Unsupervised trajectory inference using graph mining

Leen De Baets[1,2], Sofie Van Gassen[1,2], Tom Dhaene[1], and Yvan Saeys[2,3]

[1] Internet Based Communication Networks and Services (IBCN), Ghent University - iMinds, Gaston Crommenlaan 8 (Bus 201), B-9050 Gent, Belgium
[2] Data mining and Modelling for Biomedicine (DaMBi) ,VIB Inflammation Research Center, Ghent, Belgium
[3] Department of Internal Medicine - Ghent University, Ghent, Belgium

Abstract. Cell differentiation is a complex dynamic process and although the main cellular states are well studied, the intermediate stages are often still unknown. Single cell data (such as obtained by flow cytometry) is typically analysed by clustering the cells into distinct cell types, which does not model these gradual changes. Alternative approaches that explicitly model such gradual changes using seriation methods seems promising, but are only able to model a single differentiation pathway. In this paper, we introduce a new, graph-based approach that is able to model multiple branching differentiation pathways as continuous trajectories. Results on synthetic and real data show that this is a promising approach which is moreover robust to parameter changes.

1 Scientific Background

Flow cytometry offers a high-throughput platform for single cell analysis. Typically, 10 to 20 different cell surface proteins (markers) are stained with fluorochromes, enabling their detection using laser-based systems on millions of individual cells during an experiment. Traditional analysis of flow cytometry data is done by examining two-dimensional scatter plots in order to identify distinct cell populations.

However, the amount of possible scatter plots increases exponentially with the number of proteins measured, which makes it infeasible to examine them all. To alleviate this problem, alternative visualisation techniques such as SPADE [1], Visne [2] and FlowSOM [3] have been proposed. These map the multidimensional data to two-dimensional plots, incorporating the similarities of all marker dimensions. Figure 1 shows the result of a FlowSOM analysis on flow cytometry data concerning hematopoietic stem cells differentiating into common myeloid and lymphoid progenitors. While the different cell types are represented in different branches and the corresponding median fluorescence intensities for all markers are indicated, we do not infer any information about their developmental trajectory.

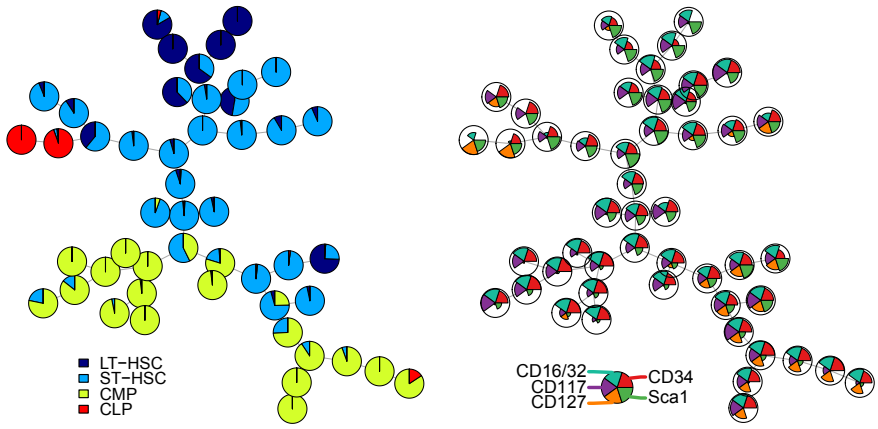


Fig. 1: FlowSOM representation of selected bone marrow cells of a wildtype mouse. Each circle represents a group of similar cells. On the left, the manual gating annotations are visualized in pie charts. The different branches in the tree correspond roughly to the different cell types. On the right, the median fluorescence intensities are indicated. Slight variations in marker intensities for a single cell type are discernable, but one cannot infer the known developmental process: the long-term hematopoietic stem cells (LT-HSC) differentiate into short-term hematopoietic stem cells (ST-HSC) and these can in turn differentiate into either common myeloid progenitor cells (CMP) or common lymphoid progenitor cells (CLP).

A schematic example of cell differentiation is given in Figure 2(a) where cells start in an immature state (state 1) and evolve through an intermediate state (state 2) to finally result in two distinct mature cell types (either state 3 or 4). While transitioning from one state to another, certain aspects of the cells change, for example increasing values of two markers when the immature cells differentiate to the intermediate state (Figure 2(b)). As cells evolve in a continuous manner with many cells being in different states, a single snapshot of the developmental system can be presented in Figure 2(c), showing only two markers for simplicity. As there is still a lot of uncertainty about the developmental trajectories that cells follow, it would be interesting to infer such trajectories automatically from the data without using any external information about the intermediate stages. This results not only in information about the differentiation state of the cells, but can also present a concise overview of marker behaviour during the differentiation, thereby providing novel hypotheses about cell differentiation and possibly revealing new intermediate cell stages.

