Computing the capacity of discrete channels using vector flows

Computing the Capacity of Discrete Channels Using Vector Flows

Guglielmo Beretta (✉)[1,2][0000−0002−4757−1965],
Giacomo Chiarot[1][0000−0002−0248−1974],
Antonio Emanuele Cinà[3][0000−0003−3807−6417], and Marcello Pelillo[1,4,5]

[1] DAIS, Ca' Foscari University of Venice, Via Torino 155, 30170 Venice, Italy
guglielmo.beretta@unive.it (✉)
[2] DAUIN, Polytechnic University of Turin, Corso Castelfidardo 34/d, 10138 Turin, Italy
[3] DIBRIS, University of Genoa, Via all'Opera Pia 13, 16145 Genoa, Italy
[4] College of Mathematical Medicine, Zhejiang Normal University, 688 Yingbin Road, Jinhua, Zhejiang Province, 321004 China
[5] European Centre for Living Technology, Ca' Bottacin, Dorsoduro 3911, Calle Crosera, 30123 Venezia, Italy

Abstract. One of the fundamental problems of information theory, since its foundation by Shannon in 1948, has been the computation of the capacity of a discrete memoryless channel, a quantity expressing the maximum rate at which information can travel through the channel. In the literature, several algorithms were proposed to estimate the channel capacity, as an analytical solution is unavailable for the general channel. We propose a novel approach based on a continuous-time dynamical system to compute the capacity. We then derive an algorithm for computing the capacity, obtained by discretizing the flow that rules the evolution of this dynamical system. In the experimental analysis, we test the performance of our algorithm when different numerical ordinary differential equation solvers are utilized for its implementation. Remarkably, the results show that the algorithm is effective in computing the capacity.

Keywords: Channel capacity · Constrained optimization · Vector flow.

1 Introduction

Discrete memoryless channels (DMCs) are among the fundamental communication systems studied in information theory. A well-known problem concerning information channels is the computation of the channel capacity, which Shannon proved [15] to be a theoretical bound on the rate at which lossless communication can occur across the channel [12]. With few exceptions [4, 6], an explicit formula to compute the capacity of a DMC is unavailable, and many authors presented algorithms aiming at an accurate capacity estimate. This is the case for the classical Blahut-Arimoto algorithm (BAA) [1,3] and some variations thereof, e.g., [13, 18], as well as for other techniques, e.g., [16].

We present a novel approach to compute the capacity of a DMC, based on a continuous-time dynamical system [9]. To this end, we first consider the standard way to recast the problem into a maximization program on the standard simplex. We then introduce a suitable ordinary differential equation (ODE), which defines a dynamical system on the standard simplex. What can be observed is that the flow ruling the evolution of the dynamics drives towards a solution of the program, leveraging an analogy with some game-theoretical models [17]. Indeed, the flow constitutes an instance of a more general continuous-time first-order interior-point method for constrained optimization [7,11]. We also present an algorithm (VFA) to produce an estimate of the channel capacity. This algorithm is based on a discrete version of the flow. Specifically, the algorithm applies numerical methods in search of a proxy for the limit point to which the continuous-time system converges.

ODEs and the associated vector flows have a long tradition in constrained optimization — see, e.g., [8] and the references therein. Notably, Faybusovich [7] presented a method to maximize smooth objective functions over polyhedra that indeed can be regarded as a generalization of our approach for some types of channels. However, to the best of our knowledge, we are the first to use a vector flow to compute the capacity of a DMC. Besides that, in contrast to [7], the objective function that we consider may happen to have differentiability issues on the boundary of the feasible set.

A longer version of the work described in the following can be found in [2]. Specifically, in the long version of this paper, we provide a more detailed description of the theoretical framework concerning the flow; we prove the theoretical results mentioned and utilized in this work; and we validate the effectiveness of the VFA on symmetric channels [6], for which an explicit formula for the capacity is known. Additionally, within this paper, we provide some new empirical results that are not discussed in [2]. Specifically, the new results concern some experiments that we have carried out to test the VFA also on DMCs that are not symmetric. Moreover, in these experiments, we compared some implementations of the VFA obtained with different ODE solvers.

