Globally Optimal Multi-Object Tracking with Splitting and Merging

Mihaela Mihaylova[1] **Jelle Piepenbrock**[1] **Johannes Textor**[1]

[1]Data Science, Institute for Computing and Information Sciences, Radboud University, Nijmegen, The Netherlands

Abstract

In multi-object tracking (MOT), the trajectories of moving objects – e.g., molecules, cells, or people – must be recovered from detected positions. MOT is often solved using network flow approaches to find globally optimal solutions. Here, we consider a version of MOT where objects can split or merge – such as cells that are moving but also dividing while being tracked. We show that this problem is NP-hard, phrase it as a Max-SAT problem, and ask: (1) To what extent can modern optimized satisfiability solvers be applied to realistic MOT problems? (2) Are the globally optimal solutions better than those generated by existing methods? Using both simulated data and real-world cell tracking data, we show that Max-SAT is a computationally costly but feasible approach that can lead to substantially improved solutions for problems of realistic size. We hope that the Max-SAT instances generated by the MOT problem can serve as practically relevant benchmarking cases for the future improvement of Max-SAT solvers.

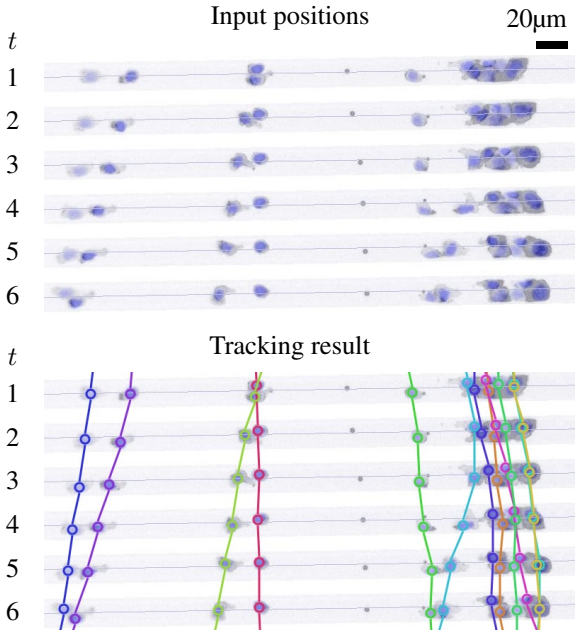

Figure 1: Example MOT problem: tracking cells (dark blue: cell nuclei, light blue: cell bodies) moving in grooves on a microchip; see Wortel et al. [2024] for details of the experimental setup. Six consecutive frames ($t \in \{1, \ldots, 6\}$) from a movie are shown vertically stacked without (top) and with (bottom) detections (points) and tracks (lines).

1 INTRODUCTION

In a *Multi-Object Tracking* (MOT) problem, we are given the positions and possibly other information such as shape, size, or current direction of movement about multiple objects in consecutive time frames. *Tracking* these objects throughout the frames means linking detections that belong to the same object (Figure 1). Applications include particles in the Large Hadron Collider [Caron et al., 2025], single molecules [Simon et al., 2026], cells [Maška et al., 2023], pedestrians in dense crowds [Parisi et al., 2021], or asteroids and other celestial bodies [Du et al., 2024]. But also problems that do not involve motion, such as tracking evolving types of bacteria in metagenomics [Brito and Alm, 2016], could be understood as instances of MOT.

Due to the generality of MOT and its many applications, the problem has been studied intensively in the artificial intelligence literature (see review by Luo et al. [2021] and references therein). The problem is often approached using network flow techniques [Wang et al., 2019] or successive shortest path algorithms [Magnusson et al., 2015]. This approach offers great flexibility and can handle many practical issues such as imperfect detections (e.g., over- and undersegmentation by object detection algorithms), objects entering or leaving the field of view, and new objects spawning or

Proceedings of the 42nd Conference on Uncertainty in Artificial Intelligence (UAI 2026), PMLR 337:4514–4527.

objects disappearing. The user defines a cost function to tune the algorithm, which can, e.g., take into account prior information on the accuracy of the underlying object detector. If the objects are detected very imprecisely, we can also treat their positions as latent [Angle et al., 2021].

An issue that has received less attention is splitting or merging of objects being tracked. Important examples of object splitting are cell division in biology [Magnusson et al., 2015, Ershov et al., 2022, Maška et al., 2023] or meteorite splitting in astronomy [Streit et al., 2020]. Both object splitting and joining can also be observed when tracking the diffusion of reacting molecules across space and time [Zhang et al., 2023]. While existing Bayesian approaches have considered this problem explicitly [Streit et al., 2020], it has to our knowledge not yet been systematically approached from a computational complexity perspective.

In this paper, we will show that allowing for object splitting fundamentally changes the complexity of the problem – essentially, it turns from a simple network flow problem to a much harder degree-restricted subgraph problem. Specifically, we offer these contributions:

1. We formulate MOT with object splitting and/or merging in terms of a maximum-weight degree-restricted subgraph, and show that this problem is NP-hard.

2. We show how to convert instances of the generalized MOT problem to partial weighted Max-SAT instances that are amenable to modern solvers.

3. We demonstrate using a large, real-world cell tracking dataset that the Max-SAT approach is currently feasible as long as there are not too many cells and options for links between cells. On such data, the Max-SAT approach performs comparably or outperforms state-of-the-art algorithms in the field.

4. We use a simulation model of dividing and moving cells to generate realistic instances of the generalized MOT problem that can be used to investigate the feasibility of the Max-SAT approach on increasingly large problems and, potentially, to generate novel benchmark instances for Max-SAT solvers.

To ensure a largely self-contained presentation, the following section defines the notation and concepts we use. Afterwards, we define the generalized MOT problem (Section 3), prove its NP-hardness (Section 4), and then show empirical results on real data (Section 5) and simulated data (Section 6). We focus on object *splitting* in this paper for simplicity, but the contributions equally apply to merging (one can be turned into the other by reversing the order of frames).

2 BACKGROUND

Graphs. A *directed graph* $G = (V, E)$ consists of a set V of *vertices* (or nodes) and a set E of *edges*, where each edge is an ordered pair of vertices; for clarity, we write an edge (u, v) as $u \to v$. A *path* in a graph is a sequence of vertices $v_1, v_2, \ldots, v_k$ such that each consecutive pair (v_i, v_{i+1}) is connected by an edge. The *indegree* of a vertex v, written as $\deg^-(v)$, is the number of edges with endpoint v, i.e., $\deg^-(v) = \#\{u \mid u \to v \in E\}$. The *outdegree* $\deg^+(v)$ is similarly defined. A *directed acyclic graph* (DAG) is a directed graph containing no directed cycles, i.e., there is no sequence of edges leading from any vertex back to itself. A *subgraph* $G' = (V', E')$ of $G = (V, E)$ is a graph where $V' \subseteq V$ and $E' \subseteq E$, with all edges in E' connecting vertices in V'.

Satisfiability. A *literal* is a Boolean variable x or its negation $\neg x$. A *clause* is a disjunction (OR) of literals, e.g., $(x_1 \vee \neg x_2 \vee x_3)$. A formula ϕ is in *conjunctive normal form* (CNF) if it is a conjunction (AND) of clauses, i.e., $\phi = C_1 \wedge C_2 \wedge \cdots \wedge C_m$. The *Boolean satisfiability problem* (SAT) asks whether there exists an assignment of truth values to variables that makes a given CNF formula evaluate to true. SAT is the canonical NP-complete problem [Cook, 1971]. In *weighted Max-SAT*, each clause has an associated nonnegative real-valued weight $w : C_i \to \mathbb{R}$, and the goal is to find an assignment maximizing the total weight of satisfied clauses. In *partial Max-SAT*, clauses are partitioned into *hard* clauses (which must be satisfied) and *soft* clauses (whose satisfaction is maximized). This can be achieved by setting the weights of each hard clause to a value greater than the sum of all values of soft clauses.

