

Efficient Computation of an Optimal Portmantout

David Renshaw

31 March 2017

1 Introduction

A *portmantout* is a string composed of sequentially overlapping words from a word set \mathbf{L} , such that each word from \mathbf{L} appears in the string at least once. For example, if $\mathbf{L} = \{\text{an, no, on}\}$, then `anon` is a portmantout for \mathbf{L} ; in fact it is the *only* portmantout for \mathbf{L} . For larger \mathbf{L} , there are often many portmantouts of differing lengths, suggesting a natural question: how short of a portmantout can we construct? Traditionally, interest in this question has centered on a particular 108,709-word word set called `wordlist.asc` [3]. The first published portmantout for `wordlist.asc` had length 630,408 [5]. Later, new methods were used to find a portmantout for `wordlist.asc` of length 537,136 and to prove any portmantout for `wordlist.asc` must have length at least 520,732 [6].

In this paper we present an efficient method for finding an *optimal* portmantout. Although the method is not guaranteed to work on all word sets, it does succeed on `wordlist.asc`, finding an optimally short portmantout of length 536,186 using less than fifteen seconds of computation time. The method is due to Anders Kaseorg, who developed it while solving a related programming puzzle [1] [2].

2 The Method

We encode the portmantout problem as a *minimum-cost network flow* problem. In general, a network flow problem has the following input data:

- A set N of nodes.
- A set A of directed arcs between nodes. Each arc $a \in A$ has an input node $a_{\text{in}} \in N$ and an output node $a_{\text{out}} \in N$.
- A supply map: each node $n \in N$ has associated an integer b_n , representing the “supply” of that node. If b_n is positive, then n is a “source” node. If b_n is negative, then n is a “sink” node.
- A cost map: each arc $a \in A$ has associated an integer c_a representing the cost per unit of flow along that arc.
- A capacity map: each arc a has a capacity $u_a \geq 0$ representing the maximum flow allowed through that arc.

Given this data, we seek a flow x on A to minimize the cost $\sum_{a \in A} c_a x_a$, subject to:

$$\sum_{a \in A, a_{\text{in}}=n} x_a - \sum_{a \in A, a_{\text{out}}=n} x_a = b_n$$
$$0 \leq x_a \leq u_a$$

As is expounded in [4], there are efficient algorithms for solving such problems. Moreover, if the supply, cost, and capacity values are all integers, then there is guaranteed to be an optimal solution vector x that is also entirely integral.

2.1 Encoding the Portmanout Problem

Given a word set \mathbf{L} , we construct a minimum-cost network flow problem as follows:

- For each word $w \in \mathbf{L}$, we create a “start” node $\llbracket w \rrbracket$ and an “end” node $|w\rangle$. If the word is redundant (i.e. is contained in some other word in \mathbf{L}) then these have zero supply. If the word is not redundant, then $\llbracket w \rrbracket$ has supply $b_{\llbracket w \rrbracket} = -1$ and $|w\rangle$ has supply $b_{|w\rangle} = 1$.

- For each $w \in \mathbf{L}$, we create an arc with zero cost from $\llbracket w \rrbracket$ to $|w\rangle$. This is called the “extra” arc. It allows us to reuse words more than once as connectors between other words.
- For each string s that is either a prefix or a suffix of any word in \mathbf{L} , we create an “affix” node \bar{s} . These nodes represent the overlap between successive words in a portmantout.
- For each $w \in \mathbf{L}$ and each affix s , if s is a prefix of w then we create an arc from \bar{s} to $\llbracket w \rrbracket$ with cost equal to the length of w minus the length of s . The cost represents the number of letters contributed to the length of the final portmantout.
- For each $w \in \mathbf{L}$ and each affix s , if s is a suffix of w then we create a zero-cost arc from $|w\rangle$ to \bar{s} .
- We create an “initial” node $|\emptyset\rangle$ with supply $b_{|\emptyset\rangle} = 1$, and for each $w \in \mathbf{L}$ an arc from $|\emptyset\rangle$ to $\llbracket w \rrbracket$ with cost equal to the length of w .
- We create one “final” node $\llbracket \emptyset \rrbracket$ with supply $b_{\llbracket \emptyset \rrbracket} = -1$, and for each $w \in \mathbf{L}$ an arc from $|w\rangle$ to $\llbracket \emptyset \rrbracket$, with cost 0.
- We set the capacity of every arc to ∞ .

