Structured Convolutional Kernel Networks for Airline Crew Scheduling

Yassine Yaakoubi [1 2] **François Soumis** [1] **Simon Lacoste-Julien** [† 2 3]

Abstract

Motivated by the needs from an airline crew scheduling application, we introduce structured convolutional kernel networks (Struct-CKN), which combine CKNs from Mairal et al. (2014) in a structured prediction framework that supports *constraints* on the outputs. CKNs are a particular kind of convolutional neural networks that approximate a kernel feature map on training data, thus combining properties of deep learning with the non-parametric flexibility of kernel methods. Extending CKNs to structured outputs allows us to obtain useful initial solutions on a flight-connection dataset that can be further refined by an airline crew scheduling solver. More specifically, we use a flight-based network modeled as a general conditional random field capable of incorporating local constraints in the learning process. Our experiments demonstrate that this approach yields significant improvements for the large-scale crew pairing problem (50,000 flights per month) over standard approaches, reducing the solution cost by 17% (a gain of millions of dollars) and the cost of global constraints by 97%.

1. Introduction

Since crew costs are the second-highest spending source for air passenger carriers, crew scheduling is of crucial importance for airlines. The crew pairing problem (CPP) searches for a minimum-cost set of anonymous feasible pairings (rotations) from the scheduled flights, such that all flights are covered exactly once, and all airline regulations and collective agreements are respected. The complexity of this problem lies in the large number of possible pairings, as the selection of pairings at minimal cost—a large integer programming problem—cannot be performed using standard solvers. Seeking to obtain an efficient algorithm for large-scale monthly CPPs (up to 50,000 flights) and building on the column generation-based solver by Desaulniers et al. (2020), Yaakoubi et al. (2020) proposed Commercial-GENCOL-DCA, an improved solver starting with an aggregation, in clusters, of flights. The initial aggregation partition permits replacing all flight-covering constraints of flights in a cluster by a single constraint, thus allowing the solver to cope with larger instances. Initial clusters can either be extracted from the initial solution (Desaulniers et al., 2020) or given separately, as in Yaakoubi et al. (2020), where authors used convolutional neural networks (CNN) to solve the flight-connection problem (a supervised multi-class classification problem). The objective of this problem is to predict the next flight that a crew follows in its schedule given the previous flight. They used CNN to harness the spatial locality (localized spatial features) and used a similarity-based input, where neighboring factors have similar features. By passing initial clusters of flights to the CPP solver, the reported reduction of solution cost averages between 6.8% and 8.52%, mainly due to the reduction in the cost of global constraints between 69.79% and 78.11%. The cost of global constraints refers to the penalties incurred when the workload is not fairly distributed among the bases in proportion to the available personnel at each base.

However, a major weakness of their approach is that they can only produce initial clusters and not an initial solution, since they use a greedy predictor making one prediction at a time (predicting sequentially the next flight given only the previous flight). This prevents the predictor from incorporating constraints on the output and the produced solutions cannot be used as initial solutions for the solver as they are not sufficiently close to being feasible. A pairing is deemed feasible if it satisfies safety rules and collective agreement rules (Kasirzadeh et al., 2017); examples include minimum connection time between two flights, minimum rest time, and maximum number of duties in a pairing. By providing an initial solution, we not only accelerate the optimization process and calculate the feasibility of proposed pairings, but we also propose clusters similar to the initial solution, thus reducing the degree of incompatibility between current solution and proposed pairings (Elhallaoui et al., 2010).

In this paper, we address this lack of constraint modeling, while still enabling the use of a convolutional architecture. For this purpose, we investigate the convolutional kernel

[1]GERAD, Polytechnique Montréal, Canada [2]Mila, Canada [3]Department of Computer Science and Operations Research, Université de Montréal, Canada † Canada CIFAR AI Chair. Correspondence to: Yassine Yaakoubi <YassineYaakoubi@outlook.com>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

network (CKN), an approximation scheme similar to CNN proposed by Mairal et al. (2014). To bypass the major limitations in Yaakoubi et al. (2020), we incorporate local constraints on the outputs (imposing that each flight has to be preceded by at most one flight). We harness the spatiotemporal structure of the CPP problem by combining kernel methods and structured prediction. The outputs start the optimizer and solve a large-scale CPP, where small savings of a mere 1% translate into an increase of annual revenue for a large airline by dozens of millions of dollars. Note that, to the best of our knowledge, we are not aware of any ML approach that can directly solve the CPP (which has complex airline-dependent costs and constraints that are not necessarily available to the ML system at train time).

We thus consider instead to use the ML system to propose good initial clusters and an initial solution for the CPP solver. The results of training on the flight-connection dataset (Yaakoubi et al., 2019), a flight-based network structure modeled as a general conditional random field (CRF) graph, demonstrate that the proposed predictor is more suitable than other methods. Specifically, it is more stable than CNN-based predictors and extensive tuning is not required, in that no Bayesian optimization (to find a suitable configuration) is needed, as we observe in our experiments. This is crucial to integrate ML into a solver for the CPP or any real-world scheduling problem. Note that unlike recurrent neural networks (RNNs) or neural networks by (Yaakoubi et al., 2020) which cannot produce initial solutions that are sufficiently close to being feasible, the proposed predictor incorporates local constraints in the learning process. Furthermore, note that while previous studies focused on using ML (either through imitation learning or reinforcement learning) to solve small-scale CO problems such as vehicle routing ($\leq$ 100 customers) and airline crew scheduling ($\leq$ 714 flights) problems, we use the proposed predictor to warm-start a monthly CPP solver (up to 50,000 flights). For an extensive literature review on using machine learning for combinatorial optimization, see Bengio et al. (2020).

Contributions. Bridging the gap between kernel methods and neural networks, we propose the structured convolutional kernel network (Struct-CKN)[1]. We first sanity check the approach on the OCR dataset (Taskar et al., 2004) yielding a test accuracy comparable to state of the art. Then, to warm-start an airline crew scheduling solver, we apply the proposed method on a flight-connection dataset, modeled as a general CRF capable of incorporating local constraints in the learning process. We show that the constructed solution outperforms other approaches in terms of test error and feasibility, an important metric to initialize the solver. The predicted solution is fed to the solver as an initial solution and initial clusters, to solve a large-scale CPP (50,000 flights). Our experiments demonstrate that this approach yields significant improvements, reducing the solution cost by 17% (a gain of millions of dollars) and the cost of global constraints by 97%, compared to baselines.

