Enabling Complexity-Performance Trade-Offs for Successive Cancellation Decoding of Polar Codes

Alexios Balatsoukas-Stimming, Georgios Karakonstantis, and Andreas Burg

Telecommunications Circuits Laboratory, EPFL, Lausanne, Switzerland.
Email: {alexios.balatsoukas, georgios.karakonstantis, andreas.burg}@epfl.ch

Abstract—**Polar codes are one of the most recent advancements in coding theory and they have attracted significant interest. While they are provably capacity achieving over various channels, they have seen limited practical applications. Unfortunately, the successive nature of successive cancellation based decoders hinders fine-grained adaptation of the decoding complexity to design constraints and operating conditions. In this paper, we propose a systematic method for enabling complexity-performance trade-offs by constructing polar codes based on an optimization problem which minimizes the complexity under a suitably defined mutual information based performance constraint. Moreover, a low-complexity greedy algorithm is proposed in order to solve the optimization problem efficiently for very large code lengths.**

I. Introduction

Channel polarization gives rise to elegant and provably optimal channel codes, called *polar codes* [1], whose decoding complexity under successive cancellation (SC) decoding is $O(N \log N)$, where N is the blocklength of the code. In implementations of channel decoders it is often desirable to trade performance for complexity in order to meet system requirements at minimal cost. These trade-off decisions can be offline, i.e., taken during the design phase of the system, or online, i.e., taken by the system during its operation. A trivial way to vary the decoding complexity is to alter the blocklength of the code. This is usually an offline decision, although some recent communications standards (e.g., IEEE 802.11n [2]) require the support of codes of various lengths. Unfortunately, using this method, polar codes do not offer a fine trade-off granularity because their blocklength is constrained to powers of two when using the simple 2×2 polarizing matrix introduced by Arıkan. Thus, any complexity reduction may lead to a large loss in performance. Moreover, due to its successive nature, the SC decoding algorithm is not amenable to online complexity tuning. Other codes which are used in modern systems, such as LDPC [3] and turbo codes [4], which are usually decoded using iterative decoding algorithms, can be tuned online by varying the number of performed iterations, according to e.g., the channel conditions. Fortunately, a simple observation allows us to trade decoding complexity for performance for SC decoding in small steps, both offline and online, without the need to change the blocklength by altering the set of channels which are used to transmit information.

Contribution: In this work, we populate the complexity-performance trade-off curve for SC decoding by formulating the frozen channel selection step of polar code construction as an optimization problem. This is achieved by reformulating the original problem of polar code construction with the objective to minimize the complexity while respecting quality constraints that represent the various dynamically changing operating conditions, or the offline system constraints. The proposed reformulation enables complexity-performance trade-offs which where not evident before. Finally, we also present a low complexity greedy algorithm which seems to approximate the original problem reasonably well.

II. Polar Codes

Following the notation of [1], we use a_1^N to denote a row vector $(a_1, \ldots, a_N)$ and a_i^j to denote the subvector $(a_i, \ldots, a_j)$. If $j < i$, then the subvector a_i^j is empty. We use $\log(\cdot)$ to denote the binary logarithm.

A. Construction of Polar Codes

Let W denote a binary input discrete and memoryless channel with input $u \in \{0, 1\}$, output $y \in \mathcal{Y}$, and transition probabilities $W(y|u)$. A polar code is constructed by applying a 2×2 *channel combining* transformation recursively on W for n times, followed by a *channel splitting* step [1]. This results in a set of $N = 2^n$ channels, denoted by $W_N^{(i)}(y_1^N, u_1^{i-1}|u_i)$, $i = 1, \ldots, N$. In principle, it is possible to compute the mutual information values $I(Y_1^N, U_1^{i-1}|U_i)$, $i = 1, \ldots, N$. In practice, finding an analytical expression turns out to be a very hard problem, except for the case of the binary erasure channel (BEC), where an exact recursive calculation is possible [1]. Methods for approximating the mutual information values in more general cases are described in [1], [5], [6]. The construction of a polar code is completed by choosing the good channels as *non-frozen* channels which carry information bits, while *freezing* the remaining channels to some known values u_i. The set of frozen channel indices is denoted by $\mathcal{A}^c$ and the set of non-frozen channel indices is denoted by $\mathcal{A}$.

