IT3708 - Project 2

Magnus Moan

Implementation

Chromosome representation

Each of my chromosomes consists of a list of depots. Each depot consist of a list of trucks, and finally each truck consists of a list of customers. I've modelled the chromosomes, depots, trucks and customers using classes. So each object of the type chromosome have a list consisting of objects of the type depot. Each depot have a constant number of trucks, and I define a truck to not be in use if it has zero customers in its customer list.

Crossover operator

My crossover operator takes as input two chromosomes, c_1 and c_2. It then follows the following steps:

1. Pick a random depot, call this d. This is the same depot in both chromosomes.
2. Choose a random truck in d in c_1, call this truck t_1.
3. Choose a random truck in d in c_2, call this truck t_2.
4. Remove the customers in t_1 from c_2.
5. Remove the customers in t_2 from c_1.
6. For each customer in t_1:
    i. Loop through all trucks in c_2.
    ii. Find the best* truck and position in that truck to insert the current customer into.
7. Repeat step 6. with t_2 instead of t_1 and c_1 instead of c_2.

*By best I mean the truck and position in that truck that will give the best fitness after inserting the current customer compared to inserting the customer into any other truck and position in that truck within the current depot, d.

Mutation operator

I've three different types of mutations:

1. Move a randomly chosen customer from a depot to another depot. The customer is inserted into the truck and position in the new depot that gives the best fitness.
2. Swap two randomly chosen customers within a truck.
3. Move a randomly chosen customer from a truck to another truck within the same depot

Type 1 is performed 90% of the times a chromosome mutates, while type 2 and 3 is performed 5% of the times a chromosome mutates.

Selection mechanism

I've used tournament selection combined with elitism for selecting parents for a new generation. The tournament selection is performed by picking two random chromosomes from the current population. In 80% of the cases only the fittest chromosome is passed on as a parent. In the other 20% of the cases both chromosomes are passed on as parents. I will explain elitism in the elitism section further down.

Parameter values

Population size: 400, Number of generations: 1500, Crossover rate: 0.9, Mutation rate: 0.1

Infeasible offsprings

Non of my mutations or my crossover function ensures that I create only feasible offspring. Instead I penalise chromosomes that have trucks that break either capacity or duration. I've chosen to penalise each unit above the limit for duration and capacity with a constant factor (10 in my implementation). Furthermore, each depot have a given number of trucks associated with it from the beginning of my algorithm. This number of trucks never increase or decrease, so I will never break the maximum number of trucks restriction. Every time I find a feasible offspring I check whether or not it has a better fitness than my current best feasible offspring, if does have a better fitness I replace the current best feasible offspring with this chromosome.

Fitness function

Fitness of a chromosome is the sum of the fitness of all its depots. The fitness of a depot is the sum of the fitness of all its trucks. The fitness of a trucks is the total distance traveled by the truck, plus any penalty for carrying either too much (breaking the capacity constraint) or having a too long duration (duration constraint). The penalty is calculated as described under "Infeasible offsprings". A lower value for fitness is considered better than a higher (might be a bit misguiding to use the word fitness if lower is better).