[1]The code is available at the following link: https://github.com/Yaakoubi/Struct-CKN

Outline. The remainder of this paper is structured as follows. Section 2 describes related methods. Section 3 presents CKNs. CRFs are outlined in Section 4. Section 5 presents Struct-CKN. Section 6 reports Computational results on OCR dataset, flight-connection dataset, and CPP.

2. Related Work

Upon a succinct review of previous work to compare available approaches in the literature to Struct-CKN, we argue for the use of the latter on the flight-connection dataset to solve CPPs.

Combining networks and energy-based models is a well-known approach since the 1990s. For instance, Bottou (2012) introduced graph transformer networks trained end-to-end using weighted acyclic directed graphs to represent a sequence of digits in handwritten character recognition. Furthermore, inspired by Q-learning, Gygli et al. (2017) used an oracle value function as the objective for energy-based deep networks, and Belanger & McCallum (2016) introduced structured prediction energy networks (SPENs) to address the inductive bias and to learn discriminative features of the structured output automatically. By assigning a score to an entire prediction, SPENs take into consideration high-order interactions between predictors using minimal structural assumptions. Nevertheless, due to the non-convexity, optimizing remains challenging, which may cause the learning model to get stuck in local optima. Another approach is to move step by step and predict one output variable at a time by applying the information gathered from previous steps. The linking between the steps is learned using a predefined order of input variable where the conditional is modeled with RNNs (Zheng et al., 2015b). Although this method has achieved impressive results in machine translation (Leblond et al., 2017), its success ultimately depends on the neural network's ability to model the conditional distribution and it is often sensitive to the order in which input data is processed, particularly in large-size graphs, as in CPPs (50,000 nodes).

In contrast to these approaches, instead of using continuous relaxation of output space variables (Belanger & McCallum, 2016), Struct-CKN uses supervised end-to-end learning of CKNs and CRF-based models. Accordingly, any of the existing inference mechanisms—from belief propagation to LP relaxations—can be applied. This allows us to naturally handle general problems that go beyond multi-label classification, and to apply standard structured loss functions (instead of extending them to continuous variables,

as in the case of SPENs). More importantly, Struct-CKN allows us to apply our method to a large-scale CRF graph containing up to 50,000 nodes. Furthermore, in contrast to methods in the literature (e.g., CNN-CRF (Chu et al., 2016), CRF-RNN (Zheng et al., 2015a), and deep structured models (Chen et al., 2015)), it has far fewer parameters, thus bypassing the need for extensive tuning.

Finally, note that in recent papers, convolutional graph neural networks (ConvGNNs) (Kipf & Welling, 2016) are used either (1) to warm-start a solver (trained under the imitation learning framework) (Owerko et al., 2020), (2) to solve the optimization problem end-to-end (Khalil et al., 2017), or (3) to guide an optimization process (variable selection in branch-and-bound Gasse et al. (2019)), and ConvGNNs might appear to be a good candidate-solution for CPPs when coupled with a CRF layer to impose constraints on the output. However, in the case of graphs with up to 50,000 nodes, the number of parameters used by ConvGNN and the computational limitations prevents us from considering it. In fact, we are not aware of any prior work where ConvGNNs were used at this scale. Furthermore, the implicit motivation for our proposed approach is an end-to-end solution method that can (1) harness the predictive capabilities of the ML predictor and the decomposition capacity of the solver (Yaakoubi et al., 2020), and (2) can be used on a standard machine with no specific resource requirements, to replace existing solvers in the industry. Future research will look into the possibility of integrating a distributed version of ConvGNNs into the proposed framework.

3. Convolutional Kernel Networks

CKN is a particular type of CNN that differs from the latter in the cost function to be optimized to learn filters and in the choice of non-linearities. We review CKNs, with the same notation as in Mairal (2016); Bietti & Mairal (2019). For further detail, see Appendix A.

3.1. Unsupervised Convolutional Kernel Networks (Mairal et al., 2014)

We consider an image $I_0 : \Omega_0 \to \mathbb{R}^{p_0}$, where p_0 is the number of channels, *e.g.*, $p_0 = 3$ for RGB, and $\Omega_0 \subset [0, 1]^2$ is a discrete set of pixel locations. Given two image patches x, x' of size $e_0 \times e_0$, represented as vectors in $\mathbb{R}^{p_0 e_0^2}$, we define a kernel $K_1(x, x') = \| x\| \| x'\| \cdot \kappa_1(\langle \frac{x}{\| x\|}, \frac{x'}{\| x'\|} \rangle)$ if $x, x' \neq 0$ and 0 otherwise, where $\|.\|$ and $\langle , \rangle$ denote the Euclidian norm and inner-product, respectively, and $\kappa_1(\langle \cdot, \cdot \rangle)$ is a dot-product kernel on the sphere. We have implicitly defined the reproducing kernel Hibert space (RKHS) $\mathcal{H}_1$ associated to K_1 and a mapping $\varphi_1 : \mathbb{R}^{p_0 e_0^2} \to \mathcal{H}_1$.

First, we build a database of n patches $\mathbf{x}_1, \dots, \mathbf{x}_n$ randomly extracted from various images and normalized to have unit ℓ_2-norm. Then, we perform a spherical K-means algorithm (Buchta et al., 2012), acting as a Nyström approximation, to obtain p_1 centroids $\mathbf{z}_1, \dots, \mathbf{z}_{p_1}$ with unit ℓ_2-norm. Given a patch $\mathbf{x}$ of I_0, the projection of $\varphi_1(\mathbf{x})$ onto $\mathcal{F}_1 := \text{Span}(\varphi_1(\mathbf{z}_1), \dots, \varphi_1(\mathbf{z}_{p_1}))$ admits a natural parametrization given in (1) where $\mathbf{Z} = [\mathbf{z}_1, \dots, \mathbf{z}_{p_1}]$, and κ_1 is applied pointwise to its arguments.

$$\Gamma_1(\mathbf{x}) := \|\mathbf{x}\| \kappa_1(\mathbf{Z}^\top \mathbf{Z})^{-1/2} \kappa_1 \left(\mathbf{Z}^\top \frac{\mathbf{x}}{\| x\|} \right) \quad \text{if } \mathbf{x} \neq 0 \text{ and } 0 \text{ o.w.} \tag{1}$$

Consider all overlapping patches of I_0. We set $M_1(z) = \Gamma_1(\mathbf{x}_z), \quad z \in \Omega_0$ where $\mathbf{x}_z$ is the patch from I_0 centered at pixel location z. The spatial map $M_1 : \Omega_0 \to \mathbb{R}^{p_1}$ thus computes the quantities $\mathbf{Z}^\top \mathbf{x}$ for all patches $\mathbf{x}$ of image I (spatial convolution after mirroring the filters $\mathbf{z}_j$), then applies the pointwise non-linear function κ_1.

