An improvement of the upper bound for GKS communication game

Ivan Petrenko*
Moscow State University

Abstract

The GKS game was formulated by Justin Gilmer, Michal Koucký, and Michael Saks in their research of the sensitivity conjecture. Mario Szegedy invented a protocol for the game with the cost of $O(n^{0.4732})$. Then a protocol with the cost of $O(n^{0.4696})$ was obtained by DeVon Ingram who used a bipartite matching. We propose a slight improvement of Ingram’s method and design a protocol with cost of $O(n^{0.4693})$.

1 Introduction

The GKS game is a three-player game where two players, Alice and Bob, play as a team against their opponent Merlin. The game has two natural parameters $n, k$. In the first phase of the game Merlin and Alice fill an initially empty string $x$ of length $n$ with bits in the following fashion: First Merlin arbitrarily picks a permutation of the set $\{1, 2, \ldots, n\}$. Then he sends indices $1, 2, \ldots, n$ one-by-one in the chosen order to Alice. Once Alice receives an index $i$, she immediately has to pick a bit to fill the $i$th position in the string, that is she chooses $x_i$. This goes on until only one empty position is left, that position is filled by Merlin, not Alice. We will commonly refer to this bit as the last bit. The now-filled string is sent to Bob, whose goal is to determine the position filled by Merlin. To this end Bob chooses a set $S$ of $k$ natural numbers. Alice and Bob win if $S$ contains the position filled by Merlin, otherwise Merlin wins. Alice and Bob can communicate before the game starts but cannot communicate during the play.

*The article was in part funded by RFBR according to the research project 19-01-00563.
A winning strategy for Alice and Bob in the $(k, n)$-game is called a $(k, n)$-strategy. The minimal $k$ for which there is a $(k, n)$-strategy is called the cost of the $n$-game.

The following proposition is very important for the study of the strategies for the GKS game:

**Proposition 1.** If there are a $(k, n)$-strategy and a $(k', n')$-strategy, then there exists a $(kk', nn')$-strategy.

The proof of this proposition can be found in [2]. This result allows to use $(k, n)$-strategies constructed for small values of $n$ to obtain upper bounds for the cost of the $n$-game for arbitrary $n$. More precisely, if a $(K, N)$-strategy exists, then the cost of the $n$-game is $O(n^{\log_N K})$, where the constant hidden in $O$-notation depends on $K, N$. Szegedy proved a $O(n^{0.4732})$ upper bound using $(5, 30)$-strategy [2]. Ingram’s $O(n^{0.4696})$ bound is derived from a $(11, 165)$-strategy [3].

## 2 Results

**Theorem 2.** There exists a $(9, 108)$-strategy.

*Proof.* The proof relies heavily on a table that is presented in Appendix A and was found by a computer search. This table consists of 220 binary strings of length 12. We call those strings codewords. In every codeword three positions are underlined, all those positions contain ones. An important property of the table is that every possible combination of three underlined positions out of $\binom{12}{3} = 220$ combinations appears exactly once. Another property that is important for us is the following:

**Proposition 3.** Let $x$ and $y$ be different codewords. Let $x'$ and $y'$ be strings obtained from $x$ and $y$, respectively, by flipping at most one non-underlined bit. Then $x' \neq y'$.

Informally this property can be understood as follows: the code corrects one error provided it appears in a non-underlined position. This is where our construction differs from that proposed by Ingram [3], who used Hamming codes that correct one error (in any position). Unfortunately, it is rather difficult to verify this property by hand and we cannot provide any proof of this fact. We verified this property by a computer program. The table itself was also found by that program.
We will now construct a \((9, 108)\)-strategy. First, Alice divides the string of length 108 into 9 blocks of length 12. Alice then fills each block independently from other blocks as follows:

1. The three first received positions in each block are filled by ones.
2. Once those three positions are filled, find the (unique) codeword in which those three positions are underlined and assign this codeword to the block.
3. The remaining positions except the last one are filled as in the assigned codeword.
4. The last position is filled so that the resulting block differs from the assigned codeword.

Observe that every block completed by Alice alone is a codeword with one error in a non-underlined position. From Proposition 3 we can conclude that every such block is different from each codeword. So Bob can receive two kinds of strings:

1. Either the received string has a block which is a codeword. In that case Bob knows that this block contains the last filled position and that that position is not underlined. Thus, he forms the set of 9 non-underlined positions in that block.
2. Or, there is no such block in the received string, that is, all the blocks are codewords with one error, and that error occurs in a non-underlined position. Then, due to Proposition 3 for each block Bob can find the assigned codeword and the position of error, that position was filled last. The set of those positions has size 9 and contains the last bit.

In both cases Bob can find a set of 9 positions containing the last bit.

\(\square\)

**Corollary 4.** *The cost of a GKS game is less then \(O(n^{\log_{108}9}) = O(n^{0.4693})\)*

This bound is derived by applying Proposition 1 to Theorem 2.

**References**

[1] Gilmer, J., Koucký, M., Saks, M. A communication game related to the sensitivity conjecture. Theory of Computing 13 (2017) pp. 1–18.

[2] Mario Szegedy. An \(O(n^{0.4732})\) upper bound on the complexity of the GKS communication game. CoRR, abs/1506.06456, 2015.