The content of this paper is articulated as follows. In Sect. 2 we review the capacity computation problem and discuss the vector flow that computes the capacity, as well as the algorithm derived from it, the VFA. Section 3 delves into the experiments that we have run. Conclusions are drawn in Sect. 4, in which we also mention a possible future line of research.

2 Vector Flow for Computing the Capacity

We now review the problem of the capacity computation and present how we tackled this problem using an appropriate vector flow ruling the evolution of a continuous-time dynamics over the standard simplex. See [2] for a more detailed treatment on the subject.

2.1 Discrete Memoryless Channels

A discrete memoryless channel (DMC) is a communication system described by a triplet $(\mathcal{X}, \mathcal{Y}, \boldsymbol{P})$, in which $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$ and $\mathcal{Y} = \{y_1, y_2, \ldots, y_m\}$ are finite alphabets called *input alphabet* and *output alphabet* respectively, whereas $\boldsymbol{P} = [p(j|i)]_{i\in[n],j\in[m]} \in \mathbb{R}^{n\times m}$ is a stochastic matrix,[6] called *transition matrix* [6], where the term $p(j|i)$ equals the probability that the symbol y_j is received as output of the channel every time the symbol x_i is given as input to the channel. The input X and output Y of the channel can be modeled as random variables having range in $\mathcal{X}$ and $\mathcal{Y}$, respectively. The *mutual information* between X and Y, here denoted as $I[X;Y]$, quantifies "the reduction in the uncertainty of X due to the knowledge of Y" [6], and equals[7]

$$I[X;Y] = \sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} \Pr(X=x, Y=y) \ln\left(\frac{\Pr(X=x, Y=y)}{\Pr(X=x)\cdot\Pr(Y=y)}\right).$$

The value of the mutual information is completely determined by the knowledge of $\boldsymbol{P}$ and of the distribution of X, since then the distributions of Y and of the random vector (X, Y) can be easily obtained: the former as a consequence of the law of total probability, the latter by definition of conditional probabilities. The *capacity* of the channel is defined as $C = \max_X I[X;Y]$, where the maximum is computed over all possible choices for the input random variable X.

2.2 Vector Flow

We now consider the standard way to formulate the problem as a maximization program over the standard simplex [4]. To this end, let $\boldsymbol{z} = (z_1, \ldots z_n)$ and $\boldsymbol{q} = \boldsymbol{q}(\boldsymbol{z}) = (q_1, \ldots q_m)$ denote the distributions of X and Y respectively, here defined as:

$$\begin{aligned} z_i &= \Pr(X = x_i), && i \in [n] \\ q_j &= \Pr(Y = y_j) = \sum_{i=1}^{n} p(j|i) z_i, && j \in [m]. \end{aligned}$$

Set $c_i = \sum_{j=i}^{m} p(j|i) \ln[p(j|i)]$ for every $i \in [n]$ and consider the function

$$I(\boldsymbol{z}) = \sum_{i=1}^{n} c_i z_i - \sum_{j=1}^{m} q_j \ln q_j. \tag{1}$$

The convention applied in the definition of c_i and $I(\boldsymbol{z})$ is that $r \ln r = 0$ in case $r = 0$. It follows that

$$C = \max_{\boldsymbol{z} \in \Delta_n} I(\boldsymbol{z}). \tag{2}$$

[6] Here and in the sequel, we set $[d] = \{\, k \in \mathbb{N} \mid 1 \le k \le d \,\}$ for every integer d.

[7] We remark that the expression given measures the mutual information in nats per transmission, and not in bits per transmission.

where $\Delta_n = \left\{ \boldsymbol{z} \in \mathbb{R}^d \;\middle|\; \boldsymbol{z} \geq \boldsymbol{0} \quad \text{and} \quad \sum_{i=1}^n z_i = 1 \right\}$ is the *standard simplex* in $\mathbb{R}^n$. Therefore, estimating the capacity amounts to finding the value of the maximization program (2). This is readily obtained if we are able to find an *optimal input distribution*, i.e, a vector $\boldsymbol{z}^* \in \Delta_n$ satisfying $I(\boldsymbol{z}^*) = C$.