B. Successive Cancellation Decoding of Polar Codes

In the SC decoding algorithm [1], decoding starts by computing an estimate of u_1, denoted by $\hat{u}_1$, based only on y_1^N. Subsequently, u_2 is estimated using $(y_1^N, \hat{u}_1)$, etc. Decisions are taken according to

$$\hat{u}_i = \begin{cases} \arg\max_{u_i \in \{0,1\}} W_N^{(i)}(y_1^N, \hat{u}_1^{i-1}|u_i), & i \in \mathcal{A}, \\ u_i, & i \in \mathcal{A}^c. \end{cases} \quad (1)$$

The channel likelihoods $W(y_i|x_i)$, $x_i \in \{0,1\}$ are combined through the stages of a decoding graph in order to calculate $W_N^{(i)}(y_1^N, \hat{u}_1^{i-1}|u_i)$, $i = 1, \ldots, N$ [1]. The decoding graph contains $N \log N$ nodes. If intermediate results are stored, then each node has to be activated only once during decoding. Thus, we need exactly $N \log N$ node-computations per codeword.

C. Complexity Reduction Through Pruning

Complexity reduction can be achieved by pruning nodes from the decoding graph whose descendant nodes at stage 0 all correspond to frozen channels, since these nodes calculate likelihoods that will never actually be used by the decoding rule [7]. For example, consider a rate-1/2 code corresponding to the decoding graph of Fig. 1 and let $u_1, u_3 \in \mathcal{A}^c$ and $u_2, u_4 \in \mathcal{A}$. In this case, (1) does not require the likelihoods $W_N^{(1)}(y_1^N|u_1)$, $u_1 = 0, 1$, and $W_N^{(3)}(y_1^N, \hat{u}_1^2|u_3)$, $u_3 = 0, 1$, to estimate u_1 and u_3, respectively. So, the corresponding node-computations at stage 0 can be pruned. However, the computations at stage 1 can not be pruned, since their results are required to estimate u_2 and u_4. If, instead, u_1, u_2 were chosen as frozen, then the computations for u_1 and u_2 at stage 0, as well as the two preceding computations at stage 1 could be pruned. In the first example we can prune two node-computations, while in the second example we can prune four node-computations. However, in the second case the error rate performance of the code will be worse since we do not allow the two best channels to carry information.

III. ENABLING COMPLEXITY-PERFORMANCE TRADE-OFFS

A. Complexity and Performance Metrics

The total number of computations that can be saved by pruning the decoding graph, denoted by c, is used as a complexity metric. Let the blocklength N and the rate $R = 1 - \frac{k}{N}$, $k \in \mathbb{N}$, $0 < k < N$ be fixed and let the mutual information values of the N channels be denoted by I_i, $i = 1 \ldots, N$. We use the sum mutual information of the set of non-frozen channels as a performance metric, i.e.,

$$m = \sum_{i \in \mathcal{A}} I_i = N \cdot I(W) - \sum_{i \in \mathcal{A}^c} I_i. \qquad (2)$$

Note that the polar code construction proposed in [1] maximizes this metric under the constraint $|\mathcal{A}| = k$ and let $m_{\max}$ denote this maximum, i.e.,

$$m_{\max} = \max_{\mathcal{A}:|\mathcal{A}|=k} \sum_{i \in \mathcal{A}} I_i. \qquad (3)$$

Since $I_i \geq 0$, $0 \leq i \leq N$, the maximization amounts to selecting the channel indices with the k largest I_i values.

B. Optimization Problem Formulation

From a complexity perspective, it is favorable to form clusters of 2^l, $l \in \mathbb{N}$, frozen channels in order to maximize pruning. In this section, we describe an optimization problem which constructs a polar code of rate R, in a way that maximizes c while ensuring that m is larger than a predefined performance constraint $m' \geq 0$. To this end, the indices of the
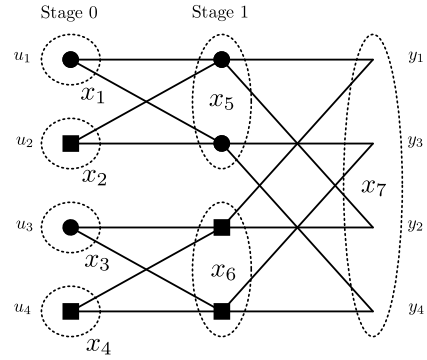

