Sur la Compilation de Jeux de Prédiction Combinatoire

Frédéric Koriche

Sur la Compilation de Jeux de Prédiction Combinatoire

Frédéric Koriche

CRIL, CNRS UMR 8188, Université d'Artois

frederic.koriche@cril.fr

Résumé

En optimisation en-ligne, le but est de choisir séquentiellement des solutions de manière à minimiser le coût moyen au cours du temps. Dès lors que l'espace de solutions faisables est décrit par des contraintes combinatoires, le problème est généralement NP-difficile. Dans cet article, nous investiguons l'idée de compiler un ensemble de contraintes combinatoires un circuit arithmétique de type `dDNNF`*, pour lequel les opérations d'optimisation linéaire et d'échantillonnage de solutions prennent un temps linéaire. Dans ce cadre de compilation, nous présentons des algorithmes efficaces pour diverses stratégies de prédiction en-ligne, en portant une attention particulière sur les approches de type « descente miroir ». Ces stratégies sont expérimentalement comparées sur plusieurs benchmarks du monde réel, pour lesquels les contraintes on été préalablement compilées en un circuit* `dDNNF`*.*

Mots Clef

Apprentissage en-ligne, compilation de connaissances, jeux de prédiction combinatoire.

Abstract

In online optimization, the goal is to iteratively choose solutions from a decision space, so as to minimize the average cost over time. As long as this decision space is described by combinatorial constraints, the problem is generally intractable. In this paper, we consider the paradigm of compiling the set of combinatorial constraints into a deterministic and Decomposable Negation Normal Form (`dDNNF`*) circuit, for which the tasks of linear optimization and solution sampling take linear time. Based on this framework, we provide efficient characterizations of existing combinatorial prediction strategies, with a particular attention to mirror descent techniques. These strategies are compared on several real-world benchmarks for which the set of Boolean constraints is preliminarily compiled into a* `dDNNF` *circuit.*

Keywords

Online learning, knowledge compilation, combinatorial prediction games.

1 Introduction

L'optimisation combinatoire est un domaine important de l'informatique et des mathématiques discrètes, avec un large spectre d'applications s'étendant depuis l'allocation de ressources et l'ordonnancement, jusqu'à la planification d'actions et les logiciels de configuration. Un des problèmes récurrents de ce domaine est de minimiser une fonction de perte linéaire $\boldsymbol{\ell}$ sur un espace combinatoire $\mathcal{S} \subseteq \{0,1\}^d$ de solutions faisables, représenté de manière compacte par un ensemble de contraintes. Dans la version *hors-ligne* de ce problème, toute l'information nécessaire à la spécification de la tâche d'optimisation est fournie à l'avance, et le but est de construire des algorithmes qui sont meilleurs que l'énumération naïve de toutes les solutions possibles. Dans la version *en-ligne* de ce problème, la fonction objectif $\boldsymbol{\ell}$ est susceptible de changer au cours du temps. Le défi est ici encore plus difficile, puisque l'algorithme d'optimisation doit choisir de manière séquentielle des solutions dans $\mathcal{S}$, afin de minimiser le coût moyen de ses choix.

De manière conceptuelle, le problème d'optimisation combinatoire en-ligne peut être vu comme un jeu répétitif entre un algorithme d'apprentissage et son environnement [1, 2]. Durant chaque tour t du jeu, l'apprenant choisit une solution faisable $\boldsymbol{s}_t$ dans l'espace $\mathcal{S}$ et, simultanément, l'environnement sélectionne un vecteur de perte $\boldsymbol{\ell}_t \in [0,1]^d$. A la fin du tour, l'apprenant subit la perte $\langle \boldsymbol{\ell}_t, \boldsymbol{s}_t \rangle = \sum_{i=1}^{d} \ell_t(i) s_t(i)$ et, selon le feedback fourni par son environnement, met à jour sa stratégie afin d'améliorer la chance de trouver de meilleures solutions durant les prochains tours.

Diverses classes de jeux de prédiction combinatoire peuvent être déclinées selon le type d'espace de décision et le type de feedback observé. Dans cet article, nous étudions les jeux à *information totale*, dans lesquels il est supposé que le feedback émis par l'environnement au tour t est le vecteur $\boldsymbol{\ell}_t$. En d'autres termes, l'apprenant connaît parfaitement la décision prise par l'environnement à la fin de chaque tour. En revanche, nous supposons que $\mathcal{S}$ peut être décrit par n'importe quelle formule `SAT`, c'est-à-dire un ensemble arbitraire de contraintes représentées par des clauses booléennes. Comme les codages `SAT` sont fréquemment utilisés dans les applications académiques et industrielles [5], notre cadre conceptuel couvre une classe importante de jeux de prédiction combinatoire.

En apprentissage en-ligne, la performance d'un algorithme de prédiction est caractérisée par deux mesures. La première est le *regret* : c'est la différence de perte cumulative entre l'algorithme de prédiction et la meilleure solution possible, qui aurait été choisie avec le recul sur l'historique du jeu. Dans cette étude, nous ne faisons aucune hypothèse sur la séquence des vecteurs de perte ; en particulier, $\boldsymbol{\ell}_t$ peut dépendre des décisions antérieures $\boldsymbol{s}_1, \cdots, \boldsymbol{s}_{t-1}$ choisies par l'apprenant. Pour ces environnements "sans oubli", pouvant être vus comme des adversaires, l'apprenant est autorisé à choisir ses décisions de manière aléatoire, et sa performance prédictive est mesurée par le *regret espéré* :

$$R_T = \mathbb{E}\left[\sum_{t=1}^{T} \langle \boldsymbol{\ell}_t, \boldsymbol{s}_t \rangle\right] - \min_{\boldsymbol{s} \in \mathcal{S}} \sum_{t=1}^{T} \langle \boldsymbol{\ell}_t, \boldsymbol{s} \rangle$$

La seconde mesure de performance est la *complexité calculatoire*, c'est-à-dire la quantité de ressources nécessaires pour trouver $\boldsymbol{s}_t$ à chaque tour t, étant donné la séquence de feedbacks observés jusqu'à présent.

1.1 Travaux Relatifs

Dans la littérature des jeux de prédiction combinatoire, trois principales stratégies ont été proposées pour atteindre un regret espéré qui est sous-linéaire en l'horizon de jeu T, et polynomial en la dimension d de l'espace de décision. La première, et sans doute la plus simple des stratégies, s'appelle *Follow the Perturbed Leader* (FPL) : à chaque tour t, l'apprenant choisit aléatoirement un vecteur de perturbation $\boldsymbol{z}_t \in \mathbb{R}^d$, et sélectionne un minimiseur de $\eta \boldsymbol{L}_t + \boldsymbol{z}_t$, où $\eta \in (0, 1]$ est un paramètre d'apprentissage et $\boldsymbol{L}_t$ est la perte cumulative : $\boldsymbol{L}_t = \boldsymbol{\ell}_1 + \cdots + \boldsymbol{\ell}_{t-1}$. Introduit par Hannan [15], et amélioré dans [18, 19], l'algorithme FPL atteint un regret espéré de $\mathcal{O}(d^{\frac{3}{2}}\sqrt{T})$.