The previous steps transform the image $I_0 : \Omega_0 \to \mathbb{R}^{p_0}$ into a map $M_1 : \Omega_0 \to \mathbb{R}^{p_1}$. Then, the CKNs involve a pooling step to gain invariance to small shifts, leading to another finite-dimensional map $I_1 : \Omega_1 \to \mathbb{R}^{p_1}$ with a smaller resolution: $I_1(z) = \sum_{z' \in \Omega_0} M_1(z') e^{-\beta_1 \|z' - z\|_2^2}, \quad z \in \Omega_1$, where β_1 is a subsampling factor. We build a multilayer image representation by stacking and composing kernels. Similarly to the first CKN layer transforming $I_0 : \Omega_0 \to \mathbb{R}^{p_0}$ to the map $I_1 : \Omega_1 \to \mathbb{R}^{p_1}$, we apply the same procedure to obtain $I_2 : \Omega_2 \to \mathbb{R}^{p_2}$, where p_2 is the number of centroids in the second layer, then $I_3 : \Omega_3 \to \mathbb{R}^{p_3}$, etc.

3.2. Supervised Convolutional Kernel Networks (Mairal, 2016)

Let $I_0^1, I_0^2, \dots, I_0^n$ be the training images with respective labels $y_1, \dots, y_n$ in {-1 ; +1} for binary classification. We also have L: $\mathbb{R} \times \mathbb{R} \to \mathbb{R}$, a convex smooth loss function. Given a positive definite kernel K on images, the classical empirical risk minimization formulation consists of finding a prediction function in the RKHS $\mathcal{H}$ associated to K by minimizing the objective $\min_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^n L(y_i, f(I_0^i)) + \frac{\lambda}{2} \|f\|_{\mathcal{H}}^2$, where the parameter λ controls the smoothness of the prediction function f with respect to the geometry induced by the kernel, hence regularizing and reducing overfitting.

After training a CKN with k layers, such a positive definite kernel $K_{\mathcal{Z}}$ may be defined as in (2) where I_k, I_k' are the k-th finite-dimensional feature maps of I_0 and I_0', respectively, and f_k, f_k' are the corresponding maps in $\Omega_k \to \mathcal{H}_k$, which have been defined in Section 3.1.

$$K_{\mathcal{Z}}(I_0, I_0') = \sum_{z \in \Omega_k} \langle f_k(z), f_k'(z) \rangle_{\mathcal{H}_k} = \sum_{z \in \Omega_k} \langle I_k(z), I_k'(z) \rangle \tag{2}$$

The kernel $K_{\mathcal{Z}}$ is also indexed by $\mathcal{Z}$, representing network parameters (subspaces $\mathcal{F}_1, \ldots, \mathcal{F}_k$, or equivalently the set of filters $Z_1, \ldots, Z_k$). Then, the formulation becomes as in (3) where $\|\cdot\|_F$ is the Frobenius norm extending the Euclidean norm to matrices and, with an abuse of notation, the maps I_k^i are seen as matrices in $\mathbb{R}^{p_k \times |\Omega_k|}$. Then, the supervised CKN formulation consists of jointly minimizing (3) w.r.t. W in $\mathbb{R}^{p_k \times |\Omega_k|}$ and with respect to the set of filters $Z_1, \ldots, Z_k$, whose columns are constrained to be on the Euclidean sphere.

$$\min_{W \in \mathbb{R}^{p_k \times |\Omega_k|}} \frac{1}{n} \sum_{i=1}^{n} L(y_i, \langle W, I_k^i \rangle) + \frac{\lambda}{2} \|W\|_F^2 \qquad (3)$$

4. Graph-Based Learning

In structured prediction, models are typically estimated with surrogate structured loss minimization, such as with structured SVM (SSVM) or CRFs. We used CRFs for the structured prediction that we briefly review below. We also tested SSVM integration instead of CRFs, but it yielded slightly worse results, see Appendices B.2 and C for details and experimental results using SSVM.

A CRF models the conditional probability of a structured output $y \in \mathcal{Y}$ given an input $x \in \mathcal{X}$ where the probability to observe y when x is observed is $p(y|x; w) \propto \exp(\langle w, F(x, y) \rangle)$, F is the feature mapping, and w is the vector of weights (to be learned). The CRF predictor of y when x is observed is: $h_w(x) = \arg\max_{y \in \mathcal{Y}} \langle w, F(x, y) \rangle$. The CRF primal problem formulation is shown in (4), where $\mathcal{L}^{CRF}$ denotes the negative log likelihood loss.

$$\min_{\omega} \lambda \|\omega\|^2 + \frac{1}{n} \sum_{i=1}^{N} \mathcal{L}^{CRF}(x_i, y_i; \omega) \qquad (4)$$

To optimize CRFs, we use stochastic dual coordinate ascent (SDCA) (Shalev-Shwartz & Zhang, 2013; 2014) as Le Priol et al. (2018) showed it yielded state-of-the-art results for CRFs. Although one can also use the stochastic average gradient (SAG) algorithm (Schmidt et al., 2015) or the online exponentiated gradient (OEG) algorithm, an advantage of SDCA over OEG (and SAG) is that it enables performing an "exact" line search with only one call to the marginalization oracle. We now rewrite (4) using the notation for the SDCA setup for multi-class classification (Shalev-Shwartz & Zhang, 2014). Denote $M_i = |\mathcal{Y}_i|$ the number of labelings for sequence i. Denote A_i the matrix whose columns are the corrected features $\{\psi_i(y) := F(x_i, y_i) - F(x_i, y)\}_{y \in \mathcal{Y}_i}$. Denote also $\Phi_i(s) := \log(\sum_{y \in \mathcal{Y}_i} \exp(s_y))$ the log-partition function for the scores $s \in \mathbb{R}^{M_i}$. Since the negative log-likelihood can be written as $-\log(p(y_i|x_i; w)) = \Phi_i(-A_i^\top w)$ (Murphy, 2012), the primal objective function to minimize over $w \in R^d$ becomes $P(w) := \frac{\lambda}{2} \|w\|_2^2 + \frac{1}{n} \sum_{i=1}^{n} \Phi_i(-A_i^\top w)$. This minimization problem has an equivalent Fenchel convex dual problem. Denote Δ_M the probability simplex over M elements. Denote $\alpha_i \in \Delta_{M_i}$ the set of dual variables for a given x_i, we define the conjugate weight function $\hat{w}$ as follows: $\hat{w}(\alpha) = \frac{1}{\lambda n} \sum_i A_i \alpha_i = \frac{1}{\lambda n} \sum_{i=1}^{n} \mathbb{E}_{y \sim \alpha_i}(\psi(y))$. We can show that $\hat{w}(\alpha^*) = w^*$ where w^* and α^* are respectively the optimal primal parameters and the optimal dual parameters. As such, we can also define the primal sub-optimality as: $P(w) - P(w^*)$.

