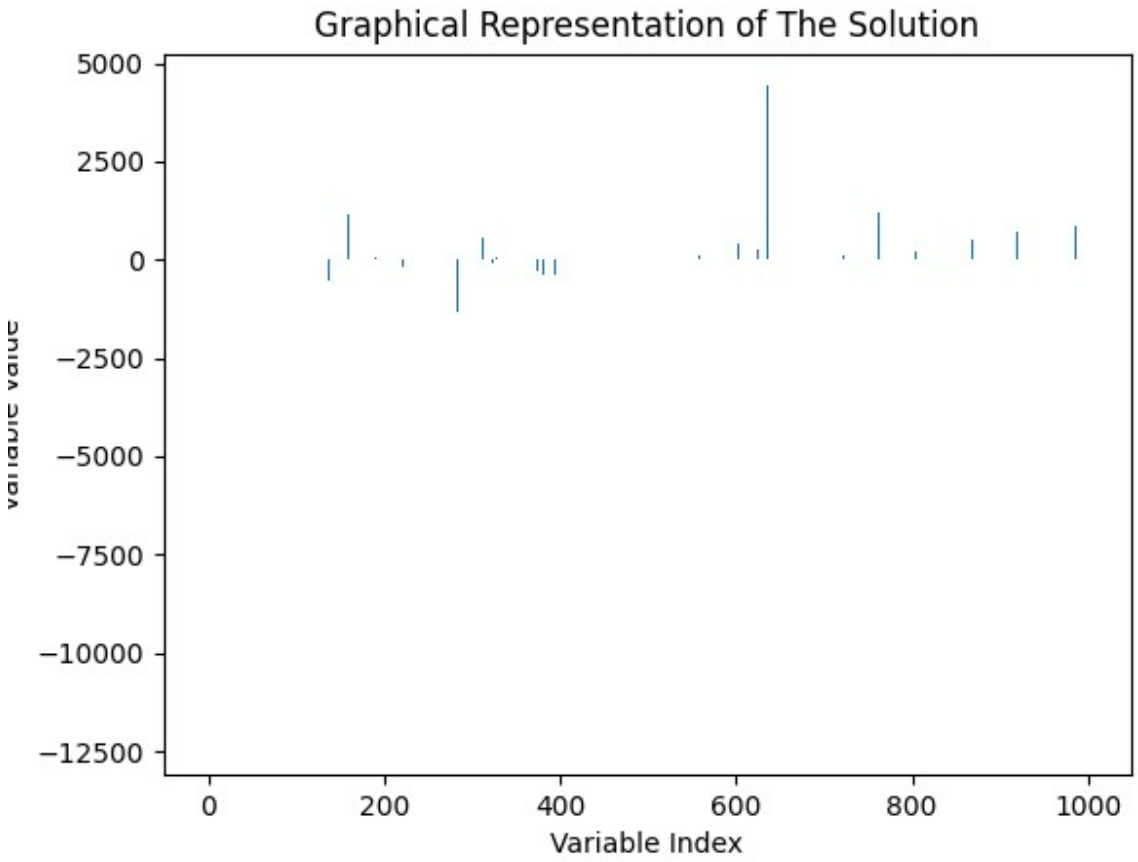
SIMULATIONS OF 10 SOLUTIONS

Solution 1

Iteration log...
Iteration: 1 Scaled dual infeas = 0.000000
Iteration: 2 Dual objective = 29.078643
Solution time without parallel: **1.0459461212158203**
Solution time with parallel: **0.002855062484741211**