La seconde stratégie est fondée sur le prédicteur "exponentially weighted average" bien connu dans le cadre de la prédiction avec avis d'experts [6]. L'idée générale est de maintenir un poids pour chaque solution faisable $\boldsymbol{s} \in \mathcal{S}$, qui décroit exponentiellement avec la perte cumulative de $\boldsymbol{s}$. Spécifiquement, à chaque tour t, l'apprenant choisit une solution $\boldsymbol{s}_t \in \mathcal{S}$ aléaoirement selon la distribution exponentielle $p_t(\boldsymbol{s}) \sim \exp(-\eta \langle \boldsymbol{L}_t, \boldsymbol{s} \rangle)$. Cette stratégie, appelée *Expanded Hedge* (EH) dans [20], atteint un regret espéré de $\mathcal{O}(d^{\frac{3}{2}}\sqrt{T})$, similaire à celui de FPL.

Enfin, la troisième stratégie appelée *Follow the Regularized Leader*, est le paradigme principalement utilisé en optimisation convexe en-ligne [16]. L'apprenant utilise ici l'enveloppe convexe de $\mathcal{S}$, notée $\mathrm{conv}(\mathcal{S})$. A chaque tour t, l'apprenant démarre en choisissant un point $\boldsymbol{p}_t \in \mathrm{conv}(\mathcal{S})$ qui minimize $\eta \langle \hat{\boldsymbol{L}}_t, \boldsymbol{p} \rangle + F(\boldsymbol{p})$, où F est une fonction de régularisation. Ensuite, $\boldsymbol{p}_t$ est décomposé en une combinaison convexe de solutions faisables dans $\mathcal{S}$, à partir de laquelle une décision $\boldsymbol{s}_t$ est choisie aléatoirement. Pour les fonctions de perte linéaire, cette stratégie est équivalent à l'algorithme *Online Stochastic Mirror Descent* (OSMD) [2, 30] qui, à chaque tour de jeu, accomplit une descente de gradient dans l'espace dual de $\mathrm{conv}(\mathcal{S})$ selon F, et projette ensuite cette solution sur l'espace primal, selon la divergence de Bregman définie sur F. En particulier, lorsque F est le régularisateur euclidien, OSMD coïncide avec la descente de gradient stochastique (SGD) [31]. Lorsque F est le régularisateur entropique, OSMD correspond à l'algorithme *Component Hedge* (CH) [20], qui atteint un regret espéré *optimal* de $\mathcal{O}(d\sqrt{T})$.

Si l'on se focalise sur la mesure de regret, l'état de l'art nous indique que peu d'innovations restent à faire dans les jeux à information totale. Cependant, le constat est bien différent dès lors que l'on tient compte des aspects calculatoires : les trois stratégies mentionnées plus haut font intervenir de puissants oracles pour prendre des décisions dans des espaces $\mathcal{S}$ représentés par des contraintes combinatoires. En particulier, la stratégie EH nécessite, à chaque itération, d'échantillonner une solution selon une famille exponentielle sur $\mathcal{S}$, ce qui est un problème #P-difficile [12]. De manière analogue, la stratégie FPL doit répétitivement résoudre un problème d'optimisation linéaire sur $\mathcal{S}$, ce qui est généralement NP-difficile [8]. Pour l'algorithme OSMD et ses spécialisations SGD et CH, la difficulté calculatoire est exacerbée par le fait que, même si l'apprenant a accès à un oracle d'optimisation linéaire, il doit accomplir à chaque tour de jeu une projection de Bregman pour laquelle les meilleurs algorithmes ont une complexité temporelle en $\mathcal{O}(d^6)$ [33] ou $\mathcal{O}(d^4)$ [24].

Bien que les jeux de prédiction combinatoire soient généralement NP-difficiles, il est dans certains cas possible d'obtenir des implémentations efficaces pour les oracles d'échantillonnage et d'optimisation. Par exemple, lorsque l'espace de décision $\mathcal{S}$ coïncide avec les bases d'un matroïde binaire, ou les appariements parfaits d'un graphe biparti, l'optimisation linéaire peut être résolue en temps polynomial, et donc des implémentations efficaces de FPL et OSMD peuvent être dérivées [17, 20, 34, 30]. D'autre part, lorsque les solutions faisables de $\mathcal{S}$ correspondent aux chemins d'un graphe orienté sans circuit (DAG), l'oracle d'échantillonnage peut être simulé par la technique de *weight pushing* [25], qui évalue récursivement la constante de partition d'une famille exponentielle sur les arêtes du DAG. A partir de cette technique, des implémentations efficaces de EH peuvent être dérivées [35, 29].

1.2 Nos Résultats

Envisager les solutions faisables comme des chemins d'un DAG n'est qu'une des nombreuses abstractions possibles qui ont été proposées dans la littérature en complexité de circuits, pour représenter des espaces combinatoires. En compilation de connaissances [11], diverses classes de circuits booléens ont été identifiées, chacune associée avec une collection de tâches d'inférence qui peuvent être accomplies en temps polynomial. Ces résultats théoriques nous incitent naturellement à nous poser la question suivante : est-il possible de *compiler* un ensemble de contraintes représentant un espace combinatoire $\mathcal{S}$ en un circuit booléen compact, pour lequel l'échantillonnage de solution et l'optimisation

linéaire sont traitables ? En regardant le processus de compilation comme une étape de "pré-processing", nous pouvons obtenir des implémentations efficaces des oracles d'échantillonnage et d'optimisation, dès lors que la taille du circuit compilé n'est pas trop grande.

Cet article vise à résoudre des jeux de prédiction combinatoire en compilant les espaces de décision en circuit dDNNF (*deterministic Decomposable Negation Normal Form*) [9]. Pour cette classe de circuits, il existe des compilateurs génériques prenant en entrée une formule SAT représentant un espace de décision $\mathcal{S}$, et retournant en sortie un circuit C qui code $\mathcal{S}$ [10, 23]. Bien que la taille de C puisse croître de manière exponentielle en la "treewidth" de la formule d'entrée, elle est souvent bien plus petite en pratique ; les compilateurs actuels sont capables de compresser des espaces combinatoires définis sur des milliers de variables et de contraintes.

Avec ces outils de compilation en main, nos contributions sont les suivantes : (i) nous montrons que pour les circuits dDNNF, l'oracle d'échantillonnage pour EH, et l'oracle d'optimisation linéaire pour FPL, prennent tous deux un temps linéaire en utilisant de simples variantes de la technique weight-pushing ; (ii) pour les stratégies SGD and CH, nous développons une méthode de projection-décomposition de Bregman qui utilise seulement $\mathcal{O}(d^2 \ln(dT))$ appels à l'oracle d'optimisation linéaire ; (iii) nous montrons expérimentalement sur des tâches de configuration séquentielle et de planification en-ligne que EH et FPL sont rapides, mais nos variants de SGD et CH sont plus efficaces pour minimiser le regret empirique.

