Investigation of the Importance of Small Odd-Set Constraints in the Maximum Weight Matching Problem

Hanna Hamilton

ISyE 6999 - Deterministic Optimization

Problem Description

A matching is a subset of a graph where at most one edge can be incident on a vertex. An example is shown in Figure 1. It illustrates a maximum weighted matching. The blue edge between e and d is chosen (with a weight of 8), along with the red edge between b and c (with a weight of 7). If any more edges are chosen, it is no longer a matching because some vertices will have more than one chosen edges. This is the matching which has the largest sum of weights.

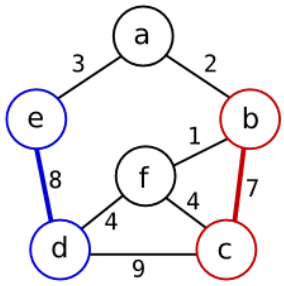

Figure 1: Matching [1]

Matchings are often used in optimization for scheduling problems. For example, one may wish

to create an optimal schedule for a basketball tournament. In this case, one must pair up the teams for the first round of games. When representing this with a graph, it is important that there is no more than one edge incident on a vertex. In other words, each team cannot be matched with more than one team at a given time.

A complete graph is an undirected graph where every pair of vertices is connected by an edge. An example is shown in Figure 2. The maximum weight matching problem consists of a com-

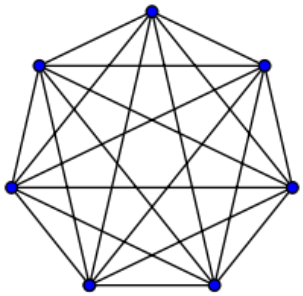

Figure 2: Complete Graph [2]

plete graph of n vertices, where n is an even number. The weights are randomly generated from a normal distribution with a mean of 0 and a variance of 1. The absolute value of all randomly generated numbers is taken before the weights are used. The weights are the only data in this problem.

This problem can be formulated as an integer program, where there is a binary decision variable for each edge. As linear programs are often easier to solve and can yield integer solutions, it is worth formulating this problem as a linear program. This can be done by simply changing each binary decision variable to a continuous decision variable with a lower bound of 0 and an upper bound of 1.

Model Description

Index Sets

i,j,k,l,m index vertices

$i = 1, ..., n$

$j = 1, ..., n$

$k = 1, ..., n$

$l = 1, ..., n$

$m = 1, ..., n$

k,l, and m will be used for constraints which will be added later.

Data

w_{ij} weight (in pounds) for the edge between vertices i and j for $j > i$

Decision Variables

x_{ij} = 1 iff edge (i,j) is in the matching, 0 otherwise for $j > i$

To solve the linear program relaxation, these variables are declared as continuous in Gurobi.

Objective

max $\sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} x_{ij}$ for $j > i$

The objective is to maximize the sum of the weights.

Constraints

1. At most one edge can be incident on a vertex: $\sum_{i=1}^{k-1} x_{ik} + \sum_{j=k+1}^{n} x_{kj} <= 1$ for all k

2. Upper bound for decision variables: $x_{ij} <= 1$ for $j > i$

3. Lower bound for decision variables: $x_{ij} >= 0$ for $j > i$

Note: For $j > i$ is used instead of for all i, j to break symmetry and to ensure that i and j are different.

Model Validation

Model validation was first done using a simple graph, which is illustrated in Figure 3. With this graph of 4 vertices and 6 edges, it is straightforward to see that the optimal solution will have selected the edge from vertex 0 to vertex 2 and the edge from vertex 1 to vertex 3 with weights 9 and 8, respectively. Although the edge from vertex 2 to vertex 3 has the largest weight, it is not selected in the optimal solution because it would have to be chosen with the edge from vertex 0 to vertex 1, giving a total sum of 15, which is less than 17. Another feasible solution selects the edge from vertex 0 to vertex 3 and the edge from vertex 1 to vertex 2 with weights 7 and 3, respectively. This gives a total sum of 10, which is less than 17. When checked in Gurobi, the optimal objective value was 17 and the same edges were selected.

