An Energy Based Scheme for Reconstruction of Piecewise Constant Signals observed in the Movement of Molecular Machines

Joachim Rosskopf†‡, Korbinian Paul-Yuan* ‡, Martin B. Plenio†, Jens Michaelis*

†Institute of Theoretical Physics Ulm University, *Institute of Biophysics Ulm University

Abstract
Analyzing the physical and chemical properties of single DNA based molecular machines such as polymerases and helicases often necessitates to track stepping motion on the length scale of base pairs. Although high resolution instruments have been developed that are capable of reaching that limit, individual steps are oftentimes hidden by experimental noise which complicates data processing. Here, we present an effective two-step algorithm which detects steps in a high bandwidth signal by minimizing an energy based model (Energy based step-finder, EBS). First, an efficient convex denoising scheme is applied which allows compression to tuples of amplitudes and plateau lengths. Second, a combinatorial optimization algorithm formulated on a graph is used to assign steps to the tuple data while accounting for prior information.

Performance of the algorithm was tested on poissonian stepping data simulated based on published kinetics data of RNA Polymerase II (Pol II). Comparison to existing step-finding methods shows that EBS is superior both in speed and precision. Moreover, the capability to detect backtracked intervals in experimental data of Pol II as well as to detect stepping behavior of the Phi29 DNA packaging motor is demonstrated.

submitted April, 2015

‡ The authors contributed equally to this article, joachim.rosskopf@uni-ulm.de, korbinian.paul@uni-ulm.de

Introduction
Single molecule measurements of molecular motors allow to study the motion of individual enzymes from comparably large steps of motor proteins like Myosin V (1) and Kinesin (2) to DNA based molecular machines which make steps on the scale of single nucleotides (3–5). Experimental techniques to study these systems range from single molecule fluorescence localization (6) to optical and magnetic tweezers (7). Most of these measurements represent the underlying dynamics as one-dimensional time series of positional changes. The enzymatic reactions which fuel this motion appear as stochastic events resulting
in step-like movements (8) covered by noise. Nowadays state of the art optical tweezers experiments allow to study the movement of enzymes with a resolution approaching single base pairs (3, 9). For example studies on the \( \varphi 29 \) bacteriophage ring ATPase (10–12) used the information from step detection data to propose a complete model of the mechanochemical cycle. However, oftentimes most analysis schemes rely on low pass smoothed data.

Indeed, the problem of finding steps is not only limited to studies of movement of enzymes but appears in a wide range of biomolecular experiments from fluorescence resonance energy transfer trajectories (13) to ion channels (14) and DNA microarrays (15) just to name a few. Consequently, there is a rich amount of signal processing techniques available to recover piece wise constant signals (PWCS) from noisy data. Due to the stochastic nature of enzymatic stepping the number of steps is often not known a priori. Therefore, different step finding algorithms have been developed (16, 17, 22, 23).

A prominent algorithm to determine steps from single molecule data is the so called t-test (18), which is based on the Student’s t-test. It locates a step when the hypothesis that two normally distributed random variables have the same mean is violated. The mean is calculated with respect to a certain time window which is an input parameter that can be eliminated by sweeping through various window sizes. The problem is that for steps on the scale of basepairs the stepsizes are oftentimes exceeded by experimental noise. Algorithms performing quite well in determining comparably large steps are compromised when applied to smaller steps on the order of basepairs. Hidden Markov models (HMM) have in the past been used to investigate the stepping of processive biomotors in situations with poor signal to noise ratio (SNR) (19, 20). In HMM the signal is modelled as a Markov process with transitions between discrete states covered by gaussian white noise. Transition probabilities of a Markov process are obtained from a maximum likelihood estimation and the step signal is reconstructed using the Viterbi algorithm (21). A HMM for processive molecular motor data requires many states to model the possible positions on the template. A solution to this is to improve computational performance by cutting the signal at a predefined amplitude and transforming positions to periodic coordinates to limit the number of states (20). HMMs proved to be excellent tools for pattern recognition in many fields. However, they are computationally demanding and rely on assumptions of the hidden stepping process and noise model.

Here, we establish an Energy Based Stepfinding algorithm (EBS) that detects steps in a two stage process. In a first stage we denoise the signal by solving the convex total variation denoising (TVDN) problem with a highly efficient and fast optimization algorithm. By characterising over - and underfitting regime the underlying steplike features can be discriminated from noise and thus our denoising method works essentially parameterless. We proceed by clustering these PWCS to increase detection performance of the actual steps. The clustering problem remains computationally expensive since it scales in general non-polynomially and traditional approaches like MCMC are still too time consuming (supplementary material). As a remedy we adopted principles of combinatorial optimization that are already in common use in the computer vision community (24). We tested EBS with simulated data that were created based on experimental data of RNA polymerase II (Pol II) movement. In particular we generated stepping data using kinetic parameters from recent experiments (25) and corrupted the signal with simulated noise typically encountered in optical tweezers. We compare the performance of EBS on the same simulated data to a t-test and the variable stepsize HMM.

The analysis reveals that our method performs faster and more accurate. We therefore applied the algorithm to detect steps in experimental data of the bacteriophage \( \varphi 29 \) packaging motor and to determine pauses of Pol II in high resolution optical tweezers experiments.
Figure 1: Flowchart of the two stage process for finding steps in noisy stepping data. The input data is first denoised via solving the convex TVDN problem. This process requires no intervention, as the regularization parameter $\lambda$ is determined automatically. This results in a lower dimensional, compressed representation of tuples (amplitude, length). This discrete data is then handed to a Graph Cut algorithm which solves a combinatorial optimization problem on a graph. The Graph Cut allows further customization by the use of regularization parameters $\rho_i$ and a pre-defined level set.

Methods

Energy Based Model

Starting from a large and noisy trajectory of motor protein movement we use an energy based step detection (EBS). EBS consists of two parts: TVDN and combinatorial clustering (CC). Both in TVDN as well as in CC we assign a scalar energy to each possible configuration of steps. The value of the energy is defined by a cost or energy loss function $E(\cdot) : \mathbb{I} \rightarrow \mathbb{R}$, where $\mathbb{I}$ is the high dimensional configuration space. The loss function incorporates some prior knowledge as well as the measured data. To reveal the steps produced by the underlying biological system hidden in noise, one has to identify piecewise constant parts in the data set. This is done by taking the $N$-element input data and creating an $N$-element output variable set. Therefore one needs to penalize variations within neighboring variables in the signal. In contrast one needs to increase the energy if the free variables deviate too much from the measured signal. This is reflected in the loss function

$$E(x, y) = \sum_{i=1}^{N} D(|x_i - y_i|) + \sum_{i=1}^{N} S(|x_i - x_{i-1}|)$$

where $y$ and $x$ are the $N$-element input data and output variable vectors respectively. The energy loss function is the conceptual baseline of our approach. It consists of terms where variables interact with the input data $D(\cdot)$, as well as nearest neighbor interaction between two adjacent variables $S(\cdot)$. Unfortunately depending on the actual shape of these terms the optimization problem can get prohibitively computationally expensive \cite{24}. Therefore, one of the design goals of EBS was to work efficiently for large data sets on commodity hardware. Therefore we chose an approach which denoises and smoothens the signal by optimizing a simple convex energy loss function. The result of this step is a lower dimensional set of tuples, each consisting of amplitudes and length of the denoised steps. Afterwards we use this lower dimensional set of tuples and optimize a more sophisticated energy functional, which is defined on a discrete level set and incorporates a step height prior. A flowchart of this two stage process is shown in figure\cite{1}.
Total Variation Denoising

In the first stage of EBS, we separate noise from the actual stepping signal. This stage works on the full and noisy 1D input data set \( y = y_1, \ldots, y_N \in \mathbb{R}^N \), which can be quite large (\( N > 10^7 \) datapoints). To denoise we use an energy loss function known as Total Variation Denoising (TVDN) problem (26),

\[
x^\star = \underset{x \in \mathbb{R}^N}{\text{argmin}} \quad q(x, y) + p(x)
\]

\[
= \underset{x \in \mathbb{R}^N}{\text{argmin}} \frac{1}{2} \sum_{i=1}^{N} |x_i - y_i|^2 + \lambda \sum_{i=1}^{N-1} |x_i - x_{i+1}|
\]

where the optimal solution \( x^\star \) represents the denoised signal. The \( \{y_i\} \) and \( \{x_i\} \) are the \( i \)-th entry of the time-discrete input and solution vector, respectively. This optimal solution is a tradeoff between prior knowledge that the enzymatic steps yield piecewise constant signals, which is introduced by \( p(x) = \lambda \sum |x_i - x_{i+1}| \), a function which penalizes introducing steps. On the other hand the term \( q(x) = \frac{1}{2} \sum |x_i - y_i|^2 \) penalizes deviations of the resulting solution from the input signal. The regularization parameter \( \lambda \) is important for the solution \( x^\star \) and controls the relative weight of the two terms.

The energy function in Eq. (2) is strictly convex, which means regardless of the input data \( y \) there exists one unique solution \( x^\star \) (see e.g. (27)). Typically TVDN is addressed by fixed-point methods (28). These methods reach the minimal theoretically possible algorithmic complexity (29). A different kind of approach (30) uses the local nature of the total variation denoising filter and provides a very fast, memory efficient, non-iterative way to solve Eq. (2). Although the theoretical complexity of this algorithm is worse compared to fixed-point methods it actually achieves competitive or even faster results on signals which exhibit piecewise constant characteristics. The implementation used here can easily handle millions of data points in a few milliseconds. The algorithm scans forward through the signal. During this sweep it tries to prolongate segments of the signal with the same amplitude, until optimality conditions derived from the TVDN problem are violated. If this happens the method backtracks to a position where a new step can be introduced, re-validates the current segment until this position and starts a new segment (supplementary material).

An open problem in the context of TVDN for step detection is how to choose the regularization parameter \( \lambda \) such that as few as possible true steps are lost (false negatives) but still the data is not overfitted (false positives). We propose a heuristic method to choose an optimal value for \( \lambda \), termed \( \lambda_h \), automatically. To motivate these heuristics we have a closer look to the two limits naturally imposed to \( \lambda \). For \( \lambda_{\text{min}} = 0 \) the TVDN algorithm perfectly reproduces the input signal such that \( x^\star = y \). The upper bound of sensible values is marked by \( \lambda = \lambda_{\text{max}} \). Above this threshold the solution of Eq. (2) is constant \( x_i^\star = \text{const} \) for all \( i \). The value of \( \lambda_{\text{max}} \) can be derived analytically from the underlying Fenchel-Rockafellar (31) problem (supplementary materials).

There exists a transition in TVDN while varying the regularization parameter from a stable minimization of total variation into the over fitting regime. Thus, by lowering \( \lambda \) from \( \lambda_{\text{max}} \) to \( \lambda_{\text{min}} \) one observes a sudden increase of steps (figure 2). The sudden increase in the number of steps marks the point when the TVDN model breaks down and the solution starts to fit noise. The breakdown also persists while varying sampling frequency or rate of steps as well as signal to noise ratio (supplementary materials). To choose \( \lambda_h \) we use a line-search algorithm which detects the sudden increase in the slope of the number of steps in the resulting signal and uses a value slightly larger for the regularization parameter \( \lambda_h \) (supplementary materials). The sole input to the algorithm is the analytically determined value of \( \lambda_{\text{max}} \). Therefore the \( \lambda_h \)-heuristic provides us with a stable means to choose the TVDN regularization parameter.
After successful TVDN of the signal $x^* \in \mathbb{R}^N$ consists of $M$ steps. A step is characterized by a discontinuity between variables of constant amplitude $a$. In a typical signal $M \ll N$. Therefore, it is beneficial to represent the signal not in the basis of indexed amplitudes $x_i^*$ anymore, but instead use tuples of amplitude $a_j$ and length $w_j$ of the signal $(a, w)_j, j \in 1, ..., M$. By this change of representation the number of elements of the data set is typically reduced from several millions to a few thousand. This increases computational efficiency due to the fact, that the complexity of following algorithms depends on the number of elements in the data set. Therefore, a compressed signal consisting of tuples opens up the possibility to apply sophisticated step-detection algorithms on the data. The tradeoff of using the tuple representation is, that in the tuple basis the optimization problem can not be formulated on $\mathbb{R}^N$. Instead it has to incorporate the length $w$ of a step and thus we change from a discrete time representation in $\mathbb{R}^N$ to a tuple representation. The problem can now be cast as a Markov Random Fields (MRF) (32) and can be tackled by a combinatorial optimization method as will be presented in the following section.
**Combinatorial Clustering**

As stated above, the input to the second stage of EBS are tuples of amplitudes and corresponding lengths \((a, w)\) of the compressed signal. Often molecular motors move in discrete steps with known step-size. These steps define a level set which can be used as additional information providing a reasonable choice of accessible template positions. The task of this stage is to cluster the tupels on this predefined set of levels, i.e. positions of the motor, such that an energy loss function is minimized. This means that a combinatorial version of an energy loss function similar to Eq. (1) has to be optimized. The length of a plateau plays the role of a weighting factor changing the contribution of a single tuple or a pair of tuples to the total energy. With this modifications a general energy loss function takes the form,

\[
E(\xi|(a, w)) = \sum_{i \in V} Q_i(\xi_i|a_i, w_i) + \sum_{(i,j) \in E} P_{i,i+1}(\xi_i, \xi_{i+1}|a_i, a_{i+1}, w_i, w_{i+1})
\]

(3)

where the possible \(\xi_i\) are taken from a predefined set of levels \(L\) which best fit prior knowledge of the problem. The value of the data term \(Q_i(\cdot)\) depends on deviations of \(\xi_i\) from the input. The pairwise term \(P_{i,j}(\cdot, \cdot)\) encodes interaction potentials between neighboring plateaus. Essentially the problem means to cluster the tupels \((a, w)\) to predefined levels, such that the joint configuration \(\xi\) minimizes \(E(\xi)\).