2.2 Recovering a Portmantout

We plug the encoded problem into a solver and get back an optimal integral flow x . Our objective now is to read off a portmantout from x by following the flow from the initial node $|\emptyset\rangle$ to the final node $\llbracket \emptyset \rrbracket$, allowing a jump from $\llbracket w \rrbracket$ to $|w\rangle$ for each non-redundant w . To formalize this objective, we create a new directed graph H with the same nodes N as before. For each unit of flow along an arc in x , we draw a distinct arc in H between the same nodes. For example, if $x_a = 3$ then we draw three distinct arcs in H from a_{in} to a_{out} . In addition, for each non-redundant $w \in \mathbf{L}$, we draw an arc from $\llbracket w \rrbracket$ and $|w\rangle$, i.e. from that word’s sink node to its source node. Finally, we draw an arc from $\llbracket \emptyset \rrbracket$ to $|\emptyset\rangle$. We then try to find an Eulerian circuit on H . That is, we look for a path that visits each arc exactly once and ends up where it started. If we can find such a circuit, then we have found an optimal portmantout and we are done.

Unfortunately, H is not guaranteed to have a Eulerian circuit. Consider, for example, the case where $L = \{\text{abyz}, \text{yzab}, \text{zxy}\}$. Then zxyzabyz , which has length 8, is the shortest possible portmantout for L , but the minimum-cost flow algorithm finds a flow of cost 7 with abyz and yzab in a cycle, producing a graph that has no Eulerian circuit. In such cases, the flow x gives a lower bound on the length of any portmantout for \mathbf{L} .

3 Future Work

Is the problem of searching for an optimally-short portmantout NP-hard in general? A plausible-looking equivalence with the Traveling Salesman Problem was presented in [5], but its reduction from TSP uses a unary encoding of the costs of edges, and therefore can cause the size of problems to grow exponentially, which apparently invalidates the proof. Can that proof be repaired? Or, perhaps, can the algorithm described in the present paper be amended to handle all cases?

References

- [1] Compounding english. <https://codegolf.stackexchange.com/questions/87311/compounding-english/87534>.
- [2] shortmantoutmost. <https://github.com/andersk/shortmantoutmost>.
- [3] wordlist.asc. <http://www.cs.cmu.edu/~tom7/portmantout/wordlist.zip>.
- [4] Ravindra K. Ahuja James B. Orlin and Thomas L. Magnanti. *Network Flows: Theory, Algorithms, and Applications*. Pearson, 1993.
- [5] Tom Murphy VII Ph.D. The portmantout. In *Proceedings of SIGBOVIK 2015*, 2015.
- [6] David Renshaw and Jim McCann. A shortmantout. In *Proceedings of SIGBOVIK 2016*, 2016.

A Appendix: An Optimal Portmantout for wordlist.asc

[illegible]

[The following text is a dense, nonsensical string of characters and punctuation, likely representing a corrupted or encrypted document. It contains no legible information.]

[illegible]

[illegible]

beginning from the end of the last century, the development of the world has entered a new era of rapid change. In this era, the pace of scientific and technological progress is unprecedented, and the changes in social structure and human life are profound. The challenges we face today are more complex and diverse than ever before. We need to have a clear understanding of the current situation and the future trends, so as to make wise decisions and take effective actions.

The first challenge is the rapid advancement of science and technology. With the advent of artificial intelligence, big data, and other cutting-edge technologies, our lives and work have been greatly impacted. These technologies bring convenience and efficiency, but they also pose new challenges. For example, how do we ensure the security and privacy of personal information? How do we deal with the ethical issues related to artificial intelligence? These are questions that require us to think deeply and find solutions.

The second challenge is the global environmental crisis. Climate change, air pollution, and other environmental problems have become serious threats to human health and the survival of the planet. We need to take urgent action to reduce greenhouse gas emissions, protect forests, and conserve water resources. Only by working together can we solve these global problems and build a sustainable future.

The third challenge is the increasing diversity of cultures and peoples. As globalization deepens, people from different backgrounds are coming into contact more frequently. This brings rich cultural exchanges, but it also leads to misunderstandings and conflicts. We need to promote mutual understanding and respect, learn from each other's strengths, and build a harmonious society where everyone can live in peace and harmony.

