Optimal Decoding of Hidden Markov Models With Consistency Constraints

Alexandre Dubray[1][0000−0002−3302−870X], Guillaume Derval[2], Siegfried Nijssen[1], and Pierre Schaus[1]

[1] Institute of Information and Communication Technologies, Electonics and Applied Mathematics (ICTEAM), UCLouvain, Belgium
{first.second}@uclouvain.be
[2] Department of Electrical Engineering and Computer Science, ULiège, Belgium
gderval@uliege.be

Abstract. Hidden Markov Models (HMM) are interpretable statistical models that specify distributions over sequences of symbols by assuming these symbols are generated from hidden states. Once learned, these models can be used to determine the most likely sequence of hidden states for unseen observable sequences. This is done in practice by solving the shortest path problem in a layered directed acyclic graph using dynamic programming. In some applications, although the hidden states are unknown, we argue that it is known that some observable elements must be generated from the same hidden state. Finding the most likely hidden state in this contrained setting is however a hard problem. We propose a number of alternative approaches for this problem: an Integer Programming (IP), Dynamic Programming (DP), a Branch and Bound (B&B) and a Cost Function Network (CFN) approach. Our experiments show that the DP approach does not scale well; B&B scales better for a small number of constraints imposed on many elements and CFNs are the most robust approach when many smaller constraints are imposed. Finally, we show that the addition of consistency constraints indeed allows to better recover the correct hidden states.

Keywords: Hidden Markov Model · Constrained Viterbi · Branch and Bound · Cost Function Networks

1 Introduction

Hidden Markov Models (HMM) are a class of probabilistic models in which it is assumed that symbols in sequences are generated independently from each other, from hidden states. For a sequence of observed data, it is assumed that there is a sequence of hidden states that generated it with a given probability; determining the hidden states that generated the symbols is here useful in understanding the data. HMMs have been used in various real-world applications such as protein structure prediction [15], trajectory mining [16], speech recognition [10] or human activity recognition [6, 9]. The decoding problem in HMMs is to find the most likely sequence of hidden states, for an observed sequence,

and is usually solved by the Viterbi algorithm [19], which has a polynomial run time. The decoding problem in HMMs can be reduced to solving the shortest path problem in a layered directed acyclic graph (DAG). Since in such graphs the shortest and longest path problems are equivalent, and the applications in Section 5 are concerned with HMMs, we will refer to this problem as the most likely path problem in the rest of this paper. However the presented methods also work for layered DAGs not associated with HMMs.

In this work we argue that in many applications, a better decoding can be found by exploiting background knowledge stating that symbols in a given sequence must have been generated from the same hidden state. Such connections between sequences are not taken into account in classical HMM decoding, in which multiple sequences are decoded independently. However, in practice such background knowledge exists. For example, in part-of-speech tagging, it is likely that within one sentence, multiple occurrences of the same uncommon word must be given the same tag. Another application can be found in the analysis of traffic data, where we consider a truck state assignment problem as an example. In this task, constraints are imposed stating that trucks in the same area at the same time must be labeled identically. Finally, in human activity recognition problems, natural consistency constraints also arise when activities are registered near to each other (e.g., same room, same sensor). To take into account the background knowledge that symbols in the sequence must originate from the same state, the Viterbi algorithm cannot be used anymore.

The rest of this paper is organized as follows. The decoding problem under constraints is presented in Section 3. Then, three of the four approaches for solving the problem are presented in more detail: Dynamic Programming, a Branch and Bound and a Cost Function Network approach. These methods are compared in Section 5 as well as the benefit of the consistency constraints. We conclude in Section 6.

2 Related Work

As we will see, the decoding problem can be seen as a problem of finding the most likely path in a DAG under logical constraints between nodes or groups of nodes. This problem has been studied in multiple contexts. In the case of HMMs, and more generally conditional random fields, Roth et al. solved the decoding problem using Integer Programming and proposed constraints useful for the semantic role labeling problem [11]. With a focus on the alignment of biological sequences, Christiansen et al. proposed a constrained version of HMMs [3]. They implemented various constraints in the PRISM language [13], but no consistency constraint between sequence elements.