2 Inférence Efficace

Pour les jeux de prédiction combinatoire considérés dans ce papier, nous supposons que l'espace de décision $\mathcal{S}$ est défini à partir d'un ensemble fini d'attributs binaires, et nous utilisons $X = \{x_1, \cdots, x_d\}$ pour caractériser l'ensemble de toutes les paires "attribut-valeur", appelées littéraux ou *indicateurs*. Une *solution* est un vecteur $\boldsymbol{s} \in \{0,1\}^d$ tel que $s(i) + s(j) = 1$ pour chque paire d'indicateurs distincts $x_i, x_j \in X$ définis sur le même attribut. Ainsi, $\|\boldsymbol{s}\|_1 = \frac{d}{2}$ pour toute solution $\boldsymbol{s} \in \mathcal{S}$.

Un circuit NNF sur X est un DAG enraciné, dont les nœuds internes sont étiquetés par $\vee$ (or-node) ou $\wedge$ (and-node), et dont les feuilles sont étiquetées par un indicateur dans X, ou bien une constante dans $\{0,1\}$. La taille de C, notée $|C|$, est donnée par le nombre d'arêtes dans C. L'ensemble des attributs apparaissant dans le sous-graphe de C, enraciné au nœud c, est noté $att(c)$.

Pour des raisons de clarté, nous supposerons que tout circuit NNF C satisfait deux propriétés : (i) tout nœud interne c dans C a exactement deux enfants, notés c_l et c_r, et (ii) $att(c_l) = att(c_r) \neq \varnothing$ pour tout or-node c de C. Un circuit NNF satisfaisant ces deux conditions est dit *lisse*. Comme il est montré dans [9], tout circuit booléen C peut être transformé en un circuit NNF lisse équivalent, dont la taille est linéaire en $|C|$.

Q	R	$\oplus$	$\otimes$	$\top$	$\bot$
maxmin	$\{0,1\}$	max	min	1	0
minsum	$\mathbb{R} \cup \{+\infty\}$	min	$+$	0	$+\infty$
sumprod	$\mathbb{R}$	$+$	$*$	1	0

TABLE 1 – Semi-anneaux commutatifs

En voyant les indicateurs comme des "portes d'entrées" et les nœuds comme des "portes de sortie", nous pouvons définir diverses tâches d'inférence sur les circuits booléens, qui dépendent du type d'entrée et de la sémantique des nœuds. Comme le suggèrent Friesen & Domingos pour les fonctions sommes-produits [13], les tâches d'inférence peuvent être caractérisées par des opérations de semi-anneaux. Rappelons ici qu'un *semi-anneaux commutatif* est un tuple $(R, \oplus, \otimes, \bot, \top)$ tel que R est un ensemble contenant les éléments $\bot$ et $\top$, $\oplus$ est un opérateur binaire associatif et commutatif sur R avec l'élément neutre $\bot$, $\otimes$ est un opérateur binaire associatif sur R avec l'élément neutre $\top$ et l'élément absorbant $\bot$, et l'opérateur $\otimes$ se distribue (à droite et à gauche) sur l'opérateur $\oplus$.

Les tâches d'inférence sur un circuit NNF C sont spécifiées par le choix d'un semi-anneau commutatif $Q = (R, \oplus, \otimes, \bot, \top)$ et d'un vecteur d'entrée $\boldsymbol{w} \in R^d$. La *sortie* d'un nœud c dans C pour Q étant donné $\boldsymbol{w}$ est notée $Q(c \,|\, \boldsymbol{w})$, et elle est récursivement donnée par

$$Q(c \,|\, w) = \begin{cases} w(i) & \text{si } c \text{ est un indicateur } x_i, \\ \top & \text{si } c \text{ est la constante } 1, \\ \bot & \text{si } c \text{ est la constante } 0, \\ Q(c_l \,|\, \boldsymbol{w}) \oplus Q(c_r \,|\, \boldsymbol{w}) & \text{si } c \text{ est le nœud } \vee, \\ Q(c_l \,|\, \boldsymbol{w}) \otimes Q(c_r \,|\, \boldsymbol{w}) & \text{si } c \text{ est le nœud } \wedge \end{cases}$$

Nous notons $Q(C \,|\, \boldsymbol{w})$ la sortie de la racine de C pour Q selon $\boldsymbol{w}$. Les semi-anneaux commutatifs jouant un rôle majeur dans cette étude sont décrits dans le tableau 1 ; maxmin, minsum, and sumprod, sont utilisés pour capturer les tâches d'inférence reliées respectivement à l'évaluation de modèle, l'optimisation linéaire, et l'échantillonnage de modèle.

2.1 Évaluation de Modèle

Étant donné un circuit NNF C sur X, l'évaluation de modèle (*model checking*) est de décider si une solution possible $\boldsymbol{s} \in \{0,1\}^d$ est vraie dans C selon la sémantique propositionnelle des nœuds. Il est clair que $\boldsymbol{s}$ est un modèle de C ssi $\mathrm{maxmin}(C \mid \boldsymbol{s}) = 1$, ce qui peut être déterminé en temps $\mathcal{O}(|C|)$. Un circuit NNF C est appelé *représentation* d'un ensemble de solutions faisables $\mathcal{S} \subseteq \{0,1\}^d$ si $sol(C) = \mathcal{S}$, où $sol(C)$ est l'ensemble des modèles de C.

Si l'on excepte l'évaluation de modèle, la totalité des tâches d'inférence dans les circuits NNF sont NP-difficiles. En effet, le langage NNF couvre la classe des formules SAT. Ainsi, nous avons besoin de contraindre cette

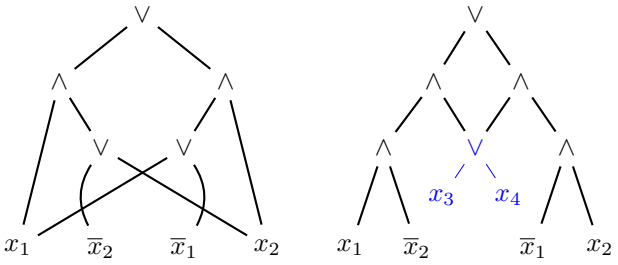

FIGURE 1 – A gauche, un circuit DNNF et à droite un circuit dDNNF.

Algorithm	Regret	Runtime
EH	$\mathcal{O}(d^{\frac{3}{2}}\sqrt{T})$	$\mathcal{O}(\lvert C\rvert)$
FPL	$\mathcal{O}(d^{\frac{3}{2}}\sqrt{T})$	$\mathcal{O}(\lvert C\rvert)$
SGD+PCG	$\mathcal{O}(d(\sqrt{T}+\ln T))$	$\mathcal{O}(d^2\lvert C\rvert \ln T)$
δ-CH+PCG	$\mathcal{O}(d(\sqrt{T}+\ln T))$	$\mathcal{O}\left(\frac{d^2\lvert C\rvert}{\delta}\ln\frac{T}{\delta}\right)$

TABLE 2 – Regret espéré et complexité temporelle (par tour) de chaque stratégie implémentée sur un circuit dDNNF C.

classe pour obtenir des formes efficaces d'optimisation et d'échantillonnage.