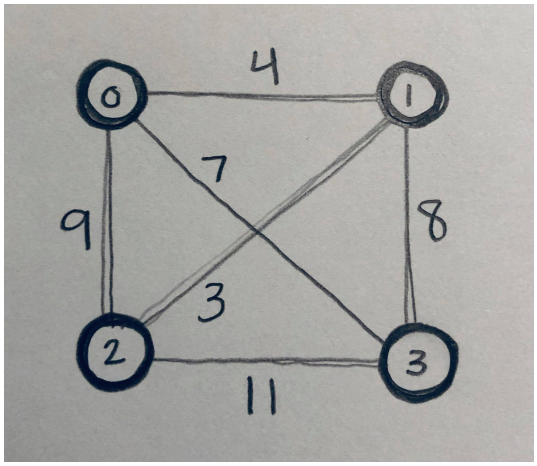

Figure 3: First Model Validation Check: Graph with 4 Vertices and 6 Edges

The second model validation check was done using a larger graph, which is illustrated in Figure 4. The number of edges increases quickly with the number of vertices. Figure 5 shows a feasible solution. By assigning weights of 2 to each of these 5 edges, and weights of 1 to all the other

edges, the model can be checked on this larger graph. When checked in Gurobi, the optimal objective value was 10 and the same edges were selected.

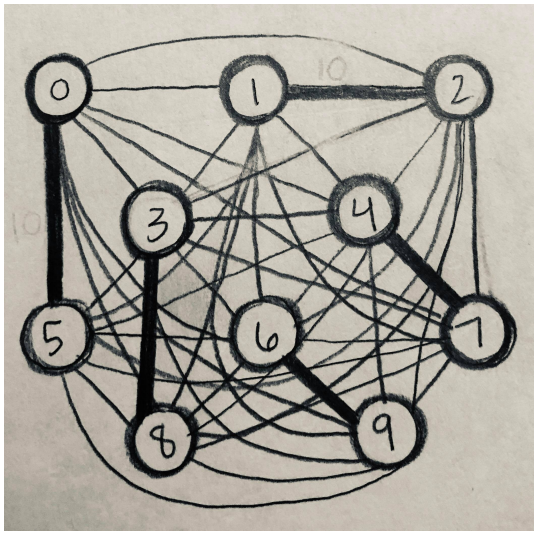

Figure 4: Second Model Validation Check: Graph with 10 Vertices and 45 Edges

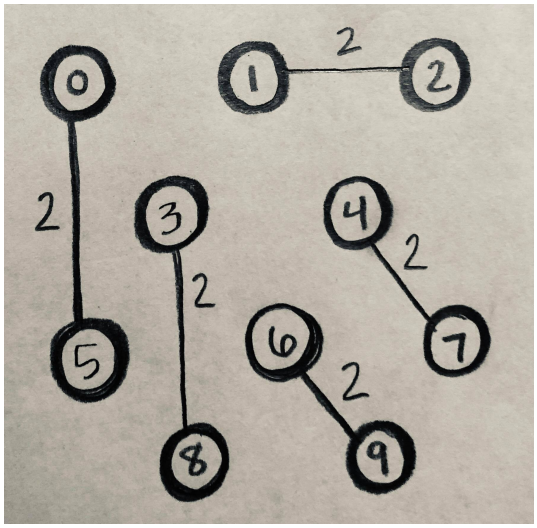

Figure 5: Second Model Validation Check: Feasible Solution

Computational Challenges

When solving this problem as a linear program, the probability of obtaining an integer solution changes with the number of vertices in the graph (n). Figure 6 shows how the probability of obtaining an optimal integer solution varies with n. Figure 7 shows how the computation time varies with n. This data was obtained by solving the linear program 100 times for each n.