We will show in this work that finding the most likely sequence of hidden states can be expressed as a weighted constraint satisfaction problem, also known as a Cost Function Network (CFN). In a CFN, the goal is to find an assignment to discrete decision variables such that a sum of functions defined on these variables

is optimized while respecting defined constraints. In this work we will rely on dedicated solvers for CFNs, such as Toulbar2 [4, 8].

In [17], for finding longest paths in a general DAG, the logical constraints are represented in a Binary Decision Diagram (BDD) and a dynamic program, taking into account the BDD nodes, is designed to find the optimal solution. In [20], consistency constraints are imposed between words to improve logical reasoning from sentences in natural language. They use Dual Decomposition [12] to solve the problem, which solves a Lagrangian relaxation of the problem; in contrast to our approach, however, this approach does not guarantee finding the optimal solution.

3 Problem Definition

In this section we formalize the problem of finding the most likely path in a layered DAG under consistency constraints. Solving this problem allows to also solve the HMM decoding problem. We first introduce the notation as well as the notions of layer and consistency constraints in a DAG, then express the problem of finding the most likely path in it.

3.1 Most Likely Path in a Layered DAG with Consistency Constraints

We define the HMM decoding problem over labeled Directed Acyclic Graphs (DAGs). Let $G = (V, E)$ be a graph with V the set of nodes and E the set of edges. Each node $v \in V$ has a label, from a set $\mathcal{L}$, denoted l_v and V is divided into T layers $L_1, \ldots, L_T$ such that $V = \bigcup_{j=1,\ldots,T} L_j$ and $L_i \cap L_j = \emptyset \ \forall i \neq j$. In each layer, no two nodes have the same label. Thus, when clear from the context, a node can be identified by its label. We denote by $e = (l, l', t) \in E$ an edge from the node with label l at layer L_t to the node with label l' at layer L_{t+1} $(1 \leq t < T)$ with weight w_e, where weights can be both positive and negative. In the HMM decoding problem, each layer has the same number of nodes representing the hidden states. An example of such a graph is shown in Figure 1.

A path in G from L_1 to L_T selects one node per layer and can be identified by the sequence of node labels on the path. More formally, let $P = \langle P_1, \ldots, P_T \rangle \in \mathcal{L}^T$ be a path from L_1 to L_T such that $P_i \in L_i$. The cost of P, is the sum of the weights of the arcs in the path: $\sum_{t=1}^{T-1} w_{(P_t, P_{t+1}, t)}$.

A consistency constraint is specified in our work by identifying a set of layers for which the same label must be selected in each layer of the path. More formally, $C = \{C_1, \ldots, C_k\}$ are k consistency constraints with $C_i = \{c_1^i, \ldots, c_{k_i}^i\} \subseteq \{1, \ldots, T\}$ and $C_i \cap C_j = \emptyset$ for $i \neq j$. The set of all constrained layers is denoted $L_C = \bigcup_{i=1}^{k} C_i$. We also define a vector $c \in \{0, \ldots, k\}^T$ that gives for each layer the index of its constraint or 0 if the layer is unconstrained. For example, in Figure 1 we have $c = \langle 0, 0, 1, 0, 2, 0, 2, 0, 1, 0, 0 \rangle$. A path P is said to be consistent

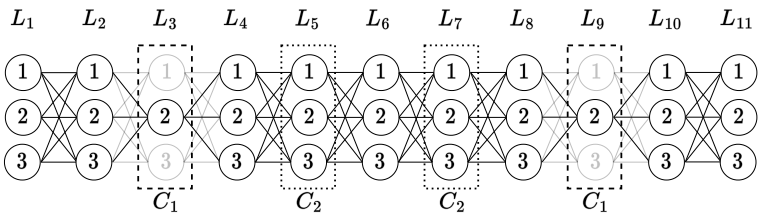

Fig. 1. Example of a layered DAG for the decoding problem in a HMM with three hidden states and two consistency constraints. The edges in the DAG are oriented from left to right and the labels on the nodes represent the hidden states. In this example, the constraint C_1 has node 2 assigned to it, hence the other nodes are faded.