2.2 Décomposabilité et Optimisation

Un circuit booléen C est *décomposable* si pour tout and-node c de C, nous avons $att(c_l) \cap att(c_r) = \varnothing$. La classe des circuits NNF décomposables est notée DNNF. Pour de tels circuits, qui sont similaires aux réseaux sommes-produits [28], nous pouvons obtenir une implémentation efficace de l'oracle d'optimisation linéaire.

Proposition 1. Soit $\mathcal{S} \subseteq \{0,1\}^d$ un espace de décision non vide représenté par un circuit DNNF C, et soit $\boldsymbol{w} \in \mathbb{R}^d$ une fonction objectif linéaire. Trouver un minimiseur de $\boldsymbol{w}$ dans $\mathcal{S}$ peut être réalisé en temps $\mathcal{O}(|C|)$.

2.3 Déterminisme et Échantillonnage

Comme le problème de comptage du nombre de modèles d'un circuit DNNF est #P-difficile [11], nous avons besoin de spécialiser cette classe afin d'obtenir une implémentation efficace de l'oracle d'échantillonnage. Dans cette perspective, un circuit NNF C est dit *déterministe* si $\mathrm{minmax}(c_l \mid \boldsymbol{s}) + \mathrm{minmax}(c_r \mid \boldsymbol{s}) \leq 1$ pour tout or-node $c \in C$ et toute solution faisable $\boldsymbol{s}$. La classe des circuits DNNF déterministes est notée dDNNF.

Proposition 2. Soit $\mathcal{S} \subseteq \{0,1\}^d$ un espace de décision représenté par un circuit dDNNF circuit C. Pour un vecteur $\boldsymbol{w} \in \mathbb{R}^d$, soit $\mathbb{P}_{\boldsymbol{w}}$ la famille exponentielle sur $\mathcal{S}$ donnée par :

$$\mathbb{P}_{\boldsymbol{w}}(\boldsymbol{s}) = \frac{\exp\langle \boldsymbol{w}, \boldsymbol{s}\rangle}{\sum_{\boldsymbol{s}'\in\mathcal{S}} \exp\langle \boldsymbol{w}, \boldsymbol{s}'\rangle}$$

L'échantillonnage $\boldsymbol{s} \sim \mathbb{P}_{\boldsymbol{w}}$ peut être fait en temps $\mathcal{O}(|C|)$.

3 Prédiction Efficace

Après une excursion dans les langages de compilation, nous sommes maintenant prêts à examiner des caractérisations efficaces de stratégies de prédiction combinatoire. Nos résultats sont résumés dans le tableau 2. En utilisant le fait que $\|\boldsymbol{s}\|_1 = d/2$, les bornes de regret pour EH and FPL peuvent facilement être dérivées de [1] and [18]. Les deux stratégies sont aussi faciles à implémenter sur des dDNNF. En effet, rappelons que EH tire, à chaque tour t, une solution faisable $\boldsymbol{s}_t \in \mathcal{S}$ aléatoirement selon la distribution $\mathbb{P}_{-\eta \boldsymbol{L}_t}$, où $\boldsymbol{L}_t = \boldsymbol{\ell}_1 + \cdots + \boldsymbol{\ell}_{t-1}$. Ainsi, par application directe de la proposition 2, cette stratégie s'exécute en temps $\mathcal{O}(|C|)$ par tour, dès lors que l'espace $\mathcal{S}$ a été compilé au préalable en un circuit dDNNF C. Pour la stratégie FPL, chaque tour t est accompli en choisissant un minimiseur $\boldsymbol{s}_t \in \mathcal{S}$ de la fonction objectif $\eta \boldsymbol{L}_t - \boldsymbol{z}_t$, où $\boldsymbol{z}_t \in \mathbb{R}^d$ est un vecteur de perturbation dont les composants sont des variables aléatoires exponentielles indépendantes. Par application de la proposition 1, FPL s'exécute en temps $\mathcal{O}(|C|)$ par tour, en utilisant un codage dDNNF C de $\mathcal{S}$, et le fait que $|C|$ est en $\Omega(d)$.

Cependant, la stratégie OSMD, et ses spécialisations SGD et CH, requièrent plus d'attention du fait qu'elles utilisent une coûteuse étape de projection-décomposition à chaque tour de jeu.

3.1 Descente Miroir Stochastique

L'idée générale de l'algorithme Online Mirror Descent (OMD) est de "suivre le leader régularisé" par le biais d'une approche primale-duale [26, 4]. Soit $\mathcal{K}$ un ensemble convexe, et notons $int(\mathcal{K})$ son intérieur. Étant donné une fonction de régularisation F définie sur $\mathcal{K}$, OMD réalise de manière itérative une descente de gradient dans l'intérieur de l'espace dual $\mathcal{K}^*$, et projette le point dual sur l'espace primal $\mathcal{K}$. La connexion entre $\mathcal{K}$ et $\mathcal{K}^*$ est assurée par les gradients ∇F et ∇F^*, où F^* est le conjugué convexe de F défini sur $\mathcal{K}^*$. L'étape de projection est capturée par la divergence de Bregman de F, qui est une fonction de la forme $B_F \colon \mathcal{K} \times \mathrm{int}(\mathcal{K}) \to \mathbb{R}$ donnée par :

$$B_F(\boldsymbol{p}, \boldsymbol{q}) = F(\boldsymbol{p}) - F(\boldsymbol{q}) - \langle \nabla F(\boldsymbol{q}), \boldsymbol{p} - \boldsymbol{q}\rangle$$

Dans le variante stochastique de OMD, introduite par Audibert et. al. [1, 2], et spécifiée dans l'algorithme 1, chaque étape de projection est réalisée sur le sous-ensemble $\mathrm{conv}(\mathcal{S})$ of $\mathcal{K}$, et le point de projection $\boldsymbol{p}_t$ est décomposé en une combinaison convexe de solutions faisables dans $\mathcal{S}$, à partir de laquelle une solution est tirée aléatoirement pour l'étape de prédiction.

