جامعة بيرزيت

BIRZEIT UNIVERSITY

Department of Electrical & Computer Engineering
ENCS3340 - Artificial Intelligence

Course Project I

Prepared by: Mumen Anbar 1212297
Prepared by: Hamza Al-Shaer 1211162
Instructor: Dr. Ismail Khater & Dr. Yazan Abu Farha
Date: May 19, 2024

Abstract

This project aims to develop a deep understanding of genetic algorithm where it was applied on some data to find a good solution as much as possible, going deep in genetic algorithm details and which steps needed and what techniques to follow, also it improved our comprehension about it and our skills in general implementation.

c++ programming language was used during this project to implement concepts since it's really helpful and has several built in STLs and templates.

Contents

1 Theory **1**

1.1 Genetic Algorithm 1

2 Procedure **2**

2.1 Formulation 2

2.2 Population Initialization 2

2.3 Crossover 2

2.4 Mutation 3

2.4.1 Random Adding 3

2.4.2 Reproduction 3

3 Results And Analysis **3**

4 Conclusion **5**

List of Figures

2.1 Chromosome Representation. 2
2.2 Decaying Exponential Distribution Applied on The Heap. 2
2.3 Growing Exponential Distribution Applied on The Heap. 3
3.1 Results Per Iterations. 3
3.2 Results Per Iterations. 4

1 Theory

1.1 Genetic Algorithm

Genetic Algorithms(GAs) are adaptive heuristic search algorithms that belong to the larger part of evolutionary algorithms. Genetic algorithms are based on the ideas of natural selection and genetics. These are intelligent exploitation of random searches provided with historical data to direct the search into the region of better performance in solution space. They are commonly used to generate high-quality solutions for optimization problems and search problems. [1]

The genetic algorithm belongs to search algorithms. We have a defined target to find and we know what it is. We create random values, called chromosome and all chromosome belong to a population. Each chromosome holds a value which can be our value to find. And like nature does, a population changes or evolves. This means in short:

1. Initialize a population (first generation).

2. Calculate their fitness (the fitness will tell how close we're to our target)

3. Find the strongest chromosomes.

4. Cross chromosomes And Create A New Population (some chromosomes we create from existing ones, some are entirely new).

5. Mutate chromosomes (this means values are changing with a slight probability). Go back to 2. until we find our target.

2 Procedure

2.1 Formulation

There are several ways to represent the formulation of the problem, the one that we've used is that dealing with each sub-job in some job as they are identical, where the chromosome has the following form shown in figure 2.1. A preinitialized map that maps between each sub-task in a job and its own machine, using this we guarantee that no sub-tasks of some job works in parallel in two different machines, leading to a valid separation of the sub-tasks over their machines.

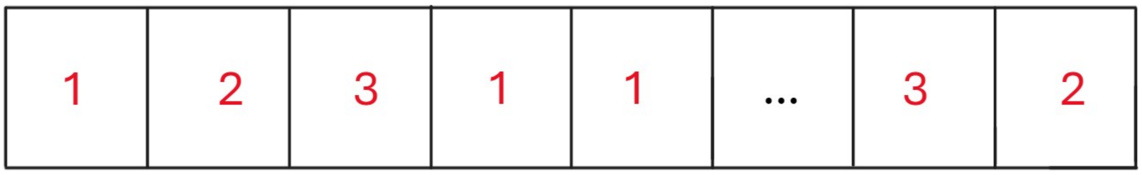


Figure 2.1: Chromosome Representation.

2.2 Population Initialization

The initial population was created by shuffling the maltiest which simulates the previous form of the chromosome, and that was done using the built in function in c++. Then, insert all of them into a heap (priority queue in c++) to sort them according to their fitness value.

2.3 Crossover

After having a population with a predefined size (let's say N), $60\% of N$ are chosen to apply the crossover operation, choosing criteria was applied using the decaying exponential distribution as figure 2.2 shows. Where we are interested in the instances with high fitness value more than the ones with small values. Also the crossover point is chosen randomly.