Measuring tracking performance. We use two standard metrics from the field [Ulman et al., 2017], CT and TF. Briefly, the **track fraction (TF)** is the average, across all ground-truth tracks, of the length of the longest continuous segment of each track that is correctly reconstructed by an algorithm, expressed as a proportion of the full length of that track. The **complete tracks** metric determines the proportion of perfectly reconstructed ground-truth tracks. A ground-truth track is considered completely reconstructed if and only if each of its detections has an assigned track point in a corresponding algorithm-generated track, and both tracks have the same length. The final CT value is defined as: $CT = \frac{2T_{\text{rc}}}{T_{\text{gt}}+T_{\text{a}}}$ where T_{rc} is the number of ground-truth tracks completely reconstructed by the algorithm, T_{gt} is the number of all ground-truth tracks, and T_{a} is the number of all algorithm-generated tracks.

3 PROBLEM DEFINITION

In this section, we formally define the MOT problem in terms of finding a maximum-weight subgraph of a given

DAG. Next, we show the encoding of MOT as a Max-SAT problem.

3.1 THE TRACKING GRAPH

We model MOT as finding a subgraph of a directed acyclic graph called the *tracking graph*. Equivalently, one could also define the problem in terms of an undirected graph (via node splitting). Our formulation corresponds to a simplified version of the construction originally proposed by Magnusson et al. [2015].

Definition 1 (Detection Set)**.** *Let $D = \bigcup_{t=1}^{T} D_t$ be a set of* detections*, where $D_t = \{d_{t,1}, d_{t,2}, \ldots, d_{t,n_t}\}$ contains all detections at time frame t; each detection refers to a position in space, i.e., $d_{t,i} \in \mathbb{R}^n$.*

We start with the most general version of the problem.

Definition 2 (Tracking Graph)**.** *Given a detection set D, the* tracking graph *$G = (V, E, w)$ is a weighted directed acyclic graph with vertex set $V \supseteq D \cup \{s, t\}$, consisting of the detection set D, other nodes representing events such as appearance and disappearance of objects, a source node s and a sink node t, and a* weight function *$w : E \to \mathbb{R}^+$ assigning nonnegative "rewards" to edges.*

Some explanation as to the vertex set V is in order. In a simple case (see Figure 2 for an example), V might contain only the detection nodes themselves, and the weight function might simply be designed to favour connections between nodes that are close, e.g., by using a kernel like

$$w(d_{t,i}, d_{t+1,j}) = \exp\left(-\|d_{t,i} - d_{t+1,j}\|^2\right)$$

However, in many realistic applications, additional nodes are also added. These nodes can then represent events like "an object enters the field of view", "an object disappears" and the cost of linking these nodes to detection nodes would represent the likelihood of such events happening (e.g., Magnusson et al. [2015]). In addition, detection nodes could be split into an "input" and "output" node, and the weight of the edge between them would then be used to model the confidence in the correctness of the detection itself (e.g., Wang et al. [2019]). Concrete examples of such "extra nodes" will be shown in Sections 5 and 6.

In practice, we may choose to remove edges whose weights are lower than a certain threshold τ. For example, if we were tracking pedestrians in a movie with one frame per second, we would likely choose to remove edges corresponding to positions more than 10 meters apart.

Definition 3 (Many-To-Many Multi-Object Tracking)**.** *Given a tracking graph $G = (V, E, w)$ and constraints $I, O : V \to 2^{\mathbb{N}}$ on the in- and outdegrees of every node, the* many-to-many multi-object tracking problem *(M2M-MOT) is to find a subgraph $G' \subseteq G$ that maximizes the edge weight:*

$$\textit{argmax}_{G'=(V',E')\subseteq G} \sum_{e \in E'} w(e)$$

while obeying given indegree and outdegree restrictions:

$$\forall v \in V' \setminus \{s, t\} : \textit{deg}^-(v) \in I(v) \textit{ and } \textit{deg}^+(v) \in O(v)$$

The in- and outdegree restrictions are unitary in many existing MOT approaches (e.g., Wang et al., 2019); i.e., each node in G' is restricted to have indegree and outdegree of at most 1 – meaning that each detection can only be assigned to a single object. This allows us to model the problem using network flow techniques such as min-cost flow with unit edge capacities [Wang et al., 2019]:

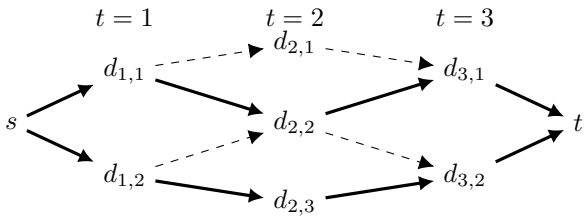

Figure 2: Simple tracking graph G consisting of detection nodes (d) and start (s) / end (t) nodes. Every path from s to t represents a trajectory, and the goal is to find a maximum-weight subgraph G' consisting of $s - t$ paths that obey a given set of degree restrictions. For example, the bold solid edges represent a maximal subgraph that fulfills the restrictions $I(d) = O(d) = \{1\}$ for all $d \in V \setminus \{s, t\}$. Some edges between detection nodes are missing; these are edges whose weights (representing likelihoods or rewards) are considered too small to be included in any solution.

Definition 4 (One-To-One MOT)**.** *An M2M-MOT with degree restrictions $I(v) = O(v) = \{1\}$ for all $v \in V' \setminus \{s, t\}$ is called a* One-to-one multi-object tracking problem *(O2O-MOT).*

By contrast, the M2M-MOT can no longer be modelled using simple network flows. In addition to the general M2M-MOT, we also introduce a more restricted version that however is already NP-hard:

Definition 5 (One-To-Many MOT)**.** *An M2M-MOT with indegree restrictions $I(v) = \{1\}$ for all $v \in V' \setminus \{s, t\}$ is called a* One-to-many multi-object tracking problem *(O2M-MOT).*

4 THEORETICAL RESULTS

We start by restating prior results.

Theorem 1. *The O2O-MOT can be solved in time* $O(|V|^2|E|)$.

Proof. The O2O-MOT has been extensively studied (see Wang et al. [2019] and references therein). It can be approached as a minimum-cost flow problem with negative weights (this construction will be discussed in more detail in Section 5); i.e., the goal is to find *any* flow (irrespective of flow value) that has the smallest (negative) cost. Such problems can be solved in time $O(m \times |V| \times |E|)$ where m is the maximum flow value. In our case, the maximum flow value is at most $|V|$ because flow is unitary (all vertex capacities are 1), which leads to the claimed complexity. □

However, the problem becomes harder by allowing objects to split and to join. This makes the problem resemble flow problems with gains or general degree-constrained subgraph problems, which were among the earliest problems shown to be NP-hard [Karp, 1972, Sahni, 1974, Shiloach, 1981].

Theorem 2. *The decision version of O2M-MOT (given a tracking graph* G *and degree restrictions* I, O*, determine if there is a subgraph* G' *of weight* $\geq k$ *that fulfills all degree restrictions) is NP-complete, even when the outdegree restrictions are limited to* $O(v) \subseteq \{1, 2\}$ *for all* $v \notin \{s, t\}$.

Corollary 1. *The decision version of M2M-MOT is NP-complete.*

We show this by reduction from a specific restricted version of SAT. We were unable to find this exact version of SAT in the literature, but its NP-hardness follows directly from a result by Tovey [1984] (see Appendix for details).

Lemma 1. *The CNF satisfiability problem remains NP-hard if every clause contains at most three variables and each variable occurs at most twice in negated form and at most twice in nonnegated form.*

Proof of Theorem 2. Membership in NP is trivial for both O2M-MOT and M2M-MOT: guess a subgraph G', check if it fulfills all restrictions and has weight k. We now show NP-hardness of O2M-MOT, which implies NP-hardness of the more general M2M-MOT.

We are given a 3-CNF SAT formula ϕ with k clauses and n variables in which each literal (negated or nonnegated version of a variable) occurs at most twice. We build a graph G consisting of 5 layers. In the first layer, we have only s. In the second layer, we have one node x_i per variable, and edges $s \to x_i$ for all i. In the third layer, we have one node per literal, and we add edges $x_i \to l_i$ and $x_i \to \neg l_i$. In the fourth layer, we have one node c_j per clause, and we add at most two edges from every literal to the clauses it appears in. In the final layer, we have only t, and we have an edge from every clause to t. These edges have weight 1, all others have weight 0.

We now set the degree restrictions as follows: $I(x_i) = O(x_i) = I(l_i) = I(\neg l_i) = I(c_j) = O(c_j) = \{1\}$; $O(l_i) = O(\neg l_i) = \{1, 2\}$.