Pour les régularisateurs habituels, le gradient $\nabla F(\boldsymbol{p}_t)$ et son dual $\nabla F^*(\boldsymbol{u}_t)$ sont facilement calculables, et nous supposerons ici que le temps dépensé pour leur construction est négligeable par rapport au temps d'exécution de l'oracle

Algorithme 1 OSMD

Entrée : espace $\mathcal{S} \subseteq \{0,1\}^d$, horizon $T \in \mathbb{Z}_+$
Paramètres : régularisateur F sur $\mathcal{K} \supseteq \text{conv}(\mathcal{S})$, scalaire $\eta \in (0,1]$

soit $\boldsymbol{u}_1 = \mathbf{0}$
pour $t = 1$ **to** T **faire**
  soit $\boldsymbol{p}_t \in \text{Argmin}_{\boldsymbol{p} \in \text{conv}(\mathcal{S})} B_F(\boldsymbol{p}, \nabla F^*(\boldsymbol{u}_t))$
  jouer $\boldsymbol{s}_t \sim \boldsymbol{p}_t$ et observer $\boldsymbol{\ell}_t$
  choisir $\boldsymbol{u}_{t+1} = \nabla F(\boldsymbol{p}_t) - \eta \boldsymbol{\ell}_t$
end pour

Algorithme 2 PCG

Entrée : $\mathcal{S} \subseteq \{0,1\}^d$, $f : \mathcal{K} \to \mathbb{R}$, $n \in \mathbb{Z}_+$
Paramètres : scalaires $\{\eta_j\}_{j=1}^n$

soit $\boldsymbol{p}_1$ un point arbitraire de $\mathcal{S}$
pour $j = 1$ **to** n **faire**
  soit $\sum_{i=1}^j \alpha_i \boldsymbol{s}_i$ une décomposition convexe de $\boldsymbol{p}_j$
  soit $\boldsymbol{s}_j^+ \in \text{Argmin}_{\boldsymbol{p} \in \text{conv}(\mathcal{S})} \langle \nabla f(\boldsymbol{p}_j), \boldsymbol{p} \rangle$
  soit $\boldsymbol{s}_j^- \in \text{Argmin}_{\boldsymbol{s} \in \{\boldsymbol{s}_1, \cdots, \boldsymbol{s}_j\}} \langle -\nabla f(\boldsymbol{p}_j), \boldsymbol{s} \rangle$
  choisir $\boldsymbol{p}_{j+1} = \boldsymbol{p}_j + \eta_j(\boldsymbol{s}_j^+ - \boldsymbol{s}_j^-)$
end pour

d'optimisation. En fait, le goulot d'étranglement de OSMD est de trouver un minimiseur $\boldsymbol{p}_t$ de $B_F(\boldsymbol{p}, \nabla F^*(\boldsymbol{u}_t))$ dans l'enveloppe convexe de $\mathcal{S}$, et de décomposer $\boldsymbol{p}_t$ en une combinaison convexe de solutions dans $\mathcal{S}$. Heureusement, selon des hypothèses raisonnables sur la courbure de B_F, cette étape de projection-décomposition peut être calculée efficacement, en s'inspirant de résultats récents sur les algorithmes d'optimisation convexe "projection-free".

De manière plus formelle, nous disons qu'une divergence de Bregman B_F possède le *nombre conditionnel* β/α si, en son premier argument, B_F est α-fortement convexe et β-lisse [1] selon la norme euclidienne $\|\cdot\|_2$. Pour de tels régularisateurs, le résultat suivant établit que l'étape de projection-décomposition peut être approximée efficacement, en exploitant la méthode *Pairwise Conditional Gradient* (PCG), qui est une variante de l'algorithme Frank-Wolfe dont la vitesse de convergence a été analysée dans [22, 14, 3].

Lemme 1. Soit $\mathcal{S} \subseteq \{0,1\}^d$ un ensemble de décision représenté par un circuit dDNNF C, et F un régularisateur sur $\mathcal{K} \supseteq \text{conv}(\mathcal{S})$ tel que B_F possède le nombre conditionnel β/α. Alors, pour tout $\boldsymbol{q} \in \text{int}(\mathcal{K})$ et $\epsilon \in (0,1)$, on peut trouver en temps $\mathcal{O}(\frac{\beta}{\alpha} \ln \frac{\beta d}{\epsilon} d^2 |C|)$ une décomposition convexe de $\boldsymbol{p} \in \text{conv}(\mathcal{S})$ telle que

$$B_F(\boldsymbol{p}, \boldsymbol{q}) - \min_{\boldsymbol{p}' \in \text{conv}(\mathcal{C})} B_F(\boldsymbol{p}', \boldsymbol{q}) \leq \epsilon$$

Notons OSMD+PCG la version raffinée de l'algorithme OSMD qui utilise PCG à chaque tour t, afin d'approximer l'étape de projection-d"composition. En plus du régularisateur F et du paramètre η, OSMD+PCG prend en entrée une séquence $\{\epsilon_t\}_{t=1}^T$ telle que

$$B_F(\boldsymbol{p}_t, \boldsymbol{q}_t) - B_F(\boldsymbol{p}_t^*, \boldsymbol{q}_t) \leq \epsilon_t$$

où $\boldsymbol{p}_t$ est le point retourné par PCG, $\boldsymbol{q}_t = \nabla F_\phi^*(\boldsymbol{u}_t)$, et $\boldsymbol{p}_t^*$ est le minimiseur de $B_F(\boldsymbol{p}, \boldsymbol{q}_t)$ sur $\text{conv}(\mathcal{S})$.

Théorème 1. Supposons que OSMD+PCG prenne en entrée un encodage dDNNF C d'un espace de décision $\mathcal{S} \subseteq \{0,1\}^d$, ainsi qu'un horizon T, et utilise un régularisateur F sur $\mathcal{K} \supseteq \text{conv}(\mathcal{S})$ tel que B_F possède le nombre conditionnel β/α, avec le paramètre d'apprentissage $\eta \in (0,1]$ et une séquence de valeurs de précision $\{\epsilon_t\}_{t=1}^T$ telle que $\epsilon_t = \gamma/t^2$ pour $\gamma > 0$. Alors OSMD+PCG atteint le regret espéré :

$$\begin{aligned} R_T \leq \sqrt{\frac{2\gamma d}{\alpha}}(\ln T + 1) + \frac{1}{\eta} \max_{\boldsymbol{s} \in \mathcal{S}} B_F(\boldsymbol{s}, \boldsymbol{p}_1^*) \\ + \frac{1}{\eta} \sum_{t=1}^T B_{F^*}(\nabla F(\boldsymbol{p}_t^*) - \eta \boldsymbol{\ell}_t, \nabla F(\boldsymbol{p}_t)) \end{aligned} \tag{1}$$

avec une complexité temporelle en $\mathcal{O}\left(\frac{\beta}{\alpha} \ln \frac{\beta d T}{\gamma} d^2 |C|\right)$.

3.2 Descente de Gradient Stochastique

L'algorithme (en-ligne) SGD est dérivé de OSMD en utilisant le régularisateur euclidien $F(\boldsymbol{p}) = \frac{1}{2}\|\boldsymbol{p}\|_2^2$. Dans ce cadre simple, l'espace primal et l'espace dual coïncident avec $\mathbb{R}^d$, et donc, $F^*(\boldsymbol{u}) = \boldsymbol{u}$, $\nabla F(\boldsymbol{p}) = \boldsymbol{p}$, et $\nabla F^*(\boldsymbol{u}) = \boldsymbol{u}$. De plus, B_F a le nombre conditionnel $1/1$, puisque $B_F(\boldsymbol{p}, \boldsymbol{q}) = \frac{1}{2}\|\boldsymbol{p} - \boldsymbol{q}\|_2^2$. Nous notons ici SGD+PCG l'instance de OSMD+PCG définie sur le régularisateur euclidien.