The proposed idea to address this problem is based on the following ordinary differential equation:

$$\dot{z}_i = z_i \left[\partial_i I(\boldsymbol{z}) - \sum_{k=1}^{n} z_k \partial_k I(\boldsymbol{z}) \right], \qquad i \in [n]. \tag{3}$$

What can be shown is that (3) provides a continuous-time first-order method for solving (2) — see [7] for a more general treatment relying on Riemannian geometry concepts. Without loss of generality, we assume that $\boldsymbol{P}$ has no null columns.[8] Under such assumption, the right-hand side in (3) as a function of $\boldsymbol{z}$ lies in $\mathcal{C}^\infty(\Omega; \mathbb{R}^n)$, where $\Omega \subset \mathbb{R}^n$ is an open superset of the the (relative) *interior* of Δ_n, i.e., of the set $\operatorname{int}(\Delta_n) = \{ \boldsymbol{z} \in \Delta_n \mid z_i > 0, \quad i \in [n] \}$. The reader may note the resemblance of (3) with the continuous-time replicator dynamics, a model pertaining to evolutionary game theory [10,17]. Indeed, as for the case of symmetric replicator dynamics with a symmetric payoff matrix, the set $\operatorname{int}(\Delta_n)$, is invariant (in the future) under the vector flow associated with (3).[9] In fact, for every choice of $\tilde{\boldsymbol{z}} \in \operatorname{int}(\Delta_n)$, there exists a unique function $\boldsymbol{z}(t)$ defined for $t \in [0, +\infty)$ that satisfies the initial value problem

$$\begin{cases} \dot{z}_i & = z_i \left[\partial_i I(\boldsymbol{z}) - \sum_{k=1}^n z_k \partial_k I(\boldsymbol{z}) \right], \quad i \in [n] \\ \boldsymbol{z}(0) & = \tilde{\boldsymbol{z}}. \end{cases} \tag{4}$$

Such function $\boldsymbol{z}(t)$, here called the *trajectory of* (3) *initialized in* $\tilde{\boldsymbol{z}}$, satisfies $\boldsymbol{z}(t) \in \operatorname{int}(\Delta_n)$ for every $t \geq 0$. Therefore, the vector flow associated with (3) rules the evolution of a continuous-time dynamical system defined on $\operatorname{int}(\Delta_n)$.

Moreover, unless $\boldsymbol{z}(t)$ is a constant function, $I(\boldsymbol{z}(t))$ increases strictly in t. This plays a role in proving the following key result.

Theorem 1 (Attaining capacity). *For every* $\tilde{\boldsymbol{z}} \in \operatorname{int}(\Delta_n)$*, the trajectory* $\boldsymbol{z}(t)$ *of* (3) *initialized in* $\tilde{\boldsymbol{z}}$ *satisfies* $I(\boldsymbol{z}(t)) \to C$ *as* $t \to +\infty$*.*

As a consequence of Theorem 1, we also get that, if the channel admits a unique optimal input distribution, then it is precisely the limit of $z(t)$ as $t \to +\infty$. Theorem 1 is strongly related to the results concerning Lyapunov functions in the replicator dynamics framework [10,17].

[8] This condition on the transition matrix entails that $q_j \neq 0$ for every $\boldsymbol{z} \in \operatorname{int}(\Delta_n)$.

[9] In case $\boldsymbol{P}$ has no zero elements, then the invariance of $\operatorname{int}(\Delta_n)$ in the past follows from the Lipschitz condition on Δ_n, as we discuss in [2]. In the general case, however, the Lipschitz condition is valid only locally in $\operatorname{int}(\Delta_n)$, and it is not possible to use the same argument.

2.3 Algorithm Derived from the Vector Flow

Motivated by Theorem 1, we devised an algorithm (VFA) based on a discrete version of the vector flow associated with (1). This algorithm aims to estimate the capacity of a DMC, as well as an optimal input distribution. We report in Algorithm 1 the pseudocode of the VFA. The algorithm requires as input the transition matrix $\boldsymbol{P}$ of the channel and a vector $\tilde{\boldsymbol{z}} \in \operatorname{int}(\Delta_n)$, as well as the parameters required by a one-step ODE solver. The choice of the solver to use is here represented with the string `method`. At each iteration, `SolverStep()` performs one step of the method selected via the string `method`, and it returns an update for $\boldsymbol{z}$. We remark that `SolverStep()` requires the right-hand side of the ODE (3), here denoted as $F(\boldsymbol{P}, \boldsymbol{z})$ to underscore the dependence on $\boldsymbol{P}$. When the solver stops, it outputs a proxy $\hat{\boldsymbol{z}}$ for an optimal input distribution, hence $\hat{C} = I(\hat{\boldsymbol{z}})$ gives an estimate for the capacity of the DMC. Once again, to underscore the dependence on $\boldsymbol{P}$, we have written $I(\boldsymbol{P}, \hat{\boldsymbol{z}})$ in place of $I(\hat{\boldsymbol{z}})$ in the pseudocode.