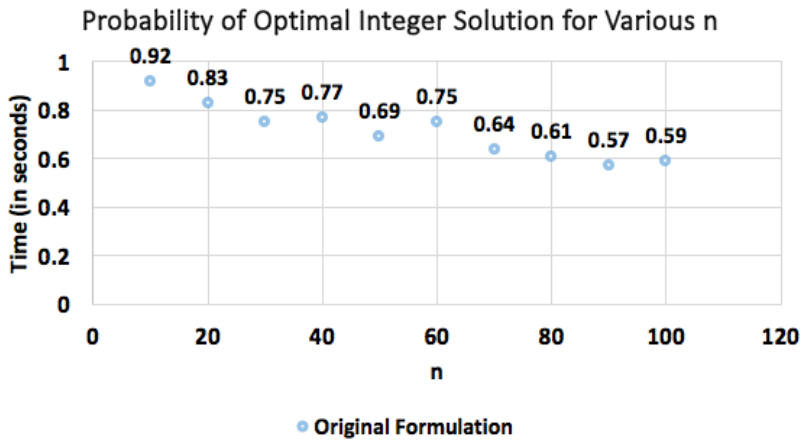

Figure 6: Probability of Optimal Integer Solution for Various n

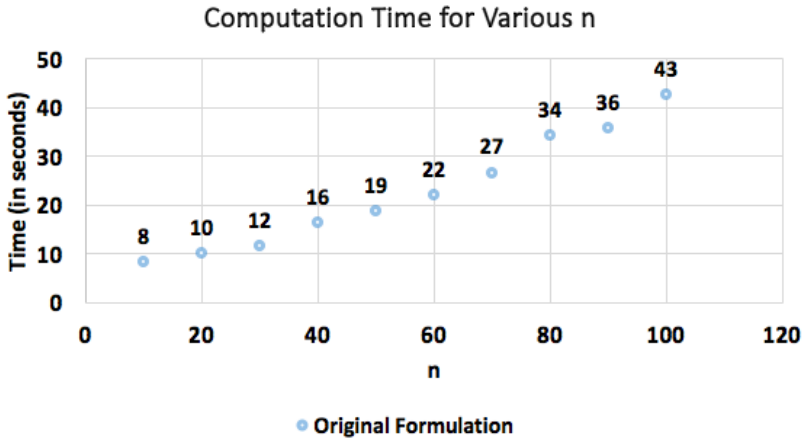

Figure 7: Computation Time for Various n (100 Trials)

From Figure 6, it is clear that as n increases, obtaining an optimal integer solution generally

becomes less probable. To improve these probabilities, the following odd-set constraints of cardinality 3 can be added to the formulation.

$x_{ij} + x_{jk} + x_{ik} <= 1$ for all $i < j < k$

With these constraints, at most one edge can exist within a cycle of 3 vertices. Figure 8 shows these probabilities (orange), as well as the probabilities from the original formulation (blue) for comparison. Figure 9 shows how the computation time varies with n. With these constraints, the computation time grows faster with n. This data was obtained by solving the linear program with the additional constraints 100 times for each n.

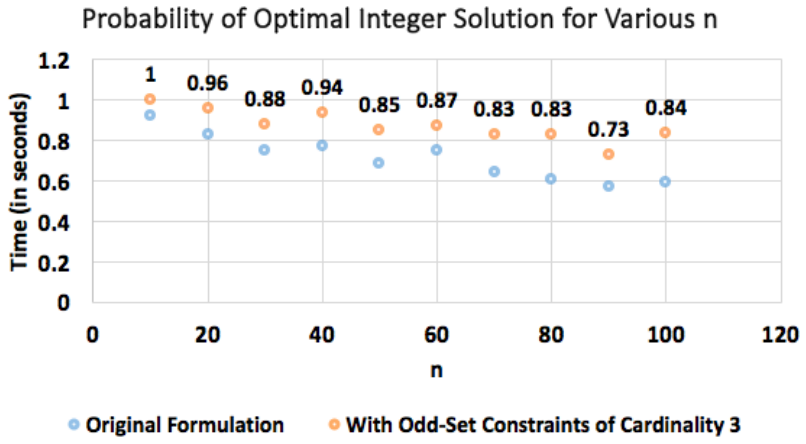

Figure 8: Probability of Optimal Integer Solution with Odd-Set Constraints of Cardinality 3 for Various n

To further improve these probabilities, the following odd-set constraints of cardinality 5 can be added to the formulation.