Le Priol et al. (2018) adapted SDCA to CRF by considering marginal probabilities over cliques of the graphical model. Because the dual variable α_i is exponentially large in input size x_i, α is replaced by $\mu = (\mu_1, \ldots, \mu_n)$, where $\mu_i \in \Pi_C^{\Delta_C}$ is the concatenation of all the clique marginal vectors for sample i. Given its state-of-the-art performance, we decided to use it, although we will compare it with other optimizers in Section 6 (see Appendix B.1 for further details).

5. The Struct-CKN Framework

As in Figure 1, the Struct-CKN model consists of two components intended to train CRFs: (1) the CKN and (2) the structured predictor, using CRF loss, and SDCA. Upon initializing the CKN layers and the structured predictor, for each iteration, we pass the input image through the CKN multilayers. The last map of CKN is passed through to the structured predictor to infer probabilities, which are employed to train CKN weights by backpropagating using rules in Mairal (2016).

Algorithm 1 Training the Struct-CKN Model

1: Initialize CKN parameters in unsupervised manner as described in Sec. 3.1
2: Initialize structured predictor and CRF model as in Alg. 2 (see Appendix B.1, steps 1-2)
3: **for** $t = 0 \ldots$ **do**
4: For each input, construct an unary feature map, as described in Sec. 3.1
5: (Optional) Center and rescale these representations to have unit ℓ_2-norm on average
6: Infer probabilities by providing image map as input to structured predictor
7: Train the structured predictor using the feature map as an input for n_{Ep} epochs
8: Use the inferred probabilities to compute the gradient by using the chain rule (backpropagation) and update the CKN weights (Mairal, 2016)
9: **end for**

To do inference for CRF models with a small number of nodes (as in Section 6.1), we use max-product belief prop-

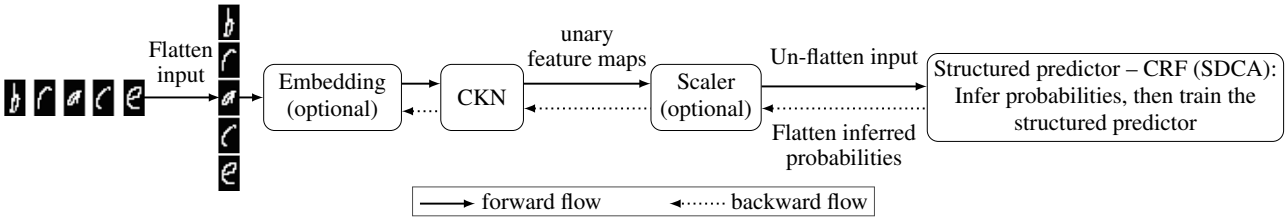

Figure 1. The Architecture Diagram for the Struct-CKN Predictor

agation, since chains can be solved exactly and efficiently. For the flight-connection dataset (as in Section 6.2), since CRF models contain 50,000 nodes, we use AD3 (alternating directions dual decomposition) (Martins et al., 2015) for approximate maximum a posteriori (MAP) inference. It allies the modularity of dual decomposition with the effectiveness of augmented Lagrangian optimization via the alternating directions method of multipliers and has some very interesting features in comparison to other message-passing algorithms. Indeed, AD3 has been empirically shown to reach consensus faster than other algorithms Martins et al. (2015) and outperforms state-of-the-art message-passing algorithms on large-scale problems. Besides, AD3 provides a library of computationally-efficient factors that allow handling declarative constraints within an optimization problem. This is particularly interesting for the CPP use case since we add a constraint to impose that each flight is preceded by one flight at most. To use SDCA (requiring a marginalization oracle) with AD3 (used to do approximate MAP), we propose a simple approximation, using MAP label estimates. See Appendix B.1 for details on integrating SDCA and AD3.

Note that an embedding layer may be used before passing the input through to the CKN layers. Indeed, for categorical variables with a large number of categories, the input matrix is sparse, making the learning process difficult, as the extracted patches have mostly null values. Furthermore, as in Chandra et al. (2017), using deep structured predictors may require using scaling. Specifically, the last map of the "deep" layer needs to be rescaled before being passed to the "structured layer". We propose to use one of the following scalers within the Scikit-learn library: Min-Max scaler, Normalizer scaler, Standard scaler, and Robust scaler. When using the SDCA optimizer, line search requires computing the entropy of the marginals. Since this is costly, in order to minimize the number of iterations, we used the Newton-Raphson algorithm. This requires storing the logarithm of the dual variable, which may be expensive, so a decent amount of memory should be allocated.

Finally, note that we use a batch-version of the Struct-CKN predictor on the flight-connection dataset (Yaakoubi et al., 2019), where one batch corresponds to one CRF model (CPP instance): (1) We initialize the CKN weights (in an unsupervised manner) and the CRF model sequentially by considering each one of the six instances separately. (2) We "flatten" the input by considering each one of the six instances separately. (3) We center and rescale the representations for all the instances at once. (4) We pass small batches of image maps as inputs to the structured predictor (e.g., 128 image maps) to infer the probabilities and train the structured predictor. Then, we use the inferred probabilities of the batch to update the CKN weights.

Table 1. Test Error on the OCR Dataset

	Test error (%)
SDCA - linear features (Le Priol et al., 2018)	12.0
LSTM (Greff et al., 2016)	4.6
CNN-CRF (Chu et al., 2016)	4.5
SCRBM (Tran et al., 2020)	4.0
NLStruct (Graber et al., 2018)	3.6
Struct-CKN	**3.4**

6. Experiments

In this section, we report the results of experiments using Struct-CKN. First, we sanity-check Struct-CKN on the standard OCR dataset in Section 6.1, showing that it's comparable to the state of the art. Then, in Section 6.2, we use the proposed predictor on the flight-connection dataset to warm-start Commercial-GENCOL-DCA.We use Pytorch (Paszke et al., 2019) to declare said model and perform operations on a 40-core machine with 384 GB of memory, and use K80 (12 GB) GPUs. The CRF model is implemented using PyStruct (Müller & Behnke, 2014), while the SDCA optimizer is implemented using SDCA4CRF. Scalers are implemented using Scikit-learn (Pedregosa et al., 2011).