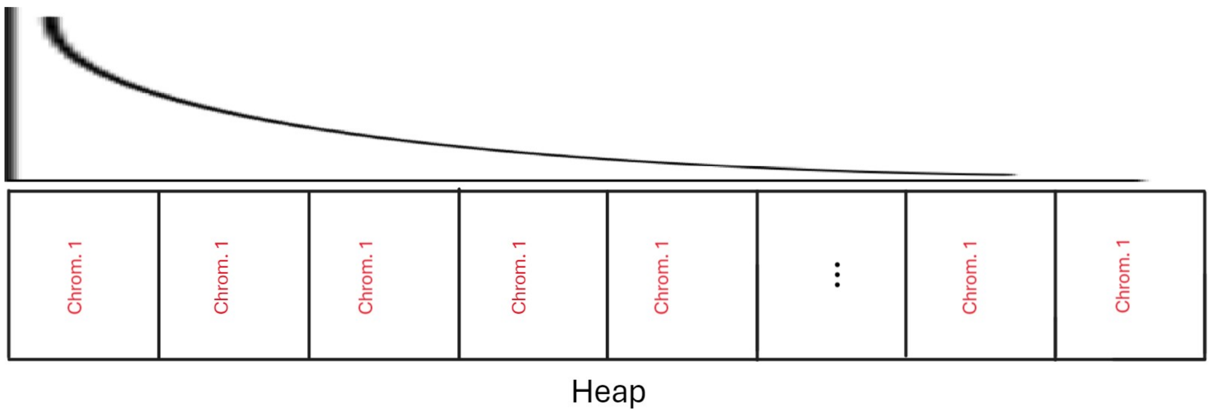


Figure 2.2: Decaying Exponential Distribution Applied on The Heap.

2.4 Mutation

After applying crossover on the population as explained before, then 15%$of N$ are chosen to apply the mutation operation, choosing criteria is the same with crossover one but with growing exponential distribution as figure 2.3 shows. Doing that to get refreshed population every dozen of running time applying the algorithm. Mutation is done by choosing distinct points with different values in the chromosome and swap them.

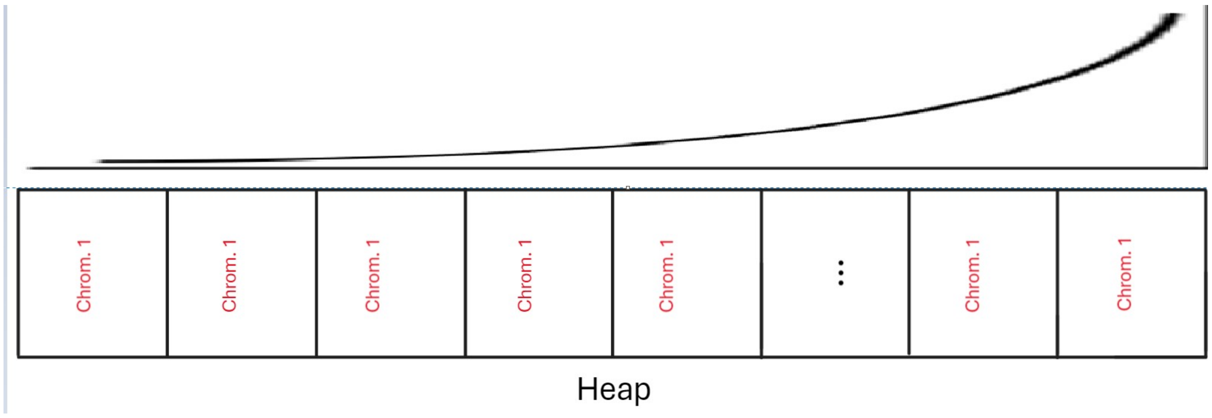


Figure 2.3: Growing Exponential Distribution Applied on The Heap.