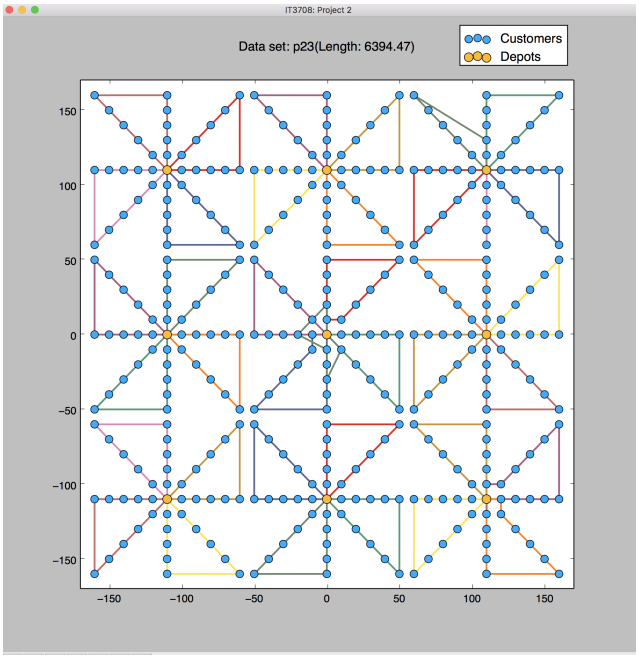
Elitism

I've chosen to implement elitism to make sure I don't lose a good solution once I've found it. In my current implementation 2% of a population is automatically part of the next population (these chromosomes are unchanged between the generations).

Solution example

Moan (vim)

```
6394.47
1   1   176.57  46   0 5 8 16 24 32 40 37 29 21 0
1   2   176.57  42   0 2 1 9 17 25 33 36 28 0
1   3   170.71  42   0 7 15 23 31 39 38 30 22 14 0
1   4   127.15  52   0 13 10 4 12 20 6 0
1   5   170.71  34   0 3 11 19 27 35 34 26 18 0
2   1   170.71  54   0 45 53 61 69 77 75 67 59 51 43 0
2   2   170.71  54   0 48 56 64 72 80 79 71 63 55 47 0
2   3   170.71  54   0 42 50 58 66 74 73 65 57 49 41 0
2   4   170.71  54   0 46 54 62 70 78 76 68 60 52 44 0
3   1   100.00  27   0 92 100 108 116 84 0
3   2   170.71  54   0 81 89 97 105 113 114 106 98 90 82 0
3   3   170.71  54   0 87 95 103 111 119 118 110 102 94 86 0
3   4   170.71  34   0 88 96 104 112 120 117 109 101 0
3   5   149.02  47   0 85 93 115 107 99 91 83 0
4   1   170.71  54   0 125 133 141 149 157 155 147 139 131 123 0
4   2   170.71  54   0 124 132 140 148 156 158 150 142 134 126 0
4   3   170.71  54   0 122 130 138 146 154 153 145 137 129 121 0
4   4   170.71  54   0 128 136 144 152 160 159 151 143 135 127 0
5   1   170.71  54   0 165 173 181 189 197 195 187 179 171 163 0
5   2   170.71  54   0 162 170 178 186 194 193 185 177 169 161 0
5   3   170.71  54   0 168 176 184 192 200 199 191 183 175 167 0
5   4   170.71  54   0 164 172 180 188 196 198 190 182 174 166 0
6   1   170.71  54   0 201 209 217 225 233 236 228 220 212 204 0
6   2   170.71  54   0 205 213 221 229 237 240 232 224 216 208 0
6   3   170.71  54   0 203 211 219 227 235 234 226 218 210 202 0
6   4   170.71  54   0 206 214 222 230 238 239 231 223 215 207 0
7   1   170.71  54   0 244 252 260 268 276 278 270 262 254 246 0
7   2   170.71  54   0 247 255 263 271 279 280 272 264 256 248 0
7   3   170.71  54   0 241 249 257 265 273 274 266 258 250 242 0
7   4   170.71  54   0 245 253 261 269 277 275 267 259 251 243 0
8   1   170.71  54   0 288 296 304 312 320 317 309 301 293 285 0
8   2   170.71  54   0 282 290 298 306 314 315 307 299 291 283 0
8   3   170.71  54   0 286 294 302 310 318 319 311 303 295 287 0
8   4   170.71  54   0 284 292 300 308 316 313 305 297 289 281 0
9   1   176.57  54   0 335 343 351 359 360 352 344 336 328 325 0
9   2   176.57  54   0 332 340 348 356 358 350 342 334 326 327 0
9   3   170.71  54   0 321 329 337 345 353 354 346 338 330 322 0
9   4   170.71  42   0 333 341 349 357 355 347 339 331 323 0
9   5   20.00   12   0 324 0
```



Dataset p23