An elegant solution can be found by mapping the problem onto a graph \(G = (V, E)\), consisting of vertices \(V\) and edges \(E\). For the simple binary case, where the tupels have to be assigned to only two levels, termed source \(s\) and terminal \(t\), both of these levels as well as all tupels represent vertices \(V\). \(E\) denotes the set of edges connecting the vertices (figure 3a) and each edge carries a capacity \(c_i \geq 0\) (figure 3b). Therefore there are two types of edges, those connecting neighboring tupels and those edges connecting tupels to levels. The capacities of the former are encoded in the pairwise term \(P_{i,j}(\cdot, \cdot)\) and the latter are represented by the data term \(Q_i(\cdot)\). In the process of assigning a level \(\xi_i\) to a tupel the Graph Cut algorithm solves the following binary decision problem: Is the assignment to level \(t\) more favorable than assignment to level \(s\) in terms of the energy loss function? In the graphical representation this assignment is represented by a cut through edges of neighboring tupels and edges between tupels and the \(s\) and \(t\) level (figure 3c).

Due to the well known min cut/max flow theorem of graph theory the optimal energy coincides with the smallest sum of capacities of the edges one has to cut from the graph to disconnect \(s\) from \(t\) (36). The cut splits the graph \(G\) in two subgraphs: The part \(S\) which is connected to the vertex \(s\) and the part \(T\) which is connected to \(t\). There exist effective algorithms which solve this problems in polynomial time for a certain set of useful energy functions (34, 35, 37).

To make min cut/max flow useable for the above described assignment of multiple different levels \(\xi\), it has to be embedded into an outer procedure. For this we use the \(\alpha\)-expansion algorithm (24, 38). It finds provably good approximate solutions by iteratively solving Graph Cut problems on graphs representing the binary decision if to alter the previous assigned level configuration or not. For a multi level problem new levels are added successively in a random order. That means, once the graph has been optimized for \(i\) levels and the new \(i + 1\)th level is introduced, \(t\) corresponds to the assignment to the predefined level set and \(s\) to the new level. Again capacities for all edges are computed. With the new graph cut, vertices in the subgraph \(S\) get assigned their new level, the other vertices connected to \(T\) keep their previously assigned level. After having introduced all levels, In order to minimize the energy even more, the assignment can be optimized by iteratively reintroducing the complete level set. This iteration stops when the overall energy is not decreasing anymore (supplementary materials).

To determine the relative importance of the terms \(Q\) and \(P\) in equation (3), we introduce the regularization parameters \(\rho_D\), \(\rho_S\) and \(\rho_P\). The data terms \(Q_i\) penalizes deviations of the proposed level amplitude
Figure 3: Cartoon representation of the graph cut algorithm. (a) the initial graph structure we use to model the step signal. The nodes \( i \in \{1 \ldots M\} \) represent the variables \( \xi_i \). (b) in a second step, energies are mapped to capacities of edges. (c) the Boykov-Kolmogorov Graph Cut algorithm (35) finds the max flow and cuts the graph into two subgraphs \( G = S \cup T \) where \( S \) is the part connected to \( s \) and \( T \) is the remaining part connected to \( t \).

\[ \xi_i \] to original tuple amplitude \( a_i \) at the vertex \( v_i \)

\[ Q_i = \frac{(1 + w_i)}{\rho_D} |\xi_i - a_i| \quad (4) \]

where \( \rho_D \) is a regularization parameter determining the importance of the data term, and \( w_i \) the weight of the current tuple.

As mentioned above, typical Graph Cut algorithms are just applicable to a limited subset of MRFs. The mapping of energies to edge capacities and graph construction of our method is constrained to submodular energy functionals (39). Submodular energies are discrete functions whose pairwise terms satisfy

\[ P_{ij}(\beta, \gamma) + P_{ij}(\alpha, \alpha) \leq P_{ij}(\beta, \alpha) + P_{ij}(\alpha, \gamma) \quad (5) \]

for arbitrary levels \( \alpha, \beta, \gamma \in \mathcal{L} \).

For the case of an equidistant level set \( P_{ij} \) consists of two different terms. The first and simpler pairwise energy uses a Potts Model (41) to increase the energy whenever two assigned levels \( \xi_i \) and \( \xi_{i+1} \) differ

\[ P_{\text{potts}}_{i,i+1} = 1 + \frac{w_i + w_{i+1}}{\rho_S}(1 - \delta(\xi_i, \xi_{i+1})) \quad (6) \]

where \( \delta(x, y) = 1 \) if \( x = y \) and \( \delta(x, y) = 0 \) else. Here \( \rho_S \) is the regularization parameter determining how large the contribution to the energy for differing adjacent levels will be. The Potts model satisfies Eq. (5) (39). The second more sophisticated contribution to the pairwise term in Eq. (4) favors level changes of specific size between adjacent sites. The potential of such a term is depicted in figure 4. The complete pairwise term \( P_{i,j} \) thus, includes prior information about step heights and is given by

\[ P_{i,i+1} = P_{\text{potts}}_{i,i+1} + \frac{1}{\rho_P} (1 - \delta(\xi_i, \xi_{i+1}))(1 - \delta(|\xi_i - \xi_{i+1}|, \sigma)) \quad (7) \]

with an expected average step height \( \sigma \) determined by the underlying process. The depth of the jump height prior potential is given by \( 1/\rho_P \). In contrary to Eq. (6) this term does not depend on the weights.
of the adjacent sites in order to not suppress small steps. Note that Eq. (7) does not strictly fulfill the submodularity condition, Eq. (5). However, recent advances make it possible to extend the class of energy loss functions to non-submodular energies (39, 42) but complicate the construction of the graph representation. We use the simple extension described in the latter reference to circumvent a submodularity violation (supplementary material).

**Simulation Method and Parameters for Analysis**

In order to quantify positional and temporal accuracy of the steps detected by EBS we use simulated data of noisy steps which are generated in a two stage process. In a first stage we generate a PWCS according to a simplified Pol II stepping model where a step is the product of an enzymatic process with a certain net rate. This model contains an elongation state with forward steps of 1bp in size generated using an effective stepping rate $k_{\text{elong}}$. We also account for backtracked states which can be entered by a backward step of 1bp (45, 46) with a rate $k_{b,1}$. In a backtracked state Pol II can step forward or backward by 1bp with the rates $k_f$ or $k_b$ respectively. Thus the model reproduces the ability to pause but does not accurately reflect the temporal order of translocation and other enzymatic processes (Supplementary Material).

Secondly, we simulate experimental noise including effects of confined brownian motion of trapped micro spheres. To accurately reflect the experiment, we take into account changes in tether length and tether stiffness due to the motion of the enzyme. We apply a harmonic description of the trapping potentials and assume that the DNA linker can be described by a spring constant $k_{\text{DNA}}$ determined by the worm like chain model (Supplementary Material).

We simulated a slow, an intermediate and a fast scenario with elongation rates of $k_{\text{elong}} = 4.1$, $k_{\text{long}} = 9.1Hz$ and $k_{\text{long}} = 25.1Hz$, respectively. For the slow scenario we generated $N = 2.5 \cdot 10^5$ data points with time increments corresponding to a $5kHz$ sampling frequency. Simulated signals of the intermediate scenario consist of $N = 10^5$ data points with $2kHz$ sampling frequency. The computed standard deviation in both scenarios is $5.5bp$ at the given sampling frequency. For the fast scenario we chose $N = 5 \cdot 10^4$ data points and $1kHz$ sampling rate. Moreover in the fast scenario we use higher noise amplitudes with a computed standard deviation of $10.0bp$ at the $1kHz$ sampling frequency.

For EBS analysis of the noisy steps we have to choose the parameters $\rho_D$, $\rho_S$ and $\rho_P$ as well as the level spacing for CC. Since our task is to optimize Eq. (5) we are only interested in relativ values of the terms. Thus, we can arbitrarily set $\rho_D = 0.01$. $\rho_S$ and $\rho_P$ are parameters that have to be defined by the user. The smoothing parameter $1/\rho_S$ has to be large enough to cluster small steps but small enough not to miss simulated steps. The value of $\rho_P$ can not be arbitrarily small since the corresponding energy term does not strictly fulfill submodularity. To this end simulated data can be used to optimize parameters such that as many steps as possible are recovered but only few false positives are created. The choice $\rho_S = 0.5$, $\rho_P = 0.01$.
\( \rho_P = 0.02 \) is optimal in this sense for the simulated signals (supplementary material). Furthermore, we use a level grid spacing of \( 1bp \) that is equal to the simulated step-size.

In order to compare different step-finding algorithms, we need to define a criterion when a detected step is correct. A detected step is classified correct whenever its temporal position lies with ±\( \Delta_{\text{window}} \) of the simulated step. We choose the window size such that \( \Delta_{\text{window}} = 1/(5 \cdot k_{\text{elong}}) \). This allows for a small temporal shift of the detected steps with respect to a simulated step. The window is small enough to avoid relevant influence on statistical properties of the detected step signal but large enough to make the step detection robust against numerical error.

**Detecting backtracked regions**

The separation between backtracked states and elongation state is important to discriminate between different stepping mechanisms. For the detected steps dwelltimes are assigned to the set of backtracked states when they lead to a backward step. A backtracked pausing interval ends when a forward step brings Pol II back to the original elongation state. At high noise and for fast steps we do not expect that our method will perfectly find all backtracking events present. For example, short backtracks can be omitted resulting in a long dwell time between two forward transitions in the detected steps. We correct for these by a statistical hypothesis testing of dwell times, assuming that forward stepping follows an exponential waiting time distribution (supplementary material). The corresponding mean dwell-time can be estimated from the dwelltime histogram of forward steps. Thus, dwelltimes which violate this hypothesis are also considered as backtracked intervals.

A state of the art method for this separation is a Savitzky Golay filtered velocity threshold pause detection (SGVT). SGVT finds backtracked regions in Savitzky-Golay SG smoothed data from histograms of instantaneous velocities \(^{(52)}\). These histograms show a pause-peak around zero velocity and an elongation-peak. One typically defines a velocity threshold by computing the mean plus one (or two) standard deviation(s) of the pause-peak which is used to characterize paused regions in transcription data. A sensible choice for typical Pol II experiments of the SG filter parameter is to use third order polynomials and a frame size of 2.5s \(^{(4)}\).
Figure 5: EBS algorithm correctly detects steps in presence of high noise. (a) noise reduction after application of TVDN. (b) step detection using combinatorial clustering. Shown is a zoomed in interval of the simulated noisy data (grey points), simulated steps (blue), denoised signal from TVDN (magenta) and detected steps after application of combinatorial clustering by Graph Cut (red).

Results & Discussion

Example of TVDN and combinatorial clustering

We developed the EBS algorithm to determine steps in the trajectories of molecular motors and tested this algorithm on simulated data of Pol II stepping using published rates (Methods and supplementary material). We first simulated data using the intermediate scenario (Methods). We simulated a trajectory of 50 s (i.e. 10^5 data points) resulting in 291 steps. At the 2kHz sampling rate noise amplitudes are much larger than the 1bp steps of the simulated step signal (figure 5). TVDN efficiently removes noise and produces a set of 587 plateaus (figure 5a). The TVDN signal approximates the simulated step signal, but often decomposes a simulated step into several smaller steps. Note, that the TVDN signal reliably follows the simulated data and some small deviations are caused by the statistical nature of the noise rather than by errors of the algorithm.

In the second step of EBS we use CC to cluster the denoised data to predefined levels of integer multiples of the known step size of 1bp. This results in a total of 176 found steps and thus surplus TVDN steps are removed (figure 5b). While the EBS algorithm visually traces the simulated data very well, it will not be able to recover all steps and thus optimization of the underlying parameters on one hand and comparison of its performance against competing algorithms is required.
Effect of including prior information in combinatorial clustering

EBS provides two possibilities to include prior information about step sizes. First, one can adjust the prior terms in Eq.(6) and Eq.(7), and secondly one can enforce a certain step size as a spacing of the level grid. We tested the impact of the prior terms on simulated data using the intermediate scenario and performed CC with different prior potential strength. We compared a 1bp-spacing of the level grid and a 1/4bp-spacing as an example of a level grid that subdivides the simulated step size of 1bp. Moreover, we increased the prior regularization parameters $1/\rho_S$ and $1/\rho_P$ starting from $1/\rho_S = 0$, while the jump-height prior parameter was varied simultaneously such that $\rho_S/\rho_P$ remained constant (supplementary material). The best result regarding the absolute number of correct steps and a small number of falsely detected steps which lie outside a certain time window around an actual step (Methods) was found for $1/\rho_S = 4, 1/\rho_P = 50$ and thus $\rho_S/\rho_P = 12.5$ (supplementary figure [18a]). This increases the number of correct steps from 34% of the steps found at vanishing prior potential to 48% for the 1/4bp level grid analysis (supplementary material). By increasing the spacing to 1bp the detection precision can be improved even further to 54% correct steps of the found steps.