The key difference to an O2O-MOT is that we allow the literal nodes to have two outgoing edges despite having only one incoming edge. This is necessary to encode satisfiability instances because the same literal could be required to render two different clauses true.

We claim that there exists a satisfying assignment for ϕ if and only if there exists a subgraph $G' \subseteq G$ fulfilling all vertex degree restrictions with weight k (the maximum possible weight of any G').

$\Rightarrow$ Start with a satisfying assignment. Take the subgraph consisting of all corresponding paths (going from variables to their values to all the clauses these occur in). Now, for each clause node with indegree > 1, delete all but one incoming edge. Then, for each literal node with outdegree 0 but indegree 1, delete the incoming edge. Finally, for all variable nodes with outdegree 0 but indegree 1, delete the incoming edge. The resulting subgraph fulfills all restrictions and has weight k.

$\Leftarrow$ Start with a subgraph G' fulfilling all restrictions. For each variable in the subgraph, at most one of the literal nodes has outgoing edges. Since G' has weight k, all clause nodes have an outgoing edge, so each clause node must also have an incoming edge connecting it to a literal. The negated version of that literal then has no edges connected to it. Hence, G' encodes a valid truth value assignment for some variables that is guaranteed to satisfy ϕ. □

Note that the graph structure we built in our proof above was quite sparse and restricted still; we only needed to have a single layer where the outdegrees were not restricted to 1, and all other degree restrictions were unitary. Therefore, the NP-hardness of this problem is not a result of dense graph structure (which could be unrealistic in many real-world tracking problems), but rather the ability to encode combinatorial complexity.

Having shown that our problem is NP-hard, we know that its solution will likely require combinatorial search. Tracking problems are often encoded as (possibly integer) linear programs [Wang et al., 2019, Bragantini et al., 2025] or quadratic programs [Chari et al., 2014]. Here, we instead convert it to weighted partial Max-SAT instances. This allows us to leverage optimized Max-SAT solvers to find globally optimal solutions.

Definition 6 (Weighted Max-SAT Encoding of M2M-MOT)**.** *An M2M-MOT problem with degree restrictions* I, O *on tracking graph* $G = (V, E, w)$ *can be encoded as a weighted partial Max-SAT instance* $\Phi = (C_1, w_1^{(\phi)}), (C_2, w_2^{(\phi)}), \ldots$ *over variables* $\{x_e \mid e \in E\}$*. For each edge* $e \in E$*, setting the variable* x_e *to true means*

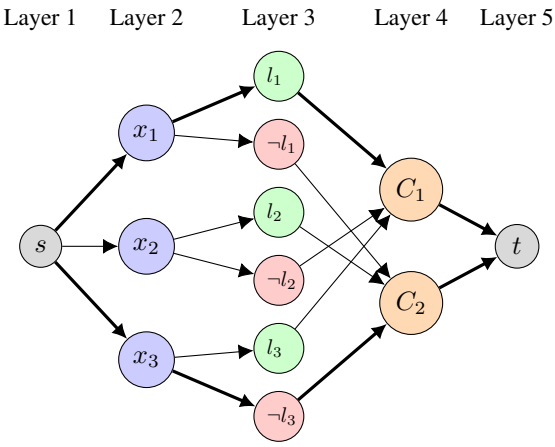

Figure 3: Graph construction for the NP-hardness reduction from SAT (Theorem 2). Example shown for formula $\phi = (x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee \neg x_3)$ with three variables and two clauses. Edges from clauses to t have weight 1; all other edges have weight 0. The bold lines depict a subgraph that fulfills all degree restrictions and corresponds to a satisfying assignment of ϕ.

that e is included in the resulting subgraph G'.

First, for each edge $e \in E$, we add a "soft" constraint

$$(x_e, w(e))$$

Then, for each node $v \in V$, we add "hard" cardinality constraints

$$\big(\textsc{CardinalityIn}\big(\{x_{u\to v} : u \to v \in E\}, I(v)\big), \Lambda\big)$$

and

$$\big(\textsc{CardinalityIn}\big(\{x_{v\to q} : v \to q \in E\}, O(v)\big), \Lambda\big)$$

where Λ is the "top value" $\Lambda := \sum_e w(e)$ and $\textsc{CardinalityIn}(X, S)$ stands for a formula that counts the number of true literals from the set X and determines whether that number is in the set S.

Note that the CardinalityIn constraints in Definition 6 can be encoded into CNF using standard cardinality encodings such as sequential counters or totalizers; these are standard operations implemented, e.g., in the PySAT framework [Ignatiev et al., 2018, 2024]. In the worst case, this transformation can lead to an exponential blowup of the formula size, e.g., if the set contains all even or all odd numbers. However, in practice, we will generally use simpler sets such as closed intervals of integers, which have smaller encodings. In this paper, we do not consider cardinality sets other than $\{1\}$ or $\{1, 2\}$. The Max-SAT solver we use in our experiments is RC2 [Ignatiev et al., 2019], which is convenient to use within the PySAT framework.

5 TRACKING NONDIVIDING CELLS

Before we apply our Max-SAT encoding to MOT problems with object splitting, we will now first ask to what extent there is already a significant cost incurred with using the Max-SAT approach on problem instances that would still have a polynomial-time solution. We will therefore consider MOT problems where objects can *not* yet split or merge. They *can*, however, enter the field of view late or leave it early. Furthermore, detection errors might lead to spurious or missing detections in some frames. These issues can still be dealt with in regular flow-based MOT approaches in addition to Max-SAT.

Specifically, we will consider data of the type shown in Figure 1, where cells were imaged moving in so-called microfluidic devices. Cells in these devices were confined to lanes of configurable width. We hand-curated 1183 of these lanes and annotated them with a "ground truth" tracking. The dataset is available in the Supporting Information, and has been generated for a study of T cell movement in confined environments [Wortel et al., 2024].

We implement and compare three approaches that differ in their graph structure and optimization strategy: global optimization by Max-SAT, global optimization by min-cost flow, and greedy optimization by a repeated shortest path algorithm. As a basis for all three approaches, we use the squared Euclidean distance to model the "cost" or "reward" (depending on the approach) of linking edges together. That is, we define a baseline weight function

$$w(d_{t,i}, d_{t+1,j}) = \tau - \|d_{t,i} - d_{t+1,j}\|^2 \qquad (1)$$

between consecutive detection nodes. Here τ is a maximum plausible movement of a cell between two consecutive frames; throughout, we set this to τ=(30µm)^2 for a frame interval of 20 seconds. All approaches impose the one-to-one degree constraints $I(d) = O(d) = \{1\}$ for every detection node $d \in D$. The three graph structures are defined below, and illustrated in Figure 4.

Max-SAT. This is our approach defined above. **Tracking graph** (G_{sat}): Given the detection set D, we use vertices $V = D \cup \{s, t\}$. The edge set is $E = \{s \to d, d \to t \mid d_{t,i} \in D\} \cup \{d_{t,i} \to d_{t+1,j} \mid \|d_{t,i} - d_{t+1,j}\|^2 < \tau\}$; i.e., we connect both s and t to every detection node, allowing tracks to begin and end at any frame, and link detections in adjacent frames if their distance is less than τ. **Weights:** For all detection nodes $d \in D$, we use $w(s \to d) = w(d \to t) = 0$. Inter-detection edges carry weight $w(d_{t,i}, d_{t+1,j}) = \tau - \|d_{t,i} - d_{t+1,j}\|^2$. **Degree restrictions:** We use the O2O-restrictions $I(d) = O(d) = \{1\}$ for all $d \in D$. The graph and degree restrictions are then encoded as a weighted partial Max-SAT instance as described above (Definition 6).