Inferring a temporal ordering is not limited to deduce the developmental chronology of cells, and occurs in many other situations, such as reconstruc-

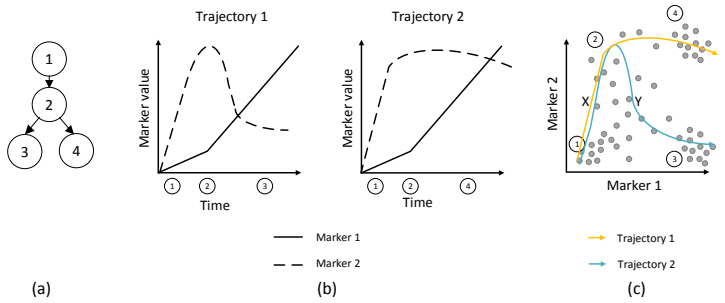


Fig. 2: This figure illustrates the concept of trajectory modelling where (a) shows a schematic overview of the developmental differentiation, (b) the marker changes during development, and (c) a snapshot of the cells following trajectories which are indicated with the coloured lines.

tion of temporal ordering of biological samples using microarray data [4]. Other related fields are seriation and ordination, which also try to find an ordering without explicitly using the order-defining property (such as time). Seriation was developed to chronologically order archaeological artefacts from numerous sites belonging to the same culture [5] and ordination orders objects so that similar objects are near each other and dissimilar objects are farther from each other, often applied on the field of community ecology [6].

The inference of differentiation pathways is a relatively new reasearch. On the one hand, there are methods working on expression data such as Monocle [7] that is an unsupervised algorithm that can infer temporal ordering in single cells based on their their expression profiles. At the other hand, there are methods working on flow cytometry data such as Wanderlust [8] that will be explained in the following section. This is the first paper that proposes a method for identifying multiple differentiation pathways in flow cytometry data.

2 Materials and Methods

In this work, we present a new computational method to identify multiple trajectories given a flow cytometry dataset. As input, we use the measured data plus a single cell selected to represent the most immature cell state and a cell selected for each mature cell state. Our algorithm forms a wrapper around the Wanderlust algorithm [8], which is capable of detecting a single trajectory where all cells must differentiate to the same mature cell type. Our method can be easily adapted to use any other approach that is capable of automatically detecting a single trajectory.

In Section 2.1, we explain shortly how Wanderlust constructs a single trajectory. In Section 2.2, we formulate our solution to find multiple trajectories in unordered data.

2.1 Inference of a single trajectory

We define a trajectory as an ordering of cells such that the cells are sorted according to their developmental order: immature cells should thus be sorted before mature cells. It is important to note that the developmental distance often does not correspond with Euclidean distance: cells that are developmentally close will typically be close in Euclidean distance as well, but the opposite is not necessarily true. An example is given in Figure 2(c) by elements X and Y. Although they are close in Euclidean distance, X is still differentiating towards the intermediate state, while Y is much closer to the mature state. The Wanderlust algorithm solves this by representing the data as a k-nearest neighbour (knn) graph such that only developmentally close cells are connected. Due to noise, it is still possible that cells get connected even though they are developmentally far apart. To handle these short circuits, an ensemble is created consisting of m graphs where in each graph each node contains only $l(< k)$ edges that are randomly chosen from the k edges present in the knn graph, resulting in m l-knn graphs.

Wanderlust creates a trajectory by ordering all cells according to their similarity to the cell selected as the most immature (the start cell). This similarity is defined as the distance in the l-knn graph and is calculated for each graph in the ensemble, resulting in l trajectories. The final trajectory is obtained by averaging each cell's distance across all m l-knn graphs.

The running time of this algorithm is dominated by the creation of the knn graph which takes $\mathrm{O}(N^2)$ if the amount of cells is N. Thus if the amount of cells increases linearly, then the running time increases quadratically.

The most restrictive assumption of this algorithm is that all cells must follow one and the same trajectory. This is a necessary assumption as the cells are ordered with respect to their distance to the starting cell of the trajectory. In the following section, we will propose an algorithm that can infer multiple trajectories.

