

Using Genetic Algorithms to Regulate the Sale of Network Coverage Areas

Brian T. Lamb[†]
Wake Forest University
Department of Computer Science
Winston-Salem, North Carolina 27109
lambbt5@wfu.edu

ABSTRACT

Combinatorial auction problems are well known to be NP-hard. To solve a combinatorial auction problem such as the FCC problem may require examination of $n!$ combinations. This process quickly becomes infeasible. This paper explores using a genetic algorithm to solve the FCC problem. While genetic algorithms are not guaranteed to provide the best answer every time, they do provide very good answers in a much shorter amount of time. This paper explores using different components of genetic algorithms to determine what combination of techniques will produce optimal results for this particular problem.

Categories and Subject Descriptors

J.4 [Computer Applications]: Economics

General Terms

Combinatorial auctions, genetic algorithms, FCC problem

1. INTRODUCTION

The FCC regulates pricing for cellular telecommunications or network zones. Organizations bid on different network coverage areas as well as any percentage of any set of network coverage areas. Therefore, with multiple organizations bidding on multiple network coverage areas or zones, determining the maximum profit is not as simple as computing the highest total amount bid. The maximum can be one or any combination of bids. With a large number of bids, determining the maximum amount for which each zone is sold is not a feasible option. Using the bids from the organizations, the FCC must determine what price to charge for each network in order to maximize profits, yet keep them affordable. One price for a network might yield a high profit, but the bidders that can afford that price might collectively

request more resources than are available. Therefore, the prices must be feasible; the collective demand of bidders that can afford that price must not exceed the amount of available resources.

Consider the simple example of four network zones, each with an allocation of 100 units. In the real world, the network zones do not all have to be the same size; however, for simplicity in this example, this assumption is made. Suppose there are four organizations placing bids for the various zones. Suppose further for each zone bid, the organization requests 40 network units. An example of a set of bids is shown in Figure 1. With each bid requesting 40 units of that network, at most two bids can be accepted without exceeding the network allotment. If the FCC sets the price of each zone to be the minimum non-zero submitted zone bid (zone 1 at 10, zone 2 at 20, zone 3 at 30, and zone 4 at 50) then no bid is excluded. Because no one bid is larger than any combination of the other three bids, two bids should be accepted. Thus the maximum possible profit is the highest two bids which is $(10 + 30 + 50)$ for bid A and $(10 + 30)$ for bid C giving a total of 130.

A: $\langle (10, 40), (0, 0), (30, 40), (50, 40) \rangle$
B: $\langle (30, 0), (20, 40), (0, 0), (0, 0) \rangle$
C: $\langle (15, 40), (0, 0), (50, 40), (0, 0) \rangle$
D: $\langle (40, 40), (0, 0), (0, 0), (0, 0) \rangle$

Figure 1: These four vectors represent sample bids. Each component is an ordered pair. The first element represents the maximum amount the bidder is willing to pay for that zone. The second element represents how many units of that zone the bidder would like to purchase. A bid of 0 for a specific zone means the bidder does not want to purchase any units of that zone.

[†]This work was partially supported by the Wake Forest University Research Fellowship Program.

Prices for the network zones are represented by vector elements. Each component of the vector is an ordered pair. The first element of the ordered pair represents the maximum amount the buyer is willing to pay for that network and the second element represents the number of units the buyer is requesting. If the final price for a particular zone is more than the buyer is willing to pay for that zone, then the buyer does not purchase any of the zones. In other words, the buyer wants to be able to purchase all zones requested or none at all.

2. APPROACHES TO THE PROBLEM

The FCC problem, essentially, is an integer knapsack problem. It is well known to be NP-hard. The input data is a finite set of pairs of integers as with the example shown in Figure 1. In this set of data, the program must choose different bids that satisfy the constraints, in this case, not exceeding the amount of network available, and simultaneously maximize prices as opposed to a greedy algorithm which is simple but may not maximize overall profits. Using known techniques to solve this is not computationally feasible if there are a large number of bids. As the number of bids increases, the resources required to solve problem increases rapidly. Looking at all possible combinations is not a feasible solution because for every n bids, there are many possible combinations.

