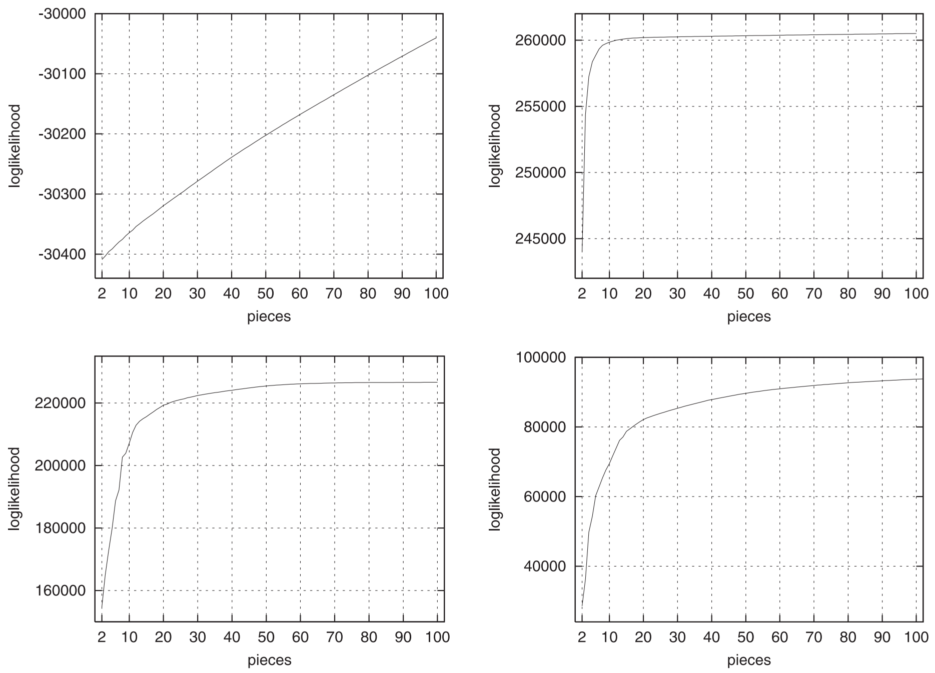

Fig. 13. Log likelihoods of the results in the trials on the generated data sets. Set 1 (top left), set 2 (top right), set 3 (bottom left), set 4 (bottom right). Except for the first set, positive log likelihoods are yielded due to the high intensities used when generating the sets

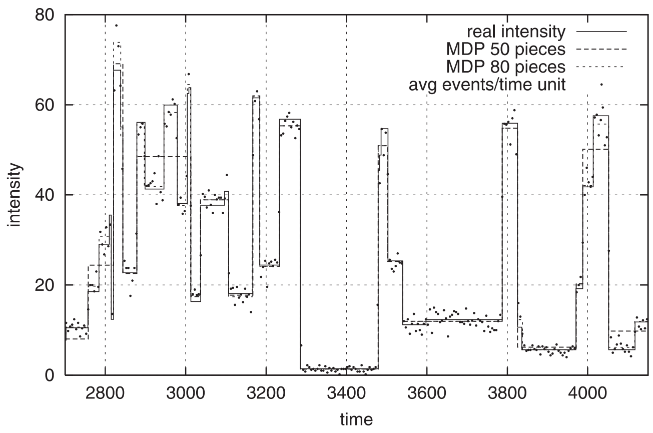

Fig. 14. The real intensity (solid line), 50- and 80-piece approximations (dashed lines) of the modified dynamic programming algorithm (MDP) and the average number of events per time unit in data (one point for each successive period of length of 5 time units)

tial changes in the data. However, the overall result is good. Figure 13, bottom left, confirms the interpretation: the functions with more than 80 pieces cannot make any considerable progress compared with the 80-piece solution. The 50-piece function yields nearly as high likelihood value.

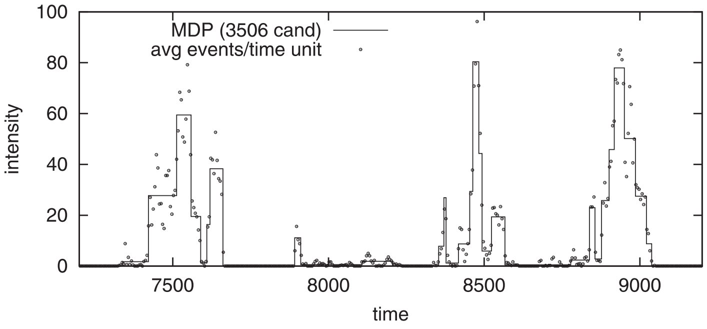

Fig. 15. A trial on dataset 4. Solution of the modified dynamic programming algorithm (MDP) with 3,506 candidates for $k = 100$ (dashed line). The points show the average numbers of events in the data per time unit in successive periods of 5 time units

When generating the fourth data set, the intensity was changed after every event. For this reason, divisions up to 200 pieces, instead of 100 pieces used above, were produced in the trials on the fourth data set.

