CSE 462 | ALGORITHM ENGINEERING SESSIONAL

The Subset Sum Problem

Group 6
Farhanaz Farheen (1505013)
Sifat Ishmam Parisa(1505016)
Salman Shamil (1505021)
Kazi Sajeed Mehrab (1505025)
Syeda Nahida Akter (1505027)

Bangladesh University of Engineering and Technology

INTRODUCTION

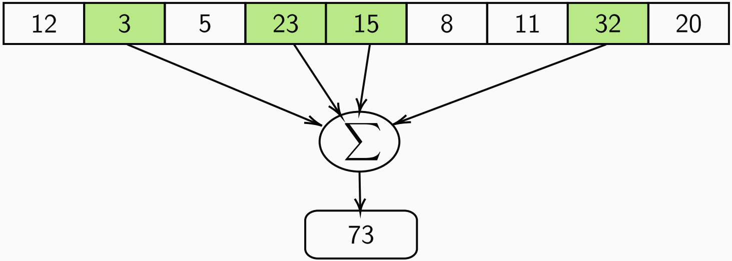
The Subset Sum Problem

Let's assume we're given a set of integers. Can we find a subset that sums up to a given target integer?

Input Set:

12	3	5	23	15	8	11	32	20

Selection of Subset:



The Subset Sum Problem

Formal definition of the problem: (Decision Version)
Given a Multiset of integers, $S = \{x_1, x_2, x_3, \cdots, x_n\}$ and a target sum W, does there exist a subset $S' \subseteq S$ such that $\sum_{x \in S'} x = W$?

The Subset Sum Problem

Formal definition of the problem: (Optimization Version)
Given a Multiset of integers, $S = \{x_1, x_2, x_3, \cdots, x_n\}$ and a target sum W, find a subset $S' \subseteq S$ so as to

maximize $Z = \sum_{x \in S'} x$,

subject to $\sum_{x \in S'} x \leq W$.

The Algorithms We Discussed

- **Exact Algorithms** - Brute-Force, Backtracking, Branch and Bound, Dynamic Programming
- **Approximation Algorithms** - A PTAS for Subset Sum Problem, An FPTAS for Subset Sum Problem
- **Heuristics and Metaheuristics** - A competitive local search based heuristic, Hill Climbing, Simulated Annealing, Genetic Algorithm

IMPLEMENTATION OF THE ALGORITHMS

The Algorithms We Implemented

- An FPTAS for Subset Sum Problem [1]
- Simulated Annealing
- Genetic Algorithm [2]

We also implemented the Dynamic Programming Algorithm for determining the optimal solution.

The Data used

- For running the algorithms, we used three categories of data (of different sizes). And each such type of set was used 10 times. This is shown in the form of a table below.

Set Size	Number of Sets
10	10
100	10
1000	10

Table: The Data

We used Python Programming language for running the algorithms.

The metrics we evaluated

For comparing the algorithms, we evaluated the following metrics.

- Average elapsed time in seconds.
- The average accuracy.

For each input sent to each algorithm, the accuracy was considered to be the ratio of obtained solution and optimal solution, i.e. Accuracy $= \frac{Solution_{obtained}}{Solution_{optimal}}$

FPTAS for Subset Sum Optimization Problem

- Let L be a list of integers. We will use the concept of 'trimming' a list.
- The idea is that if two values in L are close to each other, then for the purpose of finding an approximate solution there is no reason to maintain both of them explicitly.
- δ is a trimming parameter such that $0 < \delta < 1$.
- TRIM(L,δ) reduces L to L' such that for any $y \in L$, there exists some $z \in L'$ such that $\frac{y}{1+\delta} \leq z \leq y$.
- This procedure takes as input a set $S = \{x_1, x_2, ..., x_n\}$ of n integers, a target integer W, and an approximation parameter ϵ where $0 < \epsilon < 1$. Here, $S + x = \{x_1 + x, x_2 + x, ..., x_n + x\}$ and MERGE(L, L') returns union of L and L'.