$x_{ij} + x_{jk} + x_{kl} + x_{lm} + x_{im} <= 2$ for all $i < j < k < l < m$

With these constraints, at most two edges can exist within a cycle of 5 vertices. Figure 10 shows these probabilities (grey), as well as the probabilities from the other formulations (orange and blue) for comparison. Figure 11 shows how the computation time varies with n. With these

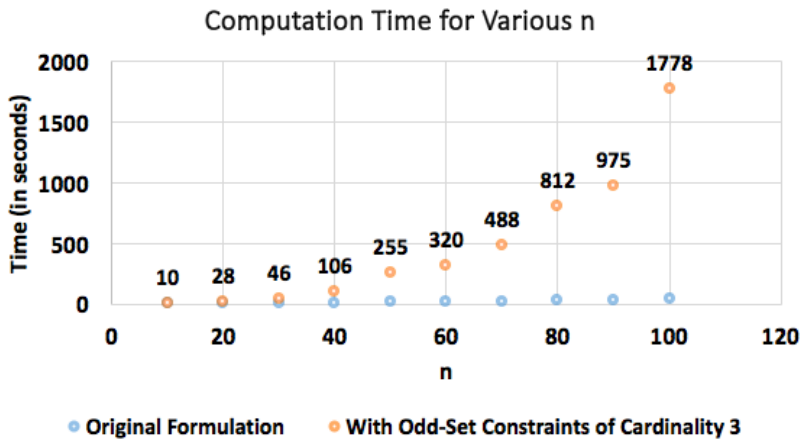

Figure 9: Computation Time with Odd-Set Constraints of Cardinality 3 for Various n (100 Trials)

constraints, the computation time grows even faster with n. This data was obtained by solving the linear program with the additional constraints 100 times for each n.

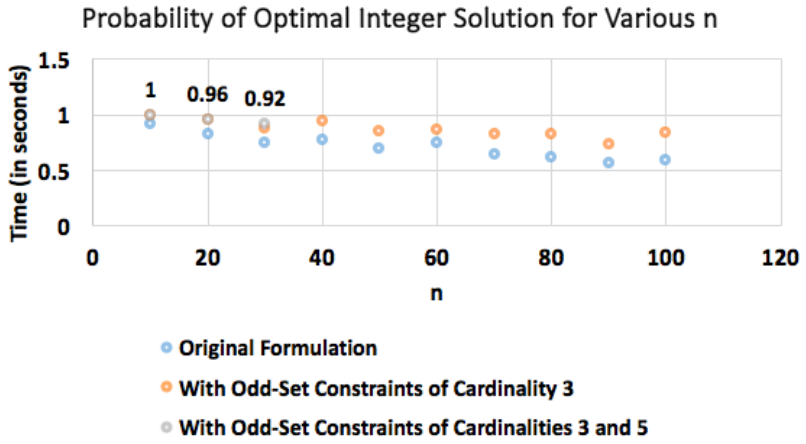

Figure 10: Probability of Optimal Integer Solution with Odd-Set Constraints of Cardinalities 3 and 5 for Various n

It is clear that the probability of obtaining an optimal integer solution generally increases as odd-set constraints are added. A trade-off is that formulations with more constraints are harder to solve. It is for this reason that the formulation with both odd-set constraints of cardinalities 3

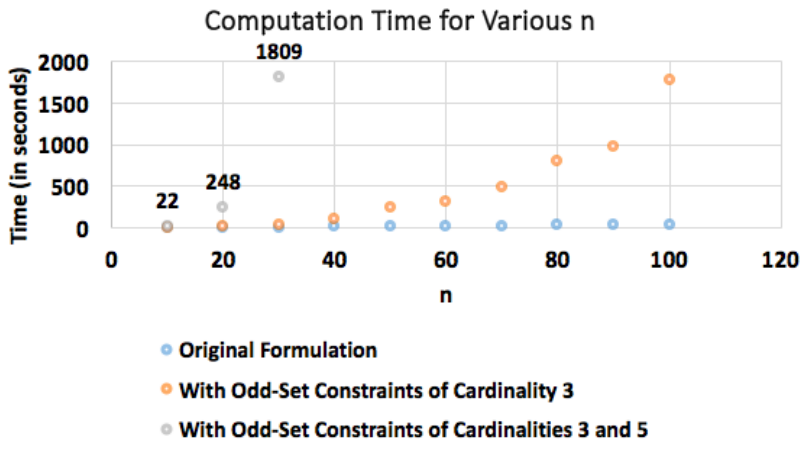

Figure 11: Computation Time with Odd-Set Constraints of Cardinalities 3 and 5 for Various n (100 Trials)

and 5 was solved for just n=10,20,30. By reducing the complexity of this formulation, one may be able to solve it for larger values of n.