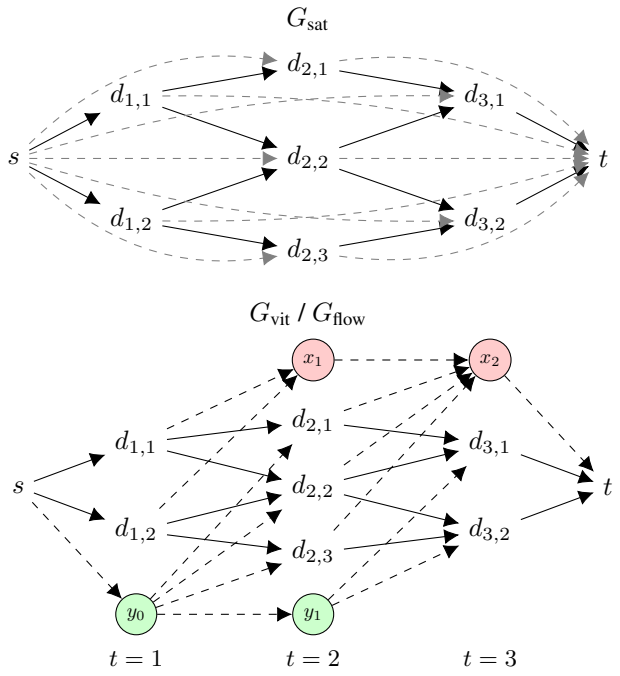

Figure 4: Graph structures for the Max-SAT (G_{sat}) and greedy/Viterbi (G_{vit}) tracking approaches. For Max-SAT, s and t connect to every detection node (gray arrows), so tracks may start or end at any frame. For Viterbi, s connects only to first-frame detections; nodes y_t and x_t are placeholders for cells appearing later (green) or leaving before the final frame (red). The same graph is used as a basis for the min-cost flow formulation – the only modification is that detection nodes are split to implement a "vertex capacity" (a standard modification in network flow algorithms) – see Appendix for details.

Viterbi. This is a greedy approach introduced by Magnusson et al. [2015], which was among the top-3 performers in the recent Cell Tracking Challenge [Maška et al., 2023]. **Tracking graph** (G_{vit}): Given a detection set $D = \bigcup_{t=1}^{T} D_t$, we use the vertex set $V = D \cup \{s, t\} \cup \{y_1, \ldots, y_{T-1}\} \cup \{x_2, \ldots, x_T\}$. That is, we augment the detection nodes with "appearance" nodes y_t (cells not entering the field of view before frame $t{+}1$) and "disappearance" nodes x_t (cells leaving the field of view at frame t). The edge set is $E = \{s \to y_1, x_T \to t\} \cup \{s \to d \mid d \in D_1\} \cup \{d \to t \mid d \in D_T\} \cup \{d_t \to x_{t+1} \mid d \in D_t, t < T\} \cup \{y_t \to d \mid d \in D_t, t > 1\} \cup \{d_{t,i} \to d_{t+1,j} \mid \|d_{t,i} - d_{t+1,j}\|^2 < \tau\}$. **Weights:** Inter-detection edges carry weight $w(d_{t,i}, d_{t+1,j}) = \|d_{t,i} - d_{t+1,j}\|^2$. All other edges carry weight τ.

The Viterbi algorithm repeatedly finds shortest paths in G_{vit} and then eliminates the detection nodes on it. This enforces the restriction that no node occurs on more than one track. The process is iterated until s and t are disconnected. While this process does not guarantee an optimal solution, it is easy and quick to implement, and has a nice *any-time* property: every path produced is a valid track. The user can stop generating more paths at any time, e.g., to inspect the ones already generated.

Since all paths in G_{vit} are of the same length, this is equivalent to finding minimum-cost paths. Note that every path π_{vit} of weight w in G_{vit} corresponds to exactly one path π_{sat} in G_{sat} of weight $(T+1)\tau - w$. Therefore, a maximum-cost subgraph $G'_{\text{sat}} \subseteq G_{\text{sat}}$ corresponds to a unique minimum-cost subgraph $G'_{\text{vit}} \subseteq G_{\text{vit}}$.

Min-cost flow: Implemented similarly to Viterbi; see Figure 4 and the Appendix for details.

In addition to these different combinatorial approaches, we also used a state-of-the-art algorithm implemented in the popular "TrackMate" package [Ershov et al., 2022] as an additional baseline. This "linear assignment problem" (LAP) algorithm is a greedy approach that first matches detections in subsequent frames to each other and then merges the resulting "tracklets" to longer tracks [Jaqaman et al., 2008]. It also has a distance cutoff τ, which we set to the same value as for the other algorithms.

Results. As expected, flow and satisfiability algorithms indeed found the same solution weight for all problems. The Viterbi approach came very close to these optimal solutions in most cases, and within 80% in all cases (Figure 5). The Viterbi approach was the fastest and solved all problems in a second or less on a MacBook Pro with an M3 processor, compared to up to 10 seconds for our (unoptimized) flow method and up to two seconds for the Max-SAT approach. However, the slightly lower solution weight did translate to a substantially reduced agreement with the ground truth annotations according to both metrics; only about half of the tracks were correctly recovered, compared to $\sim$80% for the global optimizers. The LAP approach performed slightly worse than the Viterbi algorithm on these problem instances.

In summary, these experiments showed that (1) global optimization led to significantly improved tracking performance compared to a greedy approach, and (2) the Max-SAT encoding of the tracking problem led to solvable problem instances, albeit at higher computational cost than the greedy approach. For this particular dataset, the Max-SAT solver was in fact even slightly faster than the min-cost flow approach; however, we would like to point out that we did not optimize our implementation of min-cost flow.

6 TRACKING DIVIDING CELLS

To analyze the performance of the Max-SAT approach on its intended use case, we used a simulation framework called the "cellular Potts model" [Wortel and Textor, 2021]. This simulation framework is able to generate realistically shaped

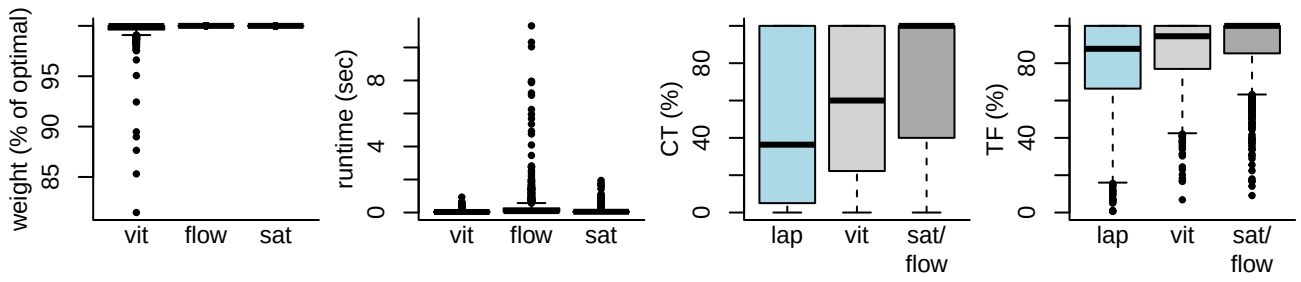

Figure 5: Evaluation results on 1183 cell tracking problems. Vit: greedy approach (Viterbi algorithm); flow: globally optimal min-cost flow solution; sat: globally optimal solution of Max-SAT encoding; lap: "Linear assignment problem", a popular algorithm implemented in TrackMate [Ershov et al., 2022]. Note that there can be several globally optimal solutions, and the solutions returned by the min-cost flow and Max-SAT approaches are not necessarily the same. Therefore there are small differences in the comparison to ground truth, but not to weight, for these two methods. All pairwise differences are statistically significant at 5% alpha (two-sided t test). LAP is not taken along in the weight comparison since it does not optimize the same weight. The runtime of LAP is consistently much less than a second. Performance metrics for Max-SAT and flow are the same.

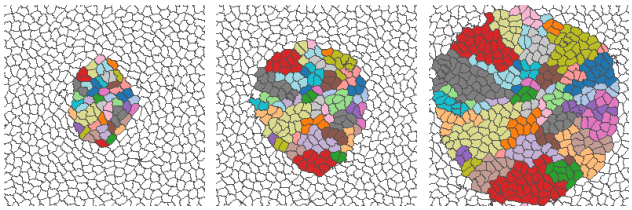

Figure 6: Example run of our simulation at 10, 40, and 80 frames. The colored cells are dividing and being tracked. Cells having the same color share the same progenitor, and should all be assigned to the same track.

cells, and can handle both cell division and cell movement. Specifically, we considered a simple 2D simulation of a growing cell population in a sheet of epithelial cells – similar to the tumor growth model of Szabó and Merks [2013] (Figure 6). In this simulation, cells grow and divide once having reached a given size, resulting in two child cells. The dividing cells are surrounded by a dense sheet of nondividing cells, which we do not track to focus our evaluation on the most difficult cells. As the dividing cells grow, they start to squeeze the surrounding cells, which start dying as a result.