FPTAS for Subset Sum Optimization Problem

Algorithm 1 TRIM(L,δ)

Input: A Sorted List $L =< y_1, y_2, ..., y_m >$, Trimming Parameter ($\delta$).

Output: A Trimmed List L'.

$L' =< y_1 >$

$last = y_1$

for j=$2...m$ **do**

  if $y_j > last \cdot (1 + \delta)$ **then**

    append y_j to L'

    last $= y_j$

end

return L'

FPTAS for Subset Sum Optimization Problem

Algorithm 2 APPROX-SUBSET-SUM(S,W,ϵ)

Input: A Set ($S = \{x_1, x_2, ..., x_n\}$), Target Sum (W), Approximation parameter (ϵ).

Output: An Approximate Solution (Weight closest to but not exceeding W).

$n = |S|$

$L_0 =< 0 >$

for i=$1, 2...n$ **do**

    L_i = MERGE($L_{i-1}, L_{i-1} + x_i$)

    L_i = TRIM($L_i, \frac{\epsilon}{2n}$)

    remove from L_i any values strictly greater than W

end

return largest value in L_n

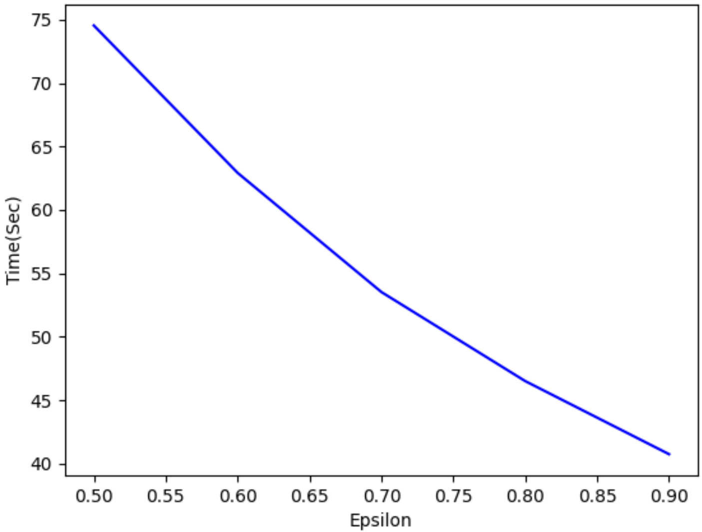


Figure: FPTAS Elapsed Time(Sec) vs. Epsilon

Simulated Annealing I

- Simulated Annealing [3] is a metaheuristic algorithm.
- It makes use of Hill Climbing while addressing the problem of local optima.

Simulated Annealing II

- Simulated Annealing keeps a probability of accepting a solution that is worse than the previous one.
- This allows the algorithm to explore more paths that might lead to the global optimum.

Simulated Annealing III

- However, if the algorithm keeps on accepting bad solutions, it might move about a lot without bringing any improvement.
- To address this, a **temperature** variable is kept. Its value is lowered with the iterations of the algorithm.
- The probability of accepting bad solutions is made proportional to the temperature.
- Therefore, as the algorithm proceeds, bad solutions are accepted less and optimization is done more.

Simulated Annealing for SSP-Optimization

Algorithm 3 Simulated-Annealing-SSP(S, W, r)

Input: Set $S = \{x_1, x_2, ..., x_n\}$, Target Sum W, integer r.
Output: An Approximate Solution (Weight closest to W).

```
S' = initial_random_subset(S)
current_residue = residue(S', W)
for i = 1...r do
    T = random_neighbor_selection(S')
    neighbor_residue = residue(T, W)
    if (neighbor_residue < current_residue) then
        S' = T
    if (residue ≥ current_residue) then
        P = calculate_probability(residue, current_residue, i)
        with probability P do S' = T
end
return (W − current_residue)
```

A few Implementation Details I

- The probability P of accepting a worse neighbor is set to e^{-X} where X is given by:

$$X = \frac{(neighbor\_residue - current\_residue)}{10^{10} * 0.8^{\frac{i}{300}}} \tag{1}$$

.