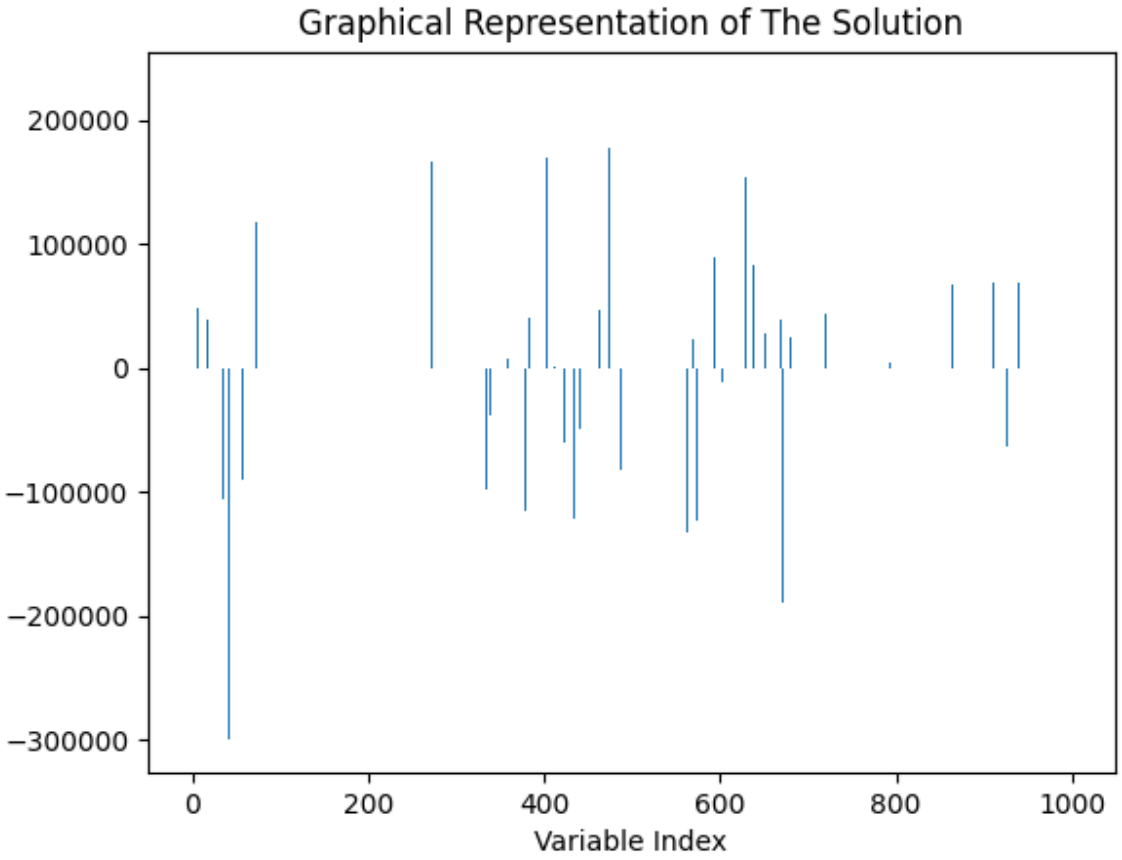
Graph of Solution 1



Solution 2:

Iteration log...
Iteration: 1 Scaled dual infeas = 0.000000
Iteration: 2 Dual objective = 184.090444
Solution time without parallel: **0.9269008636474609**
Solution time with parallel: **0.0012428760528564453**