2.4.1 Random Adding

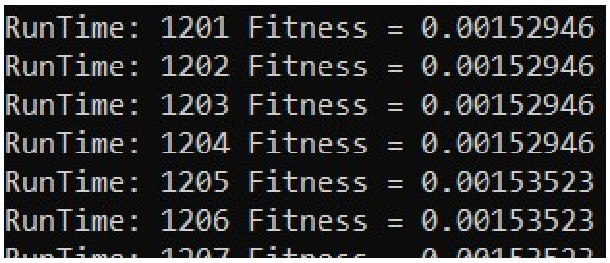
10% of the current population will be added to the new one with entirely new randomly generated chromosomes, and that's related to get out of local maxima area as much as possible.

2.4.2 Reproduction

15% of the current population will be inherited to the new one with as they are, choosing operation is done randomly to avoid patterns as possible, and that's related to get out of local maxima area as much as possible.

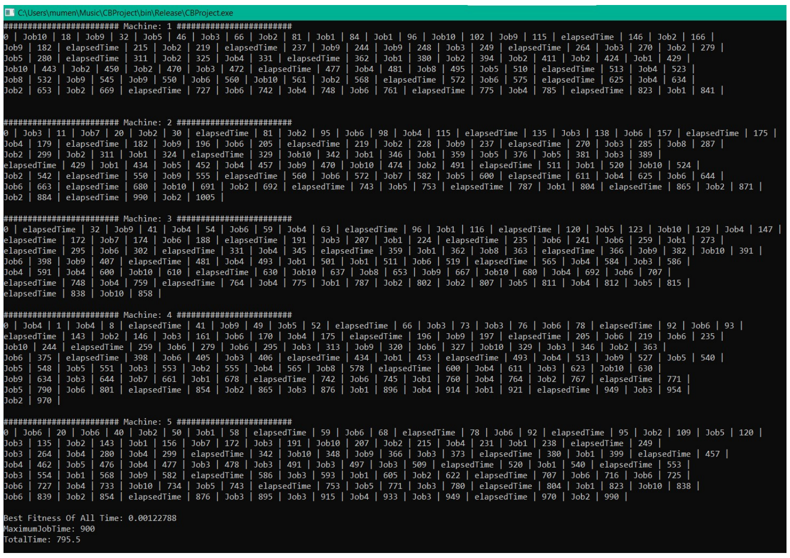
3 Results And Analysis

Testing the code with the attached sample in the (.zip) file gives a convincing results which gets updated every several number of iterations, which means that local maxima area is avoided as needed and the results satisfies our desires. As the following figure shows, the results keep changing even after 1000 iterations of the applying the algorithm and the population keeps updating.



```
RunTime: 1201 Fitness = 0.00152946
RunTime: 1202 Fitness = 0.00152946
RunTime: 1203 Fitness = 0.00152946
RunTime: 1204 Fitness = 0.00152946
RunTime: 1205 Fitness = 0.00153523
RunTime: 1206 Fitness = 0.00153523
```

Figure 3.1: Results Per Iterations.