Algorithm 1: VFA: The Vector-Flow Optimization Algorithm

Input : $\boldsymbol{P}$, transition matrix; τ, step size employed for the first step; `Niter`, maximum number of iterations; ε, parameter involved in the stopping criterion; $\hat{\boldsymbol{z}}$, starting point of the trajectory; `method`: a string enabling the selection of a numerical method for ODEs.

Output: $\hat{C}$, estimated channel capacity; $\hat{\boldsymbol{z}}$, estimated optimal input distribution; k, number of integration steps performed.

$k \leftarrow 0$
$\boldsymbol{z} \leftarrow \tilde{\boldsymbol{z}}$
$\texttt{err} \leftarrow +\infty$
while $k < \texttt{Niter} \wedge \texttt{err} \geq \varepsilon$ **do**
  $\boldsymbol{z}_{\text{new}} \leftarrow \texttt{SolverStep}(F(\boldsymbol{P}, \cdot), \boldsymbol{z}, \tau, \texttt{method})$
  $\texttt{err} \leftarrow \|\boldsymbol{z} - \boldsymbol{z}_{\text{new}}\|_1$
  $\boldsymbol{z} \leftarrow \boldsymbol{z}_{\text{new}}$
  $k \leftarrow k + 1$
$\hat{\boldsymbol{z}} \leftarrow \boldsymbol{z}$
$\hat{C} \leftarrow I(\boldsymbol{P}, \hat{\boldsymbol{z}})$
return $\hat{C}$, $\hat{\boldsymbol{z}}$, k

3 Experiments

The experiments performed have two main objectives. First, to test the VFA on a set of channels that have not been covered in the numerical experiments of [2]. Second, to investigate the usage of different fixed-step ODE solvers in the VFA. To this end, we estimated the capacity of some channels with the VFA, as well as with the BAA. Since the BAA is a well-known classical algorithm to address

this problem, we have used its estimate as a ground-truth benchmark when an explicit formula for the capacity was not available.

3.1 Dataset

We generated a dataset of stochastic matrices, where each matrix represents the transition matrix of a DMC. To build such a dataset, we first selected the number n (resp. m) of rows (resp. columns), which corresponds to the size of the input (resp. output) alphabet of the channel, in a set of 15 evenly spaced integers within the interval $[2, 100]$. For every pair (n, m) obtained in this way, we then considered a $n \times m$ transition matrix $\boldsymbol{A} = [a_{ij}]$ of a noiseless channel, i.e., such that every row is a Dirac distribution. Specifically, we set

$$a_{ij} = \begin{cases} 1 & \text{if } i = j, \\ 1 & \text{if } m < i \leq n \text{ and } j = m, \\ 0 & \text{otherwise.} \end{cases}$$

The $n \times m$ matrices in the dataset have all the form

$$\mathbf{P} = (1 - \sigma)\boldsymbol{A} + \sigma \boldsymbol{R},$$

in which $\sigma \in \{ 0, 0.5, 1 \}$ and $\boldsymbol{R}$ is a stochastic matrix with rows generated uniformly in Δ_m using a Dirichlet distribution [14]. In particular, the $n \times m$ matrices in the dataset are eleven:

- one of these matrices is $\boldsymbol{A}$, corresponding to the choice $\sigma = 0$;
- among the ten remaining matrices, five have been generated for $\sigma = 0.5$, and five for $\sigma = 1$.

3.2 ODE Solvers for the VFA

To perform the numerical integration in our experiments, we used the Python library `torchdiffeq` [5], which offers the possibility to choose among several classical methods for ODEs. The VFA was executed with three different numerical methods, each obtained by modifying one of the following explicit classical methods, which have different order of convergence: (i) the Euler method (first-order method); (ii) the midpoint method (second-order method); (iii) the 3/8-rule Runge-Kutta method (fourth-order method).