Other work on the FCC problem has looked at the Vickrey auction to yield a solution.

A more feasible way of solving this problem is to use a heuristic technique to limit the combinatorial growth. In this paper, a genetic algorithm is used to determine a set of good prices for the networks. One challenge with using a genetic algorithm is that a genetic algorithm is not guaranteed to give the best answer every time. A genetic algorithm is used to find good answers, not necessarily optimal answers, and to find those answers quickly.

A genetic algorithm is modeled after natural evolution and uses the operations of selection, crossover and mutation. The components of a genetic algorithm will be illustrated in the context of the FCC problem (for a more general discussion, the reader is referred to [3]). The prices for each zone are represented by binary chromosomes as shown in Figure 2.

$\langle 00100011, 00110010, 00001110, 01000001 \rangle$

Figure 2: This figure represents a chromosome. Each element represents a potential price for the corresponding network zone. The entire chromosome then corresponds to one bid over the entire network. For example, the third element of this vector represents the value 14 which corresponds to a maximum bid for zone 3 of \$14.

With each generation, chromosomes are selected based on fitness scores. Each chromosome represents a set of prices. The fitness score for each particular chromosome represents the profit for that set of prices. The fitness scores correspond to the probability of being selected and thus chromosomes with a higher fitness are more desirable. The fitness scores are determined by evaluating the chromosome and comparing the results to the rest of the chromosomes in that generation.

Selection occurs as many times as there are chromosomes to ensure the same number of chromosomes exist from generation to generation. Selection chooses two chromosomes for each chromosome in the generation. Single point crossover takes two selected chromosomes, splits them at a random index, and then recombines them to give a new child chromosome. For example, if the two chromosomes selected are $\langle 10100101 \rangle$, representing a price of \$165, and \langle

$00100111 \rangle$, representing a price of \$39, and the pivot point is chosen as 3, the new chromosome would be the first 5 bits of the first chromosome and the last 3 bits of the second chromosome: $\langle 10100111 \rangle$, representing a price of \$167. Single point mutation locates a random bit in the child chromosome and flips the bit. For example if the chromosome is $\langle 10010100 \rangle$, representing a price of \$148 and the bit chosen is 4, then the new chromosome would be $\langle 10011100 \rangle$, representing a price of \$156.

This evolutionary approach is a heuristic that can effectively compute a set of prices that will lead to a high profit for the FCC. Each chromosome is a set of prices and is represented in binary for crossover and mutation. The fitness score is normalized profit for the particular chromosome.

3. GENETIC ALGORITHM APPROACH

The approach for solving the FCC problem outlined in Section 2 uses a genetic algorithm to determine a good set of prices that will maximize profits. The initial prices are determined by using a random number generator to generate a random integer from one to one hundred for each network. The program uses these bids to calculate the number of networks and the number of bids. Each bid offers a price for a particular zone and the amount of units it is requesting for that zone. If a bidder does not request a particular zone, then the bidder offers a bid of zero for that zone. Generations with a population size of thirty chromosomes were examined.

3.1 Fitness

Each chromosome in this problem is an array of integer values corresponding to a set of prices for each zone. For each chromosome, the program first calculates the fitness value. Computation of a chromosome's fitness involves several steps. In the initial step the program examines whether the zone price is feasible (see Figure 3). Next, if a bid is feasible, the score of that chromosome is set to be the profit from applying that set of prices to the bids. If the bid is not feasible, each bid for each zone is examined and the minimum difference between the price for the chromosome and the price for the bid is determined. The price is then increased by that minimum amount plus one to eliminate at least one bid. This process is repeated as necessary until the prices are feasible. If there is a tie for the minimum difference in the zones, then the zone for which the price is increased is changed with a probability of 0.40. Finally, after scores have been generated for each of the chromosomes, a total score is calculated and each individual chromosome's score is divided by the total score to create the fitness value. The fitness value in this problem is the profit generated by the current chromosome divided the sum of the profits generated by all the chromosomes. These fitness values determine which chromosomes will be selected; a higher fitness value yields a higher probability of being selected. These values represent the probability distribution of a chromosome being selected during selection. If a bid cannot afford the prices in a chromosome, then that chromosome's fitness is set to zero.