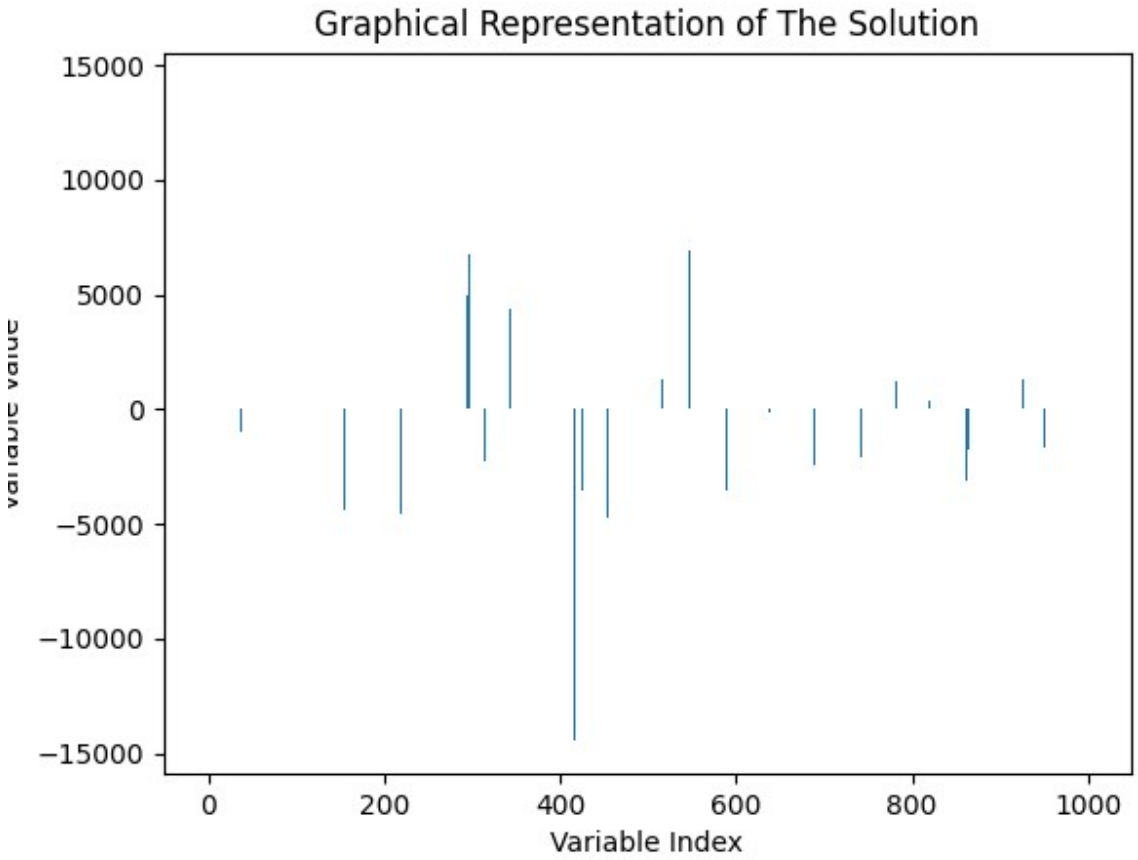
Graph of Solution 2



Solution 3

Iteration log...
Iteration: 1 Scaled dual infeas = 0.000000
Iteration: 2 Dual objective = 68.915247
Solution time without parallel: **0.8644118309020996**
Solution time with parallel: **0.0009419918060302734**

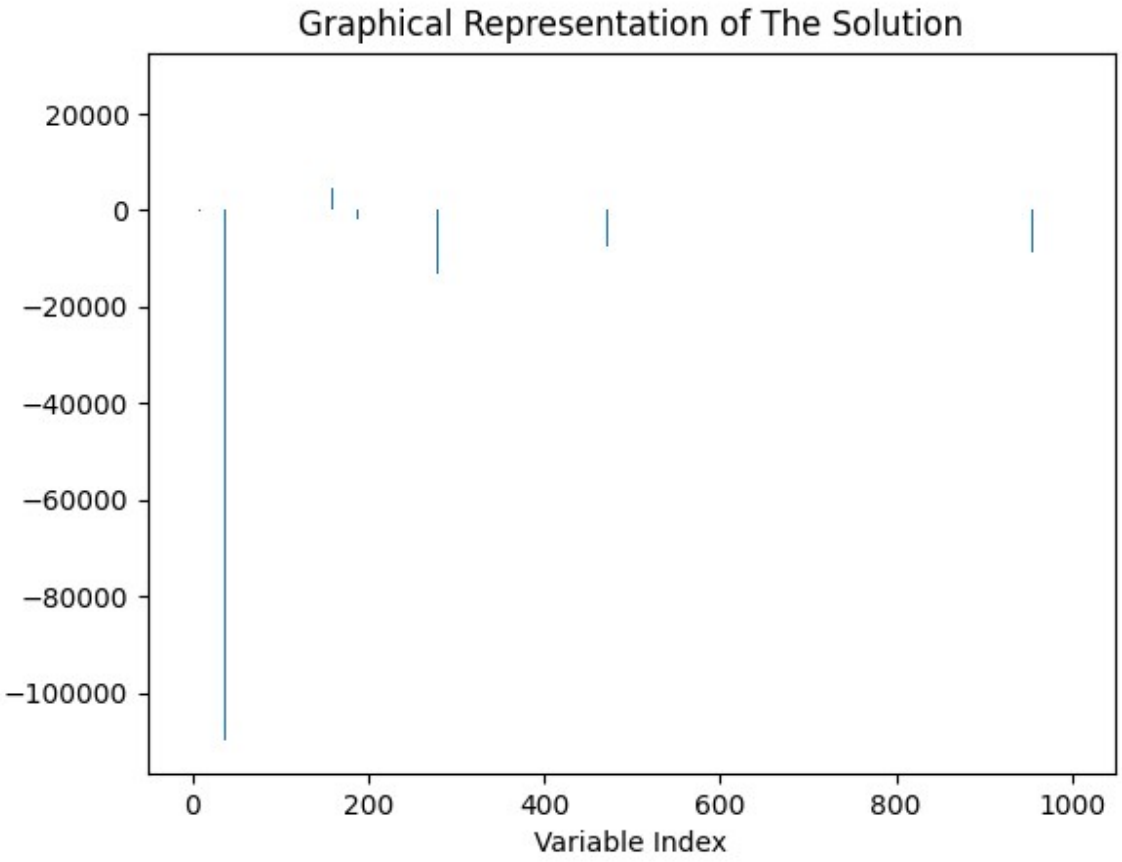
Graph of Solution 3



Solution 4

Iteration log...
Iteration: 1 Scaled dual infeas = 0.000000
Iteration: 2 Dual objective = 6.790730
Solution time without parallel: **0.8547751903533936**
Solution time with parallel: **0.0021450519561767 58**