- In this case, $0.8^{\frac{i}{300}}$ is the **temperature** for this algorithm.
- Initially the value of P will be high, so the algorithm accepts worse neighbors and explores the solution space more.
- As more iterations are run, P will decrease, and bad neighbors will be explored less.

A few Implementation Details II

- While exploring bad neighbors, we allow neighbors with sum greater than the target W.
- For such a neighbor, the residue is set to be the total sum of that subset, instead of the difference with the target W.

A few Implementation Details III

- If the target W is reached before the maximum iteration r, the program breaks and returns the exact solution.

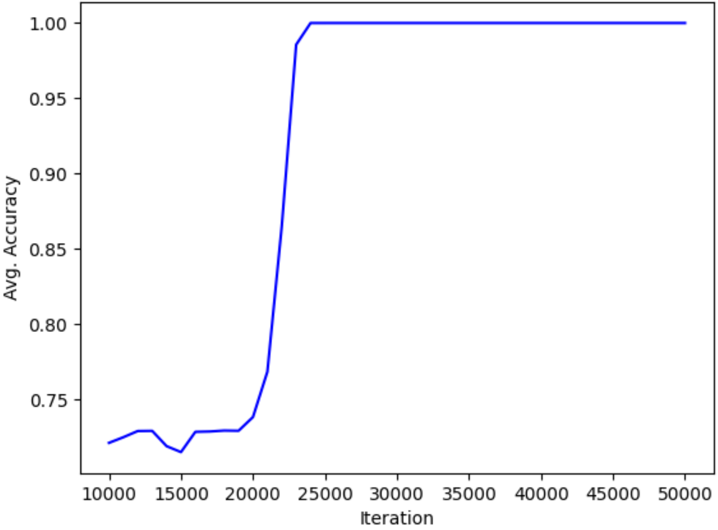


Figure: Simulated Annealing Average Accuracy vs. Iteration

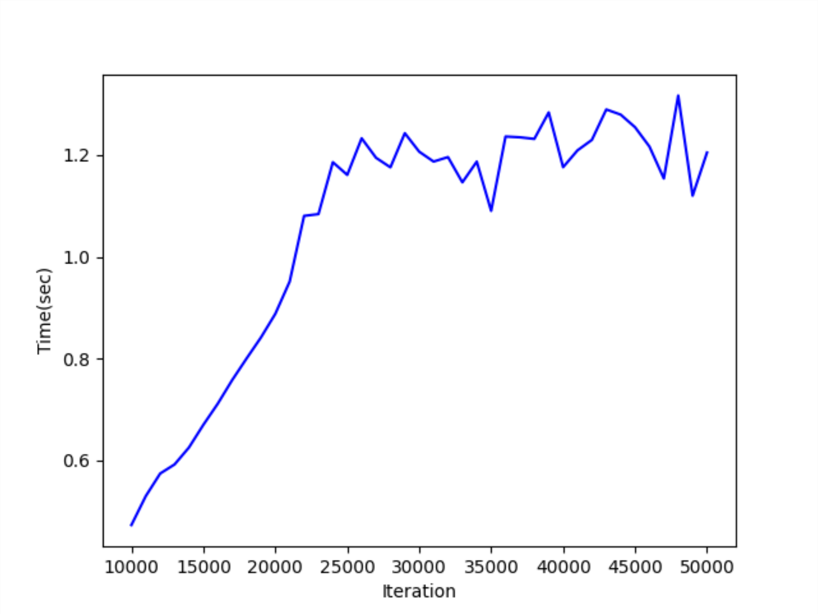


Figure: Simulated Annealing Elapsed Time(Sec) vs. Iteration

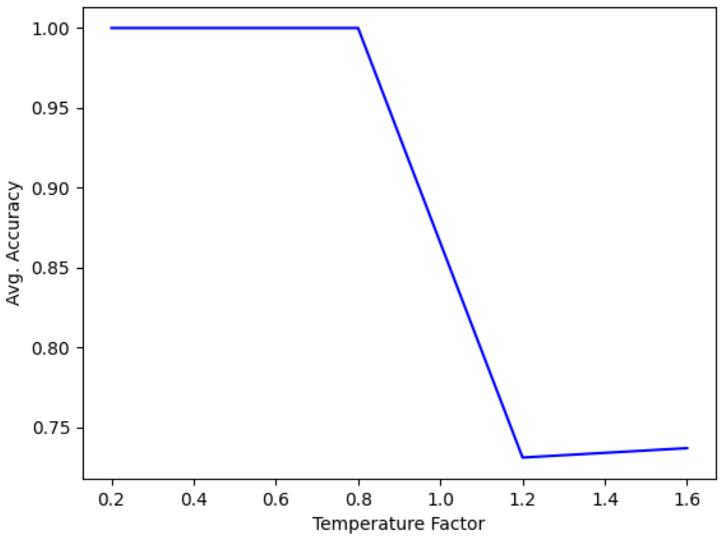
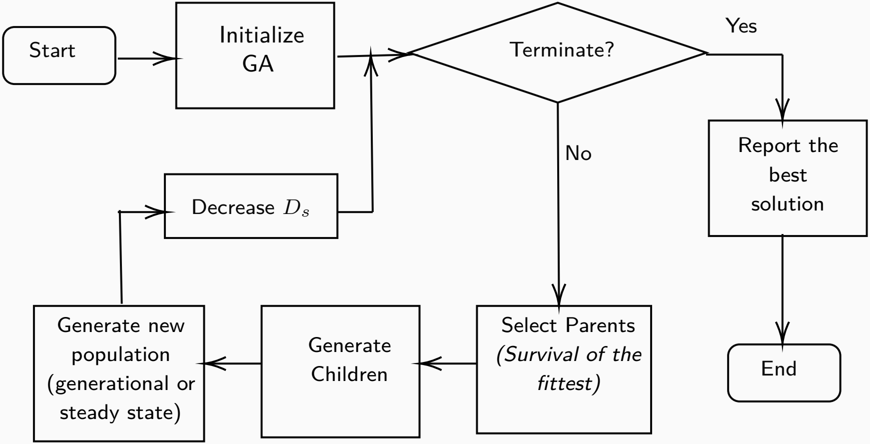


Figure: Simulated Annealing Accuracy vs. Temperature Factor

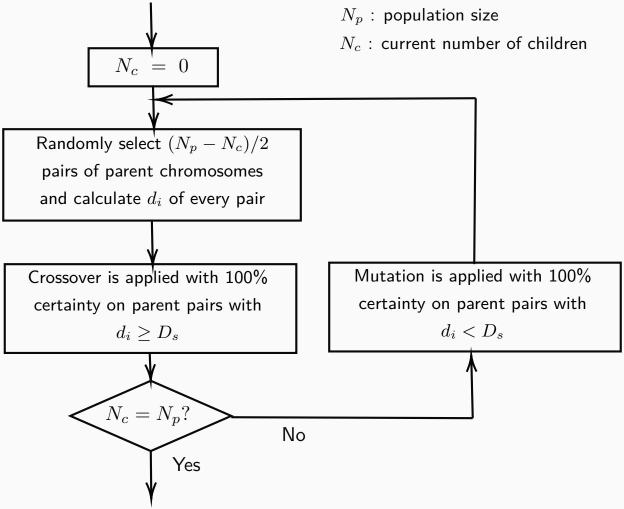
Genetic Algorithm: Key Ideas

- Global search heuristic for optimization and search problems
- Inspired by evolutionary biology concept of "Survival of the fittest"
- A population of abstract representations (genotype) of candidate solutions (individuals) evolves toward better solutions