6.1. OCR - Chain CRF

Each example in the OCR dataset (Taskar et al., 2004) consists of a handwritten word pre-segmented into characters, with each character represented as a 16×8 binary image. The task is to classify the image into one of the 26 charac-

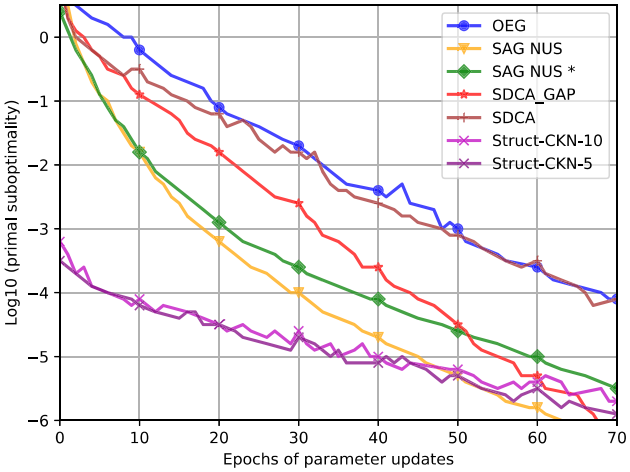

Figure 2. Comparison of Primal Sub-optimality

ters (a−z). It comes with pre-specified folds; one fold is considered the test set, while the rest as the training set, as in max-margin Markov networks (Taskar et al., 2004). Since the CRF optimizers (SAG-NUS, SAG-NUS* (Schmidt et al., 2015), SDCA, SDCA-GAP (Le Priol et al., 2018), and OEG (Schmidt et al., 2015)) yield similar test errors (11.8-12%), we only report SDCA (with linear features) in Table 1. LSTM (standard two-layer) (Greff et al., 2016) and CNN-CRF (standard two-layer) (Chu et al., 2016) yield comparable results (4.4-4.6%). Sequence Classification Restricted Boltzmann Machine (SCRBM) (Tran et al., 2020) and NLStruct (Graber et al., 2018) provide better results (4.0%, and 3.6%). However, Struct-CKN outperforms all aforementioned methods, reducing test errors to 3.40%.

Note that some structured predictors can lower test error to $1-3\%$ (Pérez-Cruz et al., 2007), such as SeaRNN (Leblond et al., 2017), which adapts RNN to the learning-to-search approach. However, such models approximate the cost-to-go for each token by computing the task loss for as many rollouts as the vocabulary size at each time step, and are thus difficult to scale to real-world datasets (with long sequences or large vocabulary), such as the flight-connection dataset (see Section 6.2). Figure 2 reports primal sub-optimality w.r.t parameter updates (see Appendix C). Struct-CKN outperforms other methods for the first 50 epochs and is comparable to other methods for subsequent epochs. We usually train predictors only for several epochs; thus, the precision of primal sub-optimality (below 10^{-5}) is negligible.

6.2. Airline Crew Scheduling Dataset - Graph CRF

This section reports results of using Struct-CKN to warm-start Commercial-GENCOL-DCA (Yaakoubi et al., 2020) and solve large-scale CPP. Section 6.2.1 presents CPP. Section 6.2.2 outlines the flight-connection prediction problem and CRF models. Section 6.2.3 reports results of predictions. Section 6.2.4 describes the optimization process and analyzes the feasibility of proposed monthly solutions. Section 6.2.5 reports results of solving CPPs using the solver.

6.2.1. Crew Pairing Problem

The CPP aims to find a set of pairings at minimal cost for each category of the crew and each type of aircraft fleet (Desaulniers et al., 1997). A flight sequence operated by a single crew forms a duty and a consecutive sequence of duty periods is named a pairing. A pairing is deemed feasible if it satisfies safety rules and collective agreement rules (Kasirzadeh et al., 2017), such as:

- minimum connection time between two consecutive flights and minimum rest-time between two duties;
- maximum number of flights per duty and maximum span of a duty;
- maximum number of landings per pairing and maximum flying time in a pairing;
- maximum number of days and maximum number of duties in a pairing.

In addition, CPPs use base constraints (referred to as global constraints) to distribute the workload fairly amongst the bases proportionally to the personnel available at each base. Penalties when the workload is not fairly distributed is called the cost of global constraints. In our approach, the predictor does not consider these very complex airline-dependent constraints as well as the solution cost and the cost of global constraints, as doing so would require more data than are currently available. Whenever a flight appears in more than one pairing, we use deadheads: one crew operates the flight, while the others are transferred between two stations for repositioning.

The CPP has been traditionally modelled as a set partitioning problem, with a covering constraint for each flight and a variable for each feasible pairing (Desaulniers et al., 1997; Kasirzadeh et al., 2017). Formally, we consider F to be a set of flights that must be operated during a given period and Ω to be the set of all feasible pairings that can be used to cover these flights. It is computationally infeasible to list all pairings in Ω when solving CPPs with more than hundreds of flights. Therefore, it is not tractable to do so in this context (CPPs with 50,000 flights). For each pairing $p \in \Omega$, let c_p be its cost and a_{fp}, $f \in F$, be a constant equal to 1 if it contains leg f and 0 otherwise. Moreover, let x_p be a binary variable that takes value 1 if pairing p is selected, and 0 otherwise. Using a set-partitioning formulation, the CPP can be modelled as follows:

$$\underset{x}{\text{minimize}} \quad \sum_{p\in\Omega} c_p x_p \tag{5}$$

$$\text{subject to} \quad \sum_{p\in\Omega} a_{fp} x_p = 1 \qquad \forall f \in F \tag{6}$$

$$x_p \in \{0,1\} \qquad \forall p \in \Omega \tag{7}$$

The objective function (5) minimizes the total pairing costs. Constraints (6) ensure each leg is covered exactly once, and constraints (7) enforce binary requirements on the pairing variables. The methodology to solve CPPs depends on the size of the airline's network, rules, collective agreements, and cost structure (Yen & Birge, 2006). Since the 1990s, the most prevalent method has been column generation inserted in branch-&-bound (Desaulniers et al., 1997). This algorithm was combined with multiple methods in Desaulniers et al. (2020) to solve large-scale CPPs. Yaakoubi et al. (2020) proposed Commercial-GENCOL-DCA with a dynamic control strategy and used CNNs to develop initial monthly crew pairings. Because the constructed solution contained too many infeasible pairings, it was passed to the solver only as initial clusters and a generic standard initial solution was used. In this work, we first use Struct-CKN to construct initial monthly crew pairings passed to the solver both as initial clusters and an initial solution. First, we use the GENCOL solver (used to assign crews to under-covered flights),[2] then we run Commercial-GENCOL-DCA. Because solvers can only handle a few thousand flights, the windowing approach is used: the month is divided into multiple windows, where each window is solved (sequentially) while flights in pairings from previous windows are frozen. GENCOL requires using two-day windows and one-day overlap period, while Commercial-GENCOL-DCA permits to use one-week windows and two-day overlap period. The latter starts with an aggregation, in clusters, of flights. The initial aggregation partition permits replacing all flight-covering constraints of the flights in a cluster by a single constraint, allowing to cope with larger instances.