Fig. 1: Decoding graph for $N = 4$ with channel groups. An optimization variable x_i is associated with each group g_i. Setting $x_i = 1$ corresponds to freezing all channels in g_i.
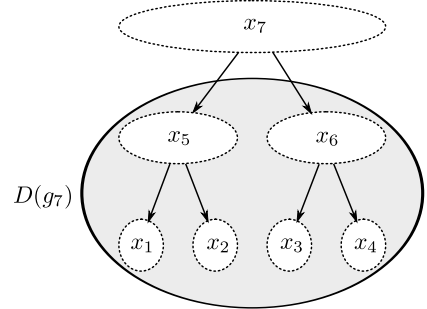

Fig. 2: Tree structure of channel groups with descendants of g_7, i.e., $D(g_7)$, and their corresponding optimization variables. If $x_7 = 1$, then $x_i = 0$ has to be enforced for all $x_i : g_i \in D(g_7)$.

N channels are grouped into clusters of $1, 2, \ldots, N$ consecutive channels as illustrated in Fig. 1, where the illustration of the groups has been spread across the stages of the data dependency graph to reduce congestion. Let the set of all the groups be denoted by $\mathcal{G}$. We have

$$|\mathcal{G}| = N \sum_{j=0}^{n} 2^{-j} = 2N - 1. \qquad (4)$$

We associate each of the groups $g_i \in \mathcal{G}$ with a binary optimization variable x_i, $i = 1, \ldots, 2N - 1$. The assignment $x_i = 1$ means that all channels contained in group i are frozen. Each group also has a cost, denoted by f_i, $i = 1, \ldots, 2N-1$. This cost is equal to the number of channel indices that are contained in g_i, i.e., $f_i = |g_i|$, and it reflects the rate loss incurred by setting $x_i = 1$. This leads to the *rate constraint*

$$\sum_{i=1}^{2N-1} f_i x_i = N - k. \qquad (5)$$

Observe that, if in the example of Fig. 1, say, $x_7 = 1$, then the cost f_7 is paid. However, due to the tree structure of the groups, f_7 includes the costs for freezing the channels in groups g_1 to g_6, So, when $x_i = 1$ for any non-leaf group, $x_i = 0$ has to be enforced for all the descendants of this group in order not to count any costs more than once. Let the descendants of group $g_i \in \mathcal{G}$ be denoted by $D(g_i)$. An

example is illustrated in Fig. 2. Let $\mathcal{M} = \{(i,j) : g_i \in \mathcal{G}\backslash\{\text{leaves}\}, g_j \in D(g_i)\}$. Since $x_i \in \{0,1\}$, the *mutual exclusiveness* constraint can be formalized as

$$x_i + x_j \leq 1, \ \forall (i,j) \in \mathcal{M}. \tag{6}$$

Moreover, we have

$$|\mathcal{M}| = N \sum_{i=1}^{\log N - 1} (\log N - i)2^{-i} = 2(\log N - 1)N + 2. \tag{7}$$

From (4) and (7), it can be seen that the number of variables grows linearly with the code length and the number of constraints in (6) grows as $N \log N$. Each group $g_i \in \mathcal{G}$ has an associated gain in the number of computations, denoted by c_i, $i = 1, \ldots, 2N-1$. This gain is the number of computations that is saved via pruning if all the channels in this group are frozen. Let $s(g_i) \in \{0, \ldots, \log N - 1\}$ denote the stage to which group $g_i \in \mathcal{G}$ corresponds. For example, in Fig. 1, group g_5 corresponds to stage 1. Then, we have

$$c_i = (s(g_i) + 1)2^{s(g_i)}, \quad i = 1, \ldots, 2N - 1. \tag{8}$$

Due to (6), no complexity gain is counted more than once. Finally, freezing the channels in group $g_i \in \mathcal{G}$ results in a loss in total mutual information, denoted by m_i, with

$$m_i = \sum_{j \in g_i} I_j, \quad i = 1, \ldots, 2N - 1. \tag{9}$$

again, due to (6), no loss is counted more than once. A *performance constraint* $m \geq m'$, $m' \geq 0$, is enforced, which can equivalently be written as