2.2 Inference of multiple trajectories

An overview of our algorithm to infer multiple trajectories is given in Figure 3. As input, we do not only use a start cell representing the most immature cell, but also one end cell for each mature state. Using this information, we infer a trajectory for each end cell. First, we assign each cell in the dataset to one or more trajectories. Once this is done, we apply the original Wanderlust algorithm

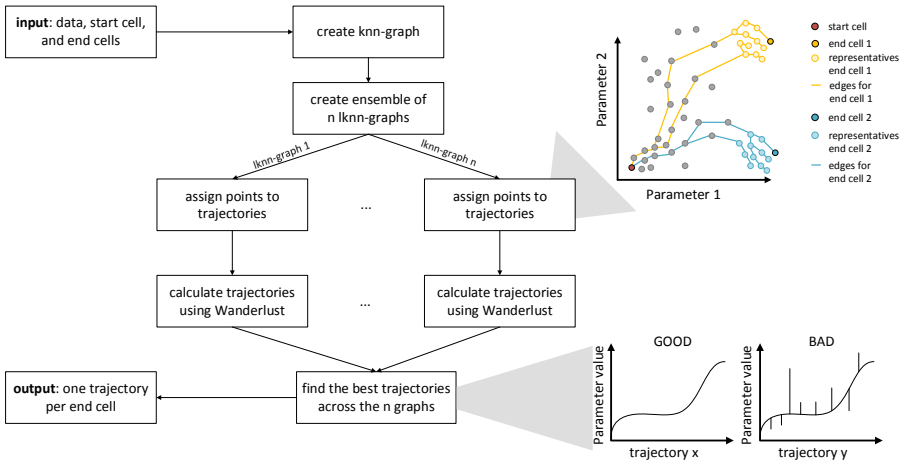


Fig. 3: This figure illustrates the proposed algorithm with a flow chart. The steps concerning the assignments of points and finding the best trajectories are elaborated in more detail with a figure.

on all cells assigned to one trajectory and aggregate the results in one final trajectory per mature state.

Assigning cells to a trajectory is done by clustering the data with k-means clustering using a large amount of clusters (overclustering). In these clusters, we find the ones to which the given end cells belong. This allows us to represent each mature state with not only one end cell, but with all the cells in the cluster: the representatives. In the next step, we determine for each representative the edges of the shortest path to the starting cell, resulting in a collection of edges for each mature state. Then for every cell, we determine the edges of the shortest path to the start cell and compare it to the edge collections of the mature states, resulting in the detection of the most similar collection. This comparison is done by determining the amount of overlap. Knowing the most similar collection, we assign the cell to the trajectory going from the starting cell to the selected end cell. When the overlap is the same for different edge collections, the cell is assigned to multiple trajectories.

Due to the ensemble of graphs, we have m possible trajectories for each mature state. To choose the best trajectory without using extra information, we add a regularisation procedure which assumes that changes between the states in the trajectory should be gradual. Due to this property, the curves representing the parameter values of the cells along the trajectory (Figure 2(b)) must be smooth. More specific, if these curves contain large jumps (see Figure 3), there is a high probability that cells are missing or that the ordering is wrong. Concretely, we calculate the smoothed curves using a median filter and calculate a score as

the difference between these curves and their smoothed version. The difference between a curve c and his smoothed version $\hat{c}$ is defined as

$$\sum_i \mid c_i - \hat{c_i} \mid$$

The trajectory with the smallest score is chosen as it differs the least from its smoothed version and thus the assumption that changes occur gradual is satisfied. This results in one trajectory per mature state. Note that indeed no extra information is used.

The running time of this algorithm is also dominated by the creation of the knn graph which takes $\mathrm{O}(N^2)$ if the amount of cells is N. Thus if the amount of cells increases linearly, then the running time increases quadratically.

The parameters for our algorithm are the amount of neighbours k for creating the knn graph, the amount of randomly selected edges l out of the k edges to reduce the effect of short circuits, and the amount of times m this sampling must happen, resulting in m l-knn graphs. For assigning points to a trajectory, the amount of clusters cl must be given.

3 Results

We used synthetic data to evaluate our algorithm and perform a sensitivity analysis for the different parameters. Additionaly, we evaluted our algorithm on real flow cytometry data.