Our experiments were done using a parallel CPM framework written by Jan Schering Schering and Textor [2026]. This implementation based on the framework "taichi" [Hu et al., 2019]. More details of the simulation model, including model equations and parameters used, are given in the Appendix.

To accommodate the dividing cells in our Max-SAT formulation, we simply change the outdegree restrictions for G_{sat} to $O(v) = \{1, 2\}$ (this assumes cells divide at most once per frame). The resulting subgraph G'_{sat} then becomes a forest on the detection nodes D. To compare this to the ground

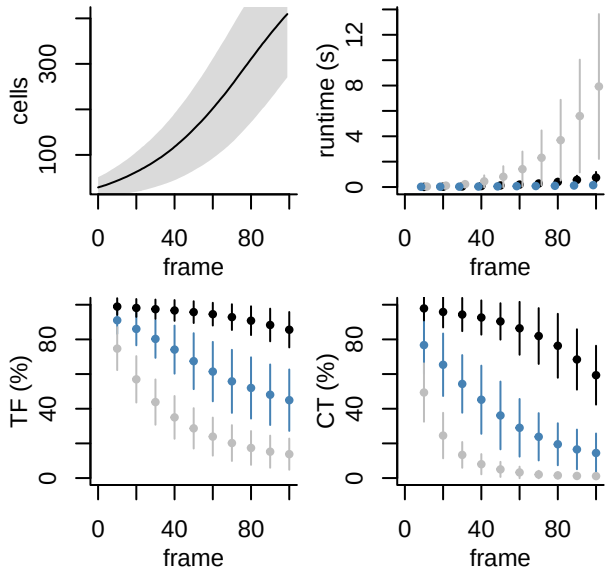

Figure 7: Runtime (top row) and performance (bottom row) metrics for Max-SAT solutions (black) of tracking dividing cells. Shading / bars depict standard error of the mean for N=100 simulations. Gray: Viterbi algorithm (does not consider cell division), blue: LAP tracker from TrackMate (does consider cell division).

truth (which is also a forest on D), we used the same CT metric and a suitably generalized version of the TF metric (replacing "longest continuous track" by "largest simply connected graph" in its definition). For the CPM data we set the squared distance threshold to τ=5000 µm^2.

We evaluated our Max-SAT tracker on 100 simulated videos in which cell numbers increased from ∼10 to ∼400 over 100 frames (Figure 7). We tracked increasingly long segments of these videos and measured runtime and TF/CT

metrics. The runtime measurements were done on an Apple MacBook Pro with 18 GB RAM and an M3 Pro processor. In these experiments, we now found that the Max-SAT tracking algorithm generally ran faster than the Viterbi algorithm, while the LAP tracker was the fastest. The quality of the Max-SAT based tracks degraded gradually, reaching ∼60% CT on full-length videos. However, this was substantially better than both the Viterbi tracker (which does not consider cell divisions) and the state-of-the-art LAP tracker, which deals with cell division in a greedy way [Ershov et al., 2022]. In summary, we were able to generate solutions of much better quality than the LAP tracker at computational cost comparable to the Viterbi tracker.

Altogether, our experiments demonstrated the feasibility of tracking tens to hundreds of dividing objects across tens to hundreds of frames with the Max-SAT approach, putting many typical application scenarios within reach [Maška et al., 2023]. On the other hand, there is clearly room for improvem/ent concerning the runtime. Many possible optimizations for future work come to mind. For example, unambiguous matches between detections could be pre-processed, and the outdegree restrictions could only be relaxed where plausible (e.g., this could depend on the current size of the cell). Further, in our current implementation we are building quite large graphs directly in Python, which is not a particularly efficient approach.

7 MAX-SAT SOLVER PERFORMANCE ON TRACKING PROBLEMS

In our previous experiments we had used the SAT solver RC2, easily available for experimentation via the "pysat" library [Ignatiev et al., 2018, 2024]. This Max-SAT solver has been surpassed in recent Max-SAT competitions, so we wondered if the already satisfactory performance of the Max-SAT approach found so far could be substantially improved by switching to other solvers.

To this end, we use three Max-SAT solvers that achieved the highest ranking in the Max-SAT 2024 competition [Berg et al., 2024]: CASHWMaxSAT-DisjCom-S6 ("cashw"; Pan et al., 2024, 2025), UWrMaxSat-SCIP ("uwr"; Karpiński and Piotrów, 2020, Piotrów, 2024), and EvalMaxSAT ("eval"; Avellaneda, 2024). To focus on only the solution time for the Max-SAT instances themselves, we saved the instances corresponding to each tracking problem (where, for the cell data, we raised the distance threshold to $\tau = 40^2 \mu m^2$ to make the instances harder) as separate files in weighted conjunctive normal form (WCNF) format. We measured the time taken for each solver to run end-to-end on each file. In addition to these Max-SAT solvers, we also took the general-purpose integer linear programming (ILP) solver "highs" along [Huangfu and Hall, 2017]. Note that using highs incurs a small conversion overhead because "highs" cannot directly read WCNF files. We performed

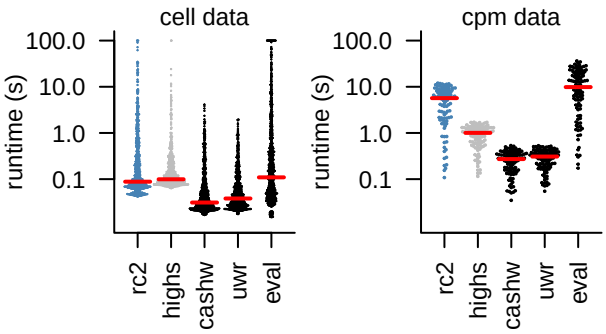

Figure 8: Runtime for our baseline Max-SAT solver (rc2), an ILP solver (highs, involving a conversion step from Max-SAT to ILP syntax), and three state-of-the-art Max-SAT solvers (cashw, uwr, eval) on Max-SAT instances generated from the cell tracking data (left) and the CPM tracking data (right). Runtime was capped at a time-out of 100 seconds. Note the logarithmic y-axis. Red bars: medians.

these experiments on an Ubuntu Linux machine with an Intel Core i9-9820X CPU (3.30GHz) so we could directly use the provided solver binaries.

The results indeed showed major differences between SAT solvers. The two highest-ranking solvers "uwr" and "cashw" (which is in fact based on "uwr") solved all simulated data (CPM) instances in less than a second, even though some of the same problems took more than 30 seconds on "eval" (which generally performed worse than our convenience RC2 solver). The ILP solver also performed quite well, being only slightly outpaced by the two fastest dedicated Max-SAT solvers despite the additional conversion step. It is possible that one could close this performance gap entirely by optimizing the setup and directly emitting the problem instance in ILP syntax.

These results indicate that production implementations of the tracking approach proposed here should use a modern Max-SAT solver to avoid unnecessary performance costs. The best available solvers were able to solve all our tracking problems in very reasonable time, and certainly faster than our implementation of the greedy Viterbi algorithm.

8 DISCUSSION AND CONCLUSIONS

In this paper, we have considered the computational complexity of a MOT problem where objects can split or merge. Despite the great flexibility of flow-based approaches to the MOT problem [Magnusson et al., 2015, Wang et al., 2019] we showed that allowing objects to split into two child objects makes this problem NP-hard even if it occurs just in a single time-frame. This warrants the application of combinatorial solution approaches like constraint satisfaction solvers.

Other approaches have in the past treated MOT as a hard combinatorial problem; examples include the Viterbi method [Magnusson et al., 2015] and the recent "Ultrack" method [Bragantini et al., 2025], which considers an even more general scenario where different detection candidates are concurrently evaluated. Ultrack uses integer linear programming (ILP) as its solution engine, and this would have been another possible candidate for this research, given the already tight connection between restricted degree subgraph problems and flow approaches for one-to-one MOT [Wang et al., 2019]. We considered that the more restricted Max-SAT formulation might open additional avenues for optimization, and would be amenable to both ILP and Max-SAT solvers. However, the Max-SAT approach requires (in our formulation) all costs to be nonnegative, whereas other authors consider mixing positive weights (rewards) and negative weights (costs) in the same problem [Wang et al., 2019]. Such a mixing of rewards and costs would not be straightforward to realize in a Max-SAT formulation.