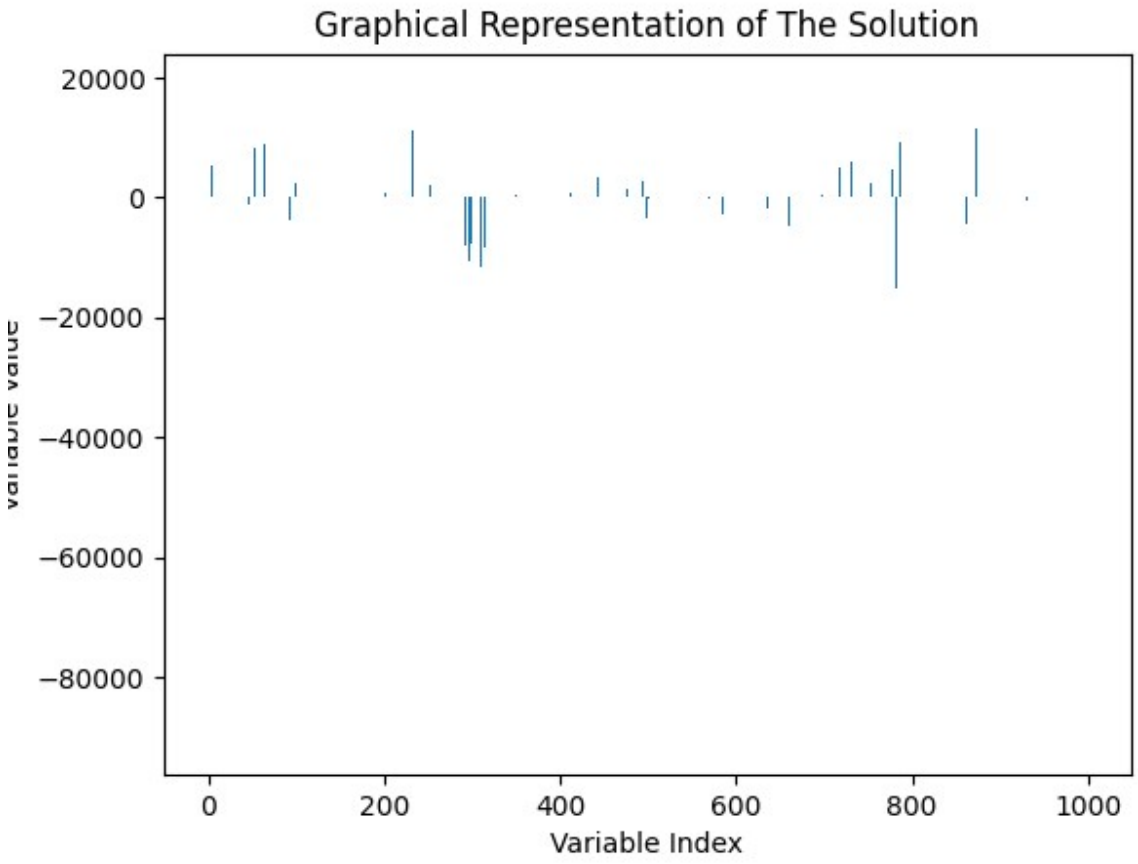
Graph of Solution 4



Solution 5

Iteration log...
Iteration: 1 Scaled dual infeas = 0.000000
Iteration: 2 Dual objective = 111.838763
Solution time without parallel: **1.0634522438049316**
Solution time with parallel: **0.0027589797973632812**