$$\sum_{i=1}^{2N-1} x_i m_i \leq N \cdot I(W) - m'. \tag{10}$$

An optimization problem which maximizes the complexity gain, while ensuring that the resulting code has rate R and satisfies the performance constraint, can be formulated as

$$\begin{aligned}
\text{maximize} \quad & \sum_{i=1}^{2N-1} c_i x_i \\
\text{subject to} \quad & \sum_{i=1}^{2N-1} f_i x_i = N - k \\
& \sum_{i=1}^{2N-1} x_i m_i \leq N \cdot I(W) - m' \\
& x_i + x_j \leq 1, \ \forall (i,j) \in \mathcal{M} \\
& x_i \in \{0,1\}, \quad i = 1, \ldots, 2N - 1
\end{aligned} \tag{11}$$

The above problem is a binary integer linear programming formulation of a multidimensional 0–1 knapsack problem [9], which is known to be NP-hard. If m' is chosen carefully so that $m' \leq m_{\max}$, then (11) is always feasible. Moreover, for $m' = m_{\max}$, the optimization problem reduces to the construction proposed by Arıkan,[1] while $m' = 0$ results in

---

[1]Note that, in this case, the solution is not necessarily unique, but each solution of (11) is also a solution of Arıkan's construction.
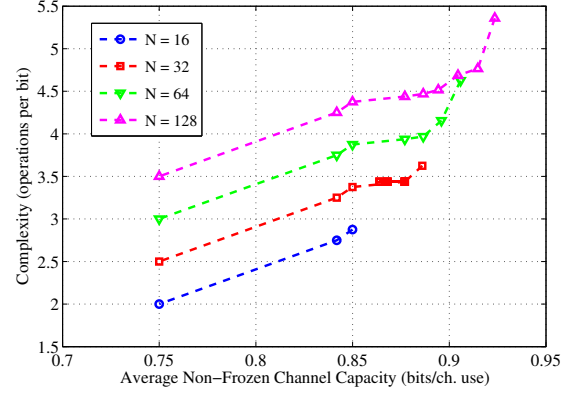

Fig. 3: Solutions of (11) for $R = 0.5$, $N = 2^n$, $n = 4, 5, 6, 7$, and transmission over a BEC(0.5).

a construction that maximizes the number of saved computations while completely disregarding performance. By varying m' between these two extremal values, various complexity-performance trade-offs can be achieved.

C. Results

Even though (11) is NP-hard, relatively small instances can still be solved by using standard branch-and-bound methods. For simplicity in calculating the mutual information values I_i, $i = 1, \ldots, N$, we present results only for the BEC(ϵ), where ϵ denotes the erasure probability. However, the proposed approach can be used for any other channel and input distribution, provided that I_i, $i = 1, \ldots, N$, are available. Moreover, given I_i, $i = 1, \ldots, N$, the complexity of (11) and of the greedy algorithm presented in Section IV does not depend on the type of channel. We assume that the capacity achieving input distribution is used, so that $I(W) = 1 - \epsilon$. In Fig. 3, we present the solutions of (11) for transmission over a BEC(0.5) with a polar code of rate $R = 0.5$ for $N = 2^4, 2^5, 2^6, 2^7$, which are obtained by solving the problem for various $0 \leq m' \leq m_{\max}$. We use the complexity in operations per bit on the vertical axis and the average mutual information on the horizontal axis. The former can be easily obtained from any solution x^* as $\frac{1}{N}\left(N \log N - \sum_{i=1}^{2N-1} c_i x_i^*\right)$, while the latter is equal to $1 + \frac{1}{RN} \sum_{i=1}^{2N-1} m_i x_i^*$.

IV. GREEDY ALGORITHM

In order to solve (11) for practically relevant blocklengths, like $2^{10} \leq N \leq 2^{20}$, in reasonable time, we present a greedy algorithm that takes advantage of the structure of the problem to provide useful solutions with negligible running time.

A. Greedy Algorithm Description

Our greedy algorithm consists of three steps, namely the *greedy maximization* step, the *feasibility* step, and the *post-processing* step. In the first step, the goal is to greedily maximize the objective function while satisfying all inequality constraints. The second step ensures that the equality constraint is also satisfied, while the last step finalizes and improves the solution. Recall that $k' = N - k$ is the number of bits that need

to be frozen. Let k'_{bin} denote the $\log N$ bit right-MSB binary representation of k' and let $k'_{\text{bin}}(j)$, $0 \le j \le \log N - 1$, denote the j-th bit of k'_{bin}. The greedy maximization step is inspired by the following observation.