Proposition 3. L'algorithme SGD+PCG atteint un regret espéré de $d(\sqrt{T} + \ln T + 1)$ avec une complexité temporelle par tour de $\mathcal{O}(d^2|C| \ln T)$ en utilisant $\eta = 1/\sqrt{T}$ et $\gamma = d/2$.

3.3 Component Hedge

L'algorithme CH est dérivé de OSMD en utilisant le régularisateur entropique $F(\boldsymbol{p}) = \sum_{i=1}^d p(i)(\ln p(i) - 1)$, pour lequel le conjugué est $F^*(\boldsymbol{u}) = \sum_{i=1}^d \exp u(i)$. Ici, on ne peut pas trouver un nombre conditionnel fini pour la divergence associée $B_F(\boldsymbol{p}, \boldsymbol{q}) = \sum_{i=1}^d p(i) \ln \frac{p(i)}{q(i)} - (p(i) - q(i))$, puisque son gradient n'est pas borné. Cette difficulté peut cependant être contournée en utilisant une astuce suggérée dans [21]. L'idée consiste à remplacer le régularisateur entropique par la fonction $F_\delta(\boldsymbol{p}) = F(\boldsymbol{p} + \boldsymbol{\delta})$, où $\delta \in (0,1)$ et $\boldsymbol{\delta} = (\delta, \cdots, \delta)$. Pour cette fonction l'espace primal est $(-\delta, +\infty)$, et puisque $F_\delta^*(\boldsymbol{u}) = F^*(\boldsymbol{u}) - \langle \boldsymbol{u}, \boldsymbol{\delta} \rangle$, l'espace dual est $\mathbb{R}^d$. Il est facile de montrer que

$$\frac{\partial F_\delta(\boldsymbol{p})}{\partial p(i)} = \ln(p(i) + \delta) \qquad \frac{\partial F_\delta^*(\boldsymbol{u})}{\partial u(i)} = e^{u(i)} - \delta$$

$$B_{F_\delta}(\boldsymbol{p}, \boldsymbol{q}) = B_F(\boldsymbol{p} + \boldsymbol{\delta}, \boldsymbol{q} + \boldsymbol{\delta}) \quad B_{F_\delta}^*(\boldsymbol{u}, \boldsymbol{v}) = B_F^*(\boldsymbol{u}, \boldsymbol{v})$$

1. Ces propriétés de courbure sont habituelles en optimisation convexe [16].

où $B_F^*(\boldsymbol{u}, \boldsymbol{v}) = \sum_{i=1}^{d} e^{v(i)}(e^{v(i)-u(i)} + v(i) - u(i) - 1)$. Enfin, puisque les dérivées partielles premières et secondes de $B_{F_\delta}^*(\boldsymbol{p}, \boldsymbol{q})$ à la coordonnée $p(i)$ sont

$$\frac{\partial B_{F_\delta}(\boldsymbol{p}, \boldsymbol{q})}{\partial p(i)} = \ln \frac{p(i) + \delta}{q(i) + \delta} \quad \frac{\partial^2 B_{F_\delta}(\boldsymbol{p}, \boldsymbol{q})}{\partial^2 p(i)} = \ln \frac{1}{p(i) + \delta}$$

il suit que B_{F_δ} possède le nombre conditionnel $1+\delta/\delta$. En effet, étant donné un point arbitraire $\boldsymbol{q} \in \mathrm{int}(-\delta, +\infty)$, soit $H_{\boldsymbol{q}}(\boldsymbol{p})$ la matrice hessienne de $B_{F_\delta}(\boldsymbol{p}, \boldsymbol{q})$ at $\boldsymbol{p} \in \mathrm{conv}(\mathcal{S})$. Alors, pour tout $\boldsymbol{z} \in \mathbb{R}^d$, les entrées de la diagonale de $H_{\boldsymbol{q}}(\boldsymbol{p})$ satisfont

$$\frac{1}{1+\delta} \leq \frac{\partial^2 B_F(\boldsymbol{p}, \boldsymbol{q})}{\partial^2 p(i)} z(i)^2 \leq \frac{1}{\delta}$$

en utilisant le fait que $p(i) \in [0, 1]$. Ainsi, $\alpha \boldsymbol{I} \preccurlyeq H_{\boldsymbol{q}}(\boldsymbol{p}) \preccurlyeq \beta \boldsymbol{I}$ pour $\alpha = 1/1+\delta$ et $\beta = 1/\delta$. Dans la suite, l'instance de OSMD+PCG qui utilise F_δ comme régularisateur est notée δ-CH+PCG.

Proposition 4. L'algorithme δ-CH+PCG atteint un regret espéré de $d(1 + 2\delta)(\sqrt{T} + \ln T + 1)$ avec une complexité temporelle par tour de $\mathcal{O}\left(d^2|C|/\delta \ln T/\delta\right)$, en utilisant $\eta = 1/\sqrt{T}$ et $\gamma = 2d(1/2 + \delta)/(1 + \delta)$.

4 Expérimentations

Afin d'évaluer la performance des différentes stratégies examinées dans la section précédente, nous avons considéré 15 instances de la SAT Library [2], décrites dans le tableau 3. Les cinq premières rangées sont des problèmes de configuration (automobile) et les rangées suivantes sont des tâches de planification. Les quatre premières colonnes du tableau indiquent le nom de l'instance SAT, le nombre d'attributs ($d/2$), le nombre de contraintes ($|\mathtt{SAT}|$), et le nombre de solutions faisables $|\mathcal{S}|$. Nous avons utilisé le compilateur D4 [3] [23] pour transformer les instances SAT en circuits dDNNF. La taille $|C|$ du circuit compilé est reportée dans la cinquième colonne.

Afin de simuler des jeux de prédiction combinatoire, nous avons utilisé le protocole suivant. Supposons que l'ensemble $X = \{x_1, \cdots, x_d\}$ des indicateurs soit trié par ordre lexicographique, de telle manière que pour tout entier impair i, le couple (x_i, x_{i+1}) code les deux configurations du même attribut. Nous construisons d'abord un vecteur $\boldsymbol{\mu}_0$ de $d/2$ variables de Bernoulli indépendantes. Pour chaque tour t du jeu, $\boldsymbol{\mu}_t$ est fixé à $\boldsymbol{\mu}_{t-1}$ avec probabilité 0.9, où tiré aléatoirement et uniformément sur $[0, 1]^{d/2}$ avec probabilité 0.1. Le feedback fourni à l'apprenant est un vecteur de perte $\boldsymbol{\ell}_t \in \{0, 1/d\}^d$ tel que $\ell_t(i) + \ell_t(i+1) = 1/d$, et $\ell_t(i) = 1/d$ avec probabilité $\mu_t(i+1/2)$ pour chaque entier impair i. Ainsi, $\ell_t(i+1) = 1/d$ avec probabilité $1 - \mu_t(i+1/2)$. Bien que ce protocole soit essentiellement stochastique, l'environnement change (en secret) $\boldsymbol{\mu}_t$ avec probabilité 0.1 à chaque tour, afin de déjouer l'apprenant.