3.1 Synthetic data

The synthetic data was created using Bézier curves representing an artificial branching structure in a three-dimensional space. We have chosen six points A, B, C, D, E and $F \in \mathbb{R}^3$. The structure starts at point A going to B where it branches to C and endpoint D. From C, it branches then further to the endpoints E and F (Figure 4(a)). All these connections are created using a quadratic Bézier curve:

$$\mathbf{B}(t) = (1-t)\left[(1-t)\mathbf{P}_0 + t\mathbf{P}_1\right] + t\left[(t-1)\mathbf{P}_1 + t\mathbf{P}_2\right], t \in [0,1] \qquad (1)$$

where P_0 and P_2 are two of the six chosen points that need to be connected in the branched structure (e.g. point B and C) and P_1 is a random point. We thus created a branched trajectory with three endpoints, as shown in Figure 4(b). Note that for each endpoint we have a trajectory that starts in A, e.g. the trajectory for endpoint E traverses A, B, C and E. Another way to visualise this branched trajectory is shown in Figure 4(c). Here, one plot is used for each trajectory and it visualises the parameter values (y-axis) of the points along the trajectory (x-axis), one curve for each feature. As we work in a three dimensional

space (amount of parameters = 3), this results in three curves on one plot. The smoothness of these curves is used to define a good trajectory.

From each curve, we uniformly sample 100 points. Noise is added by perturbing each point,

$$\mathbf{P} = \mathbf{P} + int * \mathbf{n}, int \in [0, 1] \tag{2}$$

where $\mathbf{n}$ is a noise vector ($\in \mathbb{R}^3$) containing numbers randomly sampled from a Gaussian distribution with $\mu = 0$ and $\sigma = 1$. The number int represents the intensity of the noise. It is not realistic to assume that during each stage in a trajectory the same amount of cells is available. Rather, we will have states that are very well represented and states that are rare because the cells evolve through them quickly. We model this with a random density functions f_{dens}. For each point, we generate x noisy variants, with x depending on f_{dens}.

$$x(\mathbf{P}) = f_{dens}(\mathbf{P}) \tag{3}$$

For each curve in the branched structure, we define one f_{dens} by randomly choosing four points in the range $]0, 20]$ and by interpolating these with cubic splines. We make sure that at the connection points (B and C) the densities are the same for all relevant trajectories. An example of the density functions for each trajectory is given in Figure 4(d) resulting in 3984 points representing the branched trajectory.

Evaluation. The main advantage of using synthetic data is that we know the exact underlying trajectories without noise. As such, a ground truth is available indicating which points belong to which trajectories and how they must be ordered. This underlying ground truth allows us to define two measures. First, we check if the points are assigned to the correct trajectories. From this comparison we extract the True Positive Rate (TPR or the sensitivity) and the False Positive Rate (FPR) for each trajectory, defined as:

$$TPR = \frac{TP}{TP + FN}, FPR = \frac{FP}{FP + FN} \tag{4}$$

where TP are the true positives, or points that are present in the calculated and ground truth assignments, FP the false positives or the points that are present in the calculated assignment but not in the ground truth assignment, FN the false negatives or the points that are not present in the calculated assignment but should be. The TPR must be as close to one as possible, and the FPR as close to zero.

A second criterion checks for each trajectory how well the TP are ordered according to the ground truth ordering, so we check if the points that are assigned correctly are ordered correctly. To this end, we calculate the Spearman rank correlation coefficient.

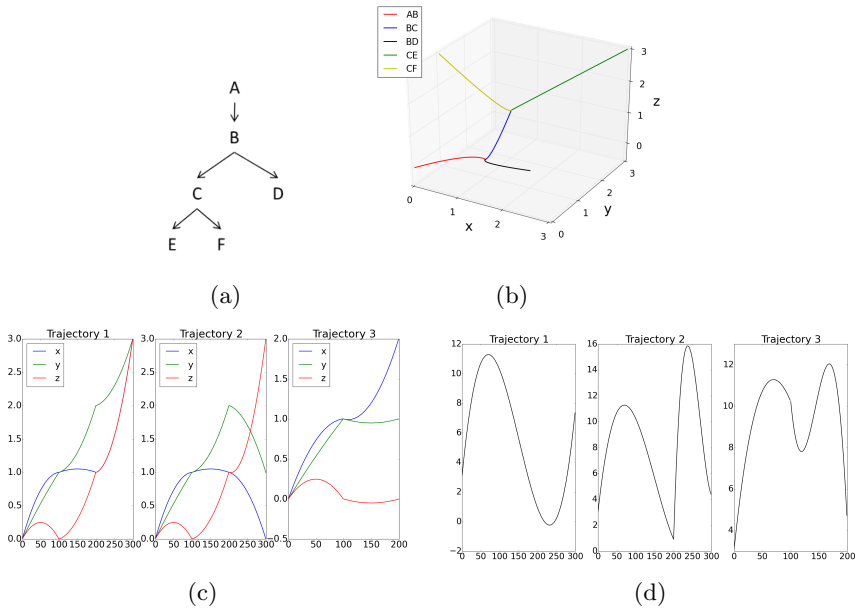


Fig. 4: This figure shows (a) a schematic branched trajectory with three endpoints, (b) the different functions $\mathbf{B}(t)$ following the branched structure in (a), (c) an alternative visualisation which shows a curve following the feature values (y-axis) of the points along the trajectory (x-axis) for each feature for each trajectory , and (d) the densities for each trajectory used for sample representation.