Proposition 1. *If there were no performance constraint present in (11), the problem could be solved exactly as follows.*

1. *Set $j = \log N - 1$ and $x_i = 0$, $1 \le i \le 2N - 1$.*
2. *If $k'_{bin}(j) = 1$, then set $x_i = 1$ for one $g_i : s(g_i) = j$, denoted by $g_{i'}$, and set $x_i = 0$ for all remaining $g_i : s(g_i) = j$. Remove all $x_i : g_i \in D(g_{i'})$ from the problem.*
3. *Set $j = j - 1$ and go to 2. until $j < 0$.*

Proof: By eliminating all $x_i : g_i \in D(g_{i'})$ from the problem at step 2, we guarantee that the mutual exclusiveness constraint is not violated. Moreover, stage $\log N - 1$ contains two groups, of which only one can be frozen, and for each group in stage j there are two groups in stage $j - 1$, so that step 2 can always be executed. We now show that any optimal solution must freeze at most one group per stage. Suppose that, for some solution, more than one groups were frozen in some stage j. Then, it is possible to replace any two frozen groups at stage j with some frozen group at stage $j+1$ without violating any constraint. Based on (8), for the complexity gains we have

$$2 \cdot \left(j2^{j-1} \right) = j2^j < (j+1)2^j, \ \forall j \ge 0, \qquad (12)$$

so this would strictly increase the objective function, meaning that the original solution could not have been optimal. Since all groups in stage j contain $2^{(j-1)}$ bits and the binary representation of k' is unique, it follows that the only way to freeze exactly k' channels by freezing at most one group per stage, thus satisfying the rate constraint, is to freeze the groups according to the pattern dictated by k'_{bin}. ∎

1) Greedy maximization step: The greedy maximization step is different than the procedure of Proposition 1 in that it makes sure that the performance constraint is satisfied. In the following procedure, k'_{bin} is again initialized to $\log N$ bit right-MSB binary representation of k', but $k'_{\text{bin}}(j) \in \mathbb{N}$.

1. Set $j = \log N - 1$ and $x_i = 0$, $1 \le i \le 2N - 1$.
2. If $k'_{\text{bin}}(j) \ge 1$, then try the following.
   2.1. Find the $g_i : s(g_i) = j$ with the smallest m_i in stage j and set $x_i = 1$.
   2.2. If $\sum_i x_i m_i \le N \cdot I(W) - m'$, then remove all $x_i : g_i \in D(g_{i'})$ from the problem, set $k'_{\text{bin}}(j) = k'_{\text{bin}}(j) - 1$, and go to 2.
   2.3. Else, set $k'_{\text{bin}}(j - 1) = k'_{\text{bin}}(j - 1) + 2$, set $x_i = 0$, and go to 3.
3. Set $j = j - 1$ and go to 2. until $j < 0$.

At step 2.3., we set $k'_{\text{bin}}(j - 1) = k'_{\text{bin}}(j - 1) + 2$ because for each group that could not be frozen at stage j due to the performance constraint, we need to freeze two groups at stage $(j-1)$ in order to (hopefully) satisfy the rate constraint. Unfortunately, there is no longer a guarantee that the procedure will be able to freeze exactly k' bits as required to satisfy the rate constraint. However, the mutual exclusiveness and performance constraints are guaranteed to be met.
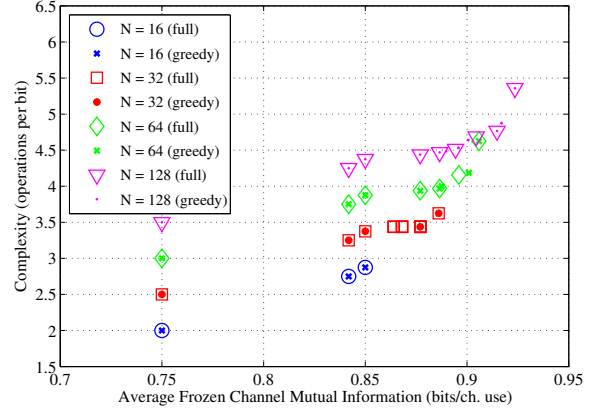

Fig. 4: Results from exact solution of (11) and of the greedy algorithm for $R = 0.5$, $N = 2^n$, $n = 4, 5, 6, 7$, and transmission over a BEC(0.5).