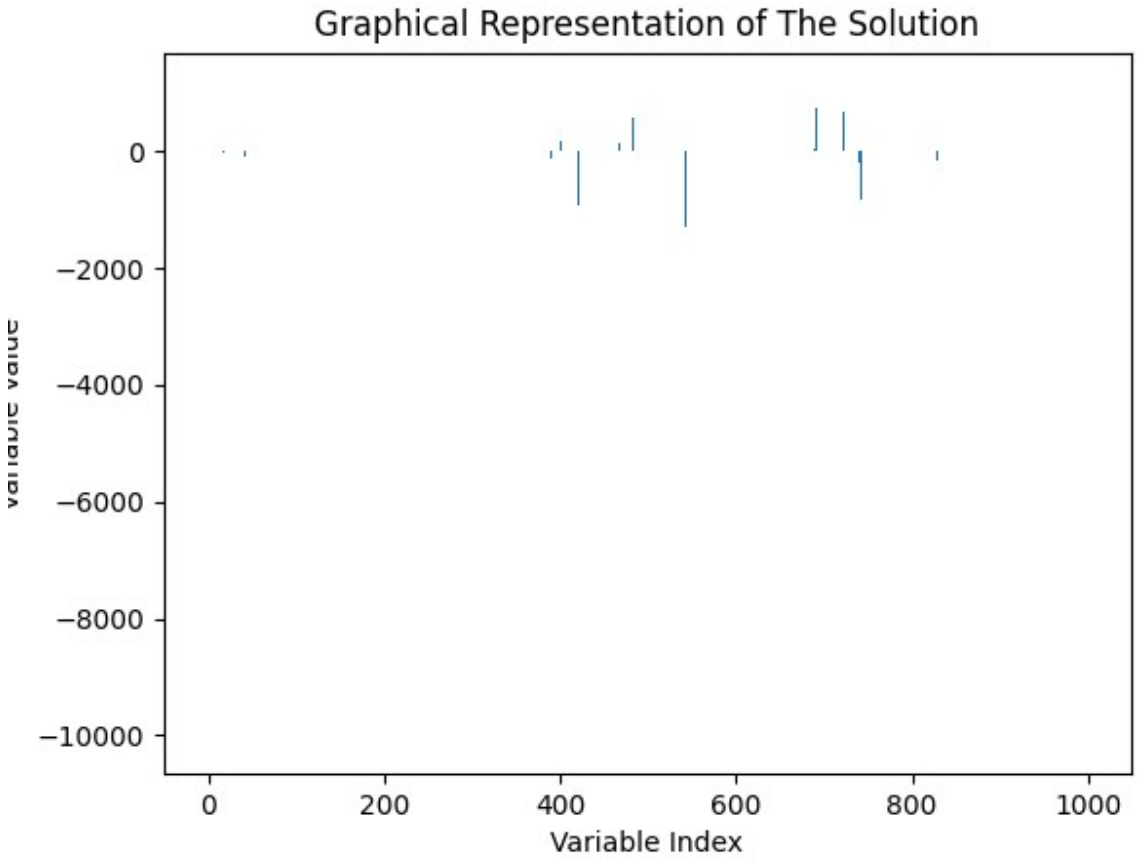
Graph of Solution 5



Solution 6

Iteration log...
Iteration: 1 Scaled dual infeas = 0.000000
Iteration: 2 Dual objective = 18.831785
Solution time without parallel: **1.1338090896606445**
Solution time with parallel: **0.0007748603820800781**

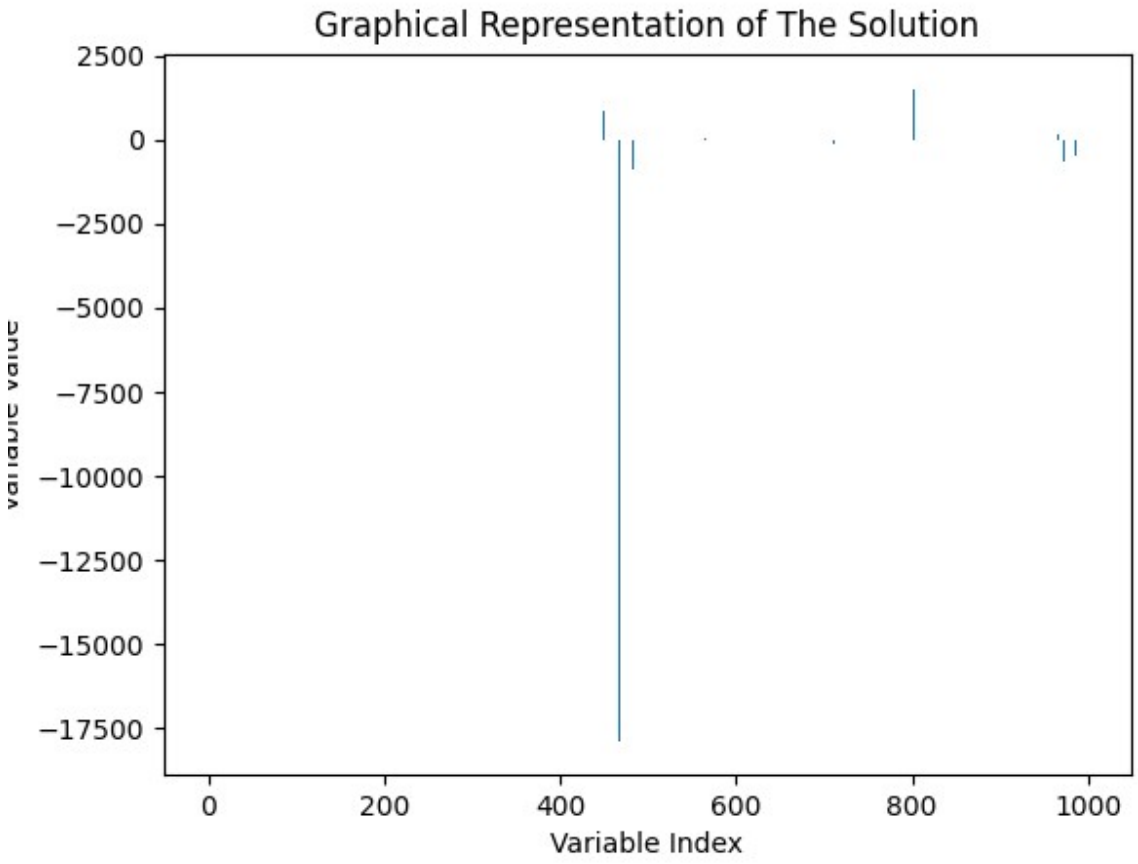
Graph of Solution 6



Solution 7

Iteration log...
Iteration: 1 Scaled dual infeas = 0.000000
Iteration: 2 Dual objective = 393.443337
Solution time without parallel: **1.1633098125457764**
Solution time with parallel: **0.0004768371582031 25**