Parameter Sensitivity. As the described algorithm has a number of parameters, the robustness of the algorithms' output needs to be investigated by comparing it for different parameter settings. In the following, we change the amount of clusters cl, the amount of edges k and l, and the starting and ending points. We do not change the amount of graphs m as we assume that if this is sufficiently large, the best trajectory will be detected no matter how many graphs are created ($m = 50$ is sufficient). We perturbed the data with 15% noise. Increasing the noise to 20% does not pose a problem, but for higher values the results deteriorate.

Sensitivity to the amount of clusters. While changing the amount of clusters, the amount of representatives for the endpoints changes. The smaller the amount of clusters, the higher the amount of points assigned to each cluster, thus the higher the amount of points representing an endpoint. If there are a lot of representatives, it is likely that a point belonging to another trajectory is incorrectly assigned to the cluster resulting in wrong assignments of points to the trajectory going to this endpoint. As the change in the amount of clusters only affects the

assignment of objects to trajectories it suffices to calculate the TPR and FPR.

In our test case, we set $k = 20$ and $l = 15$, and let the number of clusters vary between $\{4, 10, 20, 30, 40, 3950$ (amount of points in $V)\}$. The last case is equivalent to the case when there are no clusters. The TPR and FPR for each trajectory in function of the amount of clusters are shown in Figure 5(a). These indicate that when using too few clusters, this leads to the scenario described above which is visible by the fact that when using 4 clusters all elements belonging to trajectory 3, are added to trajectory 3 (TPR = 1), but at the same time we notice that the FPR is high, meaning that points belonging to trajectory 1 and 2 are also added to trajectory 3. If we plot the trajectories in the 3-dimensional space (Figure 5(b)), we indeed see that a lot of points are erroneously assigned to trajectory 3, and consequently a lot of points are missing in trajectory 1 and 2, resulting in a corresponding low TPR. When using enough clusters ($cl \geq 10$), we see that the result varies with the best being achieved when $cl = 20$. We can also see the big advantage of using clusters when comparing with the case without clusters ($cl = 3950$), namely for trajectory 1 and 2 the TPR stays approximately the same but the FPR decreases when no clusters are used. For trajectory 3 it remains approximately the same. When plotting the trajectories in the 3-dimensional space for both cases (5(c) and (d)), we see this difference: trajectory 1 and 2 improve when clusters are used, as less points from respectively branch CF and CE are wrongly assigned to them, and for trajectory 3 points are wrongly assigned whether or not clusters are used, explaining the permanent high FPR. However, when clusters are used, these wrongly assigned points are more spread across other branches. In general, the FPR for trajectory 3 is high for every possible amount of clusters indicating this is a difficult branch to detect. For the following test cases we use $cl = 20$. The usage of clusters doesn't seem to be advantageous, but their use will be justified when using real data.

Sensitivity to k and l. When changing k and l, this has consequences on both the assignment of points to a trajectory and the execution of Wanderlust. In [8], it is stated that Wanderlust is robust against changes in the parameters k and l. We recheck this by using our second evaluation metric, and the robustness of our own assignment algorithm is checked using the first evaluation metric.

In our test case, we set $cl = 20$ and let k vary between $\{10, 20, 50, 100\}$ and l between $\{5, 10, 15, 20\}$ where we guarantee that $l < k$. The Spearman rank correlation coefficient, the TPR and FPR in function of k and l are shown in Figure 6. When comparing for each trajectory, we see that a low TPR corresponds to a low Spearman rank correlation coefficient. This is logical as determining the real ordering using only a few points (low TPR) is hard, resulting in a low correlation coefficient. Otherwise, the Spearman rank correlation coefficient remains high indicating the robustness of the Wanderlust algorithm against changes in k and l.