if all the consistency constraints are respected. The problem of finding the most likely consistent path is thus formalized as follows:

$$P^\star = \arg\max_{P \in \mathcal{L}^T} \sum_{t=1}^{T-1} w_{(P_t, P_{t+1}, t)} \tag{1}$$

$$\text{s.t. } P_{c_1^i} = \ldots = P_{c_{k_i}^i} \qquad \forall C_i \in C \tag{2}$$

The importance of this problem to HMM decoding is that an instance of this problem, including the DAG and its weights, can be constructed for a specific HMM decoding problem on a sequence of symbols. Note that the problem defined by Equation (1)-(2) is NP-hard, as we showed in a technical report [5].

4 Solving the Problem

In this section, three of the four approaches to solve the problem defined by Equations (1)-(2) are explained. We omit the IP formulation as it is very similar to the one presented in [11] but with equality constraints. First a Dynamic Programming approach (DP) is introduced, followed by a Branch and Bound (B&B) method and finally a model based on cost-function networks (CFN) is presented.

4.1 Dynamic Programming

For solving the unconstrained problem, the Viterbi algorithm [19] is the classical dynamic programming approach. The recurrence relation computes the value of the most likely path from L_1 to a node $i \in L_t$ from layer L_{t-1} and stores it in a $T \times |\mathcal{L}|$ table. The entries of the table are computed as follows:

$$V[t,i] = \begin{cases} 0 & \text{if } t = 1 \\ \max_{j \in L_{t-1}} V[t-1,j] + w_{(j,i,t-1)} & \text{otherwise} \end{cases} \tag{3}$$

and the value of the most likely path is given by $\max_{i \in L_T} V[T, i]$.

This equation uses the fact that the graph is organized into layers and a path ending at layer L_t always comes from layer L_{t-1}. Thus, the most likely path to a node $i \in L_t$ is one of the most likely paths to a node in L_{t-1} plus the edge to i. However, when adding consistency constraints, this equation does not work anymore because it does not take into account consistency. We resolve this by adding assignments of labels to consistency constraints in the DP; by assigning a label to a constraint we assign the same label to all layers in the constraint. Let $P_C = \langle P_{C_1}, \ldots, P_{C_k} \rangle \in (\mathcal{L} \cup \{\perp\})^k$ be an assignment of labels to the consistency constraints with $P_{C_i} = \perp$ if no label is assigned to C_i. Then if $P_{C_i} \neq \perp$, the path from L_1 to L_T must pass through P_{C_i} for every layer L_t with $t \in C_i$. We define the assignment operator $P_C|_{j,l}$ which assigns l to P_{C_j}. The values of the most likely paths can now be stored in a $T \times |\mathcal{L}| \times k^{|\mathcal{L}|}$ table, taking into account the possible assignments of labels to the constraints. The entries in the table are computed as follows:

$$V[t, i, P_C] = \begin{cases} 0 & \text{if } t = 1 \\ \max\limits_{l \in L_{t-1}} V[t-1, l, P_C] + w_{(l,i,t-1)} & \text{if } L_{t-1} \notin L_C \\ V[t-1, P_{C_{c[t]}}, P_C] + w_{(P_{C_{c[t]}}, i, t-1)} & \text{if } L_{t-1} \in L_C \wedge P_{C_{c[t]}} \neq \perp \\ \max\limits_{l \in L_{t-1}} V[t-1, l, P_C|_{c[t],l}] + w_{(l,i,t-1)} & \text{if } L_{t-1} \in L_C \wedge P_{C_{c[t]}} = \perp \end{cases} \tag{4}$$

The first two cases of Equation (4) are the same as Equation (3) because there are no constraints to consider. However when the layer L_{t-1} is constrained, there are two situations. If there is a choice for this constraint in P_C, then in order to be consistent with P_C, the path must pass by it. In that case there is no need to consider the other nodes in the layer. However, when there is not yet a node assigned to this constraint, then every node $j \in L_{t-1}$ must be considered to compute the most likely path. In this case, the P_C vector is updated to reflect the choice made.