C:\Users\mumen\Music\CBProject\bin\Release\CBProject.exe

```
######################## Machine: 1 ########################
0 | Job10 | 18 | Job9 | 32 | Job5 | 46 | Job3 | 66 | Job2 | 81 | Job1 | 84 | Job1 | 96 | Job10 | 102 | Job9 | 115 | elapsedTime | 146 | Job2 | 166 |
Job9 | 182 | elapsedTime | 215 | Job2 | 219 | elapsedTime | 237 | Job9 | 244 | Job9 | 248 | Job3 | 249 | elapsedTime | 264 | Job3 | 270 | Job2 | 279 |
Job5 | 280 | elapsedTime | 311 | Job2 | 325 | Job4 | 331 | elapsedTime | 362 | Job1 | 380 | Job2 | 394 | Job2 | 411 | Job2 | 424 | Job1 | 429 |
Job10 | 443 | Job2 | 450 | Job2 | 470 | Job3 | 472 | elapsedTime | 477 | Job4 | 481 | Job8 | 495 | Job5 | 510 | elapsedTime | 513 | Job4 | 523 |
Job8 | 532 | Job9 | 545 | Job9 | 550 | Job6 | 560 | Job10 | 561 | Job2 | 568 | elapsedTime | 572 | Job6 | 575 | elapsedTime | 625 | Job4 | 634 |
Job2 | 653 | Job2 | 669 | elapsedTime | 727 | Job6 | 742 | Job4 | 748 | Job6 | 761 | elapsedTime | 775 | Job4 | 785 | elapsedTime | 823 | Job1 | 841 |


######################## Machine: 2 ########################
0 | Job3 | 11 | Job7 | 20 | Job2 | 30 | elapsedTime | 81 | Job2 | 95 | Job6 | 98 | Job4 | 115 | elapsedTime | 135 | Job3 | 138 | Job6 | 157 | elapsedTime | 175 |
Job4 | 179 | elapsedTime | 182 | Job9 | 196 | Job6 | 205 | elapsedTime | 219 | Job2 | 228 | Job9 | 237 | elapsedTime | 270 | Job3 | 285 | Job8 | 287 |
Job2 | 299 | Job2 | 311 | Job1 | 324 | elapsedTime | 329 | Job10 | 342 | Job1 | 346 | Job1 | 359 | Job5 | 376 | Job5 | 381 | Job3 | 389 |
elapsedTime | 429 | Job1 | 434 | Job5 | 452 | Job4 | 457 | Job9 | 470 | Job10 | 474 | Job2 | 491 | elapsedTime | 511 | Job1 | 520 | Job10 | 524 |
Job2 | 542 | elapsedTime | 550 | Job9 | 555 | elapsedTime | 560 | Job6 | 572 | Job7 | 582 | Job5 | 600 | elapsedTime | 611 | Job4 | 625 | Job6 | 644 |
Job6 | 663 | elapsedTime | 680 | Job10 | 691 | Job2 | 692 | elapsedTime | 743 | Job5 | 753 | elapsedTime | 787 | Job1 | 804 | elapsedTime | 865 | Job2 | 871 |
Job2 | 884 | elapsedTime | 990 | Job2 | 1005 |

######################## Machine: 3 ########################
0 | elapsedTime | 32 | Job9 | 41 | Job4 | 54 | Job6 | 59 | Job4 | 63 | elapsedTime | 96 | Job1 | 116 | elapsedTime | 120 | Job5 | 123 | Job10 | 129 | Job4 | 147 |
elapsedTime | 172 | Job7 | 174 | Job6 | 188 | elapsedTime | 191 | Job3 | 207 | Job1 | 224 | elapsedTime | 235 | Job6 | 241 | Job6 | 259 | Job1 | 273 |
elapsedTime | 295 | Job6 | 302 | elapsedTime | 331 | Job4 | 345 | elapsedTime | 359 | Job1 | 362 | Job8 | 363 | elapsedTime | 366 | Job9 | 382 | Job10 | 391 |
Job6 | 398 | Job9 | 407 | elapsedTime | 481 | Job4 | 493 | Job1 | 501 | Job1 | 511 | Job6 | 519 | elapsedTime | 565 | Job4 | 584 | Job3 | 586 |
Job4 | 591 | Job4 | 600 | Job10 | 610 | elapsedTime | 630 | Job10 | 637 | Job8 | 653 | Job9 | 667 | Job10 | 680 | Job4 | 692 | Job6 | 707 |