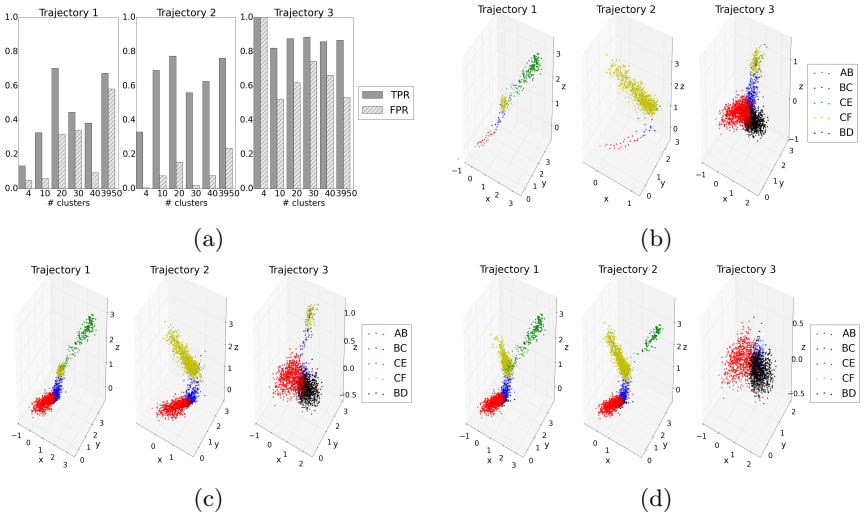


Fig. 5: This figure shows (a) the TPR and FPR concerning the assignments of points to the three different trajectories when $k = 20,$ and $l = 15$, (b) the three trajectories in the 3-dimensional space when $k = 20, l = 15,$ and $cl = 4$, (c) $k = 20, l = 15,$ and $cl = 20$, and (d) $k = 20, l = 15,$ and $cl = 3950$.

Sensitivity to start and endpoints. In the previous investigations, we assumed the real starting and endpoints to be given a priori. When working with synthetic data, this can be given without problem. But when working with real data, it is difficult to know exactly which points are the start and endpoints, i.e. know which points have no predecessor or successor in the trajectory. It is thus likely that a point with predecessors is given as starting point and points with successors as endpoints. To check if the algorithm can handle this scenario, we changed the real starting and end points. This is done by taking as start point, the point that is normally ordered as 30th, and as endpoints we pick the

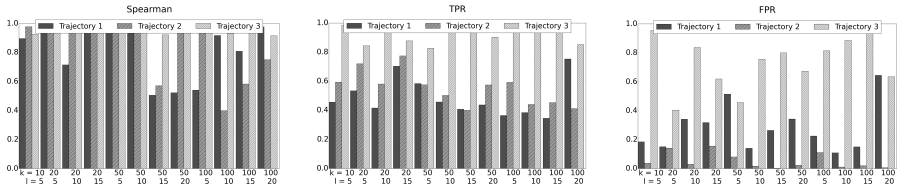
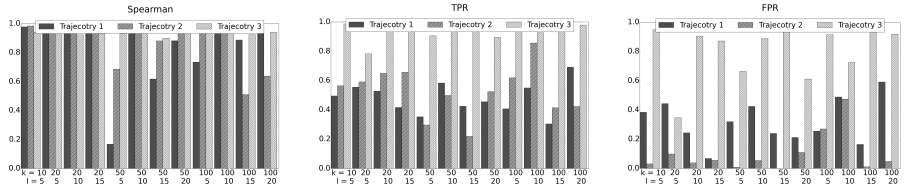


Fig. 6: This figure shows (a) the Spearman rank correlation coefficient, (b) the TPR and (c) the FPR when $cl = 20$.



(a) (b) (c)

Fig. 7: This figure shows (a) the Spearman rank correlation coefficient and (b) the TPR and (c) FPR when $cl = 20$ and the start point and ending points are altered.

points that are normally ranked 30 places before the end.

The results are given in Figure 7. If we compare these with the results obtained with the real endpoints (Figure 6), we see that the Spearman rank correlation coefficient is as good, and that the TPR values range between the same intervals, but higher FPR values are more frequent. E.g. for the case when $k = 20, l = 15$ there is clearly an increase in the FPR in Figure 7(b) when comparing to Figure 6, and at the same time we also note that the corresponding TPR increases. This translates into the fact that more points are assigned to the trajectory, both good and wrong ones.

3.2 Real data

Context. We evaluated our algorithm on the dataset visualized in Figure 1, containing 4647 bone marrow cells. These are manually analysed and are known to differentiate from long-term hematopoietic stem cells (LT-HSC) into short-term hematopoietic stem cells (ST-HSC), which can in turn differentiate into either common myeloid progenitor cells (CMP) or common lymphoid progenitor cells (CLP). The cells can thus follow two trajectories, both of which are expected to include the hematopoietic stem cells. Our data set includes measurements for five surface markers: CD34, CD16/CD32, CD117, CD127, and Sca-1. Typically only a few of these marker values are used to define each of the distinct cell types, indicated in Figure 8(a). By forming a trajectory from the unordered data, it can be checked if the known feature value changes correspond, and which other changes happen.