The prior potential strength of the smoothing term $1/\rho_S$ can not be arbitrarily large since it would remove steps in favor of fewer large steps and thus reduces the number of correctly recovered simulated steps. The strength of the jump height prior $1/\rho_P$ is also limited since this term is not strictly submodular and the approximation proposed in (42) that we have implemented only works if a negligible number of non-submodular energies arise and these terms do not dominate the total energy.

Comparison to other algorithms

In the following we compare the performance of the EBS algorithm to commonly used algorithms for detecting steps in the trajectory of motor proteins namely, a t-test (18) and the variable stepsize hidden markov model (HMM) (20). In order to quantitatively compare the results of the EBS algorithm to the t-test and variable step size HMM, we chose the slow, intermediate and fast scenarios outlined in the simulations subsection. To get statistically meaningful results we simulated 20 time traces for each scenario. Input parameters of the step-detection algorithms were adjusted for each simulation scenario (supplementary material). After the analysis the detected steps were compared to the simulated input steps (figure [5]). With properly adjusted parameters none of the algorithms tends to overfit the highly noisy data since all algorithms find fewer than the simulated steps (figure [6a]). For the slow scenario, a majority of the steps could be found. EBS finds 76% of the simulated steps. 52% of the simulated steps are found in the correct time window (correct steps, methods). The other methods found comparatively many but still slightly less correct steps (t-test: 50%, HMM: 44%). In the intermediate and fast scenario none of the algorithms were able to find the majority of the simulated steps. Thus, the simulation setting of the intermediate and fast scenario is challenging for all step detectors under consideration. In particular for the fast scenario only around half of the steps found by EBS are correct (figure [6b]). While the t-test finds a similar number of steps than EBS in the fast scenario, only 1/5 of these are correct steps, showing that many of these steps are false positives. The HMM performs similarly poor when using the fast scenario. The low performance in found steps and correct steps means that step detection accuracy is inappropriate for the lower bandwidth signal of the fast scenario and directly extracting information of the underlying enzymatic cycles of elongation from dwell time fluctuations would be error prone.
Figure 6: Performance of step detection algorithms with respect to slow scenario (blue bars), intermediate scenario (green bars) and fast scenario (dark red bars). (6a) shows in percent of the total number of simulated steps the number of detected steps. (6b) percentage of correct steps among the simulated steps. Errorbars are SEM. (6c) shows average step size histograms with 1bp binning of the detected steps of the fast scenario for t-test, HMM and EBS (from upper to lower histogram).

Furthermore, we compare the step size distributions of the detected steps for the fast scenario (figure 6c). Due to the lower bandwidth and faster stepping rates the algorithms do not reproduce the simulated step size well. In general the algorithms tend to fuse 1bp steps to steps of larger size which explains the smaller number of found steps compared to number of simulated steps. However, the step size distribution obtained by EBS resembles the expected distribution most closely, while the t-test is showing a broad distributions of step sizes and the HMM detects mostly steps of size larger than 2bp. For the slow scenario step-size histograms show a majority of the expected 1bp steps for all algorithms considered here (supplementary figure 16).

Moreover, we also compared the run-times for all three algorithms, since short run-times improve usability and allows to conduct step-finding analysis on higher bandwidth data. EBS outperformed the HMM significantly and was also faster than the t-test. Table (1) summarizes computational speed and memory consumption of the algorithms for test runs with 100s of temporal length and rate constants according to the intermediate scenario. We simulated data containing ~ 900 simulated steps and successively increased the number of data points by increasing the sampling frequency. For each sampling frequency the standard deviation of noise was constant. EBS processed $2 \cdot 10^7$ data points, simulated with 200kHz sampling frequency in 4min. The other algorithms had comparably long run times and we restricted computation times to ~ 50min and memory consumption to a limit of 1.8GB. EBS can process much more data points at comparably short run time and is essentially limited only by the size of available memory. The high bandwidth signals can be compressed very well by TVDN to a few thousand plateaus, which yields shorter run times for the subsequent CC. In contrast the t-test becomes slower at > $10^5$ data points since it has more possibilities to adapt window sizes. A limiting factor in case of a lot of data points for the HMM beside processing speed is memory consumption of the Viterbi reconstruction. Taken together the more efficient computation of EBS compared to the other algorithms allows for the analysis of high bandwidth data. This in turn increases the accuracy of step-finding. For example when using the motor kinetics of the intermediate scenario, the fraction of correct steps can be increased from 32% with 2kHz sampling rate to 52% with 200kHz.

In summary, in the slow and intermediate scenario the algorithms under consideration perform similarly in the total number of steps found as well as in the number of correct steps. In the fast scenario
Table 1: Comparison of computation efficiency of the different step-finding algorithms. \(\sim 900\) simulated steps on commodity hardware (i7-2600, 3.6GHz CPU Ubuntu System, 4GB memory). Corresponding run times were recorded in matlab for the signal of the given size. Peak memory usage, i.e. resident set size (RSS) was measured with Linux’s proc information system.

|              | t-test | HMM   | TVDN + Graph Cut |
|--------------|--------|-------|------------------|
| data points  | \(3 \cdot 10^5\) | \(2 \cdot 10^5\) | \(2 \cdot 10^7\) |
| run time/s   | 2967   | 3074  | 312              |
| peak RSS/GB  | 0.2    | 1.8   | 0.6              |

where elongation rates are faster, bandwidth is lower and noise amplitudes are higher EBS shows better results. Moreover when using EBS, the results of the fast scenario could be significantly improved by higher sampling rates which gives more data points for each plateau while still preserving comparably short run times. Here, TVDN yields a better approximation of the simulated signal (supplementary material) which improves the result of the subsequent CC. Thus, the EBS method especially excels for signals obtained from long measurement time, high bandwidth and poor signal to noise ratio.

**Application to Experimental Data**

Experimental data of Pol II transcription at saturating nucleotide concentrations yield rates comparable to the fast scenario. For these conditions the EBS as the best performing algorithm would be able to correctly detect only \(\sim 18\%\) of all steps. Therefore, in order to better test the step-finding properties of EBS on actual experimental data one would need to reduce the stepping rates, or apply the algorithm to a motor protein with larger step size. A prominent example for such a motor is the bacteriophage \(\varphi 29\) DNA packaging motor, which makes steps of \(2.5bp\) \(^{11}\). We have applied EBS to experimental stepping data of \(\varphi 29\) recorded with a bandwidth of \(2.5kHz\) using opposing forces of around \(5pN\) \(^{11, 12}\). The standard deviation of the experimental noise at this sampling frequency was found to be \(\approx 3.8bp\). For this motor at low forces of a few pN a fast burst of four \(2.5bp\) steps is followed by a long dwelltime (figure 7a). The presence of \(2.5bp\) steps had previously been identified at large forces leading to a slow down of the \(2.5bp\) steps \(^{11}\). At the forces of \(5pN\) the previously applied t-test had failed to resolve the \(2.5bp\) steps, in contrast, some of the steps are detected by EBS (figure 7). While the EBS algorithm is not able to determine a large fraction of steps of Pol II at saturating nucleotide concentrations given published noise levels, it can be used to investigate pausing of the enzyme.

We use EBS to detect pauses (Methods) in experimental data from single molecule transcription elongation data of Pol II (M. Jahnel, S. Grill Lab). The signal consists of \(N \sim 7 \cdot 10^4\) data points and was recorded with a sampling frequency of \(1kHz\). The noise amplitude has an estimated average standard deviation of \(\sim 10bp\). The pauses determined by SGVT are compared to pauses found by EBS (figure 7). When comparing the results from both algorithms one finds that most long pauses do overlap, while differences are observed for the detected short pauses. In order to get a better understanding of how well the two algorithms perform, we again use simulated data with parameters for stepping rates and sampling frequency according to the fast scenario (Supplementary material). In accordance with previously published discussions on backtracked pauses \(^{59}\) we distinguish long \((t > t_p)\) and short pauses \((t < t_p)\) by a time scale \(t_p = 1/\sqrt{k_f \cdot k_b} = 0.8s\). All simulated long pauses were found by EBS \((100\%)\) and the total length of long pauses compared to simulated long pauses was \(113\%.\) Also the SGVT found almost all
Figure 7: (a) 2.5bp substeps in ϕ29 bacteriophage data (circles) measured in an optical tweezers experiment, EBS (blue) and t-test (red). (b) Paused regions in experimental Pol II transcription elongation data. Shaded regions indicate paused intervals found by the SG-filtering method with a velocity threshold of two standard deviations of the pause peak (green) and EBS (red). 1kHz sampled transcription data (grey), SG filtered data of polynomial order 3 and frame size: 2.5s (cyan) and step detection result of our method (blue). (c) Step size histogram of detected steps by EBS. (d) Zoom into a detected pause. Shown is EBS signal (blue), SG filtered signal (black), measured data at 1kHz (black circles) and paused regions (SG: green, EBS: red).

long pauses (98%) with 94% of the total duration of simulated long pauses. Both methods did not falsely assign long pauses and thus the result of finding long pauses in step detected data and in SG filtered data largely agrees. However concerning short pauses EBS outperforms SGVT in overall recovery (EBS: 61%, SGVT: 38%) and accuracy (EBS: 8% of found pauses are false, SGVT: 43%). Especially for experiments with near base pair resolution and slow elongation rates on the order of the forward and backward stepping in backtracked states SG filtered data is not suitable to separate pauses from elongation and hence step-detection becomes the only option. For these experiments step-detection accuracy becomes better and allows the analysis of dwell time fluctuations which allows further insights into enzymatic reaction cycles such as DNA sequence dependent dynamics (60).
Conclusion & Outlook

We have presented a novel energy based step finding scheme comprised of a TVDN step followed by a CC analysis. The EBS algorithm outperforms current schemes for detecting steps or pausing events in time trajectories of molecular motors, both in precision as well as in speed and in the memory required for analysis.

A reason the proposed EBS method exhibits competitive performance is a favorable representation of information in the signal, which led to the two stage process. Here, TVDN offers a fast and unbiased denoising scheme while still preserving the step features of the underlying signal. In fact, TVDN performs often very well in tracing the actual signal even under noisy conditions.

In the context of bacterial flagellar motors another step detection algorithm that applied TVDN used rough bounds for a reasonable choice of the regularization parameter \( \lambda \). Here, we have used a drastically improved algorithm for solving the TVDN problem which provides very fast performance. This allows us to propose a heuristic to choose the regularization parameter \( \lambda \) automatically, i.e. without any freely adjustable parameters. Nonetheless, a more rigorous theoretical examination of the sudden change from over- to underfitting of TVDN which led to our heuristics remains to be done. Donoho et al. have reviewed the observation that sudden break-downs of model selection or robust data fitting occur in high-dimensional data analysis and signal processing. They further refined this finding for Compressed Sensing in, which is a class of \( l_1 \) regularized convex optimization problems. It remains an interesting question if similar theoretical statements can be established for TVDN.

We found that CC is very well suited to cluster the output of the compression. Further there are comparably fast algorithms available to solve relevant energy loss functions. In fact, we found that the Graph Cut algorithm scaled approximately quadratically in the number of tuples and linearly in the size of the predefined level set in our applications (Supplementary Material). One shortcoming regarding our implementation is the limited support of non-submodular energy loss functions. The approximation that we have implemented only works if a negligible number of non-submodular energies arise. Recent advances make it possible to extend the class of energy loss functions which are efficiently solvable with Graph Cut algorithms. With this kind of extension the penalizing energy scheme can be straightforwardly extended in an intuitive way to other prior information. For example, a histogram prior could yield a global energy term that favors certain step size and dwell time histograms.

In our analysis we used a predefined set of levels supplied to the clustering stage. If the prior knowledge of levels seems constraining, there exist methods from Multi-Model Fitting which can be used to incrementally refine the level set.

Both, the TVDN stage as well as the clustering stage, provide the possibility to harness parallelization to gain speedups. A long high bandwidth trajectory could be divided into smaller time-intervals, which could then be treated in parallel. Of course one would need to find a way to keep care of the boundaries between the intervals, e.g. by shifting the time intervals and merge the data. This extension would also make a quasi online processing of measurement data possible, where new intervals are successively ingested.

EBS was successfully applied to detect pauses by Pol II as well as \( 2.5 \)bp steps in the packaging of DNA by the bacteriophage \( \phi 29 \) motor. However, while some steps could be found, the larger the noise and smaller the step-size the fewer correct steps are found.

To make fully use of the advantages of EBS more high bandwidth data is needed. Moreover, shorter tether length, smaller beads or stiffer handles provided by DNA origami increase resolution and thus improve step detection.
In summary, the EBS method fills the gap of tools which are able to handle high bandwidth data with many data points as well as very noisy data under quite general assumptions. Regardless of the difference in TVDN and Graph Cut the energy based model provides an intuitive access for the user of the method.