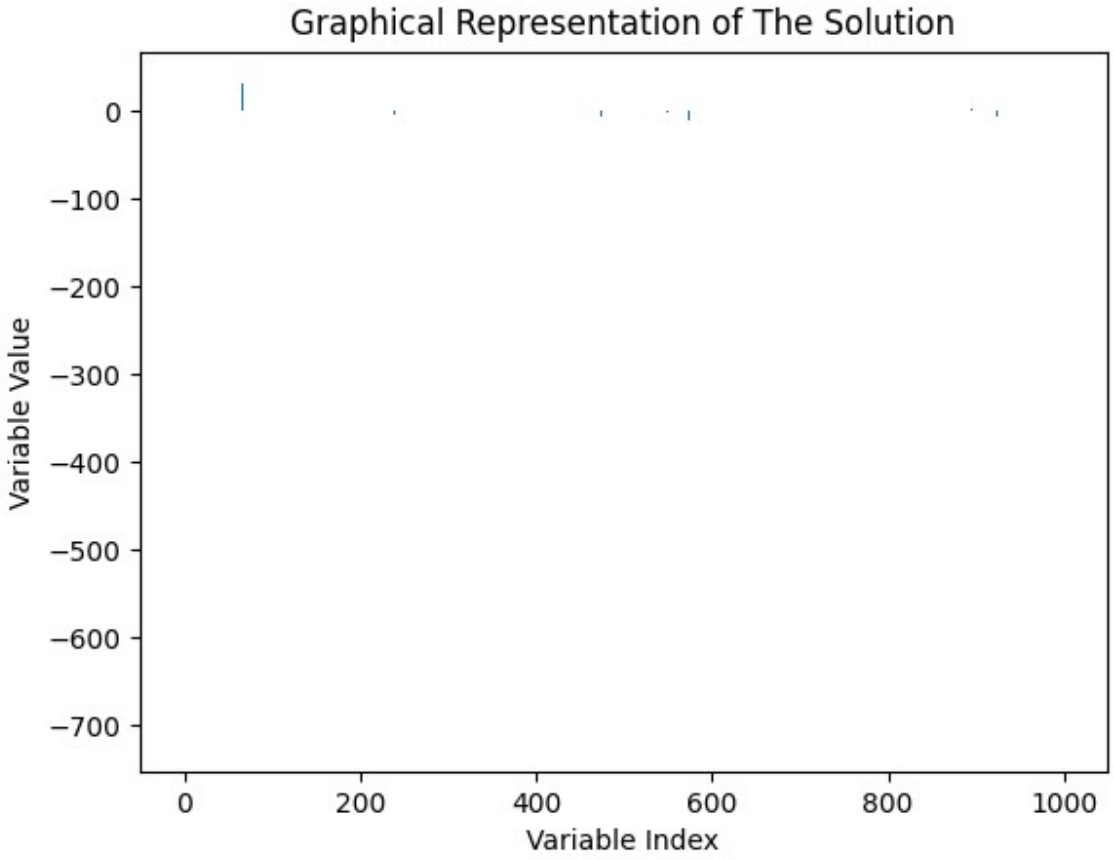
Graph of Solution 7



Solution 8

Iteration log...
Iteration: 1 Scaled dual infeas = 0.000000
Iteration: 2 Dual objective = 329.709424
Solution time without parallel: **0.8137271404266357**
Solution time with parallel: **0.00039505958557128906**