Evaluation. As the data is gated manually, we have cell annotations at our disposal. A simple way to visualise the quality of the resulting trajectories is to plot a density function which indicates the presence of a specific cell type in a specific place in the trajectory (using a sliding window approach). This leads to two plots, one for each trajecory, where the plots contain four curves representing the density of the four different cell types. This does not only show us if the cells

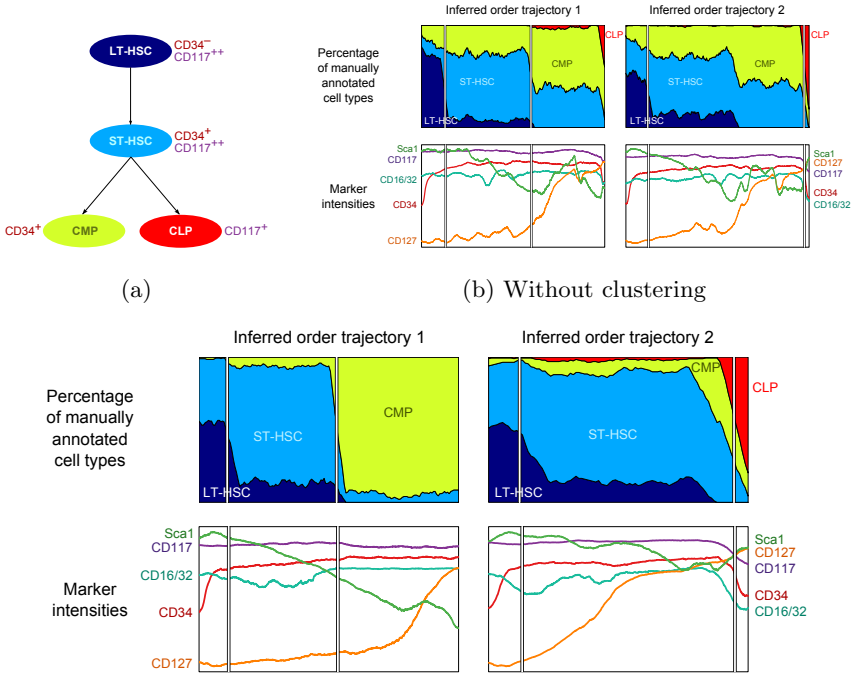
are assigned to the correct trajectories, but also if the order in which cell types are assigned is as expected. Another way to visualise the resulting trajectories is to plot curves representing the marker values of the cells along the trajectory as done in Figure 2(b). This again results in two plots, one for each trajectory, where each plot contains five curves, one for each marker value. A way to evaluate this result, is to compare these curves with the known marker presence shown in Figure 8(a). For example, we can check if CD34 is low at first and then gradually increasing when the cells differentiate from LT-HSC to ST-HSC.

Results. As input, the algorithm takes one starting cell (a LT-HSC cell) and two end cells (a CLP and a CMP cell). These are chosen from the manual analysis, relying on the assumed values for a subset of markers. The start cell is the cell with the lowest value for CD34 and the highest value for CD117, the end cell representing CMP is the cell with highest CD34 and the end cell representing CLP is the cell with the highest CD117. As parameters we used $k = 100$, $l = 15$ and $cl = 10$. Changing the parameters did not have a big influence on the general trend of the results. The result of the algorithm is given in Figure 8(b). On top, we see the density of the different cell types along the calculated trajectories. From this, we can can conclude that the algorithm takes the correct cells for each trajectory. The trajectory ending in the CMP state contains LT-HSCs, ST-HSCs and CMPs but no CLPs, and these cell types are traversed in the expected order. The trajectory ending in the CLP state does contain a few CMPs, but is also mainly as expected from the assumed differentiation path in Figure 8(a). When looking at the bottom of Figure 8(b), we can also inspect the marker changes through differentiation. As expected, we see an increase in the CD34 intensity corresponding to the transition from LT-HSC to ST-HSC.

These results might help to gain new information as well. For example, there seems to be a subset in the CMPs which gains CD127, which is not used at all in our manual gating. Often it is also very hard to know at which point a gate should be drawn if there is a gradual marker change. By looking at changes in other markers as well, a more informed decision might be taken.