2) Feasibility step: The second step of the algorithm sacrifices the objective function in a systematic step-by-step fashion until the solution is feasible, i.e., until the rate constraint is satisfied. Let k'' denote the number of additional bits that need to be frozen after the greedy maximization step is finished so that the rate constraint is satisfied, i.e., $k'' = k' - \sum_i f_i x_i$.

If $k'' > 0$, then the feasibility step starts greedily unfreezing frozen groups to free up mutual information. More and more groups are unfrozen until the total number of unfrozen groups that can be frozen at stage 0 is equal to k'' plus the number of variables in the groups that were unfrozen so far. Since during this step only groups at stage 0 are refrozen which provide the smallest complexity gain, no direct effort is made to minimize the loss in the objective function. The feasibility step starts at stage $\lceil \log k'' \rceil + 1$, because by unfreezing a group in this stage it is possible to satisfy the rate constraint in a single step, thus making an indirect effort to minimize the objective function loss. Subsequently, all stages up to $\log N - 1$ are visited, and the procedure continues with stages 0 to $\lceil \log k'' \rceil$, thus visiting all stages, if required. If $m' \le m_{\max}$, the feasibility step is guaranteed to find a feasible solution.

3) Post-processing step: The post-processing step identifies pairs of consecutive frozen groups at each stage j and replaces them with their parent group at stage $j + 1$, which improves the objective function without violating any of the constraints.

B. Results

The solutions obtained by solving (11) exactly as well as by using the greedy algorithm for various constraints and blocklength up to $N = 2^7$ and for $R = 0.50$ are compared in Fig. 4. The greedy algorithm is able to find most of the optimal solutions for small instances of the problem.

The solutions found by the greedy algorithm are presented in Fig. 5 for various blocklengths and for $R = 0.50$. For $N = 2^{20}$ the average running time of the greedy algorithm is less than 10^2 seconds, which is negligible given that the optimization is carried out offline. We observe that the rightmost part of the curve is relatively steep, thus providing favorable trade-offs. For a fixed blocklength, the codes corresponding
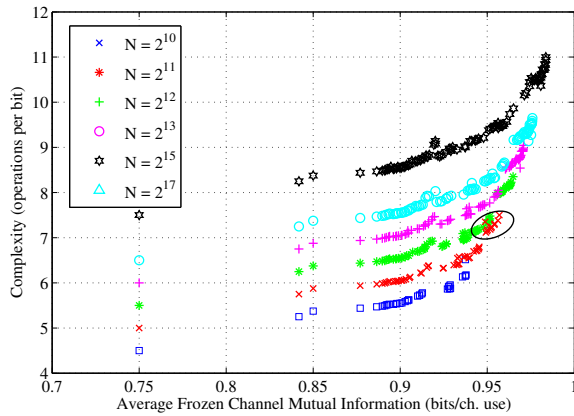
Fig. 5: Solutions of greedy algorithm for $R = 0.5$, $N = 2^n$, $n = 9, 10, 11, 12, 13, 15$, over a BEC(0.5). The circled codes for $N = 2^{10}$ are simulated in Fig. 6.

Fig. 6: BLER performance and performance metric of the useful codes for $N = 2^{10}$.

to some solution points can be chosen and stored in order to provide the system with online performance-complexity trade-offs. Moreover, during the design phase one can choose the solution with the largest performance among all blocklengths that satisfies a given complexity constraint.

V. DISCUSSION

Our choice of performance metric requires some intuitive justification. Let $Z(W)$ denote the Bhattacharyya parameter of a channel W and let $Z_i = Z(W^{(i)})$. It is known that $\sum_{i \in \mathcal{A}} Z_i$ is an upper bound on the probability of block error [1]. It was shown in [8] that, for the BEC, it is also a lower bound. Moreover, for the BEC we have $I_i = 1 - Z_i$, so by maximizing $\sum_{i \in \mathcal{A}} I_i$ one can minimize the probability of block error. Similarly, by placing a constraint on $\sum_{i \in \mathcal{A}} I_i$, we are implicitly placing a constraint on $\sum_{i \in \mathcal{A}} Z_i$, which is directly related with the probability of block error. So, for the case of the BEC, the metric that we use has an explicit relation with the probability of block error. For more general channels, one intuitively expects there to be at least an implicit relation between the two quantities. Ideally, one would like to use the probability of block error itself as a metric, but, to the best of our knowledge, this can not be described analytically as a function of $\mathcal{A}$, and especially not in a linear way which enables a simple formulation of the optimization problem.