Genetic Algorithm



Genetic Algorithm: Crossover and Mutation



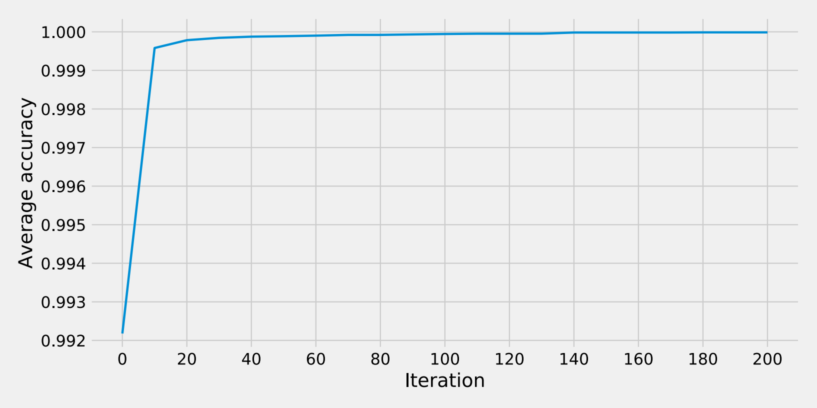


Figure: Genetic Algorithm Average Accuracy vs. Iteration

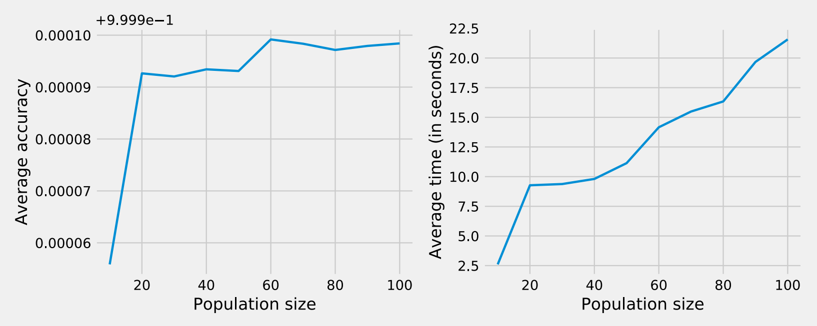


Figure: Varying the population size of Genetic Algorithm

IMPLEMENTATION AND COMPARISON

Run-time Comparison

- The average run-times (in seconds) of the algorithms were compared on the 3 categories of sets.

Set Size	FPTAS	Simulated Annealing	Genetic Algorithm
10	0.0011	0.1857	0.6841353
100	0.2604	0.0878	0.9269712
1000	41.8134	1.1680	22.178837

Table: Average Run-time Comparison

Accuracy Comparison

- The average accuracy of the algorithms were compared on the 3 categories of sets.

Set Size	FPTAS	Simulated Annealing	Genetic Algorithm
10	0.9674	1	1
100	0.9976	1	1
1000	0.9997	1	1

Table: Average Accuracy Comparison

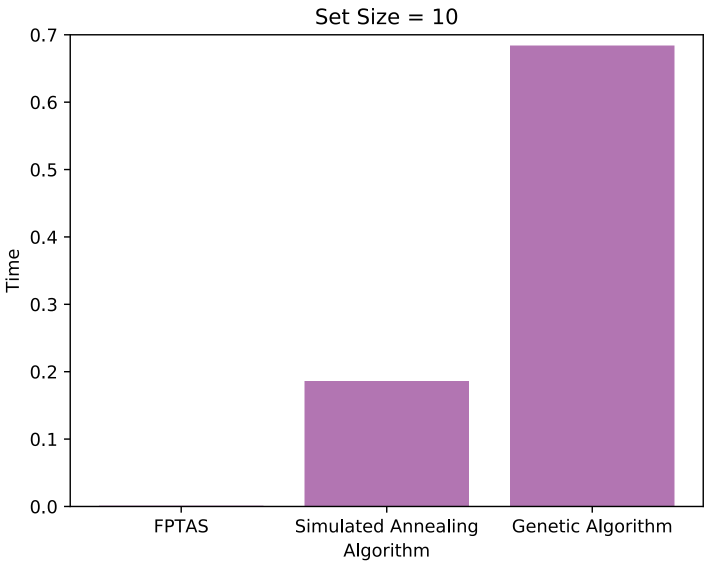


Figure: Elapsed time (sec) for three algorithms (Input Set Size = 10)