3.2 Selection

The process of selection chooses two chromosomes to be the parents of a chromosome for the next generation. These

```

for each zone
  for each bid
    if the bid is non-zero and the amount bid is larger than the current price for that zone and there are no known conflicts
      add the number of units to the total
    else if the bid is non-zero
      then add conflict
  if the total amount of units requested is larger than the available units
    keep going
  reset total to 0
if we need to continue the feasify function, this loop is going to find the smallest amount we need to adjust by
for each zone
  for each bid
    if the bid is non-zero and the bid is larger than the current price for that zone and the difference between those two
    prices is smaller than the adjusted value and greater than zero and there are no conflicts
      if there is a tie between lowest adjust value, then randomly change the zone to be the current zone based on
      the probabilistic rate of MIN_RATE
      create a random number
      if the random number is less than the minimum rate
        set the adjust value to be the current bid in the current zone - the current chromosome for the
        current zone + 1
    else
      set the adjust value to be the current bid in the current zone - the current chromosome for the
      current zone + 1
  save the old price in the chromosome
  set the new price to be the old price + the adjust value
  check total amount of units requested for each zone to see if the bids need to run the feasify function again

```

Figure 3: pseudo code for the Feasify function. The Feasify function ensures that for all the bids that can afford a given price, the total amount of network resources requested does not exceed the total amount of available resources.

are chosen based on the probability distribution of the fitness values. Chromosomes with higher fitness have a higher chance of being chosen. This is similar to the survival of the fittest concept in nature in that good sets of prices have a better chance of surviving to the next generation than weaker sets of prices. The number of chromosomes needs to remain the same from generation to generation. In this case, 30 chromosomes were used. Thus selection is done 30 times to ensure the number of chromosomes remains the same from generation to generation.

3.3 Crossover

Once two chromosomes have been chosen to act as parents, crossover is applied with a probability of 0.30 (relatively standard rate) to each chromosome. If the crossover operation occurs, then a random number is cast to determine what position in this chromosome will act as a pivot. The first part of the first chromosome is combined with the second part of the second chromosome to create the new chromosome that will be in the next generation. If crossover does not occur, then the parent chromosome with the higher fitness value is chosen to survive to the next generation.

3.4 Mutation

Next, mutation is applied with a probability of 0.01 (relatively standard rate) to each chromosome. If mutation occurs, then one bit is randomly selected and changed from a one to a zero or vice versa. If mutation does not occur, then the chromosome survives to the next generation as it exists

already. This process is continued for each chromosome in the generation until the new generation of chromosomes is complete.

3.5 The Next Generation

The process of creating a new generation will continue for at least thirty generations. After that time, the standard deviation between the prices of the chromosomes is tested for each available network zone. If the standard deviation is less than ten units (an intuitively chosen value), then the program will terminate and the set of prices that will be used will be the chromosome that has the highest score which corresponds to the highest amount paid. After the first thirty generations, the standard deviation is recalculated after every generation. To ensure the program terminates within a reasonable time period, the program will automatically quit after one hundred generations.

4. RESULTS

The problem being solved here is the sale of network zones by the FCC. Each bid requests a certain number of units for each zone and specify a maximum amount that they are willing to pay for that zone. This problem is solved using a genetic algorithm. Genetic algorithms are flexible in that modifications can be made to the operations to obtain different results. Such modifications are explained below.

Results are analyzed by running the experiment thirty times for the same set of bids and then measuring the average

amount paid and the average number of generations. Initially, the data set consists of four bids and four zones with five different sets of bids. The bids are determined to examine five different cases: all the bids are winners, one bid is a winner, two bids are winners, three bids are winners, and a completely random set of bids. The first four cases are determined in way such that the maximum possible amount paid would be easy to compute by hand. The genetic algorithm as explained in Section 3 does not produce convincing results (see Table 1). A modification is needed on the genetic algorithm to improve the results.