**Acknowledgement**

We thank Marcus Jahnel for providing the experimental Pol II data, Gheorghe Chistol for providing the experimental $\varphi 29$ phage packaging data and Jeffrey R. Moffitt and Gheorghe Chistol for MATLAB code of the t-test algorithm.

This work was supported by the EU Integrating project SIQS, the ERC Synergy grant BioQ as well as the ERC starting grant Remodeling and an Alexander von Humboldt Professorship.

Unless otherwise stated computations were performed on the computational resource bwUniCluster funded by the Ministry of Science, Research and the Arts Baden-Württemberg and the Universities of the State of Baden-Württemberg, Germany, within the framework program bwHPC.
SUPPLEMENTARY MATERIAL

An online supplement to this article can be found by visiting BJ Online at http://www.biophysj.org.

**Total Variation Denoising Algorithm**

Our implementation of 1D total variation denoising is based on the C code published together with (30). This publication also provides a detailed outline of the TVDN algorithm, description of its working principles as well as the optimality condition it adheres to.

**Determination of \( \lambda_{\text{max}} \) in TVDN**

In this section we want to show, how to determine the value of \( \lambda_{\text{max}} \) in TVDN analytically. The \( \lambda_{\text{max}} \) value determines the value of the regularization parameter \( \lambda \) in equation (2) above which the solution \( x^* \) remains constant and therefore contains no steps anymore. The information in the following subsections is twofold: First derive general expressions for the Fenchel-Rockafellar-Dual problem and the forward-backward splitting applied to TVDN, second we then derive a condition for \( \lambda_{\text{max}} \) from the Fenchel-Rockafellar dual problem and provide an analytical solution. Furthermore we give hints on the special (tridiagonal) structure of the involved operators. The definitions in the next sections follow the work of (62).

**Fenchel-Rockafellar Dual Problem and Forward-Backward splitting**

Independent of the problem of our work, we start with a function \( f(x) \) which is convex, proper, and lower semi-continuous. Then
\[
\forall u \in \mathbb{R}^n, \quad f^*(u) = \max_{x \in \mathbb{R}^n} \langle x, u \rangle - f(x)
\]

is called it’s Legendre-Fenchel dual function (62). \( f^* \) is also convex, and it holds \( (f^*)^* = f \). A further specialization is useful in the context of our work, as the TVDN problem consists of a minimization of two composed convex functions
\[
\min_{x \in \mathbb{R}^n} f(x) + g(A(x))
\]

where \( A \in \mathbb{R}^{(p \times n)} \) and the convex functions \( f : \mathbb{R}^n \to \mathbb{R} \) and \( g : \mathbb{R}^p \to \mathbb{R} \). We assume, that \( f^* \in C^1 \) and therefore there exists a Lipschitz continuous gradient.

Due to the Fenchel-Rockafellar theorem, covered in Chapter 15 of (62), the following problems are equivalent:
\[
\min_{x \in \mathbb{R}^n} f(x) + g(A(x)) = -\min_{u \in \mathbb{R}^p} f^\dagger(-A^\dagger u) + g^\dagger(u)
\]

where \( \dagger \) denotes the adjoint function. The unique solution of the primal problem \( x^* \) can be recovered from a solution of the dual problem \( u^* \), which has not to be necessarily unique.
\[
x^* = \nabla f^\dagger(-A^\dagger u^*)
\]

We use an additional assumption which is not a constraint for the TVDN problem: \( g \) is simple. That means, one can compute a closed-form expression for the so-called proximal mapping
\[
\text{prox}_{\gamma g}(x) = \argmin_{z \in \mathbb{R}^n} \frac{1}{2} \| x - z \|^2 + \gamma g(z) \quad \forall \gamma > 0.
\]
Further due to Moreau’s identity $g^\dagger$ is also simple \cite{62}.

Now having the connection between primal and dual problem at hand, this means, one has to solve again a composite problem of a convex and a simple function

$$\minimize_{u \in \mathbb{R}^p} F(u) + G(u)$$

with $F(u) = f^\dagger(-A^\dagger u)$ and $G(u) = g^\dagger(u)$.

A typical method to do Proximal Minimization is Forward-Backward splitting (see eg. Chapter 27 of \cite{62}). The dual update is given by

$$u^{(\ell+1)} = \text{prox}_{\gamma G} \left( u^{(\ell)} - \gamma \nabla F(u^{(\ell)}) \right).$$

In this update step $\gamma < L/2$, where $L$ is the Lipschitz constant. The primal iterates are given by:

$$x^{(\ell)} = \nabla F(-A^\dagger u^{(\ell)}).$$

The above general statements and theorems are taken from the tool set of Convex Analysis. For further background see e.g. \cite{31} or \cite{62}. In the following we discuss more problem specific expressions.

**Application to Total Variation Denoising**

In a continuous picture the total variation of a smooth function $\phi : \mathbb{R} \to \mathbb{R}$ is defined as

$$J(\phi) = \int \|\nabla \phi(s)\| \, ds.$$

In the discretized version one has to consider a discretized gradient operator $A : \mathbb{R}^n \to \mathbb{R}^p$ with $p = n-1$.

$$J(x) = \|Ax\| = \sum_i u_i$$

where $u_i = x_{i+1} - x_i$ and therefore $A$ taking the following form:

$$A = \begin{pmatrix}
1 & -1 & 0 & \ldots & 0 \\
0 & 1 & -1 & \vdots & \\
\vdots & \ddots & \ddots & \ddots & \\
0 & \ldots & 1
\end{pmatrix}.$$  \hspace{1cm} \text{(18)}$$

Using this and taking into account that the Divergence and Gradient operator are minus adjoint of each other ($\langle \nabla f, g \rangle = -\langle f, \nabla \cdot g \rangle$) the adjoint of the discrete gradient operator $A^\dagger$ is minus the discrete divergence:

$$A^\dagger = \begin{pmatrix}
-2 & 1 & 0 & \ldots & 0 \\
1 & -2 & 1 & \vdots & \\
0 & 1 & \ddots & \ddots & \\
\vdots & \ddots & \ddots & 1 & \\
0 & \ldots & 1 & -2
\end{pmatrix}.$$  \hspace{1cm} \text{(19)}$$
Therefore the divergence highly resembles a typical Laplace filter from signal processing. This leads for a single entry to \( u_i - u_{i-1} = x_{i+1} - 2x_i + x_{i-1} \). For the deviation of \( \lambda_{max} \) we assume the boundary conditions that \( u_0 = 0 \) and \( u_n = 0 \).

For noise removal (and to get the connection to eq. (2)) the following problem has to be solved

\[
x^* = \arg\min_{x \in \mathbb{R}^n} \frac{1}{2} \| x - y \|^2 + \lambda J(x),
\]

(20)

To make use of the material so far choose the following composition

\[
f(x) = \frac{1}{2} \| x - y \|^2 \quad \text{and} \quad g(u) = \lambda \| u \|.
\]

(21)

After that one has to translate \( f(x) \) and \( g(x) \) into their dual representations \( f^\dagger(u) \) and \( g^\dagger(u) \) by using the following relations

- For \( f(x) = 1/2 \| Ax - y \| \) and \( A \in \mathbb{R}^{n \times n} \) can be inverted then

\[
f^\dagger(u) = \frac{1}{2} \| (A^\dagger)^{-1} u + y \|^2
\]

(22)

- For \( f(x) = \| x \|_p = \sum_i (|x_i|^p)^\frac{1}{p} \) is a \( p \)-norm: Then the dual function corresponds with the indicator function \( \iota_{C} \) of the convex set \( C \):

\[
f^\dagger(u) = \iota_{\| u \| \leq 1} \quad \text{where} \quad \frac{1}{q} + \frac{1}{p} = 1
\]

(23)

Using that we get the following dual representation of the dual functions \( F(u) + G(u) \) for the TVDN problem in the Fenchel-Moreau-Rockafellar formulation.

\[
F(u) = \frac{1}{2} \| y - A^\dagger u \|^2 - \frac{1}{2} \| y \|^2 \quad \text{and} \quad G(u) = \iota_{C}(u) \quad \text{where} \quad C = \{ u : \| u \|_\infty \leq \lambda \}.
\]

(24)

The solution to the dual problem \( u^* \) can be obtained by solving

\[
u^* \in \arg\min_{\| u \| \leq \lambda} \| y - A^\dagger u \|,
\]

(25)

and by applying eq (11) the solution to the primal problem \( x^* \)

\[
x^* = y - A^\dagger u^*.
\]

(26)

What is missing for concrete expression for the forward backward iterations is first a closed form for the gradient of \( F \), which is given by

\[
\nabla F(u) = A(A^\dagger u - y).
\]

(27)

Secondly it is possible for the proximal operator of \( G \), which is the orthogonal projection on the set \( C \)

\[
\text{prox}_{\gamma G} u = \frac{u}{\max(1, \| u \| / \lambda)}.
\]

(28)
\[ \gamma < \frac{2}{\|A^\dagger A\|} = \frac{1}{4}. \]  
(29)

Inserting the above statements into the general dual update step from eq. (14), one gets the following expression:

\[
u^{(l+1)} = \text{prox}_{\gamma G} \left( u^{(l)} - \gamma \nabla F(u^{(l)}) \right)
= \frac{u^{(l)} - \gamma \nabla F(u^{(l)})}{\max \left( 1, \frac{\|u^{(l)} - \gamma \nabla F(u^{(l)})\|}{\lambda} \right)}
= \frac{u^{(l)} - \gamma A(A^\dagger u^{(l)} - y)}{\max \left( 1, \frac{\|u^{(l)} - \gamma A(A^\dagger u^{(l)} - y)\|}{\lambda} \right)}.
\]  
(30)

**Derivation of \( \lambda_{\text{max}} \) from the Proximal Iteration**

Finding a maximal regularization parameter \( \lambda \) is equal to finding a criterion, such that the dual iterations remain constant \( \forall l \)

\[ u^{(l+1)} = u^{(l)}. \]  
(31)

By using eq. (26) one can see, that this will lead to a steady state solution \( x_i^* = \text{const} \ \forall i \). For simplicity assume \( \tilde{\lambda} = \lambda/\gamma \). Starting from the proximal iteration we find that in case of \( \tilde{\lambda} \leq \|u^{(l)} - \nabla F(u^{(l)})\| \) the problem simplifies to

\[ u^{(l+1)} = u^{(l)} - Ay + AA^\dagger u^{(l)}. \]  
(32)

To satisfy the constant condition from eq. (31) the \( u^{(l)} \) has to be in the solution of:

\[ AA^\dagger u = Ay. \]  
(33)

The shape of \( AA^\dagger \) is the following

\[
AA^\dagger = \begin{pmatrix}
-3 & 3 & -1 & \ldots & 0 \\
1 & -3 & 3 & \ddots & \vdots \\
0 & 1 & \ddots & \ddots & -1 \\
\vdots & \ddots & -3 & 3 & \\
0 & \ldots & 1 & -2
\end{pmatrix}
\]  
(34)

Linear equations with a tridiagonal affine transform \( AA^\dagger \) can be efficiently solved for example an algorithm proposed by Rose (63).

Still missing is a treatment of the primal iteration step \( x^{(l+1)} = y - A^\dagger u^{(l+1)} \). The connection to the \( \lambda \) in the original TVDN problem is given such that, the Karush-Kuhn-Tucker conditions are still valid for our steady state solution (33). This means, that every \( u^{(l)} \) in the dual solution has to satisfy

\[ u_k^* \in [-\lambda, \lambda]. \]  
(35)

To ensure this, we have to choose

\[ \lambda_{\text{max}} = \|u\|_{\infty} \]  
(36)

which gives as a clear statement how to choose a maximal lambda.
Algorithm Implementing the $\lambda_h$-Heuristics

As outlined in the Methods section of the paper, we use a sudden increase of resulting steps when decreasing the regularization parameter $\lambda$ in the TVDN problem shown in eq. (2) from $\lambda_{\text{max}}$ to determine $\lambda_h$. In the following, we want to describe the heuristic method, we used to choose the value of $\lambda_h$. Starting point for the algorithm is the value of $\lambda_{\text{max}}$ on a curve like the one depicted in figure [2]. The iterative method shown in algorithm [1] approximates the point of steepest ascent in an $\lambda$-$n$ diagram, where $n$ is the number of steps, by searching an interval where the slope exceeds the slope of the secant of $\{0, \lambda_{\text{max}}\}$. The function $N(\lambda)$ counts the number of steps after the TVDN minimization for a given value of $\lambda$. This simple method gave us stable results for a variety of our test signals, either simulated or experimentally gathered. In the following section we have a closer look into the stability of the effect of sudden increase of steps.