4.2 Branch and Bound

Let $P_C \in (\mathcal{L} \cup \{\perp\})^k$ be, as for the DP, a vector of node assignments for the consistency constraint. The search starts from the vector $\langle \perp, ..., \perp \rangle$. The idea of this method is to branch on the P_{C_i} values and to compute the most likely path from L_1 to L_T while being consistent with P_C. Initially some P_{C_i} are unassigned; as long as the constraint is unassigned, we ignore the constraint and the cost is an upper bound on the optimal solution in the branch. An example is shown on Figure 1 where there are two consistency constraints and $P_C = \langle 2, \perp \rangle$. The most likely path from L_1 to L_{11} can be seen as the most likely path from L_1 to L_3, then L_3 to L_9 and finally from L_9 to L_{11}. As long as $P_{C_2} = \perp$, we ignore constraint C_2 and an upper bound on the most likely path is obtained.

In practice, a $T \times |\mathcal{L}|$ array, denoted V, is used to store the values of the most likely paths from the layers in L_c to the other layers. At the root of the search tree, the V array is filled with Equation (3) since there are no consistency

constraints imposed. When a value P_{C_i} is assigned, the whole table does not need to be recomputed. Let us look at Figure 1 as an example. When the search assigns $P_{C_1} = 2$, the layers constrained by C_1 act as new source layers. The computed values, in V, for layers L_1 to L_3 still represent the values a recursive equation computes for the most likely path from L_1 to L_3 and thus, need not be recomputed. Let us assume now that the search assigns $P_{C_2} = 1$. The values in V for L_1 to L_5 and L_9 to L_{11} are still valid, and only the values from L_6 to L_8 need to be updated.

Notice that when all the edges have a negative weight, as for the HMM decoding problem, then the values in the V array can be computed between consistency constraints, even if not assigned. In the example in Figure 1, the consequence is that when P_{C_1} is set to 2, the values from L_3 are only computed until L_5 and not L_9. Since the edges only have negative weights, this gives a less tight upper bound on the optimal solution, but is faster to compute.

4.3 Cost Function Networks

In Cost Function Networks (CFNs) [4], a set of functions is defined, each of which maps a subset of the assignment in P_c to a cost. The goal is to find an assignment of P_C such that the sum of the function's cost (evaluated on the assignment) is minimal. We model our problem in CFNs by dividing the graph into segments between successive constrained layers; a function is defined on each of these segments. These functions map the choice for the constrained layers at the start and end of the segment (i.e., a partial assignment of P_C) to the value of the most likely path on the segment, consistent with P_C. For a full assignment of P_C, the sum of the most likely path on the segments gives the value of the most likely path in the full graph.

More formally, let $L_t, L_{t'}$ such that $c[t] \neq c[t']$ and $\nexists\ t'' : t < t'' < t' \wedge c[t] \neq c[t'']$ be two successive constrained layers of different consistency constraints. A function $f_{t,t'} : \mathcal{L} \times \mathcal{L} \mapsto \mathbb{R}$ is defined on the segment between L_t and $L_{t'}$, mapping each choice of $c[t]$ and $c[t']$ to the value of the most likely path from L_t to $L_{t'}$ consistent with the choices. For every node $u \in L_t$ and $v \in L_{t'}$, a simple dynamic program finds the value of the most likely path between u and v and stores it in a $\mathcal{L} \times \mathcal{L}$ table. Since the table fully defines the function, $f_{t,t'}$ is used to refer to the function as well as its table of values and $f_{t,t'}(P_C)$ refers to the value associated with the choices for L_t and $L_{t'}$ in P_C.