For a given classical method mentioned, the corresponding modified method is also a one-step method. At every step, the output of the modified method is obtained by first performing one step of the classical method, which produces as output some $\boldsymbol{z}_{\text{classical}} \in \mathbb{R}^n$, and then by applying the following two operations:

1. Every negative component of $\boldsymbol{z}_{\text{classical}}$ is set to zero;
2. The resulting vector is normalized with respect to the L^1-norm.

If $\boldsymbol{z}_{\text{classical}}$ has at least one positive component, then the two operations described perform a projection onto Δ_n, and so the output is an element of Δ_n. We have introduced this additional projection to fix some numerical stability issues encountered, as well as to prevent the floating point arithmetic from pushing the dynamics outside Δ_n. For both the VFA and the BAA, we used $\|\boldsymbol{z}^{(k-1)} - \boldsymbol{z}^{(k)}\|_1 < 10^{-4}$ as stopping condition, where $\boldsymbol{z}^{(k)}$ denotes the k-th point of the discrete trajectory, and we also set 10 000 as the maximum number of iterations. These two choices correspond to the initializations $\varepsilon = 10^{-4}$ and $\texttt{Niter} = 10\,000$ in Algorithm 1.

An additional remark is due for both the modified midpoint method and the modified 3/8-rule Runge-Kutta method. Whenever either of these methods is selected to run the VFA, some issues may occur *during* a solver step. In fact, it may happen that an integration step queries an evaluation of $F(\boldsymbol{P}, \cdot)$ in some $\boldsymbol{z}$ that is *not* in the domain of $F(\boldsymbol{P}, \cdot)$, hence resulting in an error. In contrast, this cannot happen with the modified Euler method, since the classical Euler method is a Runge-Kutta method that uses just one stage, unlike the other two classical methods. We have managed to prevent errors of this type by reducing the step size τ, although this solution has some drawbacks, that we discuss in Sect. 3.3. In the experiments, we used a different value for step size τ depending on the numerical method selected. Specifically, we have used $\tau = 1$ for the modified Euler method, $\tau = 0.5$ for the modified midpoint method, and $\tau = 0.33$ for the modified 3/8-rule Runge-Kutta method.

3.3 Results and Discussion

In the case of noiseless channels, which correspond to $\sigma = 0$, the capacity can be computed exactly and equals $\ln(\min\{n, m\})$ nats per transmission. Notably, the experimental results show that both the BAA and the VFA can estimate this theoretical result with relative error in the order of 10^{-7}. No meaningful differences have been encountered with the three solvers used for the VFA. In contrast, for $\sigma > 0$, a formula for computing the capacity is not available. Therefore, the results obtained with the BAA provide a valuable benchmark for evaluating the results of the VFA.

For every channel considered, we have computed the signed relative error with the formula $(C_{\text{BA}} - \hat{C})/C_{\text{BA}}$, where C_{BA} and $\hat{C}$ denote the estimate obtained with the BAA and with the VFA respectively. We report in Fig. 1, Fig. 2 and Fig. 3, for each pair (n, m), the average of the signed relative error computed over the channels having an $n \times m$ transition matrix. To better highlight what happens as the level of noise increases, these figures also display the case $\sigma = 0$. The figures show that the average signed relative error is always below 5×10^{-4}, demonstrating that the VFA produces an estimate that is comparable with that of BAA. Moreover, for $\sigma > 0$ we have encountered that the modified Euler method yields a negative signed relative error, which happens if and only if $C_{\text{BA}} < \hat{C}$, i.e., whenever the VFA estimates the capacity better than the BAA. In contrast, the other two methods yield an estimate that is slightly worse than the BAA estimate.