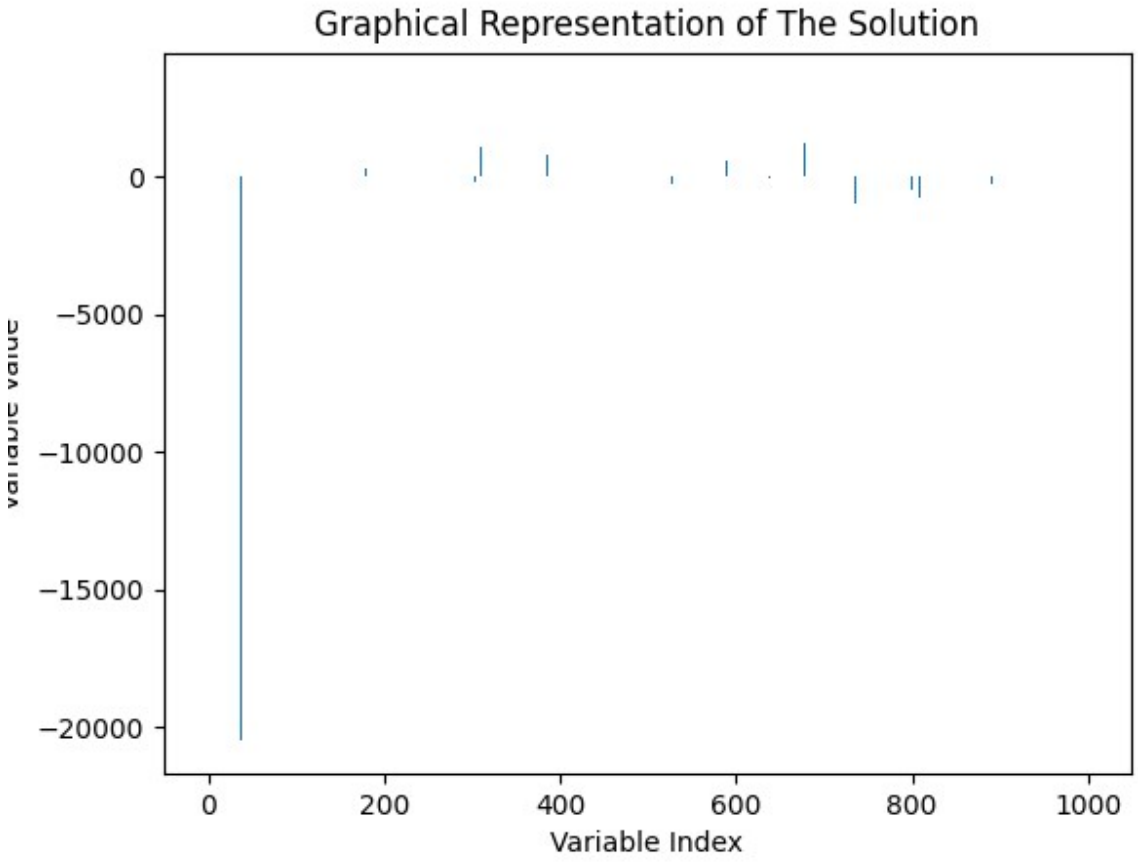
Graph of Solution 8



Solution 9

Iteration log...
Iteration: 1 Scaled dual infeas = 0.000000
Iteration: 2 Dual objective = 200.787303
Solution time without parallel: **0.8188397884368896**
Solution time with parallel: **0.0005838871002197266**