elapsedTime | 748 | Job4 | 759 | elapsedTime | 764 | Job4 | 775 | Job1 | 787 | Job2 | 802 | Job2 | 807 | Job5 | 811 | Job4 | 812 | Job5 | 815 |
elapsedTime | 838 | Job10 | 858 |

######################## Machine: 4 ########################
0 | Job4 | 1 | Job4 | 8 | elapsedTime | 41 | Job9 | 49 | Job5 | 52 | elapsedTime | 66 | Job3 | 73 | Job3 | 76 | Job6 | 78 | elapsedTime | 92 | Job6 | 93 |
elapsedTime | 143 | Job2 | 146 | Job3 | 161 | Job6 | 170 | Job4 | 175 | elapsedTime | 196 | Job9 | 197 | elapsedTime | 205 | Job6 | 219 | Job6 | 235 |
Job10 | 244 | elapsedTime | 259 | Job6 | 279 | Job6 | 295 | Job3 | 313 | Job9 | 320 | Job6 | 327 | Job10 | 329 | Job3 | 346 | Job2 | 363 |
Job6 | 375 | elapsedTime | 398 | Job6 | 405 | Job3 | 406 | elapsedTime | 434 | Job1 | 453 | elapsedTime | 493 | Job4 | 513 | Job9 | 527 | Job5 | 540 |
Job5 | 548 | Job5 | 551 | Job3 | 553 | Job2 | 555 | Job4 | 565 | Job8 | 578 | elapsedTime | 600 | Job4 | 611 | Job3 | 623 | Job10 | 630 |
Job9 | 634 | Job3 | 644 | Job7 | 661 | Job1 | 678 | elapsedTime | 742 | Job6 | 745 | Job1 | 760 | Job4 | 764 | Job2 | 767 | elapsedTime | 771 |
Job5 | 790 | Job6 | 801 | elapsedTime | 854 | Job2 | 865 | Job3 | 876 | Job1 | 896 | Job4 | 914 | Job1 | 921 | elapsedTime | 949 | Job3 | 954 |
Job2 | 970 |

######################## Machine: 5 ########################
0 | Job6 | 20 | Job6 | 40 | Job2 | 50 | Job1 | 58 | elapsedTime | 59 | Job6 | 68 | elapsedTime | 78 | Job6 | 92 | elapsedTime | 95 | Job2 | 109 | Job5 | 120 |
Job3 | 135 | Job2 | 143 | Job1 | 156 | Job7 | 172 | Job3 | 191 | Job10 | 207 | Job2 | 215 | Job4 | 231 | Job1 | 238 | elapsedTime | 249 |
Job3 | 264 | Job4 | 280 | Job4 | 299 | elapsedTime | 342 | Job10 | 348 | Job9 | 366 | Job3 | 373 | elapsedTime | 380 | Job1 | 399 | elapsedTime | 457 |
Job4 | 462 | Job5 | 476 | Job4 | 477 | Job3 | 478 | Job3 | 491 | Job3 | 497 | Job3 | 509 | elapsedTime | 520 | Job1 | 540 | elapsedTime | 553 |
Job3 | 554 | Job1 | 568 | Job9 | 582 | elapsedTime | 586 | Job3 | 593 | Job1 | 605 | Job2 | 622 | elapsedTime | 707 | Job6 | 716 | Job6 | 725 |
Job6 | 727 | Job4 | 733 | Job10 | 734 | Job5 | 743 | elapsedTime | 753 | Job5 | 771 | Job3 | 780 | elapsedTime | 804 | Job1 | 823 | Job10 | 838 |
Job6 | 839 | Job2 | 854 | elapsedTime | 876 | Job3 | 895 | Job3 | 915 | Job4 | 933 | Job3 | 949 | elapsedTime | 970 | Job2 | 990 |

Best Fitness Of All Time: 0.00122788
MaximumJobTime: 900
TotalTime: 795.5
```

Figure 3.2: Results Per Iterations.

4 Conclusion

During this project, we've seen deeply how genetic algorithm might be applied, what stages it goes in and their specifications, improved our implementation and problem solving skills. We really look forward to improve our knowledge in local search algorithms and methods to solve problems such Np-Hard ones which are really frequent these days.

References

[1] Genetic algorithm. Accessed on 2023-05-19. [Online]. Available: https://www.geeksforgeeks.org/genetic-algorithms