The 100-piece solution with 3,506 candidates is illustrated in Fig. 15. The large variation of the real intensity is captured in broad outline by the function with 100 levels in quite a convincing way. A more detailed view is given in Fig. 8 in Sect. 6.1.

Not surprisingly, Fig. 13, bottom right, indicates a slower increase of the likelihoods in the case of the fourth data set compared with the values in the other trials.

References

Arjas E (1989) Survival models and Martingale dynamics. Scand J Stat 16:177–225

Bernardo J, Smith A (1994) Bayesian Theory. Wiley, Chichester

Brooks S, Giudici P (1999) Diagnosing convergence of reversible jump MCMC algorithms. In: Bernardo J, Berger J, Dawid A, Smith A (eds) Bayesian Statistics 6. Oxford University Press, Oxford, pp 733–742

Chib S, Greenberg E (1995) Understanding the Metropolis–Hastings algorithm. Am Stat 49:327–335

Eerola M, Mannila H, Salmenkivi M (1998) Frailty factors and time-dependent hazards in modeling ear infections in children using Bassist. In: Proceedings of XIII symposium on computational statistics (COMPSTAT '98). Physica-Verlag, Heidelberg, pp 287–292

Gamerman D (1997) Markov chain Monte Carlo. Stochastic simulation for Bayesian inference. Texts in Statistical Science. Chapman and Hall, London

Gelman A, Carlin J, Stein H, Rubin D (1995) Bayesian data analysis. Texts in Statistical Science. Chapman, New York

Green P (1995) Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. Biometrika 82:711–732

Guralnik V, Srivastava J (1999) Event detection from time series data. In: Proceedings of the Fifth ACM SIGKDD international conference on knowledge discovery and data mining (KDD-1999). San Diego, CA, USA, pp 33–42

Guttorp P (1995) Stochastic modeling of scientific data. Stochastic modeling series. Chapman and Hall, London

Hawkins D (1976) Point estimation of parameters of piecewise regression models. Journal Roy Stat Soc Ser C 25:51–57

Hätönen K, Klemettinen M, Mannila H et al (1996) TASA: Telecommunication alarm sequence analyzer, or "how to enjoy faults in your network." In: Proceedings of the 1996 IEEE network operations and management symposium (NOMS'96). Kyoto, Japan, pp 520–529

Mannila H, Salmenkivi M (2001) Finding simple intensity descriptions from event sequence data. In: Proceedings of the seventh ACM SIGKDD international conference on knowledge discovery and data mining (KDD-2001). San Francisco, CA, USA, pp 341–346

Tierney L (1994) Markov chains for exploring posterior distributions. An Stat 22:1701–1728.

Author biographies

Marko Salmenkivi (born 1967) is a postdoctoral researcher in the Basic Research Unit of Helsinki Institute of Information Technology, a joint research unit of University of Helsinki and Helsinki University of Technology. He received his Ph.D. in 2001. His research areas are data mining, data analysis and bioinformatics.

Heikki Mannila (born 1960) is the research director of the Basic Research Unit of Helsinki Institute of Information Technology, a joint research unit of University of Helsinki and Helsinki University of Technology. He is also a professor of computer science at Helsinki University of Technology. He received his Ph.D. in 1985 and has been a professor at the University of Helsinki, senior researcher in Microsoft Research, Redmond, research fellow at Nokia Research Center and a visiting researcher at Max Planck Institute for computer science and at Technical University of Vienna. His research areas are data mining, algorithms and databases. He is the author of two books and over 120 scientific publications. He is a member of the Finnish Academy of Science and Letters and editor-in-chief of the journal Data Mining and Knowledge Discovery. Dr. Mannila is the recipient of ACM SIGKDD Innovation Award 2003.

Correspondence and offprint requests to: Marko Salmenkivi, Helsinki Institute for Information Technology, Basic Research Unit, Department of Computer Science, P.O. Box 26, 00014 Helsinki, Finland. Email: Marko.Salmenkivi@cs.helsinki.fi