Let $L_{t_1}, \ldots, L_{t_m}$ be all the constrained layers. Without loss of generality, we assume that they are sorted in chronological order so that $t_1 < t_2, t_2 < t_3, \ldots$ Let $F = \{f_{start}, f_{end}, f_{t_1,t_2}, \ldots, f_{t_{m-1},t_m}\}$ be the functions, as defined above, for each segment of the graph and two additional special functions. The $f_{start} : \mathcal{L} \mapsto \mathbb{R}$ function maps, for each choice for $c[t_1]$, the value of the most likely path from L_1 to L_{t_1}. The function f_{end} is defined in the same way for the layer L_{t_m} to L_T. The value of a path P_C, which we wish to optimize, is then given by

$$f_{start}(P_C) + f_{end}(P_C) + \sum_{i=1}^{m-1} f_{t_i,t_{i+1}}(P_C). \tag{5}$$

Proportion of constraints	0.00	0.10	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90	1.00
DP	6.00	360.10	324.40	299.30	278.60	255.70	232.70	211.90	192.20	173.90	157.80
IP	588.20	841.67	911.33	926.60	927.20	1007.20	1001.25	1169.67	1061.67	1141.00	1047.00
B&B	**2.63**	**3.59**	**3.59**	**3.55**	**3.43**	**3.42**	**3.34**	**3.31**	**3.27**	**3.21**	**2.93**
CFN	-	35.94	34.28	32.43	30.92	28.80	26.85	25.15	23.00	20.80	18.93

Table 1. Execution time in seconds of the methods in function of the proportion of constraints in the model on the truck state assignment problem. The entries for the CFN method represent the time needed to compute the functions plus the optimization time by Toulbar2.

Dedicated solvers for CFN are designed to find the assignment to P_C such that the value of Equation (5) is minimal. From this optimal assignment we can easily recover the solution using a dynamic program.

5 Experimental Results

In this section we analyze the run time of the methods presented in Section 4 on two different HMM applications with different characteristics in terms of sequence lengths and number of consistency constraints. We finish this section by analyzing, on a third application, the impact of the consistency constraints on the output of the decoding problem. The IP is solved with the Gurobi solver [7] and for the B&B we use the variation of the algorithm that supports only negative weights, as we experiment only with HMMs and it gives, in our experiments, the best results.[3]. For the CFN method the Toulbar2 solver [14] is used.

Truck Trajectory Mining HMMs have been used to identify activity stops in truck trajectories [16]. Four hidden states represent if the truck is driving, in a traffic jam, resting or doing work-related actions. In this context, it is natural to assume that trucks located in similar geographical areas do the same activity. Four consistency constraints are created based on the type of point (stop or driving) in some geographical areas (e.g., rest areas, highways).

We experiment on a data set of trajectories of trucks described in [1], which contains roughly 6 million data points (and thus as many layers in the graph). We successively kept a given percentage of each constraint in order to evaluate the impact of the constraints size on the run time.

Table 1 shows the run time of the methods with different proportions of states included in constraints, where constraints are larger if they involve more states. The run time of the DP and IP methods both increases with the size of the constraints. For the DP method, more choices must be propagated through the recursion while in the IP model there are more linear constraints. The run time of the B&B method is stable with the constraint size. The size of the constraints only impacts the computation of the V array in each node of the

[3] The source code and the data sets can be found at https://github.com/AlexandreDubray/consistent-viterbi.

dataset	conll2000					treebank					brown				
Number of layers	25 9104					100 676					1 161 192				
number of constraints	2	3	4	5	6	2	3	4	5	6	2	3	4	5	6
DP	153.2	128.7	O.O.M	O.O.M	O.O.M	56.0	1132.0	O.O.M	O.O.M	O.O.M	1047.4	O.O.M	O.O.M	O.O.M	O.O.M
IP	97.6	137.6	86.0	128.1	130.6	34.6	33.4	32.1	32.2	47.6	485.3	480.9	790.1	O.O.M	O.O.M
B&B	**19.6**	**30.75**	290.25	T.O.	T.O.	14.95	66.94	116.93	777.78	T.O	**81.66**	1054.34	T.O.	T.O.	T.O.
CFN	52.33	72.56	**72.22**	**72.59**	**71.88**	**7.64**	**29.58**	**31.2**	**29.8**	**29.93**	294.71	**301.91**	**293.4**	**294.31**	**297.66**