Algorithm 1 Outline of our line-search algorithm to determine $\lambda_h$

1: $\lambda, n \leftarrow \lambda_{\text{max}}, N(\lambda_{\text{max}})$
2: $\lambda^+, n^+ \leftarrow \frac{\lambda_{\text{max}}}{2}, N(\lambda_{\text{max}}/2)$
3: $\delta_{\text{start}} \leftarrow \frac{|N(0) - N(\lambda_{\text{max}})|}{\lambda_{\text{max}}}$
4: while less than max. iterations do
5: $\delta \leftarrow \frac{|n^+ - n^-|}{\lambda^+ - \lambda^-}$
6: if $\delta > \delta_{\text{start}}$ then
7: break
8: end if
9: $\lambda, n \leftarrow \lambda^+, n^+$
10: $\lambda^+, n^+ \leftarrow \frac{\lambda^+}{\rho}, N(\lambda^+)$
11: end while
12: return $\lambda_h \leftarrow \lambda^+$

Stability and Scalability of $\lambda_h$-Heuristics

The $\lambda_h$-heuristic is the starting point of finding steps which are corrupted by noise and here we analyze the applicability of this scheme on simulated data. In general we do not expect that this scheme returns good results for arbitrarily large noise amplitudes or sampling frequencies on the order of stepping rates. The dependencies on noise amplitudes and sampling frequencies for poisson distributed steps (forward stepping with rate constant $10Hz$) covered by noise can be best summarized in the following phase diagrams (figure [8] and [9]). As for the data shown in figure [2] we compute the number of produced steps after TVDN for different denoising parameter $\lambda$. For a signal of $100s$ length with $1016$ poisson distributed steps we vary the sampling frequency and keep the standard deviation of noise constant at $4.4bp$ (figure [8]). For each sampling frequency the number of produced steps is normalized to the number of simulated data points. Furthermore, we vary the standard deviation of noise and keep the sampling frequency constant $6kHz$ (figure [9]).

In the overfitting regime (white), the number of steps of the denoised signal equals the number of data points. At $\lambda/\lambda_{\text{max}} = 1$ the denoised signal is constant without any steps. At a sampling frequency $f = 10kHz$ the number of steps as a function of $\lambda$ has a clear transition between overfitting and underfitting and resembles the data shown in figure [2](green). In this case $\lambda_h = 30.2$ ($\log(\lambda_h/\lambda_{\text{max}}) = -14.1$)
Figure 8: Over fitting transition depends on sampling frequency. Each step signal has 1016 poissonian distributed steps sampled with different frequencies and covered by noise. The signals are 100s long.

As the sampling frequency is lowered the transition is shifted more and more towards $\lambda_{max}$. Below a sampling frequency of 100Hz the number of produced steps are gradually increasing until there are as many steps as data points, as was already observed for the data in figure 2 (red curve). If the sampling frequency is this low, the $\lambda_h$-heuristic is not applicable anymore since TVDN breaks down and just imitates noise. At 100Hz there are on average 10 data points for each plateau. Since steps are poissonian distributed many steps have plateaus that consist of less than 10 data points and are thus hardly distinguishable from noise.

For a single step signal without noise a highest value $\tilde{\lambda}$ of $\lambda$ such that TVDN preserves the step of height $\Delta$ and plateau length $w$ is $\tilde{\lambda} = \Delta \cdot w/2$ (23). At the given stepping rate of 10Hz, 20% of the steps have shorter dwell times than 20ms. Applying this argument to the signal with 10kHz sampling frequency (figure 8), $\tilde{\lambda} < 34$ (i.e. $\log(\tilde{\lambda}/\lambda_{max}) < -14.0$) and thus our $\lambda_h$-heuristic yields a $\lambda$ that is able to preserve steps with longer dwell times.

For decreasing SNR we get a similar shift of the phase boundary towards $\lambda_{max}$ for worse SNR, fig. 9.

Mapping of Energies on Edge-Capacities

In the process of assigning a level $\xi_i$ to vertex $v_i$ the above mentioned Graph Cut algorithm solves a binary decision problem, whether the assignment of a new level is more favorable in terms of the energy loss function or not. The binary outcome of the decision is reflected in the graph structure by introducing two special vertices, where $t$ is associated with keeping the old and $s$ with assigning the proposed level. The energy values of the data term $Q_i$ as well as the pairwise term $P_{i,i+1}$ and their different combinations of keeping the current level or assigning a new level are mapped to capacities of edges in the MRF. In this section this mapping is explained stemming theoretical foundations outlined by Kolmogorov et. al. in (34).

In figure 10 the situation for the data term is depicted. Here the mapping is easy, as the energy for a single variable $v_i$ for the current level $E_0$ is mapped to the Edge $A$. The energy $E_1$ for a new level is mapped to the edge $B$. 
Figure 9: Influence of SNR on over fitting transition. Each step signal has 980 poissonian distributed steps with different noise amplitudes at 6kHz. Every signal is 100s long and consists of $6 \cdot 10^5$ data points.

Figure 10: Situation in an MRF concerning a single variable $v_i$ and edges $A, B$ to special variables $s$ and $t$ relevant for the data term.

Figure 11: Two neighbouring variables in an MRF and edges relevant for the pairwise term. Here the two $s$ vertices represent the same vertex in the graph and are just drawn separated to make the diagram look nicer. The same is true for the $t$ vertices.

The situation for the pairwise term $P_{i,j}$ is more complicated and depicted in figure 11. Here two variables $v_i$ and $v_{i+1}$ are involved which leads to four different energy combinations $E_{0,0}, E_{0,1}, E_{1,0}, E_{1,1}$ are possible. Here $E_{0,0}$ is associated with the energy value if both variables get assigned a new level. In contrast $E_{1,1}$ represents the energy of both variables keeping their current levels. The two other combinations represent the case when one variable keeps the current label and the other gets the new level assigned. $E_{0,1}$ the variable $i + 1$ keeps its level, for $E_{1,0}$ this is the case for the variable $i$.

The four energies can be represented in the following way

$$
\begin{bmatrix}
E_{0,0} & E_{0,1} \\
E_{1,0} & E_{1,1}
\end{bmatrix}
= \begin{bmatrix}
a & b \\
c & d
\end{bmatrix}
= \begin{bmatrix}
a & a \\
d & d
\end{bmatrix}
+ \begin{bmatrix}
0 & b - a \\
-c - a & 0
\end{bmatrix},
$$

(37)
The first summand on the right hand side is mapped to terminal capacities. This means that the capacity $a$ is associated with the edge $C$, and the capacity $d$ with the edge $B$. The second summand maps to the edge $E$ and gets the capacity $b - a + c - d$.

At this point the above mentioned strategy to circumvent a violation of submodularity is applied if $E_{0,0} + E_{1,1} > E_{0,1} + E_{1,0}$. Then in turn $E_{0,1}$ and $E_{1,0}$ is increased and $E_{0,0}$ is decreased by a small amount until the submodularity condition Eq. (5) is satisfied. Details and limitation of this approach can be found in (42).

**α-Expansion Algorithm Outline**

Finding a solution $\xi^*$ that minimizing eq.(3) is a problem that is in general NP-hard to solve for $|\mathcal{L}| \geq 3$. The iterative alpha-expansion algorithm outlined in algorithm 2 finds provably good approximate solutions to this problem. In each iteration the algorithm updates or moves the current labeling $\xi'$ if it has found a better configuration. To achieve this, in each iteration, a new, randomly chosen label $\alpha \in \mathcal{L}$ is introduced and each site $v_i$ has the choice to stay with the previous label or adopt the new proposed label $\alpha$. The binary optimization problem is solved via a Graph Cut (line 4 of algorithm 2). This step is called $\alpha$-expansion due to the fact, that the number of nodes with the label $\alpha$ assigned could grow during this phase. The $\alpha$-expansion algorithm was initially published by Boykov et al. in (24).

**Algorithm 2 α-Expansion outline**

1: $\xi' \leftarrow$ arbitrary labeling of sites
2: while not converged do
3:     for all $\alpha \in \mathcal{L}$ do
4:         $\xi^\alpha \leftarrow \text{argmin}_\xi E(\xi, \xi')$
5:         if $E(\xi^\alpha) < E(\xi')$ then
6:             $\xi' \leftarrow \xi^\alpha$
7:         end if
8:     end for
9: end while

Together with this publication we provide an open-source software package implementing the EBS method as well as the simulations.

The implementation reflects the stages of the EBS method described above. It consists of four executables. First there is a program implementing the lambda heuristics to determine the optimal TVDN regularization parameter $\lambda_h$. Second the package provides a denoising program which removes noise by solving the TVDN problem. Third a level generator makes it easy to create a set of equidistant levels. And fourth we have a program which clusters to a set of predefined levels. For maximum flexibility each of the programs exchanges data via files, and can be recomposed as needed. The documentation shipped with the code explains the data format and which parameters each of the binaries actually

![https://github.com/qubit-ulm/ebs](https://github.com/qubit-ulm/ebs)

![https://github.com/qubit-ulm/ebs_simulation](https://github.com/qubit-ulm/ebs_simulation)
Figure 12: Mean run time performance of combinatorial optimization step for 10 simulated noisy step signals. (a) Graph Cut computation time versus number of tupels for a label grid of 800 levels. Second order polynomial fit to computation times (red curve). (b) Graph Cut computation time versus label grid size for a fixed system size of 750 tupels. Linear scaling of performance with increasing number of levels in the label grid set. The error bars are SEM.

needs. The programs are implemented in C++ and run on Unix as well as on Windows systems. We also provide examples how to use the programs from MATLAB or Python. The implementation depends on the Boost.Graph libraries (53) to construct the graph structure as well as to solve the min-cut/max-flow problem and reuses the TVDN algorithm from Laurent Condat (30).

Scaling of Graph Cut

When analyzing high-bandwidth noisy time traces of the movement of molecular motors the CC step often limits run time performance. Most of all, performance is influenced by system size, i.e. the number of tupels and number of levels in the label grid set. To analyse the scaling behaviour for these two influences numerically, we simulated 10 noisy poisson step signals for each system size and label grid set and record computation times. fig. (12) shows mean and standard deviations as error bars. In fig.(12a) system size was increased from 250 to 3000 tupels and the number of levels offered to the combinatorial optimization problem was kept constant to around 800 levels. In this case computational time is expected to scale mostly with the complexity of the Boykov-Kolmogorov max flow algorithm which has a worst case complexity of $O(|edges| \cdot |nodes|^2 \cdot C)$ (35). Where $C$ is the cost of the minimal cut, $|edges|$ and $|nodes|$ are respectively the number of edges and nodes in the graph. For the type of graphs considered here, for each additional tupel in the input data set we have to add two edges which would give a worst case complexity of roughly $O(N^3 \cdot C)$. However, computation times fit well to a quadratic function meaning that for our signals the scaling behaviour is better than the worst case complexity (figure 12 red curve). The second case is shown in fig.(12b). When the system size is fixed (here: 750 tupels) and the number of labels increases (here: from $10^3$ to ca. $10^4$) by refining the label grid subsequently, the corresponding run times increase linear. This is in agreement with the theory behind multi label graph cut problems (35). The $\alpha$-expansion offers new labels one by one in a random order until all labels were used and the iteration stops. Thus the observed linear scaling in the number of labels is also expected from theory.
It is important to point out that due to the TVDN compression the expression above is an improvement for this type of step signals (high bandwidth, number of data points $\sim 10^5$ but comparably few steps $< 1000$) compared to the Fourier transform accelerated HMM implementation (20): $O(m \cdot n^2 N \cdot \log_2 m)$ where $m$ is the number of position states, $n$ the number of molecular states and $N$ the number of data points. Moreover, the direct comparison of run times and memory consumption given in the main text shows that our algorithm is advantageous regarding computational resources compared to existing algorithms.

Comparison of Graph Cut and MCMC

Since Markov Chain Monte Carlo (MCMC) methods are standard techniques to optimize an energy functional with Pott’s model terms like Eq. (7), we compare the Graph Cut method with a Metropolis Hastings (MH) sampling and simulated annealing (SA) optimization algorithm (61). In each iteration we randomly generate a proposal assignment of labels. The new assignment of a site is accepted or rejected according to the standard MH rules. Moreover a logarithmic temperature schedule is used for SA. The temperature parameter is introduced as commonly done: $p(x) \propto \exp(-E(x)/T)$. If an accepted proposal has smaller energy than all previous ones it becomes the new configuration that minimizes Eq. (7). To compare the quality of the step detection result we computed the energy, Eq. (3) with prior terms Eq. (7) for the energy minimizing solutions of Graph Cut and MCMC method, Fig. (13). For a system size below 350 tuples, computation times of the Graph Cut algorithm were always below 10s. Since MCMC is computationally more complex longer computation times were used for MCMC, i.e. 45min which allowed for 12 iterations of a SA temperature cycle. Each cycle consists of 20 subsequent cooling steps and in each step we iterate through 20000 proposals. Inspite of the significantly higher computational cost the MCMC solutions always have higher energies compared to Graph Cut and the excess energy increases for larger systems. This shows that MCMC returns increasingly worse solutions compared to the Graph Cut technique when the number of input data grows for fixed computation time. As expected, Graph Cut shows an approximately linear increase in energy with linearly increasing system size.

To conclude, the plain MCMC algorithm used here is conceptually simpler than Graph Cut but computationally more expansive and also less suited to cluster the denoised steps optimally according to an energy functional. This finding in one dimension is not surprising, since similar observations had been made in 2D image analysis (24).