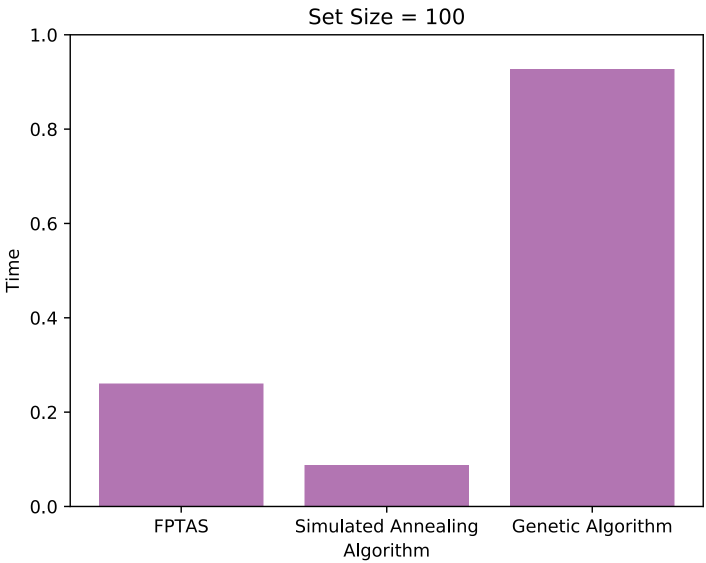


Figure: Elapsed time (sec) for three algorithms (Input Set Size = 100)

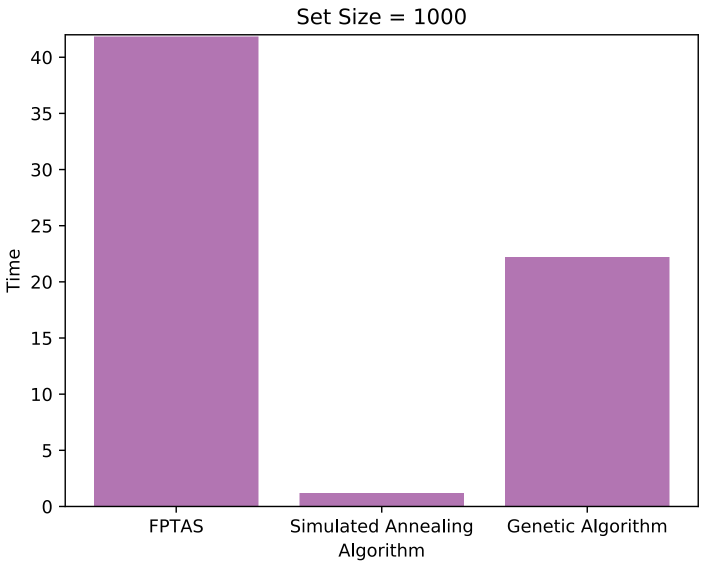


Figure: Elapsed time (sec) for three algorithms (Input Set Size = 1000)

REFERENCES

References I

- H. Kellerer, R. Mansini, U. Pferschy, and M. G. Speranza, "An efficient fully polynomial approximation scheme for the subset-sum problem," *Journal of Computer and System Sciences*, vol. 66, no. 2, pp. 349–370, 2003.
- R. L. Wang, "A genetic algorithm for subset sum problem," *Neurocomputing*, vol. 57, pp. 463–468, 2004.
- S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *science*, vol. 220, no. 4598, pp. 671–680, 1983.

THANK YOU!