Table 2. Run time in seconds of the methods in function of the number of consistency constraints for the POS tagging problem. Timeout has been set to 1 hour and is indicated by T.O. while out of memory errors are indicated by O.O.M.

search tree. As the whole array still needs to be computed in order to have a feasible solution, the impact is limited. Finally, the run time of the CFN method decreases with the size of the constraints. In that case, the run time is dominated by the computation of the local functions F. Once computed, Toulbar2 is able to find the optimal solutions in a few milliseconds. With more constraints, the segments are shorter and thus faster to compute, which makes the overall approach faster.

Overall the B&B method is the fastest on this data set because there are few constraints and few choices per constraint. Thus even if the CFN approach is much better than the DP and IP, the time needed to compute the local functions F makes it slower than B&B.

Part of Speech Tagging The goal of this application is to assign to each word of a sentence, or text, a part of speech (POS) tag. The NLTK Python package [2] provides data sets of sentences with annotated POS. We experiment on three data sets with the 12 universal POS tags and consistency constraints are imposed on layers with the same POS tag.

Table 2 shows the run time of the methods in function of the number of consistency constraints. First, let us note that only the CFN method is able to solve the problem for all numbers of constraints on all data sets. The B&B and DP methods both time-out or reach a memory limit quickly as the number of constraints increases. For the DP method, with more constraints, the number of constraint choices to propagate increases exponentially. For the B&B the search space becomes too large and the upper bound is not strong enough to prune large part of the search space to make the approach tractable. The IP methods can handle more constraints but, on the brown data set, which is larger, the amount of memory needed to model the problem is too large. For these three methods, the run time increases with the number of constraints which is expected.

On the contrary, the run time of the CFN is stable with the number of constraints and the method is the most efficient for these data sets. Adding new constraints has little impact on the time needed to compute F since it is done by computing the values between successive constrained layers (i.e., all layers of the graph are processed $|\mathcal{L}|^2$ times). In addition to that, Toulbar2 is very efficient at finding the optimal solution, in few milliseconds. Hence the total run time of the CFN method is stable with the number of constraints.

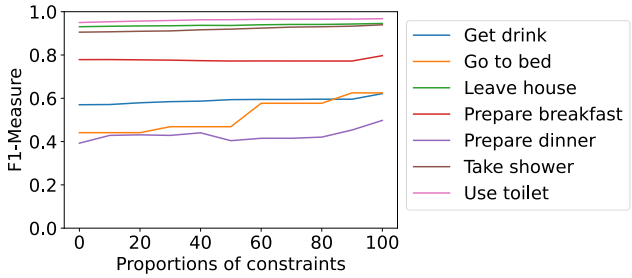

Fig. 2. F1-Measure in function of the proportion of constraints for each activity

Human Activity Recognition Finally, in this section we analyze the impact of the consistency constraints on the output of the decoding problem, using a real-world data set for Human Activity Recognition (HAR). In HAR the goal is to find which activities a person is doing based on inputs from sensors which can be placed on the person (e.g., a smartwatch) or in their environment (e.g., light sensors in the house). We use the annotated data sets as described in [18] for this experiment. These data sets provide the activities (based on the activation of sensors in their house) made by three persons for multiple days.

The F1-Measure per activity is shown in Figure 2 for one of the houses (the results are similar for the other houses). The F1-Measure was computed, for a proportion of the constraints, following the same methodology as in [9]. It can be seen that the activities are better recovered as the proportion of constraints increases. The biggest impact is on the activities that are not well recovered using a classical decoding algorithm (e.g. "Go to bed", "Prepare dinner"). The activities that have a high F1 measure when there are no constraints also benefit from the constraints, but in a less marked way.