First Approach: Removing Variables for Edges With Small Weights

As an optimal solution to this problem is more likely to choose the edges which have larger weights, the first approach for reducing the complexity was to only create a variable for an edge if its corresponding weight was greater than or equal to 1.

Second Approach: Adding Less Odd-Set Constraints

As not all constraints are binding at an optimal solution, the second approach for reducing the complexity was to only add constraints that are likely to be binding at an optimal solution. For the odd-set constraints of cardinality 3, this meant only adding them if any 2 weights were greater than or equal to 1. For the odd-set constraints of cardinality 5, this meant only adding them if any 3 weights were greater than or equal to 1. The logic is that these cases need to be looked out for more than other cases because the optimization model will want to choose edges with larger weights.

Third Approach: Only Using Odd-Set Constraints When Integer Solution Not Found

The odd-set constraints serve the purpose of helping find an optimal integer solution, but at the cost of increasing the computation time. Perhaps the odd-set constraints can be only used when the optimal solution is not integer. The third approach for reducing the complexity was to first try solving with the original formulation and then re-solve with the added constraints if the optimal solution is not integer.

Comparison of Approaches

Figures 12 and 13 compare the results between the 3 approaches for reducing the complexity of the model with odd-set constraints of cardinality 3. Figures 14 and 15 compare the results between the 3 approaches for reducing the complexity of the model with odd-set constraints of cardinalities 3 and 5. Both models were affected similarly by the 3 approaches. The first approach resulted in lower computation times, but at the cost of lower probabilities of obtaining optimal integer solutions. The second approach resulted in lower computation times and no change in the probabilities of obtaining optimal integer solutions. The third approach resulted in significantly lower computation times and no change in the probabilities of obtaining optimal integer solutions. This makes sense because many trials which used the third approach did not involve any odd-set constraints. In these cases, the optimal integer solution was found without them.

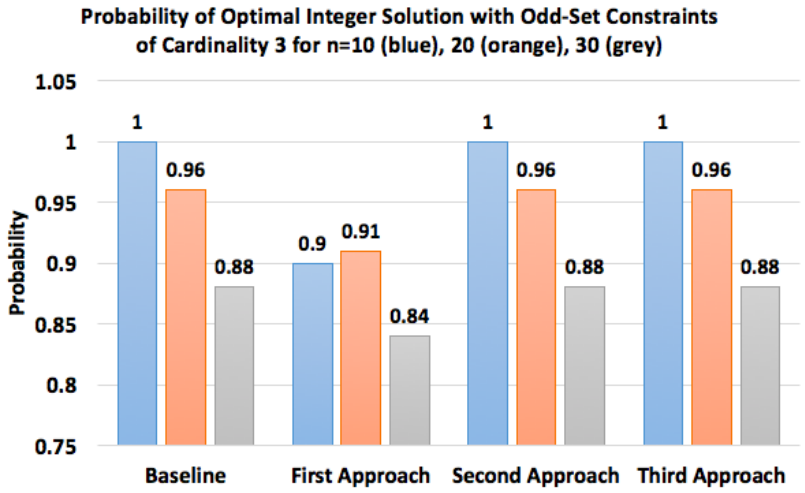

Figure 12: Probability of Optimal Integer Solution with Odd-Set Constraints of Cardinality 3 for n=10 (blue), 20 (orange), 30 (grey)