Noisy step simulations

Single base pair steps are typically exceeded by noise fluctuations and most of the time it is not possible to judge by eye whether an algorithm correctly positioned steps. Therefore simulated data is necessary to show and compare the performance of step detection algorithms. We generate noisy steps in two stages as outlined in Fig. (14). First, we generate a PWCS according to a simplified version of the linear ratchet model of Pol II (25). This model contains elongation and backtracked states and reproduces the ability to pause (45, 46). During elongation, 1bp forward steps are generated with an effective rate of $k_{\text{elong}}$. This effective rate includes the process of translocation, NTP insertion and pyrophosphate release. In our model catalysis, bond formation and PPi release are summarized by a rate $k_3$. Furthermore, the NTP-binding net rate is $k_2 = c_{\text{NTP}} \cdot k_3/K_D$ and the translocation net rate $k_1 = k_+ \cdot k_2/(k_- + k_2)$. $c_{\text{NTP}}$ is the NTP concentration, $K_D = 9.2 \mu M$ the dissociation constant, $k_+ = 88 Hz$ is the forward translocation rate of Pol II and $k_- = 680 Hz$ backward translocation. The values of these constants are known from
Figure 13: Graph Cut MCMC comparison. Energies of optimal solutions with increasing system size for MCMC and Graph Cut algorithms.

experiments (25). The elongation rate is then determined by \( k_{\text{elong}} = (1/k_1 + 1/k_2 + 1/k_3)^{-1} \).

With a rate of \( k_b = 5Hz \) the motor makes a backward step of identical size as the forward step and thus enters the backtracked state. The enzyme can further backtrack by a rate \( k_b = 1.3Hz \) or return to the original state with a rate \( k_f = 1.3Hz \) (figure 15). The rates corresponding to a forward step (\( k_+, k_f \)) or backward step (\( k_b, k_b \)) are modified under external forces according to \( k(F) = k(0) \cdot \exp(\pm F \cdot 0.17nm/(k_bT)) \), where \( k_bT = 4.11pN \cdot nm \) and the plus sign in the exponent applies to rates of forward steps. Simulations were computed for an assisting force of 6.5pN. At this force forward and backward diffusion rates are \( k_b = 3.8Hz, k_b = 1.0Hz \) and \( k_f = 1.7Hz \), in accordance with the kinetic model. For numerical simulation purposes the rates above are devided by the simulation’s time increment to yield dimensionless quantities.

The transitions between elongation and backtracked states are generated using the Gillespie stochastic simulation scheme (47) for a single enzyme. Dwell times are sampled from an exponential distribution according to the respective rates.

In a second step, we simulated experimental noise including effects of confined brownian motion of trapped micro spheres. To accurately reflect the experiment, we take into account changes in the tether length and in the tether stiffness due to motion of the enzyme. We apply a harmonic description of the trapping potentials and assume that the DNA linker can be described by a spring constant \( k_{\text{DNA}} \) determined by the worm like chain model (49).

To formulate the equation of motion of two trapped micro spheres tethered by DNA we choose the coordinate system such that the enzyme moves in \( x \)-direction. Furthermore we assure that drag coefficients \( \gamma_i \) and the trapping stiffness \( k_{\text{trap},i} \) are identical in both traps. With this the effective DNA length \( x \) can be described by the following equation.

\[
\gamma \dot{x} = -k \cdot x + F_T(t)
\]

where \( k = k_{\text{trap}} + 2k_{\text{DNA}} \), \( k_{\text{DNA}} \) is the DNA stiffness and \( \gamma \) is the drag coefficient. \( F_T(t) \) is the thermal force which is treated as gaussian white noise: \( \langle F_T(t) \rangle = 0 \) and \( \langle F_T(t) F_T(t') \rangle = 2k_BT \gamma \delta(t - t') \).
Figure 14: Step and noise simulation procedure. State transition model and corresponding stepping rates determine the probability distribution from which a PWCS (first inset) is sampled. In a second step noise is simulated with the PWCS as the mean (second inset).

Figure 15: Simplified stepping model of RNAP II with an elongation and backtracked states.

Eq. (38) describes a so called Ornstein-Uhlenbeck process and can be solved and simulated by standard techniques of stochastic differential equations (48) which is shown in the next subsection. Eq. (38) was derived for the static situation without positional changes. However, a molecular motor which is attached between micro spheres by a DNA-tether will change the tether length during its activity. Thus, $k_{\text{DNA}}$ is also changing and can be computed using the worm-like chain model (49).

In the simulations we use a trap stiffness of $k_{\text{trap}} = 0.25pN/nm$, a drag coefficient of $\gamma = 0.8 \cdot 10^{-5} pN \cdot s/nm$ corresponding to beads with 850nm diameter and an initial length of $L = 3kbp$ for the DNA tether.

We simulated a slow, an intermediate and a fast scenario which differ by stepping speed, sampling frequency, number of data points and noise amplitudes. Sampling frequencies and number of data points of the slow scenario are $f = 5kHz$ and $N = 2.5 \cdot 10^5$ points, for the intermediate scenario: $f = 2kHz$ and $N = 10^5$ points and for the fast scenario: $f = 1kHz$ and $N = 5 \cdot 10^4$. The elongation rate $k_{\text{elong}}$ of the slow scenario $k_{\text{elong}} = 4.1Hz$ can be expected at a NTP concentration of $c_{\text{NTP}} = 7mM$. Since backtracking becomes more likely at these NTP concentrations we limited analysis to simulated data that shows a net forward translocation. This excludes analysis of simulated data which exhibits only backtracked states. Elongation rates of the intermediate ($k_{\text{elong}} = 9.1Hz$) and fast scenario ($k_{\text{elong}} = 25.8Hz$) are expected at $c_{\text{NTP}} = 20mM$ and $c_{\text{NTP}} = 1000mM$ respectively. The standard deviation of noise amplitudes are directly computed from the noisy input data. This is done by subtracting the simulated step signal from the noisy steps and computing the standard deviation of the remaining signal. In both scenarios, slow and intermediate, the computed standard deviation is $5.5bp$ at the given sampling frequency. For the fast scenario we choose $N = 5 \cdot 10^4$ data points and $1kHz$ sampling rate. Moreover in the fast scenario we use higher noise amplitudes with a computed standard deviation of $10.0bp$ at the $1kHz$ sampling frequency.
Finally, for all three scenarios 20 data sets were simulated and analyzed. Table 2 gives an overview over the simulation parameters.

Table 2: Overview of simulation parameters. Shown is elongation rate \( k_{\text{elong}} \), corresponding NTP concentration \( c_{\text{NTP}} \) and rate constants of the backtracking state. Moreover, the standard deviation of noise \( \sigma_n \), sampling frequency \( f \) and number of data points \( N \) is given.

| scenario: | \( k_{\text{elong}} / \text{Hz} \) | \( c_{\text{NTP}} / \text{mM} \) | \( k_b \) / Hz | \( k_f \) / Hz | \( \sigma_n / \text{bp} \) | \( f / \text{kHz} \) | \( N \) |
|----------|-------------------------------|----------------|-------------|-------------|----------------|----------------|-------|
| slow     | 4.1                           | 7              | 3.8         | 1.0         | 1.7            | \( \sim 6 \)  | 5     | \( 2.5 \cdot 10^5 \) |
| intermediate | 9.1                        | 20             | 2.3         | 1.0         | 1.7            | \( \sim 6 \)  | 2     | \( 1 \cdot 10^5 \) |
| fast     | 25.8                          | 1000           | 2.3         | 1.0         | 1.7            | \( \sim 10 \) | 1     | \( 5 \cdot 10^4 \) |

Simulating beads in a harmonic optical trap

As described above we account for confined brownian motion of trapped beads in a dual trap optical tweezers. A harmonic description of trapping potentials is applied and we assume the DNA linker can be described by a WLC model with a spring constant \( k_{\text{DNA}} \). In the following we briefly show the derivation of eq.(38) and its solution. We focus on the x-coordinates of two beads trapped in different optical traps and tethered by DNA. The equation of motion of such a system of reads (57):

\[
\gamma \dot{x} = -k x + F_T(t)
\]  

(39)

where \( x = (x_1, x_2) \) is the x-coordinate of first and second bead. Furthermore drag coefficient, stiffness and thermal force are:

\[
\gamma = \begin{pmatrix} \gamma_1 & 0 \\ 0 & \gamma_2 \end{pmatrix}, \quad \kappa = \begin{pmatrix} k_{\text{trap}, 1} + k_{\text{DNA}} & -k_{\text{DNA}} \\ -k_{\text{DNA}} & k_{\text{trap}, 2} + k_{\text{DNA}} \end{pmatrix}, \quad F_T(t) = \begin{pmatrix} F_{T, 1}(t) \\ F_{T, 2}(t) \end{pmatrix}
\]

The thermal force fulfills the gaussian white noise propertie: \( \langle \xi_i(t) \rangle = 0 \) and \( \langle F_{T, i}(t) F_{T, j}(t') \rangle = 2k_B T \gamma \delta_{ij} \delta(t - t') \) The relative coordinate \( \tilde{x} = x_2 - x_1 \) which will be called \( x \) in the following can be simplified by assuming that \( \gamma_1 = \gamma_2 = \gamma \) and \( k_{\text{trap}, 1} = k_{\text{trap}, 2} = k_{\text{trap}} \) to:

\[
\dot{x} = -f_c \cdot x + \frac{1}{\gamma} F_T
\]  

(40)

Where \( f_c = (k_{\text{trap}} + 2k_{\text{DNA}}) / \gamma \) is the corner frequency of the system. \( k_{\text{DNA}} = k_{\text{DNA}}(F, L) \) depends on force and length of the DNA tether and is calculated from the wormlike chain model (58). During enzyme stepping \( k_{\text{DNA}} \) has to be updated repeatedly with respect to the external parameters force \( F \) and length \( L \). Eq.(40) describes a so called Ornstein-Uhlenbeck process (OU) and can be solved and simulated by standard techniques of stochastic differential equations (43). From Eq.(40) it can be seen that for timescales slower than the corner frequency \( f_c \) noise behaves essentially as white noise. For faster timescales than \( f_c \) noise rather has characteristics of brownian motion. In the following, the simulation of an Ornstein-Uhlenbeck process is described. We rewrite Eq.(40) in Ito form:

\[
dx = -k \cdot x \cdot dt + \sqrt{2D} \cdot dW_t
\]  

(41)
Figure 16: Histogram of step sizes of the easy scenario 16a and the intermediate scenario 16b for t-test (blue), HMM (green) and EBS (red).

where $k = (\kappa + 2k_{DNA})/\gamma$, $D$ is the diffusion constant and $dW_t$ infinitiesmall describes brownian motion. For a finite time interval $\Delta W_t = \int_{t-h}^{t} dW_t'$ describes a standard normal distributed random variable $N(0, h)$, with standard deviation $\sigma = \sqrt{h}$. Eq. (41) just describes a gaussian random variable with mean $\mu$ and variance $\sigma^2$ (68):

$$x_t \in N(\mu, \sigma^2) = N(x_{t-1}e^{-\gamma t}, \frac{D}{\gamma} (1 - e^{-2\gamma t})),$$

and a random path can be straightforwardly simulated starting from an initial position $x_0$.

**Details of algorithm comparison**

To achieve best results for the three simulation scenarios we need to tune the external parameters of the t-test, the HMM and the EBS algorithm. While the latter is described in the main text we will briefly explain how to adapt the other two algorithms to yield as many correct steps as possible but also to have a large fraction of correct steps among the found steps.

For the t-test a minimum step size of 0.3nm and a shortest dwell of 10ms was used. Moreover, the t-test threshold was 0.01, the binomial threshold 0.005 and the maximal number of iterations was 100.

The HMM analysis was conducted with maximally 100 iterations for maximum likelihood estimation of transition probabilities. More iterations did not give better results and fewer iterations ($\leq 10$) could not optimize the log-likelihood properly (data not shown). For the slow scenario 85 states were used and for the intermediate and fast scenario we used 140. To prevent memory overflow in the intermediate scenario, we performed box car averaging to reduce the number of data points by a factor of two. Furthermore, a grid spacing of $1/2$ bp was used which proved to be better than a 1bp spacing. Since the HMM level grid has to be aligned by using noisy data as an input, a two times smaller grid spacing showed better results. In contrast the situation is advantegous in case of combinatorial optimization which constructs the level grid on already denoised data.

To complete the performance results of the algorithm comparison (figure 6) the step size histograms of the step detection result for easy and intermediate scenario are given, fig. (16). For slow stepping rates the step size histograms resemble the simulated step size $\pm 1bp$ quite well, fig. (16a). However, a deviation from the 1bp steps can be already seen in the intermediate scenario for the t-test and HMM, fig. (16b).
Figure 17: Cumulative fraction of found steps for the EBS in the intermediate scenario. Plotted is the cumulative number of found steps/simulated steps of the signal plotted in Fig. (5) for different dwell times. The number of detected steps compared to simulated ones is smaller for short dwell time steps. Dwell time histograms with a binning of 87.3 ms were determined for the detected and simulated steps respectively. The number of detected steps for each dwell time was divided by the corresponding number of simulated steps and cumulatively summed up. In total 60.5% of the number of simulated steps were found.

For EBS the majority of the detected steps are 1bp in size in both scenarios. The more difficult situation in the intermediate scenario is reflected by the larger fraction of 2bp steps (figure 16b, red).

Remarks on Example of TVDN and combinatorial clustering

In order to get temporal information of the missing steps in the example given in the Results & Discussion section, fig. (5), we compare the dwell time histograms of the simulated and detected steps. The cumulative fraction of found steps for a certain dwell time shows that steps with short dwell times are omitted with higher probability, fig. (17).