Our evaluation on real-world data showed that at least for problems of moderate size (tens of detections in tens of frames, a scenario previously considered infeasible for combinatorial optimization [Jaqaman et al., 2008]), there is not necessarily a steep performance cost incurred by using a modern Max-SAT solver instead of a flow-based approach. The same experiments also showed that seemingly modest differences between global optima and those found by greedy algorithms can translate into substantial differences in solution accuracy – so it may well be worth it. While these experiments were run on rather sparse graphs, we still expect the execution time to become an issue when scaling to, say, thousands of detections in hundreds of frames. However, when attempting to construct tracks on non-sparse detection graphs, the greedy algorithm may be expected to drift even further from the global optima. Due to this, the Max-SAT formulation could potentially have an advantage in cases where high detection uncertainty or low time resolution create a need for an approach that takes into account the global characteristics of the specific tracking problem.

For flow-based MOT, specific optimizations have been shown to yield major performance benefits [Wang et al., 2019]. Since the Max-SAT instances generated by our MOT formulation likewise have special structural properties (e.g., all soft clauses consist of a single literal), it is conceivable that solvers could exploit those properties to solve such instances much faster. We would be thrilled if our research would trigger future activity in this area.

All tracking problem instances analyzed in this paper turned out to be easily solvable by the best current Max-SAT solvers. Larger datasets with more ambiguity might generate harder instances that perhaps pose new challenges for these solvers. In this context, it could be interesting to investigate the performance of anytime Max-SAT solvers, which rather than finding the best solution are given a fixed time budget and try to find the best possible solution within that constraint. Anytime solvers are in a separate category in the recurring Max-SAT competitions and generally based on different techniques [Berg et al., 2024].

Deep-learning-based object detectors greatly improved detection-based MOT when they became available; for example, the "Cellpose" model [Stringer et al., 2020] is trained on vast microscopy datasets to recognize and segment cells. A more recent generation of models, including SAM3 [Carion et al., 2025], are capable of directly tracking objects in movies upon visual or text prompts. In the cell tracking field, the combination of Cellpose and SAM3 specifically is rapidly gaining popularity [Pachitariu et al., 2025]. Does this mean that detection-based linking algorithms will become a thing of the past? While time will tell, we would like to point out an important caveat. MOT is usually followed by downstream analyses of parameters such as mean square displacement or track straightness. Choices made during tracking to resolve ambiguity invariably influence and bias the results of these downstream analyses (e.g., if we construct a cost function that assumes our cells move in a Brownian fashion, our analysis may show support for this assumption). With transparent, interpretable cost functions, such biases can be analyzed and mitigated, which can be critical for scientific applications. Indeed, new combinatorial approaches continue to be developed [Betjes et al., 2025, Bragantini et al., 2025]. Perhaps a next generation of deep learning approaches will deliver the best of both worlds by combining the power of foundation models like SAM3 with the controllability of detection-based MOT.

Author Contributions

JT designed the research with input from JP and MM. MM provided the cell-tracking data, contributed code for the evaluation, conducted the evaluation on the cell-tracking data together with JT, contributed the description of the evaluation methodology in the Background section, and provided feedback on the manuscript. JP designed the Max-SAT encoding with input from JT and provided feedback on the manuscript. JT supervised the project, wrote the manuscript draft, wrote the final Max-SAT encoding, and conducted the evaluation on simulated data.

Acknowledgements

We would like to thank Sander Geurts for conducting initial experiments that inspired us to pursue this project further. We also thank Jan Schering for making his parallel implementation of the Cellular Potts Model available for us for this project. Claude Code (Opus 4.5 and Sonnet 4.6) was used for proofreading, to generate initial drafts of TikZ graphics, and for coding assistance.

References

R. Angle, Roy Streit, and Murat Efe. A low computational complexity JPDA filter with superposition. *IEEE Signal Processing Letters*, 28:1031–1035, 2021. ISSN 1558-2361. doi: 10.1109/lsp.2021.3082040.

F. Avellaneda. EvalMaxSAT 2024. In *MaxSAT Evaluation 2024: Solver and Benchmark Descriptions*, page 8, 2024. URL https://hdl.handle.net/10138/584878.

Jeremias Berg, Matti Järvisalo, Ruben Martins, Andreas Niskanen, and Tobias Paxian, editors. *MaxSAT Evaluation 2024: Solver and Benchmark Descriptions*, Department of Computer Science Report Series B, 2024. University of Helsinki, Department of Computer Science.

Max A. Betjes, Rutger N. U. Kok, Sander J. Tans, and Jeroen S. van Zon. Cell tracking with accurate error prediction. *Nature Methods*, 22(11):2400–2410, October 2025. ISSN 1548-7105. doi: 10.1038/s41592-025-02845-6.

Jordão Bragantini, Ilan Theodoro, Xiang Zhao, et al. Ultrack: pushing the limits of cell tracking across biological scales. *Nature Methods*, 22(11):2423–2436, August 2025. ISSN 1548-7105. doi: 10.1038/s41592-025-02778-0.

Ilana L. Brito and Eric J. Alm. Tracking strains in the microbiome: Insights from metagenomics and models. *Frontiers in Microbiology*, 7, May 2016. ISSN 1664-302X. doi: 10.3389/fmicb.2016.00712.

Nicolas Carion, Laura Gustafson, Yuan-Ting Hu, et al. SAM 3: Segment anything with concepts, 2025. URL https://arxiv.org/abs/2511.16719.

Sascha Caron, Nadezhda Dobreva, Antonio Ferrer Sánchez, José D. Martín-Guerrero, Uraz Odyurt, Roberto Ruiz de Austri Bazan, Zef Wolffs, and Yue Zhao. Trackformers: in search of transformer-based particle tracking for the high-luminosity lhc era. *The European Physical Journal C*, 85(4), April 2025. ISSN 1434-6052. doi: 10.1140/epjc/s10052-025-14156-3.

Visesh Chari, Simon Lacoste-Julien, Ivan Laptev, and Josef Sivic. On pairwise cost for multi-object network flow tracking. *CoRR*, abs/1408.3304, 2014. URL http://arxiv.org/abs/1408.3304.

Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, page 151–158. ACM, 1971. doi: 10.1145/800157.805047.

Zhenhong Du, Hai Jiang, Xu Yang, Hao-Wen Cheng, and Jing Liu. Deep learning-assisted near-earth asteroid tracking in astronomical images. *Advances in Space Research*, 73(10):5349–5362, May 2024. ISSN 0273-1177. doi: 10.1016/j.asr.2024.02.048.

Dmitry Ershov, Minh-Son Phan, and Joanna W. Pylvänäinen. TrackMate 7: integrating state-of-the-art segmentation algorithms into tracking pipelines. *Nature Methods*, 19 (7):829–832, June 2022. ISSN 1548-7105. doi: 10.1038/s41592-022-01507-1.

Yuanming Hu, Tzu-Mao Li, Luke Anderson, Jonathan Ragan-Kelley, and Frédo Durand. Taichi: a language for high-performance computation on spatially sparse data structures. *ACM Transactions on Graphics*, 38 (6):1–16, November 2019. ISSN 1557-7368. doi: 10.1145/3355089.3356506.

Q. Huangfu and J. A. J. Hall. Parallelizing the dual revised simplex method. *Mathematical Programming Computation*, 10(1):119–142, December 2017. ISSN 1867-2957. doi: 10.1007/s12532-017-0130-5.

Alexey Ignatiev, António Morgado, and João Marques-Silva. PySAT: A Python toolkit for prototyping with SAT oracles. In *SAT*, pages 428–437, 2018. doi: 10.1007/978-3-319-94144-8_26.

Alexey Ignatiev, António Morgado, and João Marques-Silva. RC2: an efficient maxsat solver. *Journal on Satisfiability, Boolean Modelling and Computation*, 11(1):53–64, 2019.

Alexey Ignatiev, Zi Li Tan, and Christos Karamanos. Towards universally accessible SAT technology. In *SAT*, pages 4:1–4:11, 2024. doi: 10.4230/LIPICS.SAT.2024.16.

Khuloud Jaqaman, Dinah Loerke, Marcel Mettlen, Hirotaka Kuwata, Sergio Grinstein, Sandra L Schmid, and Gaudenz Danuser. Robust single-particle tracking in live-cell time-lapse sequences. *Nature Methods*, 5(8):695–702, July 2008. ISSN 1548-7105. doi: 10.1038/nmeth.1237.