2. www.cs.ubc.ca/~hoos/SATLIB/index-ubc.html
3. www.cril.univ-artois.fr/KC/d4.html

Avant d'examiner les résultats, donnons quelques indications sur le choix des paramètres. Pour les algorithmes FPL et EH, nous avons utilisé les valeurs de η reportées dans [1] et [18]. Concernant SGD+PCG et δ-CH+PCG, nous avons utilisé les valeurs de η et γ déterminées par notre analyse théorique : les valeurs de $\{\eta_t\}$ pour PCG ont été calculées par recherche binaire, comme il est préconisé dans [14], et la valeur de δ a été fixée à $1/\ln d$ afin de garder une complexité temporelle quadratique pour δ-CH+PCG. Enfin, nous avons fixé l'horizon T à 10^3, et nous avons fixé une journée pour le temps limite total de calcul pour l'apprentissage.

Dans nos expérimentations, le regret est mesuré par la différence de perte cumulative entre la stratégie étudiée et la meilleure solution faisable, obtenue par recul sur le jeu à l'horizon T. Cette mesure établie en moyenne sur 10 simulations, et divisée par T pour donner un regret empirique moyen. Les résultats correspondants sont donnés dans les quatre dernières colonnes du tableau 3. Le symbole "$-$" indique que l'apprenant n'a pas réussi à accomplir les T tours en une journée. Concernant le regret, SGD+PCG et δ-CH+PCG sont meilleurs que EH et FPL, ce qui confirme nos résultats théoriques. Nous mentionnons au passage que SGD+PCG et δ-CH+PCG sont remarquablement stables. Par contraste, FPL exhibe une grande variance.

Concernant les temps de calcul, EH et FPL sont sans surprise plus rapides que SGD+PCG et δ-CH+PCG. Pour les très grandes instances c129-fr, c140-fc, c163-fw, et log-5, nous avons constaté que EH et FPL réalisent chaque tour sur approximativement une minute, alors que les algorithmes OSMD+PCG requièrent plusieurs minutes par tour, à cause des ressources dépensées pour approximer la projection de Bregman. Cependant, il est important de souligner que la vitesse de convergence de PCG est, en pratique, bien plus rapide que l'analyse théorique en $\tilde{\mathcal{O}}(d^2|C|)$. En fait, SGD+PCG et δ-CH+PCG peuvent traiter pratiquement toutes les instances de problèmes de planification en moins d'une minute par tour. Nous avons observé que, pour la plupart des instances, SGD+PCG est légèrement plus rapide que δ-CH+PCG ; c'est particulièrement le cas pour les grands domaines où les faibles valeurs de δ ont un impact sur le temps de calcul. En résumé, SGD+PCG offre le meilleur compromis entre précision et temps de calcul ; puisque toutes les solutions sont denses ($\|\boldsymbol{s}\|_1 = d/2$), la différence de précision entre SGD+PCG et δ-CH+PCG n'est pas significative.

5 Perspectives

En lumière des résultats obtenus dans cette étude, une perspective naturelle de recherche est d'étendre notre cadre à d'autres classes de jeux de prédiction combinatoire. Notamment, le mode *semi-bandit* semble à portée de main. L'extension de EH en mode-semi-bandit, appelée EXP2 [2], utilise un ajustement des poids pour estimer le vecteur de perte à chaque itération. Par simple application de la proposition 2, cet ajustement peut être calculé en temps linéaire sur des circuits dDNNF. De manière analogue, l'extension semi-bandit de FPL exploite une méthode

Name	$d/2$	$\lvert$SAT$\rvert$	$\lvert\mathcal{S}\rvert$	$\lvert C\rvert$	EH	FPL	SGD+PCG	δ-CH+PCG
c129-fr	1888	6245	$1.33\ 10^{166}$	$1.42\ 10^{6}$	1023 ± 181 (65s)	1435 ± 321 (62s)	-	-
c140-fc	1828	4267	$5.74\ 10^{151}$	$6.94\ 10^{5}$	864 ± 175 (22s)	915 ± 185 (22s)	-	-
c163-fw	1815	3580	$2.97\ 10^{140}$	$8.93\ 10^{5}$	798 ± 87 (24s)	972 ± 104 (21s)	-	-
c169-fv	1411	637	$3.22\ 10^{15}$	$7.20\ 10^{2}$	18 ± 6 (<1s)	26 ± 10 (<1s)	9 ± 1 (4s)	9 ± 1 (5s)
4-step	165	396	$8.34\ 10^{4}$	$1.29\ 10^{2}$	3 ± 2 (<1s)	5 ± 3 (<1s)	2 ± 1 (2s)	2 ± 1 (2s)
5-step	177	459	$8.13\ 10^{4}$	$5.80\ 10^{1}$	3 ± 1 (<1s)	3 ± 1 (<1s)	1 ± 0.9 (<1s)	1 ± 0.8 (<1s)
log-1	939	3742	$5.64\ 10^{20}$	$9.43\ 10^{2}$	46 ± 5 (<1s)	51 ± 8 (<1s)	7 ± 3 (4s)	7 ± 1 (5s)
log-2	1337	24 735	$3.23\ 10^{10}$	$1.16\ 10^{4}$	16 ± 3 (2s)	19 ± 6 (2s)	9 ± 3 (62s)	9 ± 2 (78s)
log-3	1413	29 445	$2.79\ 10^{11}$	$4.96\ 10^{3}$	19 ± 4 (1s)	21 ± 4 (1s)	9 ± 3 (12s)	8 ± 3 (12s)
log-4	2303	42529	$2.34\ 10^{28}$	$9.47\ 10^{4}$	77 ± 11 (2s)	94 ± 27 (2s)	10 ± 4 (72s)	11 ± 3 (81s)
log-5	2701	29483	$7.24\ 10^{38}$	$4.62\ 10^{5}$	121 ± 33 (18s)	147 ± 52 (17s)	-	-
tire-1	352	1022	$8.29\ 10^{36}$	$1.37\ 10^{3}$	74 ± 5 (<1s)	67 ± 7 (<1s)	3 ± 2 (7s)	3 ± 2 (8s)
tire-2	550	1980	$7.39\ 10^{11}$	$7.26\ 10^{2}$	20 ± 2 (<1s)	24 ± 4 (<1s)	5 ± 2 (3s)	5 ± 1 (3s)
tire-3	577	1984	$2.23\ 10^{11}$	$6.31\ 10^{3}$	20 ± 5 (1s)	19 ± 7 (1s)	6 ± 2 (18s)	6 ± 2 (20s)
tire-4	812	3197	$1.03\ 10^{14}$	$3.97\ 10^{3}$	27 ± 3 (1s)	28 ± 3 (1s)	8 ± 3 (7s)	7 ± 3 (10s)

TABLE 3 – Résultats expérimentaux pour les différentes stratégies sur des instances SAT codées par des circuits dDNNF.

d'échantillonnage géométrique pour estimer les vecteurs de perte [27]. À nouveau, cette méthode itérative peut être implémentée en temps linéaire (par itération) en utilisant la proposition 1. L'adaptation de OSMD aux semi-bandits est cependant moins immédiate : même si l'extension de CH atteint encore un regret espéré optimal dans ce contexte, son utilisation pratique reste limitée à cause de la projection de Bregman. Une question ouverte intéressante ici est de trouver si une combinaison de CH avec PCG est capable, en mode semi-bandit, d'atteindre un regret quasi-optimal en temps quadratique. Naturellement, le mode *bandit* pur est encore plus difficile. Sur ce point, Sakaue et. al. [32] ont obtenu de premiers résultats en utilisant des OBDDs pour une implémentation efficace de l'algorithme COMBBAND [7]. Mis à part ce résultat très récent, tout reste à faire dans les bandits combinatoires, notamment avec des circuits dDNNF, qui sont plus succincts que les OBDDs.