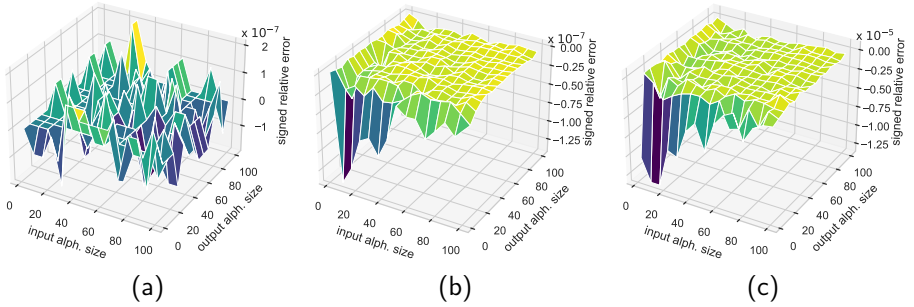

Fig. 1. Signed relative error on the capacity estimate of the VFA, using the modified Euler solver, against the BAA. Different values of noise are considered: (a) $\sigma = 0$; (b) $\sigma = 0.5$; (c) $\sigma = 1$.

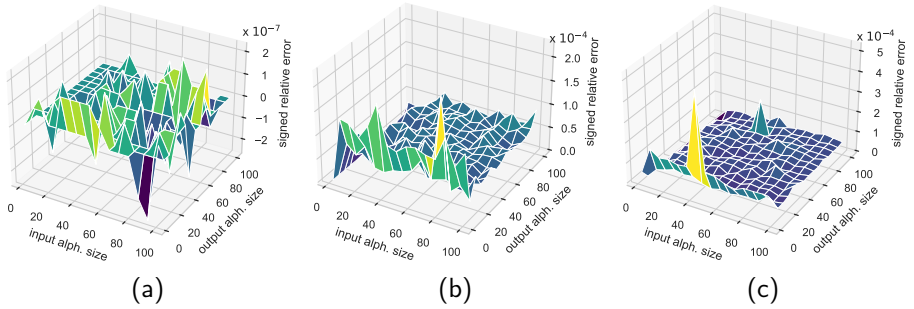

Fig. 2. Signed relative error on the capacity estimate of the VFA, using the modified midpoint solver, against the BAA. Different values of noise are considered: (a) $\sigma = 0$; (b) $\sigma = 0.5$; (c) $\sigma = 1$.

In summary, the experiments demonstrate that the VFA can effectively be used to compute the capacity and that the ODE solvers used effectively maximize the objective function $I(\cdot)$. However, we recall that we have used different values of τ, depending on the ODE solver selected. What emerges from the experiments is that the modified Euler solver not only performs better than the other two solvers in terms of accuracy of the capacity estimate, but it also requires fewer steps to halt. Besides that, there is also another reason to prefer the modified Euler solver, which is the computational cost per step. Indeed, each step of the Euler method requires just one evaluation of the function $F(\cdot, \boldsymbol{P})$. In contrast, the midpoint and the 3/8-rule method require two and four evaluations of $F(\cdot, \boldsymbol{P})$ at every step. We conclude that there is no advantage in using these higher-order methods.

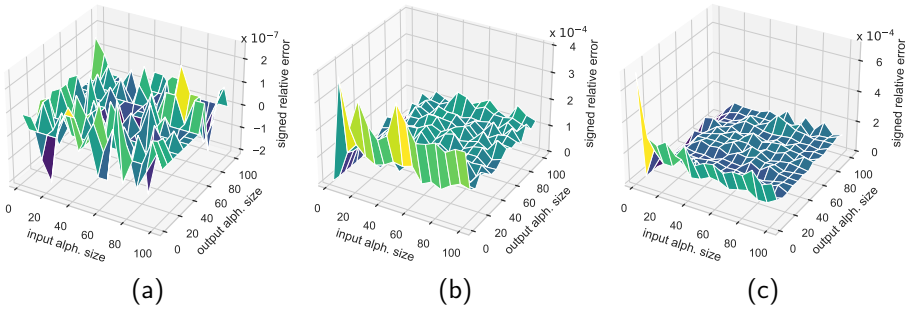

Fig. 3. Signed relative error on the capacity estimate of the VFA, using the modified 3/8-rule Runge-Kutta solver, against the BAA. Different values of noise are considered: (a) $\sigma = 0$; (b) $\sigma = 0.5$; (c) $\sigma = 1$.