Richard M. Karp. *Reducibility among Combinatorial Problems*, page 85–103. Springer US, 1972. ISBN 9781468420012. doi: 10.1007/978-1-4684-2001-2_9.

Michał Karpiński and Marek Piotrów. Incremental encoding of pseudo-boolean goal functions based on comparator networks. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 519–535. Springer, 2020.

Wenhan Luo, Junliang Xing, Anton Milan, Xiaoqin Zhang, Wei Liu, and Tae-Kyun Kim. Multiple object tracking: A literature review. *Artificial Intelligence*, 293:103448, April 2021. ISSN 0004-3702. doi: 10.1016/j.artint.2020.103448.

Klas E. G. Magnusson, Joakim Jalden, Penney M. Gilbert, and Helen M. Blau. Global linking of cell tracks using

the viterbi algorithm. *IEEE Transactions on Medical Imaging*, 34(4):911–929, April 2015. ISSN 1558-254X. doi: 10.1109/tmi.2014.2370951.

Martin Maška, Vladimír Ulman, Pablo Delgado-Rodriguez, et al. The cell tracking challenge: 10 years of objective benchmarking. *Nature Methods*, 20(7): 1010–1020, May 2023. ISSN 1548-7105. doi: 10.1038/s41592-023-01879-y.

Marius Pachitariu, Michael Rariden, and Carsen Stringer. Cellpose-sam: superhuman generalization for cellular segmentation. May 2025. doi: 10.1101/2025.04.28.651001.

S. Pan, Y. Wang, S. Cai, J. Li, W. Zhu, and M. Yin. CASHWMaxSAT-disjcad: Solver description. In *MaxSAT Evaluation 2024: Solver and Benchmark Descriptions*, page 25, 2024. URL https://hdl.handle.net/10138/584878.

Shiwei Pan, Yiyuan Wang, and Shaowei Cai. An efficient core-guided solver for weighted partial maxsat. In James Kwok, editor, *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI-25*, pages 2647–2656. International Joint Conferences on Artificial Intelligence Organization, 8 2025. doi: 10.24963/ijcai.2025/295. Main Track.

Daniel R. Parisi, Alan G. Sartorio, Joaquín R. Colonnello, Angel Garcimartín, Luis A. Pugnaloni, and Iker Zuriguel. Pedestrian dynamics at the running of the bulls evidence an inaccessible region in the fundamental diagram. *Proceedings of the National Academy of Sciences*, 118(50), December 2021. ISSN 1091-6490. doi: 10.1073/pnas.2107827118.

M. Piotrów. UWrMaxSat entering the MaxSAT evaluation 2024. In *MaxSAT Evaluation 2024: Solver and Benchmark Descriptions*, pages 27–28, 2024. URL https://hdl.handle.net/10138/584878.

Sartaj Sahni. Computationally related problems. *SIAM Journal on Computing*, 3(4):262–279, December 1974. ISSN 1095-7111. doi: 10.1137/0203021.

Jan Schering and Johannes Textor. TaichiCPM: A parallel GPU-based CPM framework built on TaichiLang, 2026. URL https://doi.org/10.5281/zenodo.21258321.

Yossi Shiloach. Another look at the degree constrained subgraph problem. *Information Processing Letters*, 12 (2):89–92, April 1981. ISSN 0020-0190. doi: 10.1016/0020-0190(81)90009-0.

François Simon, Chris H. Wiggins, and Lucien E. Weiss. Analyzing single-molecule dynamics with both complex types of motion and complex transition kinetics: Benchmarking of exatrack. January 2026. doi: 10.64898/2026.01.22.700663.

Roy Streit, Robert Blair Angle, and Murat Efe. Tracking a variable number of objects. In *Analytic Combinatorics for Multiple Object Tracking*, pages 81–112. Springer, 2020.

Carsen Stringer, Tim Wang, Michalis Michaelos, and Marius Pachitariu. Cellpose: a generalist algorithm for cellular segmentation. *Nature Methods*, 18(1):100–106, December 2020. ISSN 1548-7105. doi: 10.1038/s41592-020-01018-x.

András Szabó and Roeland M. H. Merks. Cellular Potts modeling of tumor growth, tumor invasion, and tumor evolution. *Frontiers in Oncology*, 3, 2013. ISSN 2234-943X. doi: 10.3389/fonc.2013.00087.

Craig A. Tovey. A simplified np-complete satisfiability problem. *Discrete Applied Mathematics*, 8(1):85–89, April 1984. ISSN 0166-218X. doi: 10.1016/0166-218x(84)90081-7.

Vladimír Ulman, Martin Maška, Klas E. G. Magnusson, et al. An objective comparison of cell-tracking algorithms. *Nature Methods*, 14(12):1141–1152, December 2017. ISSN 1548-7105. doi: 10.1038/nmeth.4473.

Congchao Wang, Yizhi Wang, Yinxue Wang, Chiung-Ting Wu, and Guoqiang Yu. *muSSP: efficient min-cost flow algorithm for multi-object tracking*. Curran Associates Inc., Red Hook, NY, USA, 2019.

Inge MN Wortel and Johannes Textor. Artistoo, a library to build, share, and explore simulations of cells and tissues in the web browser. *eLife*, 10, 2021. doi: 10.7554/elife.61288.

Inge MN Wortel, Jérémy Postat, Mihaela Mihaylova, Mauricio Merino, Aanya Bhagrath, Aysha Cerf, Maryl Harris, Lin Wouters, Lucas E Wiebke, Daniel R Parisi, Judith N Mandl, and Johannes Textor. Cooperative motility emerges in crowds of t cells and prevents jamming. October 2024. doi: 10.1101/2024.10.21.618803.

Zhengfu Zhang, Hongbo Chen, Ming Hu, and Dapeng Wang. Single-molecule tracking of reagent diffusion during chemical reactions. *Journal of the American Chemical Society*, 145(19):10512–10521, 2023.

A PROOF OF LEMMA 1

This follows from an argument by Tovey [1984]. We are given an arbitrary SAT formula ϕ in CNF. First, as is well known, we can transform ϕ into 3-CNF, that is, an equivalent formula ϕ' in which each clause contains exactly 3 variables. Next, we can transform ϕ' to a new formula ϕ'' in which no variable occurs more than three times [Tovey, 1984]. The new formula ϕ'' contains new clauses that contain only two variables, which for our purpose is fine. Now, if any variable occurs only in negated or only in nonnegated form in ϕ'', we can ignore it when determining satisfiability; that is, there is another formula ϕ''' that is satisfiable if and only if ϕ'' is satisfiable and does not contain any variable in only negated or only nonnegated form. This proves Lemma 1.

B DETAILS OF CELL TRACKING DATASET

In our cell tracking dataset, we have tracked cells moving in horizontal channels on microfabricated devices. In each video, multiple channels are visible and recorded at the same time (see Figure 1). The length of the videos varies, as does the width of the channels and the number of cells in each channel. Table 1 shows some statistics about numbers of cells, frames, and detections in each of the 46 videos in the dataset.

C DETAILS FOR MIN-COST FLOW ALGORITHM

This approach departs from the same graph structure as used by the Viterbi approach, but finds the global optimum via a minimum-cost flow formulation. Essentially, this means we need to modify the weights such that large negative values represent high rewards, and compute a flow with "vertex capacity" such that only 1 unit can pass through each vertex. **Tracking graph** (G_{flow}): We build G_{flow} from G_{vit} by splitting each detection node d into an in-node d^- and an out-node d^+ joined by an internal edge of weight 0. All incoming edges into d are redirected to d^-; all outgoing edges originate from d^+. More formally, $V = \{d^-, d^+ \mid d \in D\} \cup \{s, t\} \cup \{y_1, \ldots, y_{T-1}\} \cup \{x_2, \ldots, x_T\}$ and $E = \{s \to y_1, x_T \to t\} \cup \{s \to d^- \mid d \in D_1\} \cup \{d^+ \to t \mid d \in D_T\} \cup \{d_t^+ \to x_{t+1} \mid d \in D_t, t < T\} \cup \{y_t \to d^- \mid d \in D_t, t > 1\} \cup \{d_{t,i}^+ \to d_{t+1,j}^- \mid \|d_{t,i} - d_{t+1,j}\|^2 < \tau\} \cup \{d^- \to d^+ \mid d \in D\}$. **Weights:** Edge weights between subsequent detection nodes are set to $\|d_{t,i} - d_{t+1,j}\|^2 - \tau$; all other edge weights are set to 0. I.e., we use the negation of the corresponding weights in G_{sat}. **Capacities:** To enforce the input and output restrictions, we set a unit capacity on the inner-detection edges: $c(d^- \to d^+) = 1$. All other edges have unlimited capacity.