Effect of prior information in combinatorial clustering

To analyze the impact of the prior terms and level grid spacing on step detection quality we performed CC with different prior potential strength and level grid spacing (figure 18a). We varied the prior regularization parameters $1/\rho_S$ and $1/\rho_P$ starting from $1/\rho_S = 0$ to a maximum of $1/\rho_S = 6$, while the jump-height prior parameter was varied simultaneously such that $\rho_S/\rho_P = 12.5$ remained constant. By increasing the prior regularization parameters the precision of step detection can be increased (triangles, fig. 18a). Furthermore, precision can be improved by choosing a level grid with a spacing of the simulated step size of 1bp (squares, fig. 18a).

The variation of the prior term also effects step size histograms (figure 18b). When no prior terms are present ($\rho_S = \rho_P = 0$) the detected step-size is oftentimes smaller than the simulated step-height (figure 18b, upper panel). Optimization of the regularization parameters as well as an increase in level spacing
improves the precision of the EBS algorithm as shown in the histograms of detected step-sizes (figure 18b, middle and lower panel).

Application of EBS to find pauses in experimental transcription data

In the following we discuss the determination of pauses in experimental Pol II data as an example of further post processing of the detected steps and compare EBS based pause finding and SGVT on simulated data.

For the simulated Pol II steps dwell times are assigned to a pause when they lead to a backward step. The corresponding pause ends when a forward step brings Pol II back to the elongation state. For the detected steps this criterion also applies, however unlikely long dwells are also considered as pauses, since the algorithm will not perfectly find all steps present. Given the limited bandwidth (1kHz), high speed (saturating NTP concentration) of the enzyme and noise (standard deviation $\sim 10bp$) in the traces step detection performance should be similar to the fast scenario in our algorithm comparison. One can expect that mostly very fast steps are lost (figure 17), i.e. fused to large steps. On the other hand that means that also short backtracks are likely to be skipped and instead a longer dwell time between two forward steps is returned by the algorithm. Nevertheless, these longer dwells can be identified based on statistical hypothesis testing. Assuming that dwell times of forward stepping $\langle \tau_{\text{forward}} \rangle$ follow an
exponential waiting time distribution, we calculate the mean dwell time of forward steps to estimate the probability distribution.

\[ \langle \tau_{\text{elongation}} \rangle \sim \langle \tau_{\text{forward}} \rangle = \frac{1}{N} \sum_{i=1}^{N} \tau_{\text{forward}} \]  

(43)

Since not all backtracked pauses are discovered, this estimate of \( \langle \tau_{\text{elongation}} \rangle \) also contains longer dwell times at a skipped pause. Thus \( \langle \tau_{\text{forward}} \rangle \) can be larger than \( \langle \tau_{\text{elongation}} \rangle \) and should be taken as an upper bound for the actual mean waiting time.

Furthermore we assume that forward steps obey an exponential distribution of the following form:

\[ p(\tau) = \frac{1}{\langle \tau \rangle} exp\left(-\tau/\langle \tau \rangle\right) \]  

(44)

Under these assumptions we can define a confidence level to discriminate between normal dwell times of elongation and unlikely long dwell times which are caused by undetected backtracks.

The confidence level can be adjusted by comparing recovered pauses to simulated backtracked pauses. A good compromise is found when most of the pauses are recovered and none or only very few of them are wrongly found.

To this end we simulated 10 data sets with stepping rates and sampling frequency of the fast scenario and a computed noise amplitude of \( \sim 6 \text{bp} \). The simulated data is processed by EBS and the paused regions are identified according to the criterion described above. We also identify paused regions by SGVT with a threshold of two standard deviations of the pause peak as described in the methods section. SGVT sometimes returns very short pauses which are not related to simulated ones and are presumably caused by high noise affecting the filtered signal. Thus we exclude pauses smaller than 10ms in the SGVT analysis. Pauses found by EBS were always larger than 10ms and thus there was no need for such an additional post-processing step. For each detected pause we identify if it is a correctly found one by checking whether it coincides with a simulated pause. We also take into account that, either two detected pauses which are close but separated could overlap with a simulated pause, or that a single detected pause could cover two very close but separated simulated ones. Having identified how many pauses are correct, we can compute the recall (i.e. the number of correctly found pauses devided by the number of simulated pauses), the precision (i.e. the number of correctly found pauses devided by the number of found pauses) and the false discovery rate (FDR, i.e. number of wrongly found pauses devided by the number of found pauses). Moreover we compare the total cumulated length of all detected pauses to the total cumulated length of simulated ones. This value is relevant since a correct determination of the total length of pauses is important for determining pause-free velocities which are computed by excluding the paused intervals from the measured data. Table 3 shows mean and standard deviation of recall, precision, FDR and total length for long (\( t > t_p \)) and short pauses (\( t < t_p \)) where the threshold for determining a long pause is \( t_p = 0.8s \) (Results & Discussion). Although in the fast scenario step detection performance is inappropriate for further dwell time analysis, finding pauses still works well, fig. 19 and table 3.
Table 3: Detection of short and long backtracks in simulated data by EBS and SG filter. Shown is the number of correctly detected backtracks divided by the number of simulated backtracks (recall), the number of correctly detected backtracks divided by the number of found backtracked regions (precision) and the false discovery rate (FDR, number of false positives divided by number of found backtracked regions). Moreover, the total length of detected backtracks divided by the total length of simulated backtracks is given. Backtracks with a detected duration $< 10\text{ ms}$ were excluded. The uncertainties for recall, precision and FDR are SEM.

|           | recall/\% | precision/\% | FDR/\% | total length/\% |
|-----------|-----------|--------------|--------|-----------------|
| **short pauses:** |           |              |        |                 |
| SG filter  | 38 ± 7    | 57 ± 7       | 43 ± 8 | 70              |
| EBS        | 61 ± 4    | 98 ± 2       | 8 ± 2  | 91              |
| **long pauses:** |           |              |        |                 |
| SG filter  | 98 ± 1    | 100          | 0      | 94              |
| EBS        | 100       | 100          | 0      | 113             |

Figure 19: Backtracked pause detection in simulated data. Shown is the noisy input signal (circles), the simulated step signal (red), the SG filtered signal (black) and the detected step signal from EBS (blue). Pauses in simulated data are highlighted in green and paused regions in step detected data are indicated by the blue shaded areas.

References

1. Mehta A.D., R.S. Rock, M. Rief, J.A. Spudich, M.S. Mooseker, R.E. Cheney, 1999 Myosin-V is a processive actin-based motor nature 400:590-593
2. Carter N.J., R.A. Cross, 2005 Mechanics of the kinesin step Nature 435, 308-312
3. Abbondanzieri, E.A., W.J. Greenleaf, J.W. Shaevitz, R. Landick, and S.M. Block 2005. Direct observation of base-pair stepping by RNA polymerase. Nature 438: 460465.
4. Galburt, E., 2007, Backtracking determines the force sensitivity of RNAP II in a factor-dependent manner nature 446(7137):820-823
5. Michaelis J., and B. Treutlein, 2013, Single-molecule studies of RNA polymerases Chem. Rev. 113: 8377-8399
6. Yildiz A. and P.R. Selvin, 2005, Fluorescence Imaging with One Nanometer Accuracy: Application to Molecular Motors Acc. Chem. Res. 38(7):574-82.
7. Neuman K.C., and Nagy A., 2008 Single-molecule force spectroscopy: optical tweezers, magnetic tweezers and atomic force microscopy Nature Methods 5, 491 - 505
8. Kolomeisky A.B., M.E. Fisher, 2007, Molecular Motors: A Theorists Perspective Annu Rev Phys Chem 58:675-695
9. Heller I., T. P. Hoekstra, G. A. King, E. J. G. Peterman, and G. J. L. Wuite, 2014 Optical Tweezers Analysis of DNAProtein Complexes Chem. Rev., 114 (6), 30873119
10. Chemla Y.R., K. Aathavan, J. Michaelis, S. Grimes, P.J. Jardine, D.L. Anderson, and C. Bustamante, 2005, Mechanism of Force Generation of a Viral DNA Packaging Motor Cell, 122 (5), 683-692
11. Moffitt J.R., Y.R. Chemla, K. Aathavan, S. Grimes, P.J. Jardine, D.L. Anderson, and C. Bustamante, 2009, Intersubunit coordination in a homomeric ring ATPase nature 457:446-450
12. Chistol, G., S. Liu, C.L. Hetherington, J.R. Moffit, S. Grimes, P.J. Jardine, and C. Bustamante. 2012. High Degree of Coordinationand Division of Labor among Subunits in a Homomeric Ring ATPase Cell, 151:1017-1028
13. McKinney S.A., C. Joo, and T. Ha, 2006, Analysis of Single-Molecule FRET Trajectories Using Hidden Markov Modeling Biophysical Journal 91(5):1941-1951
14. Venkataramanan Land F.J. Sigworth, 2002, Applying Hidden Markov Models to the Analysis of Single Ion Channel Activity Biophys J. 82(4): 19301942.
15. A.L. Drobsheer, 2003, Specificity assessment from fractionation experiments (SAFE): a novel method to evaluate microarray probe specificity based on hybridisation stringencies. Nucleic Acids Research vol. 31
16. Kerssemakers J. W. J., E. L. Munteanu, L. Laan, T. L. Noetzel, M. E. Janson and M. Dogterom, 2006 Assembly dynamics of microtubules at molecular resolution Nature 442, 709-712
17. Milescu L. S., Yildiz A.,Selvin P. R. and F . Sachs, 2006 Extracting Dwell Time Sequences from Processive Molecular Motor Data Biophys Journal 91(9): 31353150
18. Carter, B. C., M. Vershinin, and S. P. Gross. 2008. A comparison of step-detection methods: how well can you do? Biophys. J. 94: 306319.
19. Milescu LS, A. Yildiz, S.R. Selvin, F. Sachs, 2006, Maximum likelihood estimation of molecular motor kinetics from staircase dwell-time sequences. Biophysical Journal 91(4):1156-68
20. Müllner, F . E., S. Syed, ., F . J. Sigworth. 2010. Improved hidden Markov models for molecular motors, part 1: basic theory. Biophys. J. 99:36843695.
21. Forney, G.D . The viterbi algorithm Proceedings of the IEEE Vol. 61 Issue:3, 268 - 278
22. Little M.A., N.S. Jones, 2011,Generalized methods and solvers for noise removal from piecewise constant signals: Part I Background theory Proceedings of the Royal Society A, 467(2135):3088-3114, doi: 10.1098/rspa.2010.0671
23. Little, M. A., B. C. Steel, Fan Bai, Y. Sowa, T. Bilyard, D. M. Mueller, R. M. Berry, and N. S. Jones 2011 Steps and Bumps: Precision Extraction of Discrete States of Molecular Machines Biophys. J. 101:477-485
24. Boykov, Yuri, Olga Veksler, and Ramin Zabih. "Fast approximate energy minimization via graph cuts." Pattern Analysis and Machine Intelligence, IEEE Transactions on 23.11 (2001): 1222-1239.
25. Dangkulwanich M., T. Ishibashi, S. Liu, M. L Kireeva, L. Lubkowska, M. Kashlev, C.J. Bustamante, 2013, Complete dissection of transcription elongation reveals slow translocation of RNA polymerase II in a linear ratchet mechanism. eLife 2013;2:e00971
26. Rudin, Leonid I., Stanley Osher, and Emad Fatemi. "Nonlinear total variation based noise removal algorithms." Physica D: Nonlinear Phenomena 60.1 (1992): 259-268.
27. Boyd, Stephen, and Lieven Vandenberghe. "Convex optimization." Cambridge university press, 2004.
28. Vogel, Curtis R., and Mary E. Oman. "Iterative methods for total variation denoising." SIAM Journal on Scientific Computing on 17.1 (1996): 227-238.
29. Beck, Amir, and Marc Teboulle. "Fast gradient-based algorithms for constrained total variation image denoising and deblurring problems." Image Processing, IEEE Transactions on 18.11 (2009): 2419-2434.
30. Condat, Laurent. "A Direct Algorithm for 1-D Total Variation Denoising." Signal Processing Letters, IEEE on 20.11 (2013): 1054-1057.
31. Rockafellar, R. Tyrrell. "Convex analysis". No. 28. Princeton university press, 1997.
32. Li, Stan Z. "Markov random field modeling in image analysis”. Vol. 26. London: Springer, 2009.
33. Ishikawa, Hiroshi. "Exact optimization for Markov random fields with convex priors." Pattern Analysis and Machine Intelligence, IEEE Transactions on 25.10 (2003): 1333-1336.
34. Kolmogorov, Vladimir, and Ramin Zabin. "What energy functions can be minimized via graph cuts?.” Pattern Analysis and Machine Intelligence, IEEE Transactions on 26.2 (2004): 147-159.
35. Vladimir Kolmogorov. "Graph Based Algorithms for Scene Reconstruction from Two or More Views" PhD thesis, Cornell University, September 2003.
36. Papadimitriou, Christos H., and Kenneth Steiglitz. "Combinatorial optimization: algorithms and complexity”. Courier Dover Publications, 1998.
37. Y. Boykov, V. Kolmogorov. "An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision.” Pattern Analysis and Machine Intelligence, IEEE Transactions on 26(9):1124-1137, Sep 2004.
38. A. Delong, A. Osokin, H. N. Isack, Y. Boykov. "Fast Approximate Energy Minimization with Label Costs.” In CVPR, June 2010.
39. Kolmogorov, Vladimir, and Carsten Rother. "Minimizing nonsubmodular functions with graph cuts-a review.” Pattern Analysis and Machine Intelligence, IEEE Transactions on 29.7 (2007): 1274-1279.
40. Lovsz, L. "Submodular functions and convexity.” Mathematical Programming The State of the Art. Springer Berlin Heidelberg, 1983. 235-257.
41. Potts, Renfrey Burnard. "Some generalized order-disorder transformations.” Mathematical Proceedings of the Cambridge Philosophical Society on Vol. 48. No. 01. Cambridge University Press, 1952.
42. Carsten Rother, Sanjiv Kumar, Vladimir Kolmogorov, and Andrew Blake "Digital tapestry [automatic image synthesis].” Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on. Vol. 1. IEEE, 2005.
43. Winkler, Gerhard. Image analysis, random fields and Markov chain Monte Carlo methods: a mathematical introduction. Vol. 27. Springer, 2003.
44. Brooks, S. P., and B. J. T. Morgan. "Optimization using simulated annealing." The Statistician (1995): 241-257.
45. Komissarova N., M. Kashlev, 1997, RNA Polymerase Switches between Inactivated and Activated States By Translocating Back and Forth along the DNA and the RNA. J. Biol. Chem. 272:15329-15338.
46. Shaeftiz J.W., E.A. Abbondanzieri, R. Landick, and S.M. Block, 2003, Backtracking by single RNAPolymerase molecules observed near-base-pair resolution nature vol. 426:684-687
47. D.T. Gillespie, 1977, Exact Stochastic Simulation of Coupled Chemical Reactions The Journal of Physical Chemistry, Vol. 81, No. 25
48. Kloeden, P.E., and E. Platen. 1992. Numerical Solution of Stochastic Differential Equations. Springer Science & Business Media.
49. Boustamante C, J.F . Marko, E.D. Siggia, S. Smith, 1994, Entropic Elasticity of λ-Phage DNA Science 265:1599-1600
50. David L. Donoho, et. al., Observed Universality of Phase Transitions in High-Dimensional Geometry, with implications for modern data analysis and signal processing, Phil. Trans. R. Soc. A 13 November 2009 vol. 367 no. 1906 4273-4293
51. David L. Donoho, et. al., The Noise-Sensitivity Phase Transition in Compressed Sensing, IEEE Transactions on Information Theory. Letters, Vol. 57
52. Neuman, K.C., 2003, Ubiquitous Transcriptional Pausing Is Independent of RNA Polymerase Backtracking.
35. Siek, Jeremy G., Lie-Quan Lee, and Andrew Lumsdaine. "Boost Graph Library: User Guide and Reference Manual", The. Pearson Education, 2001.
36. Larson, M.H., J. Zhou, C.D. Kaplan, M. Palangat, R.D. Kornberg, R. Landick and S.M. Block, 2012 Trigger loop dynamics mediate the balance between the transcriptional fidelity and speed of RNA polymerase II PNAS
37. Chib, Siddhartha, and Edward Greenberg. "Understanding the metropolis-hastings algorithm." The American Statistician 49.4 (1995): 327-335.
38. Moffit J.R., C. Bustamante, 2014, Extracting signal from noise: kinetic mechanisms from a Michaelis-Menten-like expression for enzymatic fluctuations FEBS Journal 281, 4985-17
39. Moffit, J.R., Y.R. Chemla, D. Izhaky, and C. Bustamante. Differential detection of dual traps improves the spatial resolution of optical tweezers PNAS
40. Marko J.F., Siggia E.D. 1995. Stretching DNA, Macromolecules 28:8759-8770
41. Depken M., E.A. Galburt, S.W. Grill 2009, The Origin of Short Transcriptional Pauses Biophysical Journal Volume 96 21892193
42. Robert L.B., M. Fulbright, M. D. Wang, 2007 Mechanochemical Kinetics of Transcription Elongation Phys. Rev. Let. 98, 068103
43. Kirkpatrick S., C. D. Gelatt, Jr., M. P. Vecchi, 1983, Optimization by Simulated Annealing Science Vol. 220, Nr. 4598
44. Bauschke, Heinz H., and Patrick L. Combettes. "Convex analysis and monotone operator theory in Hilbert spaces." Springer, 2011.
45. Rose, Donald J. "An algorithm for solving a special class of tridiagonal systems of linear equations." Communications of the ACM 12.4 (1969): 234-236.
46. Boros, Endre, P. L. Hammer, and X. Sun. "Network flows and minimization of quadratic pseudo-Boolean functions", Vol. 4, No. 8. Technical Report RRR 17-1991, RUTCOR, 1991.
47. Hammer, Peter L., Pierre Hansen, and Bruno Simeone. "Roof duality, complementation and persistency in quadratic 01 optimization." Mathematical programming 28.2 (1984): 121-155.
48. Izzak, Hossam, and Yuri Boykov. "Energy-based geometric multi-model fitting." International journal of computer vision 97.2 (2012): 123-147.
49. Pfützer E., C. Wachauf, F. Kilchherr, B. Pelz, W.M. Shih, M. Rief and H. Dietz, 2013 Rigid DNA Beams for High-Resolution Single-Molecule Mechanics Angewandte Chemie Vol 52, Issue 30, 77667771
50. Donald L. Ermak and J. A. McCammon, 1978 Brownian dynamics with hydrodynamic interactions J. Chem. Phys. 69, 1352; doi: 10.1063/1.436761
List of Figures