Table 1: Profit is the average profit \pm the standard deviation for 30 test cases. Generations is the average number of generations computed until the standard deviation of bids for each chromosome is less than some arbitrarily chosen value for 30 test cases \pm the standard deviation for 30 test cases. These cases were run for 4 bids and 4 zones. 4 winners corresponds to all 4 bids being accepted, 1 winner corresponds to only 1 bid being accepted, etc. Random is a series of 4 bids where the bid and the units of the zone requested were initially randomly chosen by the computer but remained the same thereafter for all 30 trials. The maximum profit for each case was computed manually: 4 winners = \$170.0, 1 winner = \$90.0, 2 winners = \$120.0, and 3 winners = \$175.0. This definition of winners applies to Tables 1, 2, 3, and 4.

	Profit	Generations
1 winner	45.93 \pm 16.25	32.03 \pm 4.79
2 winners	54.83 \pm 9.02	30.40 \pm 0.80
3 winners	101.43 \pm 10.94	30.77 \pm 1.56
4 winners	93.00 \pm 17.33	34.23 \pm 13.66
random	111.30 \pm 15.15	30.97 \pm 1.68

4.1 Adjusted Mutation Rate

The first modification increases the mutation rate from one percent to twenty percent and allows the rate to diminish from generation to generation but never going below 0.01. As the number of generations increase, the prices begin to converge. Thus by having a high mutation rate, it will take longer for the values to converge. Therefore, with each subsequent generation, the mutation rate is decreased by one percent until it is down to one percent at which point it will remain at one percent. This improves the profits slightly but the results are still not convincing (see Table 2).

Table 2: Mutation rate began at 0.20 and decreased by 0.10 until the mutation rate was 0.10 where it stayed for the duration of the trial.

	Profit	Generations
1 winner	58.13 \pm 10.98	30.30 \pm 1.64
2 winners	63.80 \pm 5.35	30.47 \pm 1.74
3 winners	119.87 \pm 16.49	30.37 \pm 0.89
4 winners	114.13 \pm 22.77	35.57 \pm 17.93
random	123.20 \pm 18.88	30.33 \pm 0.96

4.2 Tournament Selection

Next, selection is examined to see if any modifications can be made to select chromosomes with higher fitness. To improve upon this, a process called tournament selection is used. Instead of choosing two chromosomes for selection, four chromosomes are chosen and then of those four, the top two are chosen for selection. For this process, the resulting profits do not increase significantly from their initial values and therefore are still not convincing (see Table 3).

Table 3: 4 chromosomes are chosen during selection and of those 4, the 2 with the highest fitness are kept for crossover and mutation.

	Profit	Generations
1 winner	42.77 \pm 15.73	33.47 \pm 12.86
2 winners	58.90 \pm 7.75	30.40 \pm 0.77
3 winners	111.63 \pm 15.44	30.27 \pm 0.74
4 winners	109.67 \pm 19.93	30.73 \pm 0.87
random	115.90 \pm 16.71	30.57 \pm 1.19

4.3 Annealing

Another problem is that the genetic algorithm is stuck on a local maximum. Simulated annealing is implemented to improve the results by moving off the local maximums. For each chromosome, a random number δ from one to ten is cast. δ will represent by how much the chromosome will change. Next, another random number is cast to determine if the price will increase or decrease. The price is then increased or decreased by δ . If by reducing the price by δ results in a negative price, the price is then reset to zero. Next, each network was tested to determine which price, if changed, will give the highest amount paid. If no price change will create a higher amount paid than the initial amount paid, then the chromosome remains unchanged. This is done by calling the same function used to determine fitness for each individual chromosome. However, this increases the complexity of the algorithm and makes runtime noticeably slower. Also, the results do not justify this increase in complexity as the results were no better than previous methods. (see Table 4).