4 Conclusion

Estimating the capacity of a DMC is still a hot topic in current research. Within this paper, we have presented a solution to this problem that is based on a vector flow, an instance of a first-order continuous-time method for constrained optimization over the standard simplex. Using ODE solvers to discretize the flow, we have derived an algorithm, the VFA, from the continuous-time dynamics. We have conducted some experiments to test the VFA on channels, and we have investigated the performance of VFA when combined with different ODE solvers. The results demonstrate the effectiveness of the VFA in estimating the channel capacity. Besides that, the results show that the modified Euler method is the best option among the ODE solvers considered. No benefit in the capacity estimate quality has been found that may compensate for the higher computational cost of the other solvers. It is still an open question whether the performance of the VFA can be improved by using numerical methods that have not been considered in this paper. To approach this problem, we believe that a natural and promising path involves extending the study to adaptive numerical methods.

Acknowledgments. Guglielmo Beretta's scholarship is funded jointly by Ca' Foscari University of Venice and by the Polytechnic University of Turin. The authors thank PhD Sebastiano Vascon for the useful discussion about this work.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Arimoto, S.: An algorithm for computing the capacity of arbitrary discrete memoryless channels. IEEE Trans. Inf. Theory **18**(1), 14–20 (1972). https://doi.org/10.1109/TIT.1972.1054753

2. Beretta, G., Chiarot, G., Cinà, A.E., Pelillo, M.: Vector flows and the capacity of a discrete memoryless channel. Preprint at https://arxiv.org/abs/2312.16472 (2023)
3. Blahut, R.: Computation of channel capacity and rate-distortion functions. IEEE Trans. Inf. Theory **18**(4), 460–473 (1972). https://doi.org/10.1109/TIT.1972.1054855
4. Boche, H., Schaefer, R.F., Poor, H.V.: Algorithmic computability and approximability of capacity-achieving input distributions. IEEE Trans. Inf. Theory **69**(9), 5449–5462 (2023). https://doi.org/10.1109/TIT.2023.3278705
5. Chen, R.T.Q.: Torchdiffeq (2018). https://github.com/rtqichen/torchdiffeq
6. Cover, T.M., Thomas, J.A.: Elements of Information Theory. Wiley, Hoboken, NJ, USA, 2nd edn. (2006)
7. Faybusovich, L.: Dynamical systems which solve optimization problems with linear constraints. IMA J. Math. Control Inf. **8**(2), 135–149 (1991). https://doi.org/10.1093/imamci/8.2.135
8. Helmke, U., Moore, J.B.: Optimization and Dynamical Systems. Springer, London (1994)
9. Hirsch, M., Smale, S., Devaney, R.: Differential Equations, Dynamical Systems, and an Introduction to Chaos. Academic Press, Cambridge, MA, USA (2013)
10. Hofbauer, J., Sigmund, K.: Evolutionary Games and Population Dynamics. Cambridge University Press, Cambridge, U.K. (1998)
11. Luenberger, D.G., Ye, Y.: Linear and Nonlinear Programming. Springer, Cham (2016)
12. MacKay, D.J.C.: Information Theory, Inference, and Learning Algorithms. Cambridge University Press, Cambridge, U.K. (2003)
13. Matz, G., Duhamel, P.: Information geometric formulation and interpretation of accelerated Blahut-Arimoto-type algorithms. In: Information Theory Workshop. pp. 66–70. IEEE (2004). https://doi.org/10.1109/ITW.2004.1405276
14. Ng, K.W., Tian, G., Tang, M.: Dirichlet and Related Distributions. Wiley Series in Probability and Statistics, Wiley, Hoboken, NJ, USA (2006)
15. Shannon, C.E.: A mathematical theory of communication. Bell System Technical Journal **27**(3), 379–423 (1948). https://doi.org/10.1002/j.1538-7305.1948.tb01338.x
16. Tope, M.A., Morris, J.M.: A PAC-bound on the channel capacity of an observed discrete memoryless channel. In: 55th Annual Conference on Information Sciences and Systems (CISS). pp. 1–6. IEEE (2021). https://doi.org/10.1109/CISS50987.2021.9400323
17. Weibull, J.: Evolutionary Game Theory. MIT Press, Cambridge, MA, USA (1995)
18. Yu, Y.: Squeezing the Arimoto–Blahut algorithm for faster convergence. IEEE Trans. Inf. Theory **56**(7), 3149–3157 (2010). https://doi.org/10.1109/TIT.2010.2048452