The real data also emphasises the need for the usage of clusters for representing the endpoints. This is indicated in Figure 8(c) where the density of the different cell states and the parameter values of the objects along the calculated trajectory is shown when no clusters are used. Comparing this with the previous result in Figure 8(b), we see several mistakes are made, especially for trajectory 1 where many CMP cells are assigned in the CLP differentiation, resulting in a high FPR (Figure 8(c) top left).

Note that for example in the top left of Figure 8(b) the length of the interval where the CLP density is high, is very small in comparison to the length of the interval where the ST-HSC density is high. This has the simple reason that there are just far more ST-HSC-cells present in the data than CLP-cells (2192



(a) (b) Without clustering

(c) With clustering

Fig. 8: This figure shows (a) the theoretical branched trajectory, (b) the density of the different cell states and the feature values of the cells along the calculated branch going to CMP (on the left), and to CLP (on the right) when using $k = 100, l = 15$, and no clusters, and (c) the results when using 10 clusters. Only with clustering, the separation between the CMPs and CLPs is detected correctly.

versus 122). This phenomenon is also present in the bottom left of Figure 8(b) where the feature values for CLP-cells are only shown in a short interval. A nice extension would thus be to automatically detect the different cell states and rescale the axis.

4 Conclusion

In this paper, we proposed a graph-based approach to infer multiple trajectories from unlabelled data. This is done by extending the Wanderlust algorithm [8] that was only able to model a single trajectory. Our algorithm extends Wanderlust by first assigning each cell to one or multiple trajectories. On each separate trajectory, Wanderlust is then executed. Results indicate that the assignment of

the cells to the different trajectories and the ordering of the cells works well.

Future work includes finding the branch points and using this information to avoid wrong assignments. Knowing the branch points would also be beneficial for a better representation of the trajectories with a non-uniform density as this would allow rescaling of the trajectory. It is also important to take into account the fact that not all cells in the data might belong to the trajectories. While assigning the cells, such noisy data should be removed. This is a challenging extension that must be made as it will allow us to make use of data sources where not everything is known.

Acknowledgments

We would like to thank Lianne van de Laar and Bart Lambrecht for providing a biologically relevant dataset to test our algorithm. Sofie Van Gassen is funded by the Flanders Agency for Innovation by Science and Technology (IWT).

References

1. P. Qiu, E.F. Simonds, S.C. Bendall, K.D. Gibbs Jr, R.V. Bruggner, M.D. Linderman, K. Sachs, G.P. Nolan and S.K. Plevritis. "Extracting a cellular hierarchy from high-dimensional cytometry data with SPADE". *Nature biotechnology*, vol.29, no.10, pp. 886–891, 2011.
2. E.D. Amir, K.L. Davis, M.D. Tadmor, E.F. Simonds, J.H. Levine, S.C. Bendall, D.K. Shenfeld, S. Krishnaswamy, G.P. Nolan and D. Pe'er. "viSNE enables visualization of high dimensional single-cell data and reveals phenotypic heterogeneity of leukemia". *Nature biotechnology*, vol.36, no.6, pp. 545-552, 2013.
3. S. Van Gassen, B. Callebaut, M.J. Van Helden, B.N. Lambrecht, P. Demeester, T. Dhaene, and Y. Saeys. "FlowSOM: Using selforganizing maps for visualization and interpretation of cytometry data". *Cytometry Part A*, 2015.
4. Magwene, Paul M., Paul Lizardi, and Junhyong Kim. "Reconstructing the temporal ordering of biological samples using microarray data." *Bioinformatics* 19.7 (2003): 842-850.
5. Liiv, Innar. "Seriation and matrix reordering methods: An historical overview." Statistical Analysis and Data Mining: *The ASA Data Science Journal 3.2* (2010): 70-91.
6. Ter Braak, Cajo JF, and O. F. R. Van Tongeren. *Data analysis in community and landscape ecology.* Cambridge University Press, 1995.
7. Trapnell, Cole, et al. "The dynamics and regulators of cell fate decisions are revealed by pseudotemporal ordering of single cells." *Nature biotechnology 32.4* (2014): 381-386.
8. Bendall, Sean C., et al. "Single-cell trajectory detection uncovers progression and regulatory coordination in human B cell development." *Cell* 157.3 (2014): 714-725.
9. Trapnell, Cole, et al. "The dynamics and regulators of cell fate decisions are revealed by pseudotemporal ordering of single cells." *Nature biotechnology* (2014).
10. Seita, J., and Weissman, I. L. (2010). Hematopoietic stem cell: selfrenewal versus differentiation. *Wiley Interdisciplinary Reviews: Systems Biology and Medicine*, 2(6), 640-653.