6 Conclusions and Future Work

In many applications using Hidden Markov Models, consistency constraints between sequences can be found but are not used in the classical decoding algorithm. In this work, we formalized this problem as finding the most likely path in a layered directed acyclic graph with consistency constraints on the layers of the graph. We proposed an Integer Programming (IP), a Dynamic Program, a Branch and Bound (B&B) and a Cost Function Network method to solve the problem. We showed that Branch and Bound scales better for a few large constraints, while the CFN is better for many smaller constraints. Finally, our experiments on a real-world human activity recognition data set showed the benefit of consistency constraints.

In this work we focused on consistency constraints, imposing that the same node is selected between different layers. However in some applications, it might be acceptable to have sets of nodes that can appear together in the layers of a

consistency constraint (e.g., a non-activity stops and a rest stop, in the Truck Trajectory Mining problem). The impact of additional logical constraints on the Branch and Bound method could also be investigated.

References

1. Adam, A., Finance, O., Thomas, I.: Monitoring trucks to reveal belgian geographical structures and dynamics: From gps traces to spatial interactions. Journal of Transport Geography (2021)
2. Bird, S., Klein, E., Loper, E.: Natural language processing with Python: analyzing text with the natural language toolkit. " O'Reilly Media, Inc." (2009)
3. Christiansen, H., Have, C.T., Lassen, O.T., Petit, M.: Inference with constrained hidden markov models in prism. Theory and Practice of Logic Programming (2010)
4. Cooper, M.C., De Givry, S., Sánchez, M., Schiex, T., Zytnicki, M., Werner, T.: Soft arc consistency revisited. Artificial Intelligence (2010)
5. Dubray, A., Derval, G., Nijssen, S., Schaus, P.: On the complexity of the shortest path problem in a layered directed acyclic graph with consistency constraints (2022), 2078.1/264677
6. Fallmann, S., Kropf, J.: Human activity recognition of continuous data using hidden markov models and the aspect of including discrete data. In: UIC (2016)
7. Gurobi Optimization, LLC: Gurobi Optimizer Reference Manual (2022), https://www.gurobi.com
8. Hurley, B., O'sullivan, B., Allouche, D., Katsirelos, G., Schiex, T., Zytnicki, M., Givry, S.d.: Multi-language evaluation of exact solvers in graphical model discrete optimization. Constraints (2016)
9. Kabir, M.H., Hoque, M.R., Thapa, K., Yang, S.H.: Two-layer hidden markov model for human activity recognition in home environments. IJDSN (2016)
10. Rabiner, L.R.: A tutorial on hidden markov models and selected applications in speech recognition. Proceedings of the IEEE (1989)
11. Roth, D., Yih, W.t.: Integer linear programming inference for conditional random fields. In: ICML (2005)
12. Rush, A.M., Sontag, D., Collins, M., Jaakkola, T.: On dual decomposition and linear programming relaxations for natural language processing (2010)
13. Sato, T., Kameya, Y.: Prism: a language for symbolic-statistical modeling. In: IJCAI (1997)
14. Schiex, T., de Givry, S., Sanchez, M.: Toulbar2—an open source weighted constraint satisfaction solver. URL https://toulbar2.github.io/toulbar2 (2006)
15. Sonnhammer, E.L., Von Heijne, G., Krogh, A., et al.: A hidden markov model for predicting transmembrane helices in protein sequences. In: Ismb (1998)
16. Taghavi, M., Irannezhad, E., Prato, C.G.: Identifying truck stops from a large stream of gps data via a hidden markov chain model. In: ITCS (2019)
17. Takeuchi, F., Nishino, M., Yasuda, N., Akiba, T., Minato, S.i., Nagata, M.: Bdd-constrained a* search: A fast method for solving constrained shortest-path problems. IEICE TRANSACTIONS on Information and Systems (2017)
18. Van Kasteren, T., Noulas, A., Englebienne, G., Kröse, B.: Accurate activity recognition in a home setting. In: UbiComp (2008)
19. Viterbi, A.: Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. IEEE transactions on Information Theory (1967)
20. Yoshikawa, M., Mineshima, K., Noji, H., Bekki, D.: Consistent ccg parsing over multiple sentences for improved logical reasoning. arXiv preprint (2018)