1. Flowchart of the two stage process for finding steps in noisy stepping data. The input data is first denoised via solving the convex TVDN problem. This process requires no intervention, as the regularization parameter \( \lambda \) is determined automatically. This results in a lower dimensional, compressed representation of tuples (amplitude, length). This discrete data is then handed to a Graph Cut algorithm which solves a combinatorial optimization problem on a graph. The Graph Cut allows further customization by the use of regularization parameters \( \rho_i \) and a pre-defined level set.

2. Breakdown of the TVDN model dependent on \( \lambda/\lambda_{max} \). The number of produced steps and plateaus vs. different \( \lambda \). For small \( \lambda/\lambda_{max} \) solving the TVDN problem reproduces the input signal and the number of steps equals the number of data points. For big \( \lambda/\lambda_{max} \) the number of steps is significantly lower. The point \( \lambda_h/\lambda_{max} \) (black) before the number of steps increases suddenly marks the value of the TVDN regularization parameter that we choose in our heuristic. Plotted is a constant signal with added gaussian white noise (red), a signal with exponentially distributed dwell times and gaussian white noise (green), and the same signal without white noise (blue).

3. Cartoon representation of the graph cut algorithm. 3a) the initial graph structure we use to model the step signal. The nodes \( i \in \{1...M\} \) represent the variables \( \xi_i \). 3b) in a second step, energies are mapped to capacities of edges. 3c) the Boykov-Kolmogorov Graph Cut algorithm (35) finds the max flow and cuts the graph into two subgraphs \( \mathcal{G} = S \cup T \) where \( S \) is the part connected to \( s \) and \( T \) is the remaining part connected to \( t \).

4. Level prior potential depending on level difference \( \xi_i - \xi_{i+1} \) designed for a stepping process of known step size.

5. EBS algorithm correctly detects steps in presence of high noise. 5a) noise reduction after application of TVDN. 5b) step detection using combinatorial clustering. Shown is a zoomed in interval of the simulated noisy data (grey points), simulated steps (blue), denoised signal from TVDN (magenta, 5a) and detected steps after application of combinatorial clustering by Graph Cut (red, 5b).

6. Performance of step detection algorithms with respect to slow scenario (blue bars), intermediate scenario (green bars) and fast scenario (dark red bars). (6a) shows in percent of the total number of simulated steps the number of detected steps. (6b) percentage of correct steps among the simulated steps. Errorbars are SEM. (6c) shows average step size histograms with 1bp binning of the detected steps of the fast scenario for t-test, HMM and EBS (from upper to lower histogram).

7. (a) 2.5bp substeps in \( \varphi29 \) bacteriophage data (circles) measured in an optical tweezers experiment, EBS (blue) and t-test (red). (b) Paused regions in experimental Pol II transcription elongation data. Shaded regions indicate paused intervals found by the SG-filtering method with a velocity threshold of two standard deviations of the pause peak (green) and EBS (red). 1kHz sampled transcription data (grey), SG filtered data of polynomial order 3 and frame size: 2.5s (cyan) and step detection result of our method (blue). (c) Step size histogram of detected steps by EBS. (d) Zoom into a detected pause. Shown is EBS signal (blue), SG filtered signal (black), measured data at 1kHz (black circles) and paused regions (SG: green, EBS: red).
8 Over fitting transition depends on sampling frequency. Each step signal has 1016 poissonian distributed steps sampled with different frequencies and covered by noise. The signals are 100s long. .................................................. 22
9 Influence of SNR on over fitting transition. Each step signal has 980 poissonian distributed steps with different noise amplitudes at 6kHz. Every signal is 100s long and consists of 6·10^5 data points. .................................................. 23
10 Situation in an MRF concerning a single variable \( v_i \) and edges \( A, B \) to special variables \( s \) and \( t \) relevant for the data term. .................................................. 23
11 Two neighbouring variables in an MRF and edges relevant for the pairwise term. Here the two \( s \) vertices represent the same vertex in the graph and are just drawn seperated to make the diagram look nicer. The same is true for the \( t \) vertices. .................................................. 23
12 Mean run time performance of combinatorial optimization step for 10 simulated noisy step signals. 12a Graph Cut computation time versus number of tupels for a label grid of 800 levels. Second order polynomial fit to computation times (red curve). 12b Graph Cut computation time versus label grid size for a fixed system size of 750 tupels. Linear scaling of performance with increasing number of levels in the label grid set. The error bars are SEM. .................................................. 25
13 Graph Cut MCMC comparison. Energies of optimal solutions with increasing system size for MCMC and Graph Cut algorithms. .................................................. 27
14 Step and noise simulation procedure. State transition model and corresponding stepping rates determine the probability distribution from which a PWCS (first inset) is sampled. In a second step noise is simulated with the PWCS as the mean (second inset). .................................................. 28
15 Simplified stepping model of RNAP II with an elongation and backtracked states. .................................................. 28
16 Histogram of step sizes of the easy scenario 16a and the intermediate scenario 16b for t-test (blue), HMM (green) and EBS (red). .................................................. 30
17 Cumulative fraction of found steps for the EBS in the intermediate scenario. Plotted is the cumulative number of found steps/simulated steps of the signal plotted in Fig.(5) for different dwell time times. The number of detected steps compared to simulated ones is smaller for short dwell time steps. Dwell time histograms with a binning of 87.3ms were determined for the detected and simulated steps respectively. The number of detected steps for each dwell time was devided by the corresponding number of simulated steps and cumulatively summed up. In total 60.5% of the number of simulated steps were found. .................................................. 31
18 Prior terms and level grid regularize combinatorial clustering. 18a Relative frequencies of correct steps among the number of detected steps (precision) as a function of prior potential strength \( 1/\rho_S (\rho_S/\rho_P = 0.08 \) is kept constant) for simulated data using the intermediate scenario. Shown is the precision for clustering with a level grid of 1/4bp spacing (black triangles) and with a spacing of 1bp (grey squares). 18b step size histograms of detected steps with a label grid of 1/4bp without prior terms (blue), with a spacing of 1/4bp and prior terms (green) and with a spacing of 1bp and prior terms of the same strength (red). The computed precision corresponding to the three histograms is encircled with the respective color, Fig.(18a). .................................................. 32
19 Backtracked pause detection in simulated data. Shown is the noisy input signal (circles), the simulated step signal (red), the SG filtered signal (black) and the detected step signal from EBS (blue). Pauses in simulated data are highlighted in green and paused regions in step detected data are indicated by the blue shaded areas. .................................................. 34
List of Tables

1. Comparison of computation efficiency of the different step-finding algorithms. \(\sim 900\) simulated steps on commodity hardware (i7-2600, \(3.6GHz\) CPU Ubuntu System, \(4GB\) memory). Corresponding run times were recorded in matlab for the signal of the given size. Peak memory usage, i.e. resident set size (RSS) was measured with Linux’s proc information system.

2. Overview of simulation parameters. Shown is elongation rate \(k_{\text{elong}}\), corresponding NTP concentration \(c_{\text{NTP}}\) and rate constants of the backtracking state. Moreover, the standard deviation of noise \(\sigma_n\), sampling frequency \(f\) and number of data points \(N\) is given.

3. Detection of short and long backtracks in simulated data by EBS and SG filter. Shown is the number of correctly detected backtracks divided by the number of simulated backtracks (recall), the number of correctly detected backtracks divided by the number of found backtracked regions (precision) and the false discovery rate (FDR, number of false positives divided by number of found backtracked regions). Moreover, the total length of detected backtracks divided by the total length of simulated backtracks is given. Backtracks with a detected duration \(< 10ms\) were excluded. The uncertainties for recall, precision and FDR are SEM.