Moreover, in principle, it is possible that a solution of (11) contains a very bad channel in $\mathcal{A}$. This would lead to a catastrophic failure of the code, resulting in a block error rate (BLER) close to 1. This problem can be circumvented by adding the following additional constraints to (11)

$$(1 - x_i) \cdot h_i = 0, \ i = 1, \ldots, 2N - 1, \tag{13}$$

where $h_i = 1$ if $g_i \in \mathcal{G}$ contains a channel with $I_i \le m''$, where m'' is chosen as the lowest acceptable mutual information of the channels used for the information bits, and $h_i = 0$ otherwise. However, we have observed in simulations that the useful codes (a code is said to be *useful* if it lies on the Pareto frontier of the set of obtained solutions) have performance which degrades gracefully with decreasing values
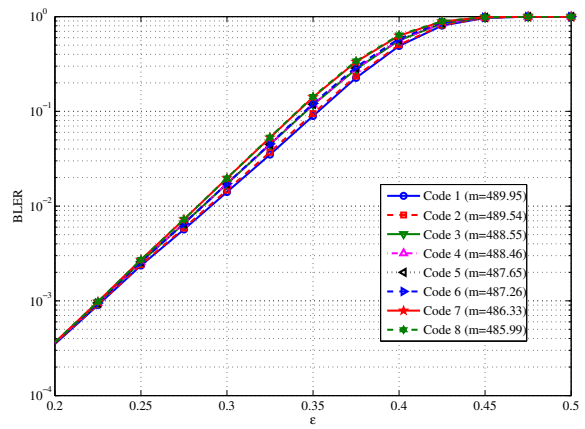
of the performance metric. An example of this behavior for $N = 2^{10}$ can be seen in Fig. 6, where code 1 corresponds to the construction in [1], while codes 2 to 8 provide different performance-complexity trade-offs (annotated in Fig. 5).

VI. CONCLUSION

In this paper, we showed how to achieve fine-grained trade-offs between complexity and performance of SC decoding of polar codes by reformulating the frozen channel selection step of the standard polar code construction procedure as a 0-1 knapsack problem. Moreover, we described a low-complexity greedy algorithm, which is tailored to fit our specific knapsack problem instance. The greedy algorithm was used to approximately solve the optimization problem in order to construct polar codes of blocklength up to $N = 2^{20}$.

REFERENCES

[1] E. Arıkan, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Trans. Inf. Theory*, vol. 55, no. 7, pp. 3051–3073, July 2009.
[2] *IEEE 802.11n-2009 – Amendment 5: Enhancements for Higher Through-put*, IEEE Std., Oct. 2009.
[3] R. G. Gallager, *Low-Density Parity-Check Codes*. Cambridge, MA: MIT Press, 1963.
[4] C. Berrou and A. Glavieux, "Near optimum error correcting coding and decoding: Turbo-codes," *IEEE Trans. Commun.*, vol. 44, no. 10, pp. 1261–1271, Oct. 1996.
[5] R. Pedarsani, S. Hassani, I. Tal, and E. Telatar, "On the construction of polar codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT 2011)*, Aug. 2011, pp. 11–15.
[6] I. Tal and A. Vardy, "How to construct polar codes," arXiv:1105.6164, 2013.
[7] A. Alamdar-Yazdi and F. R. Kschischang, "A simplified successive-cancellation decoder for polar codes," *IEEE Comm. Letters*, vol. 15, no. 12, pp. 1378–1380, Dec. 2011.
[8] M. Bastani Parizi and E. Telatar, "On the correlation between polarized BECs," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT 2013)*, July 2013.
[9] A. Fréville, "The multidimensional 0–1 knapsack problem: An overview," European Journal of Operational Research, vol. 155, no. 1, pp. 1–21, May 2004.