[3] Ingram, Devon. An Upper Bound on the GKS Game via Max Bipartite Matching, arXiv:1712.01149, 2017.
## Appendix

| 1. 111000100000 | 2. 111110000000 | 3. 111110110110 |
| 4. 110101011011 | 5. 110010100001 | 6. 110000111000 |
| 7. 110010001011 | 8. 110000000100 | 9. 110100000100 |
| 10. 110001100011 | 11. 101110100010 | 12. 101101110010 |
| 13. 101001101100 | 14. 111001110110 | 15. 101001101110 |
| 16. 101000001000 | 17. 101100000110 | 18. 111000011000 |
| 19. 101000000101 | 20. 110110110110 | 21. 100101000111 |
| 22. 110011010100 | 23. 110100101011 | 24. 110111011010 |
| 25. 110100111100 | 26. 11110111011 | 27. 110010000110 |
| 28. 110011101101 | 29. 101110111101 | 30. 110011010000 |
| 31. 100010001100 | 32. 100110011010 | 33. 110010010110 |
| 34. 100001000011 | 35. 100011100000 | 36. 100001010011 |
| 37. 101110101101 | 38. 100001001010 | 39. 010101010010 |
| 40. 111001001001 | 41. 111010111011 | 42. 100110111100 |
| 43. 100101001000 | 44. 100000100100 | 45. 101001100010 |
| 46. 100000100110 | 47. 100011110101 | 48. 100000111111 |
| 49. 101110111101 | 50. 101100001111 | 51. 101001001111 |
| 52. 100011011001 | 53. 100100011111 | 54. 100100110011 |
| 55. 101011111111 | 56. 011110101010 | 57. 011010100011 |
| 58. 011001000010 | 59. 011101000011 | 60. 011100010101 |
| 61. 011000001100 | 62. 011110101111 | 63. 011000001111 |
| 64. 011000111101 | 65. 011010010100 | 66. 110101000000 |
| 67. 110111110001 | 68. 010100110000 | 69. 011100011000 |
| 70. 010101000011 | 71. 010101110001 | 72. 010100001000 |
| 73. 010011001000 | 74. 111011110101 | 75. 011010001100 |
| 76. 111011100000 | 77. 110111000000 | 78. 010011101111 |
| 79. 110100100011 | 80. 010001111010 | 81. 010001000000 |
| 82. 010111011001 | 83. 010110001110 | 84. 010011010001 |
| 85. 010111011111 | 86. 010011111000 | 87. 010000010000 |
| 88. 010110101100 | 89. 010111100111 | 90. 010001110001 |
| 91. 111000011111 | 92. 010101011000 | 93. 010000011010 |
| 94. 111101010001 | 95. 110010000010 | 96. 111000101010 |
| 97. 101001101111 | 98. 011100100110 | 99. 010000100101 |
| 100. 010000010111 | 101. 111110010110 | 102. 101110010011 |
| 103. 011111000011 | 104. 101100010000 | 105. 011000101010 |
| 106. 011111010001 | 107. 101100000111 | 108. 011010000111 |
| 109. 011111010000 | 110. 010101010000 | 111. 101110101010 |

4
| 112. 011111001011 | 113. 011011000101 | 114. 001010000010 |
|---------------------|---------------------|---------------------|
| 115. 001011110001 | 116. 001011000000 | 117. 001110111000 |
| 118. 001001000101 | 119. 011111111101 | 120. 001101010010 |
| 121. 001011000011 | 122. 101100110101 | 123. 001100110100 |
| 124. 001011101101 | 125. 001000101011 | 126. 001100111111 |
| 127. 110001110000 | 128. 011010101000 | 129. 000100010011 |
| 130. 001000111010 | 131. 101100011100 | 132. 001000011110 |
| 133. 001010011011 | 134. 001001100110 | 135. 111100101101 |
| 136. 011001110011 | 137. 000111000110 | 138. 001101010100 |
| 139. 000110011000 | 140. 000111111111 | 141. 001110000100 |
| 142. 010110000110 | 143. 000110110011 | 144. 011011010100 |
| 145. 001101011001 | 146. 000110100001 | 147. 101101000100 |
| 148. 110110000011 | 149. 000111000001 | 150. 000101111110 |
| 151. 000100111001 | 152. 000111000110 | 153. 100101110010 |
| 154. 001110101001 | 155. 100110011011 | 156. 001000011010 |
| 157. 000100010010 | 158. 001110010001 | 159. 000110000110 |
| 160. 100100101111 | 161. 110110101001 | 162. 100110001110 |
| 163. 111110000101 | 164. 101110100011 | 165. 010011100010 |
| 166. 000111110000 | 167. 100011001010 | 168. 000011010100 |
| 169. 101011100111 | 170. 000011001111 | 171. 101101100000 |
| 172. 000020111000 | 173. 101010101001 | 174. 000020100110 |
| 175. 00020100001 | 176. 001011010000 | 177. 011011011000 |
| 178. 010010110010 | 179. 010010010001 | 180. 100111011000 |
| 181. 001011111011 | 182. 011010001001 | 183. 110010100111 |
| 184. 011101011111 | 185. 001100011111 | 186. 011101111000 |
| 187. 000011101001 | 188. 000001011010 | 189. 000011111010 |
| 190. 000001100011 | 191. 100011111010 | 192. 100001011100 |
| 193. 111101111110 | 194. 000001111101 | 195. 001110011000 |
| 196. 000110110100 | 197. 110001010011 | 198. 010001011111 |
| 199. 000001000101 | 200. 000101011111 | 201. 101010111001 |
| 202. 000000110100 | 203. 010101111011 | 204. 100000011001 |
| 205. 010101011010 | 206. 101010101100 | 207. 100000101101 |
| 208. 010000101110 | 209. 010110110101 | 210. 000100010111 |
| 211. 111010011110 | 212. 010010111111 | 213. 000000011001 |
| 214. 001110110110 | 215. 110000010101 | 216. 000100010111 |
| 217. 010110011100 | 218. 0000000001111 | 219. 000101001011 |
| 220. 110100000111 |