Tracking is now done by finding a minimum-cost flow (of any flow value) in G_{flow}. This is done by repeatedly searching for flow-augmenting minimum-cost paths in the residual graph, until no such path of negative cost can be found anymore. The tracks can then be read off the flow network by moving along saturated inter-detection edges. By construction, such a min-cost flow corresponds to a unique max-cost subgraph $G'_{\text{sat}} \subseteq G_{\text{sat}}$.

D DETAILS OF CELLULAR POTTS SIMULATION FRAMEWORK

The cellular Potts model (CPM) is an established framework in computational biology that is used to simulate diverse scenarios including morphogenesis, tumor growth, the movement of immune cells, and many others. For details on the CPM, we refer to Wortel and Textor [2021] and references therein. Here we only give a brief, largely informal description of the key model components and parameter settings. We also provide code in the supplementary material.

A CPM is similar to a cellular automaton (CA) in that it consists of a lattice whose elements can have different values and interact with their neighbours. However, whereas a "cell" in a CA is a single lattice site, cells in the CPM consist of *multiple* lattice sites with the same value (typically an integer).

A simulation in the CPM consists of a series of so-called "copy attempts" in which we repeatedly consider two random connected lattice sites i, j and consider what would happen if the value at i would be copied to j ("copy attempt"). This happens by considering an energy function (Hamiltonian) H, which maps lattice states to energy values, both before and after each copy attempt, leading to an energy difference ΔH. The copy attempt is then executed with Boltzmann probability

$$P_{\text{copy}} = \begin{cases} 1 & \Delta H \leq 0 \\ \exp(-\Delta H / T) & \Delta H > 0 \end{cases} \tag{2}$$

where T (temperature) is a parameter; we use the value $T = 20$. The fundamental time unit in the simulation is a "Monte Carlo step" (MCS), which is a number of copy attempts equal to the number of lattice points. The randomness in this simulation leads to fluctuations of the cell shape and random movements.

In our simulation, we use the following energy function:

$$H = \underbrace{\sum_{\text{neighbours i,j}} J\left(\sigma(i), \sigma(j)\right) \cdot \delta(i,j) +}_{\text{adhesion}}$$

$$\underbrace{\sum_{\text{cells } \sigma > 0} \lambda_V(\sigma) \left(V(\sigma, t) - V_{\text{target}}(\sigma)\right)^2 +}_{\text{volume conservation}}$$

Video	Channels	Frames	Cells / channel			Detections / channel		
			Min	Max	Mean	Min	Max	Mean
1	18	181	1	12	3.6	58	917	341.4
2	26	181	1	10	2.7	9	660	157.0
3	27	271	1	19	5.9	57	2700	781.1
4	31	271	1	13	4.0	42	1305	415.9
5	30	271	1	10	5.0	13	954	436.5
6	30	180	1	15	6.2	19	1609	558.0
7	30	180	3	19	9.9	64	2050	806.9
8	21	271	1	7	3.0	58	1248	317.8
9	29	271	1	14	4.0	29	2925	638.7
10	16	271	2	13	7.6	47	2740	1332.9
11	29	271	1	12	5.4	42	2072	779.3
12	29	271	1	26	9.2	119	5007	1536.0
13	13	181	1	5	2.6	27	864	278.2
14	14	271	1	7	2.0	32	984	280.4
15	31	180	1	29	12.2	13	3506	1206.5
16	28	180	1	27	9.1	22	3536	1017.3
17	24	181	1	7	3.6	8	514	253.4
18	27	271	1	16	4.3	30	1664	452.4
19	30	271	1	16	7.1	74	2059	713.8
20	31	180	1	37	17.8	93	4171	1894.0
21	27	180	1	15	5.1	26	898	370.4
22	16	271	1	3	1.9	67	697	271.0
23	27	271	1	8	2.7	5	813	251.9
24	29	180	1	23	5.2	31	2792	467.9
25	17	271	1	8	2.3	36	1023	357.9
26	27	271	1	16	7.2	71	1694	673.6
27	31	180	3	29	14.7	95	3559	1307.7
28	23	180	1	10	3.9	21	1001	298.2
29	24	180	1	8	3.4	39	1274	316.8
30	26	180	1	16	7.0	35	1748	655.9
31	27	180	1	12	4.7	70	1624	425.0
32	17	271	1	5	1.8	5	444	215.4
33	34	271	1	7	3.1	71	544	228.0
34	11	271	1	3	1.4	34	586	231.5
35	30	271	1	17	6.4	68	2646	887.0
36	33	271	4	44	20.3	126	8157	2376.2
37	28	180	1	12	6.5	66	1271	597.4
38	13	271	1	9	3.5	132	1558	451.8
39	31	180	1	15	6.8	95	1333	509.1
40	27	271	1	12	5.1	37	944	451.8
41	31	180	1	18	6.7	59	1904	649.0
42	31	180	2	9	5.0	58	919	414.3
43	20	180	6	21	11.4	465	3452	1319.7
44	29	120	1	13	4.2	12	957	209.3
45	30	180	1	8	3.4	31	1347	497.5
46	30	180	1	23	7.7	24	2938	797.9
Total	1183	10226			7629			800130

Table 1: Summary statistics for our cell tracking dataset. Overall, the dataset contains 7629 hand-checked ground truth tracks across 800,130 annotations.

$$\underbrace{\sum_{\text{cells } \sigma>0} \lambda_P(\sigma)\left(P(\sigma,t) - P_{\text{target}}(\sigma)\right)^2}_{\text{perimeter conservation}}$$

The *adhesion* constraint defines to what extent cells make contact with each other. We use the following matrix:

$$J = \begin{pmatrix} 40 & 100 \\ 100 & 100 \end{pmatrix}$$

These numbers imply that nondividing cells prefer to stay in contact with each other rather than getting in contact with dividing cells.

The volume and perimeter constraint define cell size and shape (a small perimeter/volume ratio will generate a roughly circle-shaped cell). We do not differentiate between the two cell types here and use $V = 152, P = 145$. The multipliers λ_V, λ_P define the relative importance of these constraints; we use $\lambda_V = 50, \lambda_P = 2$.

We have now defined the entire simulation except for three processes:

Initialization. We use a grid of size 256×256, and seed single-pixel cells 12 pixels per dimension apart. We assign all cells to the type "nondividing", except for the middle cell.

Cell division. For each dividing cell, we consider once per Monte Carlo step whether a division takes place. If the cell's current volume is 80% of its target volume V, then the cell is split into two daughter cells with a probability (division rate) of 0.005. Splitting happens by computing the major and minor axes of the cell pixel's coordinates, and splitting along the minor axis such that half of the pixels get assigned to each child.

Cell death. For each nondividing cell, we consider once per Monte Carlo step whether the cell dies due to a lack of space. If the volume of the cell is less than 80% of its target, we kill the cell (by setting all its lattice sites to 0) with probability 0.2.

Both cell division and cell death are only applied after a "burnin phase" of 25 MCS, allowing all nondividing cells enough time to reach their target volume. Cells are tracked starting after 750 MCS and for an additional 1000 MCS. Cell coordinates (centers of mass) are recorded every 10 MCS, yielding 100 frames in total per simulated video. Coordinates are multiplied by 10 to make cell-cell distances similar to those in our real data.

For this project we used a parallelized implementation of the CPM currently being developed in our group. The source code is available on Zenodo Schering and Textor [2026].

E GUIDE TO SUPPLEMENTARY MATERIAL

As supplementary material to this manuscript, we provide the following:

- The cell tracking dataset used in Section 5 (file `tracks/all_gt.csv.gz`).
- Tracks of the simulated dividing cells used in Section 6 (file `tracks/dividing-cell.csv.gz`).
- CSV files containing the performance measurements shown in the figures (folder `data/`)
- Python implementations of the Viterbi, Max-SAT, and Min-cost flow algorithms used in Section 5 along with the evaluation scripts we used to measure their performance on the cell tracking data (folder `scripts/`)