6.2.2. Prediction Problem Formulation

We aim to provide the CPP solver with an initial solution and initial clusters. Using the flight-connection dataset built in Yaakoubi et al. (2019), the flight-connection prediction problem is a multi-class classification problem, formulated as follows: "Given the information about an incoming flight in a specific connecting city, choose among all possible departing flights from this city the one that the crew should follow" (see Appendix D). Thus, each input contains information on the previous flight performed by the crew as well as information on all possible next flights (up to 20 candidates), sorted by departure time, and the output (class) is the rank of the flight performed by the crew in past solutions. In Yaakoubi et al. (2019), the authors propose a similarity-based input where neighboring factors have similar features, allowing the use of CNNs. We extend their approach with our Struct-CKN model, thus starting the CPP solver with an initial solution, and not only the initial clusters.

The training set in the flight-connection dataset consists of six monthly crew pairing solutions (50,000 flights per month) and the test set is a benchmark that airlines use to decide on the commercial solver to use. Each flight is characterized by the cities of origin and destination, the aircraft type, the flight duration, and the departure and arrival time. For each incoming flight, the embedded representation of the candidate next flights is concatenated to construct a similarity-based input, where neighboring factors have similar features. The intuition is that each next flight is considered a different time step, enabling the use of convolutional architecture across time. Yaakoubi et al. (2019) compare multiple predictors on the flight-connection dataset and empirically confirm this intuition (see Appendix D).

The dataset is used to define a pairwise CRF on a general graph where each example consists of a flight-based network structure with nodes corresponding to flights and *arcs representing the feasibility of two flights being successive*. Each flight corresponds to a node connected to nodes that are possible successors/predecessors; the true label is the rank of the next flight in the set of sorted possible successors. We impose local constraints to the output by imposing that each flight has to be preceded by at most one flight (using a XOR constraint, which contributes $-\infty$ to the potential). Unlike for the OCR dataset where "max-product" is used for belief propagation, in this section, we use AD3 (alternating directions dual decomposition) (Martins et al., 2015) for approximate maximum a posteriori (MAP) inference.

6.2.3. Results on the Flight-connection Dataset

As Struct-CKN has fewer hyperparameters than CNNs, we observed that Struct-CKN is more stable than CNNs in our experiments, in that it does not requires Bayesian optimization to find a good architecture, thus justifying our use of CKN (see Appendix C). Table 2 reports the number of parameters and the test error on the flight-connection dataset using (1) CNNs and Bayesian optimization to search for the best configuration of hyperparameters (Yaakoubi et al., 2020); (2) standard CNN-CRF (with non-exhaustive hyperparameter tuning) (Chu et al., 2016); and (3) Struct-CKN. Struct-CKN outperforms both CNN and CNN-CRF while having far fewer parameters (97% fewer parameters). While Bayesian optimization can be used to fine-tune hyperparameters, this is not feasible in a real-case usage scenario, as practitioners cannot perform it each time new data become

[2] http://www.ad-opt.com/optimization/why-optimization/column-generation/

Table 2. Test Error on the Flight-connection Dataset (Yaakoubi et al., 2019)

	Test error (%)	# parameters
CNN (Yaakoubi et al., 2020)	0.32	459 542
CNN-CRF (Chu et al., 2016)	0.38	547 542
Struct-CKN	**0.28**	**15 200**

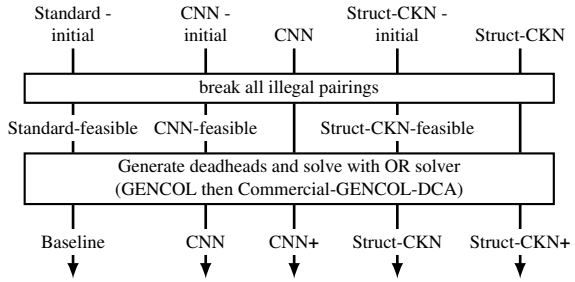

Figure 3. The Optimization Process for the CPP

available and the need for extensive fine-tuning makes a predictor impossible to integrate into any scheduling solver.

6.2.4. Construction and Feasibility of a Monthly Solution

As in Figure 3, we compare four approaches to construct monthly pairings. The first approach is a standard monthly solution, called *"Standard - initial"*, a "cyclic" weekly solution (from running the optimizer on a weekly CPP) rolled to cover the whole month (Desaulniers et al., 2020). A cyclic solution is where the number of crews in each city is the same at the beginning and end of the horizon. In the second approach, CNNs predict the flight-connection probabilities. Then, using the same heuristics as in Yaakoubi et al. (2020) (see Appendix D.3), we build a monthly crew pairing called *"CNN - initial"*. In the third and fourth approaches, CNN-CRF (Chu et al., 2016) and Struct-CKN are used to build monthly crew pairings called *"CNN-CRF - initial"*, and *"Struct-CKN - initial"*. Then, we break all illegal pairings and freeze the legal sub-part in initial crew pairings, resulting in *"Standard - feasible"*, *"CNN - feasible"*, *"CNN-CRF - feasible"* and *"Struct-CKN - feasible"*. We generate deadheads on all flights and pass it to the solver.

Note that to reproduce the following results, we obtain the commercial dataset from Yaakoubi et al. (2019), which cannot be distributed because it contains too much flight-data information sensitive to airlines' operations. Since the first step of the CPP solver can solve up to several thousand flights, we are constrained to use two-day windows. Since pairings extend to over two days, it cannot "fix" the mispredicted and illegal pairings. Therefore, the more illegal pairings the constructed solution contains, the longer it will take the solver to find a suitable final solution. Furthermore, when the constructed solution is highly infeasible, it can only be proposed as initial clusters as in all past research (e.g., Yaakoubi et al. (2019; 2020)) and not as an initial solution. In this case, a generic standard initial solution is used, and since the initial solution and the initial clusters are different, an adaptation strategy is required to adapt the proposed clusters of the current window to the solution of the previous window. This is a major limitation of past research since the explored neighborhood needed to be large enough to reach a good LP (linear programming) solution but small enough to maintain a small number of fractional variables permitting to have an efficient heuristic branch-&-bound. Thus, not only can we conclude that the primary metric of interest in our case is the feasibility of the constructed monthly solution. But, it also becomes crucial to propose a feasible pairing solution that can be proposed both as initial clusters and as an initial solution, therefore bypassing the adaptation strategy and permitting to reduce the resolution time (by reducing the neighborhood in which to explore in order to find a suitable solution).