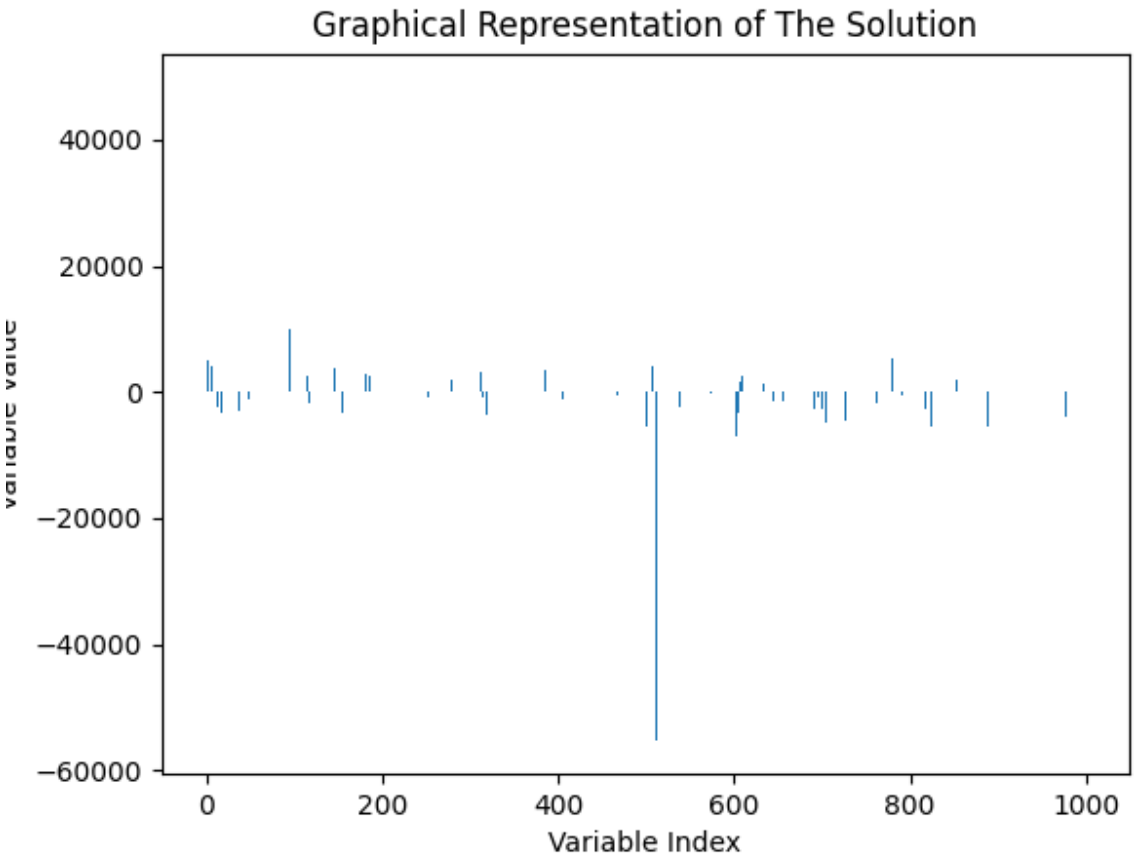
Graph of Solution 9



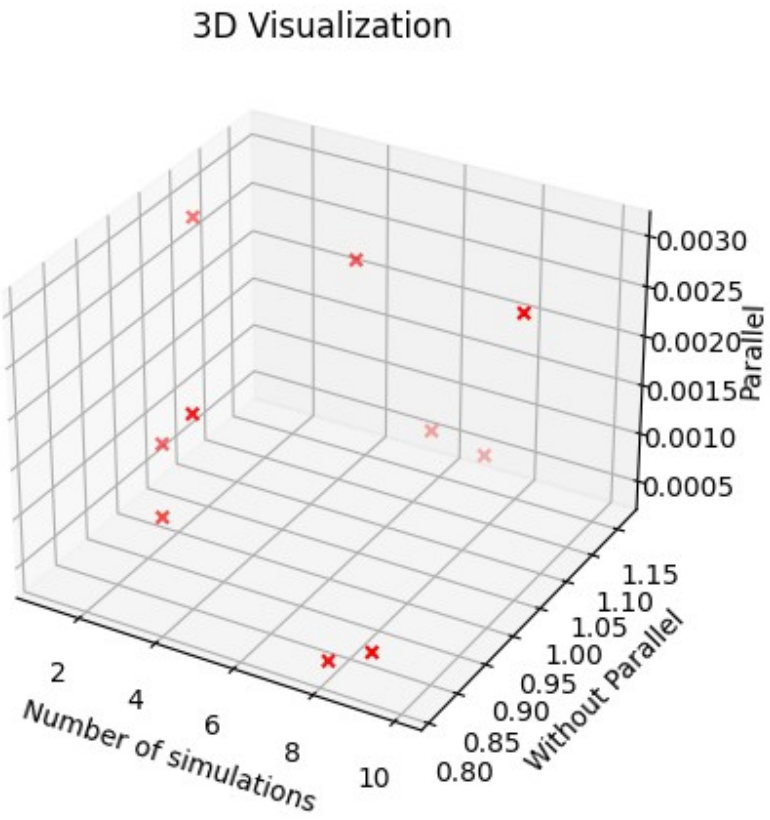
Solution 10

Iteration log...
Iteration: 1 Scaled dual infeas = 0.000000
Iteration: 2 Dual objective = 147.929004
Solution time without parallel: **1.0041818618774414**
Solution time with parallel: **0.0030488967895507812**

Graph of Solution 10



3-D Visualization of Solutions



Integer Linear Programming (ILP) is a mathematical optimization technique that involves optimizing a linear objective function subject to linear equality and inequality constraints, where the decision variables are restricted to be integers. In an integer linear programming problem, the goal is to find the values of the decision variables that maximize or minimize the objective function while satisfying all the constraints. For an ILP with thousand variables, it means the problem has thousand decision variables that are subject to integer constraints. These decision variables could represent various quantities, such as the number of units produced or the amount of resources used depending on the specific problem modeled.

Parallel computing can significantly impact the performance of solving an integer linear programming problem with a large number of variables. In general, parallel computing involves dividing a large computational problem into smaller sub-problems that can be solved simultaneously on multiple processors, thereby reducing the overall computational time required to solve the problem.When solving an integer linear programming problem with a thousand variables, the problem can be divided into smaller sub-problems,

each of which can be solved independently on different processors. This approach can lead to a significant reduction in computational time, as each processor is working on a smaller part of the problem, allowing for faster solution times.

For the above simulations, we can see that solving an integer linear programming problem with a thousand variables can be computationally expensive and time-consuming without parallelization. As the problem size and complexity increase with the number of variables. However, parallelization can still provide a significant performance boost, especially for large-scale problems.