In conclusion, the world is full of opportunities and challenges. We must stay calm and rational, analyze the situation objectively, and take proactive measures. Only through continuous learning and innovation can we overcome all difficulties and achieve our common goals. Let us unite our efforts and create a better world for ourselves and for future generations.

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

1
 2
 3
 4
 5
 6
 7
 8
 9
 10
 11
 12
 13
 14
 15
 16
 17
 18
 19
 20
 21
 22
 23
 24
 25
 26
 27
 28
 29
 30
 31
 32
 33
 34
 35
 36
 37
 38
 39
 40
 41
 42
 43
 44
 45
 46
 47
 48
 49
 50
 51
 52
 53
 54
 55
 56
 57
 58
 59
 60
 61
 62
 63
 64
 65
 66
 67
 68
 69
 70
 71
 72
 73
 74
 75
 76
 77
 78
 79
 80
 81
 82
 83
 84
 85
 86
 87
 88
 89
 90
 91
 92
 93
 94
 95
 96
 97
 98
 99
 100
 101
 102
 103
 104
 105
 106
 107
 108
 109
 110
 111
 112
 113
 114
 115
 116
 117
 118
 119
 120
 121
 122
 123
 124
 125
 126
 127
 128
 129
 130
 131
 132
 133
 134
 135
 136
 137
 138
 139
 140
 141
 142
 143
 144
 145
 146
 147
 148
 149
 150
 151
 152
 153
 154
 155
 156
 157
 158
 159
 160
 161
 162
 163
 164
 165
 166
 167
 168
 169
 170
 171
 172
 173
 174
 175
 176
 177
 178
 179
 180
 181
 182
 183
 184
 185
 186
 187
 188
 189
 190
 191
 192
 193
 194
 195
 196
 197
 198
 199
 200
 201
 202
 203
 204
 205
 206
 207
 208
 209
 210
 211
 212
 213
 214
 215
 216
 217
 218
 219
 220
 221
 222
 223
 224
 225
 226
 227
 228
 229
 230
 231
 232
 233
 234
 235
 236
 237
 238
 239
 240
 241
 242
 243
 244
 245
 246
 247
 248
 249
 250
 251
 252
 253
 254
 255
 256
 257
 258
 259
 260
 261
 262
 263
 264
 265
 266
 267
 268
 269
 270
 271
 272
 273
 274
 275
 276
 277
 278
 279
 280
 281
 282
 283
 284
 285
 286
 287
 288
 289
 290
 291
 292
 293
 294
 295
 296
 297
 298
 299
 300
 301
 302
 303
 304
 305
 306
 307
 308
 309
 310
 311
 312
 313
 314
 315
 316
 317
 318
 319
 320
 321
 322
 323
 324
 325
 326
 327
 328
 329
 330
 331
 332
 333
 334
 335
 336
 337
 338
 339
 340
 341
 342
 343
 344
 345
 346
 347
 348
 349
 350
 351
 352
 353
 354
 355
 356
 357
 358
 359
 360
 361
 362
 363
 364
 365
 366
 367
 368
 369
 370
 371
 372
 373
 374
 375
 376
 377
 378
 379
 380
 381
 382
 383
 384
 385
 386
 387
 388
 389
 390
 391
 392
 393
 394
 395
 396
 397
 398
 399
 400
 401
 402
 403
 404
 405
 406
 407
 408
 409
 410
 411
 412
 413
 414
 415
 416
 417
 418
 419
 420
 421
 422
 423
 424
 425
 426
 427
 428
 429
 430
 431
 432
 433
 434
 435
 436
 437
 438
 439
 440
 441
 442
 443
 444
 445
 446
 447
 448
 449
 450
 451
 452
 453
 454
 455
 456
 457
 458
 459
 460
 461
 462
 463
 464
 465
 466
 467
 468
 469
 470
 471
 472
 473
 474
 475
 476
 477
 478
 479
 480
 481
 482
 483
 484
 485
 486
 487
 488
 489
 490
 491
 492
 493
 494
 495
 496
 497
 498
 499
 500
 501
 502
 503
 504
 505
 506
 507
 508
 509
 510
 511
 512
 513
 514
 515
 516
 517
 518
 519
 520
 521
 522
 523
 524
 525

[illegible]