Table 3. Characteristics of Monthly Solutions

	#pairings	Cost ($\times 10^8$)	% infeasible pairings
Standard - initial	6 525	37.93	50.56
Standard - feasible	3 226	29.75	
CNN - initial	4 883	24.13	21.05
CNN - feasible	3 855	16.58	
CNN-CRF - initial	4 567	21.15	12.12
CNN-CRF - feasible	4 010	16.15	
Struct-CKN - initial	4 515	20.29	**11.07**
Struct-CKN - feasible	**4 015**	16.46	

Table 3 summarizes the computational results on the feasibility and characteristics of constructed monthly pairings. First, breaking all illegal pairings in *"Standard - initial solution"* removes 50.56% of the pairings, while that in *"CNN - initial solution"* led to removing 21.05%. Note that even though the test error is low for CNNs, due to the large number of infeasible pairings, running the optimization using this initial solution is problematic. Breaking all illegal pairings in *"CNN-CRF - initial solution"* removes 12.12% of the pairings, while only 11.07% are removed from *"Struct-CKN - initial solution"*. Clearly, although Struct-CKN has 97% fewer parameters than other methods and its hyperparameters are not exhaustively fine-tuned, it outperforms other methods in terms of test error and feasibility. In what follows, we provide the constructed monthly pairings to the solver and compare the resulting solutions when using the baseline solution (Desaulniers et al., 2020), CNN (Yaakoubi et al., 2020) and the proposed approach (Struct-CKN).

Table 4. Computational Results for Monthly Solutions

	Solution cost ($\times 10^6$ \$)	Cost of global constraints ($\times 10^5$)	Number of deadheads	Total time (hours)
Baseline (Desaulniers et al., 2020)	20.64	21.27	992	45.92
CNN (Yaakoubi et al., 2020)	18.88	4.66	1014	95.72
Struct-CKN	**18.68**	**4.20**	**915**	64.48
CNN+	18.62	3.34	997	126.62
Struct-CKN+	**17.15**	**0.59**	**583**	41.44

6.2.5. Results on the Crew Pairing Problem

As in Figure 3, the baseline solution (*"Baseline"*) for the monthly CPP is obtained by feeding the solver initial clusters from *"Standard - feasible"* (Desaulniers et al., 2020). The previous state-of-the-art was obtained by feeding the solver initial clusters from *"CNN - feasible"*, yielding a solution called *"CNN"*. Instead of CNNs, we can use Struct-CKN to propose **both initial clusters and an initial solution** from *"Struct-CKN - feasible"* to the solver, yielding a monthly solution called *"Struct-CKN"*. To work on finding the best monthly solution possible and overcome the limitations of using the windowing approach, we feed the solution obtained from *"Struct-CKN"* to the solver (again) as initial clusters and initial solution, yielding *"Struct-CKN+"*. Because we cannot use the constructed monthly pairing as an initial solution, when using CNNs, we claim that re-running the optimization using the solution *"CNN"* as initial clusters does not improve the monthly solution much. To support our claim, we feed solution *"CNN"* to the solver as initial clusters, yielding *"CNN+"* (see Figure 3). See Appendix D.5 for detailed statistics of the optimization process.

Table 4 reports computational results for the final monthly solution. Note that we do not report variances since the CPP solver is deterministic. Struct-CKN outperforms both Baseline and CNN reducing the solution cost (in millions of dollars) and cost of global constraints by 9.51% and 80.25%, respectively, while also being 33% faster than CNN. By re-running the optimization, on the one hand, CNN+ does not improve the solution much. On the other hand, the Struct-CKN+ solution yields the best statistics, reducing solution cost and cost of global constraints by 16.93% and 97.24%, respectively. More interestingly, the number of deadheads (see Section 6.2.1) is reduced by 41.23%, compared to Baseline. Therefore, we can conclude that proposing a feasible monthly solution both as initial clusters and as an initial solution allowed us to achieve better results in less time and to provide the possibility to re-optimize the solution and improve it further. Thus, this permits to update the training solutions, suggesting that Struct-CKN can be further optimized and that further research to avoid the windowing approach and use a one-month window can present better results than the current version of solver.

7. Conclusion

Seeking an initial solution to a crew pairing solver, this study proposes Struct-CKN, a new deep structured predictor. Its supervised use outperforms state-of-the-art methods in terms of primal sub-optimality of the structured prediction "layer" and test accuracy on the OCR dataset. The proposed method is then applied on a flight-connection dataset, modeled as a general CRF capable of incorporating local constraints in the learning process. To warm-start the solver, we use Struct-CKN to propose initial clusters and an initial solution to the solver, reducing the solution cost by 17% (a gain of millions of dollars) and the cost of global constraints by 97%, compared to baselines. Future research will look into combining deep structured methods with various operations research methods and designing new reactive/learning metaheuristics that learn to guide the search for better solutions in real-time.

Acknowledgements

We are thankful to the anonymous reviewers and the meta-reviewer for their valuable comments that improved the quality of this work. We also would like to thank Andjela Mladenovic, Gina Arena, Joey Bose, Mehdi Abbana Bennani, and Stephanie Cairns for their constructive comments regarding this work. We thank Iban Harlouchet for his involvement during the first phase of the project. This work was supported by IVADO, a Collaborative Research and Development Grant from the Natural Sciences and Engineering Research Council of Canada (NSERC), by the Canada CIFAR AI Chair Program, and AD OPT, a division of IBS Software. We would like to thank these organizations for their support and confidence. Simon Lacoste-Julien is a CIFAR Associate Fellow of the Learning in Machines & Brains program.

References

Belanger, D. and McCallum, A. Structured prediction energy networks. In *International Conference on Machine Learning*, 2016.

Bengio, Y., Lodi, A., and Prouvost, A. Machine learning for combinatorial optimization: a methodological tour d'horizon. *European Journal of Operational Research*, 2020.

Bietti, A. and Mairal, J. Group invariance, stability to deformations, and complexity of deep convolutional representations. *JMLR*, 2019.

Bottou, L. Stochastic gradient descent tricks. In *Neural networks: Tricks of the trade*. Springer, 2012.

Buchta, C., Kober, M., Feinerer, I., and Hornik, K. Spherical k-means clustering. *Journal of Statistical Software*, 2012.