Table 4: The amount annealing changes the chromosome is a random number between 1 and 10. The decision to add or subtract that amount from the current gene is chosen randomly. If the new value yields a higher profit, the chromosome is set to be this new value.

	Profit	Generations
1 winner	41.67 \pm 31.27	31.27 \pm 2.59
2 winners	56.83 \pm 7.83	31.37 \pm 2.89
3 winners	106.70 \pm 14.40	32.20 \pm 4.96
4 winners	95.73 \pm 18.64	36.00 \pm 12.05
random	109.07 \pm 19.04	31.37 \pm 4.09

4.4 Minimum Bids

One recurring problem is that some winning prices have values of zero for certain networks even though there are non-zero bids for those networks. The next idea is to specify a minimum value for each network. To ensure that this would not be an unfair process, the minimum value for each network is set to be the smallest non-zero bid from the set of

Table 5: Z is minimum bids per zone, M is change mutation rate, A is annealing, and T is tournament selection. These values represent the average of the 5 cases: 4 winners, 1 winner, 2 winners, 3 winners, and random. For each set of bids, the top row is the average profit and the bottom row is the average number of generations. The shaded regions represent the best results obtained from the different cases.

	Z	MZ	TZ	AZ
(4, 4)	99.4 ± 20.19	126.4 ± 8.81	123.39 ± 11.69	118.79 ± 15.62
	30.39 ± 1.48	30.26 ± 1.11	33.33 ± 6.56	30.34 ± 0.94
(4, 16)	731.45 ± 95.62	761.17 ± 39.60	775.52 ± 72.46	975.71 ± 47.06
	65.67 ± 21.83	62.39 ± 14.74	66.29 ± 17.27	60.07 ± 16.36
(8, 4)	123.47 ± 23.05	232.74 ± 35.68	172.49 ± 60.74	250.76 ± 19.66
	41.97 ± 12.59	61.57 ± 14.91	50.47 ± 23.66	53.35 ± 17.82
(8, 16)	555.05 ± 120.27	680.67 ± 92.83	619.69 ± 89.42	768.13 ± 84.82
	55.95 ± 16.52	65.70 ± 10.92	58.29 ± 10.26	81.91 ± 7.17
(16, 16)	848.40 ± 101.47	648.54 ± 48.35	1020.13 ± 83.65	1030.75 ± 97.33
	77.80 ± 14.92	44.52 ± 13.91	77.29 ± 10.22	92.09 ± 13.35
(64, 32)	4733.49 ± 207.76	4025.59 ± 201.99	5665.07 ± 240.14	4708.44 ± 195.72
	100.00 ± 0.00	68.06 ± 24.98	99.51 ± 2.67	100.00 ± 0.00
	MTZ	MAZ	TAZ	MTAZ
(4, 4)	91.59 ± 14.51	127.98 ± 7.66	128.31 ± 7.19	114.02 ± 2.90
	30.32 ± 0.38	30.02 ± 0.11	33.77 ± 5.83	30.56 ± 0.69
(4, 16)	792.28 ± 44.41	765.55 ± 19.50	776.05 ± 86.27	805.25 ± 22.38
	55.89 ± 20.01	60.35 ± 10.80	71.01 ± 71.01	57.81 ± 15.96
(8, 4)	230.30 ± 35.10	261.23 ± 14.32	258.99 ± 18.70	276.21 ± 11.56
	62.45 ± 22.65	60.72 ± 15.84	56.95 ± 19.13	67.84 ± 10.94
(8, 16)	731.13 ± 110.17	778.67 ± 27.50	852.50 ± 81.52	852.28 ± 36.69
	66.07 ± 10.60	83.09 ± 6.97	83.40 ± 8.76	82.79 ± 7.28
(16, 16)	723.63 ± 55.71	835.58 ± 63.10	1199.67 ± 126.75	894.79 ± 72.52
	43.83 ± 11.49	61.91 ± 6.68	92.19 ± 11.45	61.46 ± 3.93
(64, 32)	4712.63 ± 233.13	4006.97 ± 207.11	5636.67 ± 232.88	4706.74 ± 238.20
	44.44 ± 17.57	64.48 ± 24.94	99.71 ± 1.61	48.69 ± 20.87