Références

[1] J-Y. Audibert, S. Bubeck, and G. Lugosi. Minimax policies for combinatorial prediction games. In *Proceedings of the 24th Annual Conference on Learning Theory (COLT 2011)*, pages 107–132, 2011.

[2] J-Y. Audibert, S. Bubeck, and G. Lugosi. Regret in online combinatorial optimization. *Mathematics of Operations Research*, 39(1) :31–45, 2014.

[3] M. A. Bashiri and X. Zhang. Decomposition-invariant conditional gradient for general polytopes with line search. In *Advances in Neural Information Processing Systems 30, (NIPS 2017)*, pages 2687–2697, 2017.

[4] A. Beck and M. Teboulle. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operational Research Letters*, 31(3) :167–175, 2003.

[5] A. Biere, M. Heule, H. van Maaren, and T. Walsh. *Handbook of Satisfiability*. IOS Press, 2009.

[6] N. Cesa-Bianchi and G. Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, 2006.

[7] N. Cesa-Bianchi and G. Lugosi. Combinatorial bandits. *Journal of Computer and System Sciences*, 78(5) :1404–1422, 2012.

[8] N. Creignou, S. Khanna, and M. Sudan. *Complexity Classification of Boolean Constraint Satisfaction Problems*. SIAM Monographs on Discrete Mathematics and Applications, 2001.

[9] A. Darwiche. Decomposable negation normal form. *Journal of the ACM*, 48(4) :608–647, 2001.

[10] A. Darwiche. A compiler for deterministic, decomposable negation normal form. In *Proceedings of the 18th National Conference on Artificial Intelligence, (AAAI 2002)*, pages 627–634, 2002.

[11] A. Darwiche and P. Marquis. A knowledge compilation map. *Journal of Artificial Intelligence Research (JAIR)*, 17 :229–264, 2002.

[12] M. E. Dyer, L. A. Goldberg, and M. Jerrum. The complexity of weighted Boolean #CSP. *SIAM Journal of Computing*, 38(5) :1970–1986, 2009.

[13] A. L. Friesen and P. M. Domingos. The sum-product theorem : A foundation for learning tractable models. In *Proceedings of the 33nd International Conference on Machine Learning, (ICML 2016)*, pages 1909–1918, 2016.

[14] D. Garber and O. Meshi. Linear-memory and decomposition-invariant linearly convergent conditional gradient algorithm for structured polytopes. In *Advances in Neural Information Processing Systems 29, (NIPS 2016)*, pages 1001–1009, 2016.

[15] J. Hannan. Approximation to Bayes risk in repeated play. *Contributions to the Theory of Games*, 3 :97–139, 1957.

[16] E. Hazan. Introduction to online convex optimization. *Foundations and Trends in Optimization*, 2(3-4) :157–325, 2016.

[17] D. P. Helmbold and M. K. Warmuth. Learning permutations with exponential weights. *Journal of Machine Learning Research*, 10 :1705–1736, 2009.

[18] M. Hutter and J. Poland. Adaptive online prediction by following the perturbed leader. *Journal of Machine Learning Research*, 6 :639–660, 2005.

[19] A. T. Kalai and S. Vempala. Efficient algorithms for online decision problems. *Journal of Computer and System Sciences*, 71(3) :291–307, 2005.

[20] W. M. Koolen, M. K. Warmuth, and J. Kivinen. Hedging structured concepts. In *Proceedings of the 23rd Conference on Learning Theory (COLT 2010)*, pages 93–105, 2010.

[21] W. Krichene, S. Krichene, and A. M. Bayen. Efficient bregman projections onto the simplex. In *Proceedings of the 54th IEEE Conference on Decision and Control, (CDC 2015)*, pages 3291–3298, 2015.

[22] S. Lacoste-Julien and M. Jaggi. On the global linear convergence of frank-wolfe optimization variants. In *Advances in Neural Information Processing Systems 28 (NIPS 2015)*, pages 496–504, 2015.

[23] J-M. Lagniez and P. Marquis. An improved decision-dnnf compiler. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence, (IJCAI 2017)*, pages 667–673, 2017.

[24] Y. T. Lee, A. Sidford, and S. C. Wong. A faster cutting plane method and its implications for combinatorial and convex optimization. In *IEEE 56th Annual Symposium on Foundations of Computer Science, (FOCS 2015)*, pages 1049–1065, 2015.

[25] M. Mohri. General algebraic frameworks and algorithms for shortest-distance problems. Technical Report 981210-10TM, AT & T Labs-Research, 1998.

[26] A. S Nemirovski and D. B. Yudin. *Problem Complexity and Method Efficiency in Optimization*. J. Wiley and Sons, 1983.

[27] G. Neu and G. Bartók. Importance weighting without importance weights : An efficient algorithm for combinatorial semi-bandits. *Journal of Machine Learning Research*, 17 :154 :1–154 :21, 2016.

[28] H. Poon and P. M. Domingos. Sum-product networks : A new deep architecture. In *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence (UAI 2011)*, pages 337–346, 2011.

[29] H. Rahmanian and M. K. Warmuth. Online dynamic programming. In *Advances in Neural Information Processing Systems 30, (NIPS 2017)*, pages 2824–2834, 2017.

[30] A. Rajkumar and S. Agarwal. Online decision-making in general combinatorial spaces. In *Advances in Neural Information Processing Systems 27, (NIPS 2014)*, pages 3482–3490, 2014.

[31] H. Robbins and S. Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3) :400–407, 1951.

[32] S. Sakaue, M. Ishihata, and S-I Minato. Efficient bandit combinatorial optimization algorithm with zero-suppressed binary decision diagrams. In *Proceedings of the 21st International Conference on Artificial Intelligence and Statistics*, 2018.

[33] D. Suehiro, K. Hatano, S. Kijima, E. Takimoto, and K. Nagano. Online prediction under submodular constraints. In *Proceedings of the 23rd International Conference on Algorithmic Learning Theory (ALT 2012)*, pages 260–274, 2012.

[34] E. Takimoto and K. Hatano. Efficient algorithms for combinatorial online prediction. In *Proceedings of the- 24th International Conference on Algorithmic Learning Theory (ALT 2013)*, pages 22–32, 2013.

[35] E. Takimoto and M. K. Warmuth. Path kernels and multiplicative updates. *Journal of Machine Learning Research*, 4 :773–818, 2003.