Chandra, S., Usunier, N., and Kokkinos, I. Dense and low-rank gaussian crfs using deep embeddings. In *IEEE International Conference on Computer Vision*, 2017.

Chen, L.-C., Schwing, A., Yuille, A., and Urtasun, R. Learning deep structured models. In *International Conference on Machine Learning*, 2015.

Chu, X., Ouyang, W., Li, h., and Wang, X. Crf-cnn: Modeling structured information in human pose estimation. In *Advances in Neural Information Processing Systems 29*, pp. 316–324. Curran Associates, Inc., 2016.

Desaulniers, G., Desrosiers, J., Dumas, Y., Marc, S., Rioux, B., Solomon, M. M., and Soumis, F. Crew pairing at Air France. *European journal of operational research*, 97(2), 1997.

Desaulniers, G., Lessard, F., Mohammed, S., and François, S. Dynamic constraint aggregation for solving very large-scale airline crew pairing problems. *Les Cahiers du GERAD*, G–2020(21), 2020.

Elhallaoui, I., Metrane, A., Soumis, F., and Desaulniers, G. Multi-phase dynamic constraint aggregation for set partitioning type problems. *Mathematical Programming*, 123(2), 2010.

Gasse, M., Chételat, D., Ferroni, N., Charlin, L., and Lodi, A. Exact combinatorial optimization with graph convolutional neural networks. In *Advances in Neural Information Processing Systems*, 2019.

Graber, C., Meshi, O., and Schwing, A. Deep structured prediction with nonlinear output transformations. In *Advances in Neural Information Processing Systems*, 2018.

Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., and Schmidhuber, J. Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems*, 28 (10), 2016.

Gygli, M., Norouzi, M., and Angelova, A. Deep value networks learn to evaluate and iteratively refine structured outputs. *International Conference on Machine Learning*, 2017.

Kasirzadeh, A., Saddoune, M., and Soumis, F. Airline crew scheduling: models, algorithms, and data sets. *EURO Journal on Transportation and Logistics*, 2017.

Khalil, E., Dai, H., Zhang, Y., Dilkina, B., and Song, L. Learning combinatorial optimization algorithms over graphs. In *Advances in Neural Information Processing Systems*, 2017.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. *International Conference on Learning Representations*, 2016.

Kumar, S., August, J., and Hebert, M. Exploiting inference for approximate parameter learning in discriminative fields: An empirical study. In *International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*. Springer, 2005.

Lacoste-Julien, S., Jaggi, M., Schmidt, M., and Pletscher, P. Block-coordinate frank-wolfe optimization for structural svms. *International Conference on Machine Learning*, 2012.

Lacoste-Julien, S., Jaggi, M., Schmidt, M., and Pletscher, P. Block-coordinate frank-wolfe optimization for structural svms. *International Conference on Machine Learning*, 2013.

Le Priol, R., Piché, A., and Lacoste-Julien, S. Adaptive stochastic dual coordinate ascent for conditional random fields. *Conference on Uncertainty in Artificial Intelligence*, 2018.

Leblond, R., Alayrac, J.-B., Osokin, A., and Lacoste-Julien, S. Searnn: Training rnns with global-local losses. *International Conference on Machine Learning*, 2017.

Mairal, J. End-to-end kernel learning with supervised convolutional kernel networks. In *Advances in Neural Information Processing Systems*, 2016.

Mairal, J., Koniusz, P., Harchaoui, Z., and Schmid, C. Convolutional kernel networks. *Advances in Neural Information Processing Systems*, 2014.

Martins, A. F., Figueiredo, M. A., Aguiar, P. M., Smith, N. A., and Xing, E. P. Ad3: Alternating directions dual decomposition for map inference in graphical models. *Journal of Machine Learning Research*, 2015.

Müller, A. C. and Behnke, S. Pystruct: learning structured prediction in python. *J. Mach. Learn. Res.*, 15, 2014.

Murphy, K. P. *Machine learning: a probabilistic perspective*. MIT press, 2012.

Owerko, D., Gama, F., and Ribeiro, A. Optimal power flow using graph neural networks. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style,

high-performance deep learning library. *Advances in Neural Information Processing Systems 32*, 2019.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2011.

Pérez-Cruz, F., Ghahramani, Z., and Pontil, M. Kernel conditional graphical models. In *Predicting structured data*, 2007.

Scharstein, D. and Pal, C. Learning conditional random fields for stereo. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2007.

Schmidt, M., Babanezhad, R., Ahmed, M., Defazio, A., Clifton, A., and Sarkar, A. Non-uniform stochastic average gradient method for training conditional random fields. In *AISTATS*, 2015.

Shalev-Shwartz, S. and Zhang, T. Stochastic dual coordinate ascent methods for regularized loss minimization. *Journal of Machine Learning Research*, 2013.

Shalev-Shwartz, S. and Zhang, T. Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization. In *International Conference on Machine Learning*, 2014.

Taskar, B., Guestrin, C., and Koller, D. Max-margin markov networks. In *Advances in Neural Information Processing Systems*, 2004.

Tran, S. N., Garcez, A. d., Weyde, T., Yin, J., Zhang, Q., and Karunanithi, M. Sequence classification restricted boltzmann machines with gated units. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

Yaakoubi, Y. *Combiner intelligence artificielle et programmation mathématique pour la planification des horaires des équipages en transport aérien*. PhD thesis, Polytechnique Montréal, 2019.

Yaakoubi, Y., Soumis, F., and Lacoste-Julien, S. Flight-connection prediction for airline crew scheduling to construct initial clusters for OR optimizer. *Les Cahiers du GERAD*, G–2019(26), 2019.

Yaakoubi, Y., Soumis, F., and Lacoste-Julien, S. Machine learning in airline crew pairing to construct initial clusters for dynamic constraint aggregation. *EURO Journal on Transportation and Logistics*, 2020. ISSN 2192-4376. doi: 10.1016/j.ejtl.2020.100020.

Yen, J. W. and Birge, J. R. A stochastic programming approach to the airline crew scheduling problem. *Transportation Science*, 40(1), 2006.

Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., Huang, C., and Torr, P. H. Conditional random fields as recurrent neural networks. In *Proceedings of the IEEE international conference on computer vision*, 2015a.

Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., Huang, C., and Torr, P. H. Conditional random fields as recurrent neural networks. In *Proceedings of the IEEE international conference on computer vision*, 2015b.

Zhong, P. and Wang, R. Using combination of statistical models and multilevel structural information for detecting urban areas from a single gray-level image. *IEEE transactions on geoscience and remote sensing*, 45(5), 2007.