bids for that network. Initially, the minimum difference of several of the networks is zero because several of the zone prices have been set to be the smallest non-zero amount bid for that zone. The feasibility function will select different networks each time in order to ensure fairness. At the end of each generation, a function is called that looks through the chromosomes and readjusts the prices to make sure they are not less than the minimum bid values for the zones. This process increases the amount paid for 4 winners, 2 winners, and 3 winners but not for 1 winner and random (see Table 6).

Table 6: The minimum price charged for each zone is the smallest amount bid for each zone.

	Profit	Generations
1 winner	39.53 ± 14.39	30.50 ± 1.53
2 winners	60.33 ± 8.71	30.30 ± 0.65
3 winners	155.50 ± 2.58	30.13 ± 0.35
4 winners	95.73 ± 18.64	36.00 ± 12.05
random	71.63 ± 75.28	30.93 ± 4.56

4.5 Combinations of Methods

Next several combinations of these methods are examined to see if a certain combination of these methods would give better results. For this case with four bids and four zones, the combination of minimum bids per network, adjusting the mutation rate, and annealing produces the best results of the experiments ran. Because the number of networks and

the number of bids is determined by the program as they are initially read in, scaling the problem up to include more bids and more networks did not require any additional programming. As the number of networks and zones increased, the combination of all four techniques outlined above: adjust the mutation rate, tournament selection, annealing, and minimum bids per zone generally produce the best results. The one method that whenever applied tends to produce significant results was minimum bids per zone. For each case where that method was applied, the average amount paid tends to be higher than other cases where it was not applied (see Table 5).

5. CONCLUSION

Using a genetic algorithm to determine the prices, significant profits can be attained in a feasible amount of time. A genetic algorithm is not guaranteed to consistently get the best answer. At best, one hopes to arrive at good answers. By using the methods outlined above, not only are the answers significant, but those answers can be arrived at in a reasonable amount of time.

With the exception of the 4 bids and 4 zones case, all of the best winners involved using annealing. Of those winners, only one did not use tournament selection. Using an adjusted mutation rate worked well for small cases but not for large cases. In the cases where tournament selection alone is comparable to tournament selection and annealing combined, tournament selection alone should be used because

annealing dramatically increases run time. The 64 bid case that used annealing took around one hour to run 30 test cases. All other cases took less than 5 minutes to run 30 test cases. It can be concluded from this experiment that a genetic algorithm can provide good answers to this problem in a reasonable amount of time.

Currently, there are no restrictions on the bidders. A next step for this project would be for a bidder to specify a set amount of money that they are willing to spend and to specify which zones they would like to bid on. The program would be allowed flexibility in how to divide the total money for each zone to ensure that the bidder still wins but doing it in a way that will maximize the profit.

The largest case explored was 64 bids and 32 zones. Running 30 test cases with annealing resulted in a run time of about an hour which equates to about 2 minutes per case. A next step for this project would be to test cases with more bids and more zones to see how that affects run time.

6. REFERENCES

- [1] M. Bykowsky, R. Cull, and J. Ledyard. Mutually Destructive bidding: The FCC Auction Design Problem *Social Science Working Paper*, California Institute of Technology 916 (1995)
- [2] H. Deitel and P. Deitel. *C++ how to program 3rd ed.* Prentice-Hall, Inc., Upper Saddle River, NJ, 2001.
- [3] J. L. Jones and G. J. Koehler. Combinatorial auctions using rule-based bids. *Decision Support Systems*, 34(1):59–74, 2002.
- [4] M. Mitchell. *An introduction to genetic algorithms.* The MIT Press, Cambridge, MA, 1999.
- [5] S. Russell and P. Norvig. *Artificial intelligence: a modern approach. 2nd ed.* Pearson Education, Inc., Upper Saddle River, NJ, 2003.