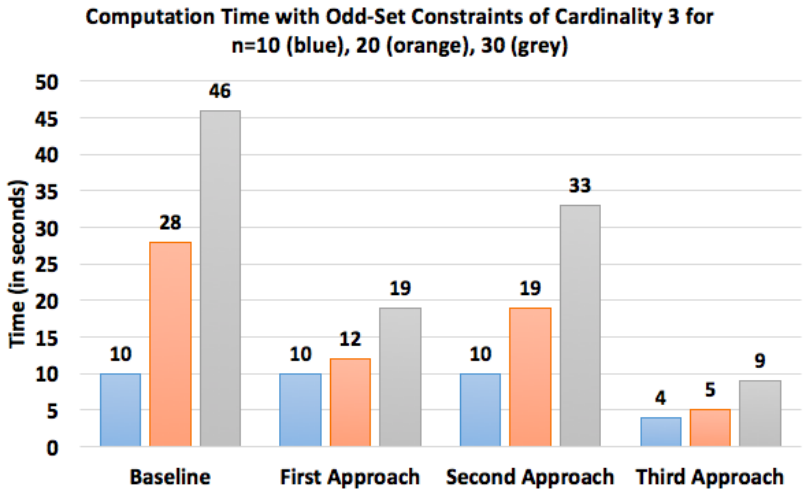

Figure 13: Computation Time with Odd-Set Constraints of Cardinality 3 for n=10 (blue), 20 (orange), 30 (grey) (100 Trials)

2

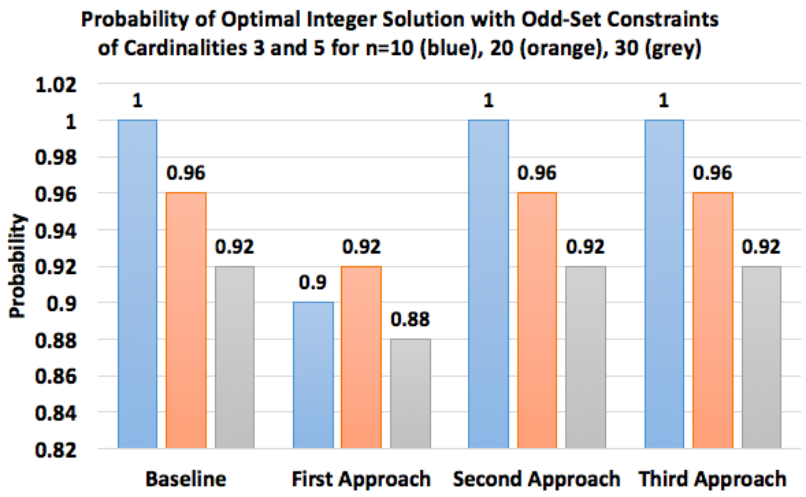

Figure 14: Probability of Optimal Integer Solution with Odd-Set Constraints of Cardinalities 3 and 5 for n=10 (blue), 20 (orange), 30 (grey)

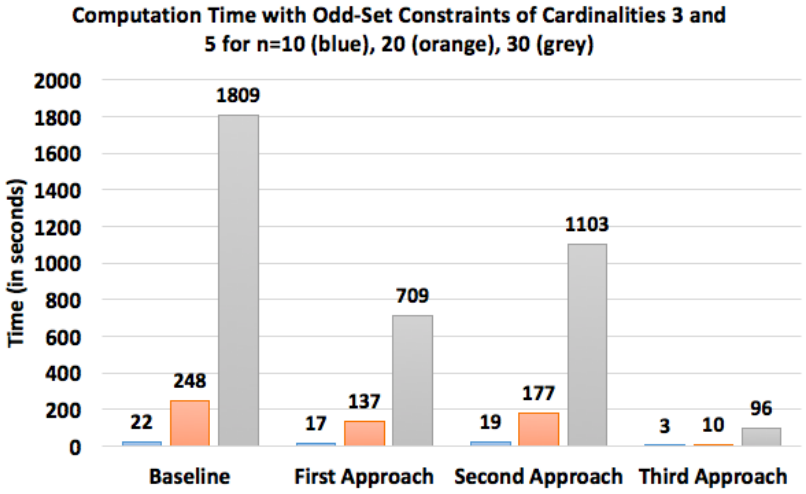

Figure 15: Computation Time with Odd-Set Constraints of Cardinalities 3 and 5 for n=10 (blue), 20 (orange), 30 (grey) (100 Trials)

Data

Numpy's random.randn(n,n) generated random numbers from a standard normal distribution and returned them in an nxn Numpy array. The absolute value of all randomly generated numbers was taken before the data was used. To see if and how the performance of the model would

change by generating data differently, standard normal without absolute value, standard uniform, and standard exponential were tested. The resulting performances are shown in Figures 16 and 17.

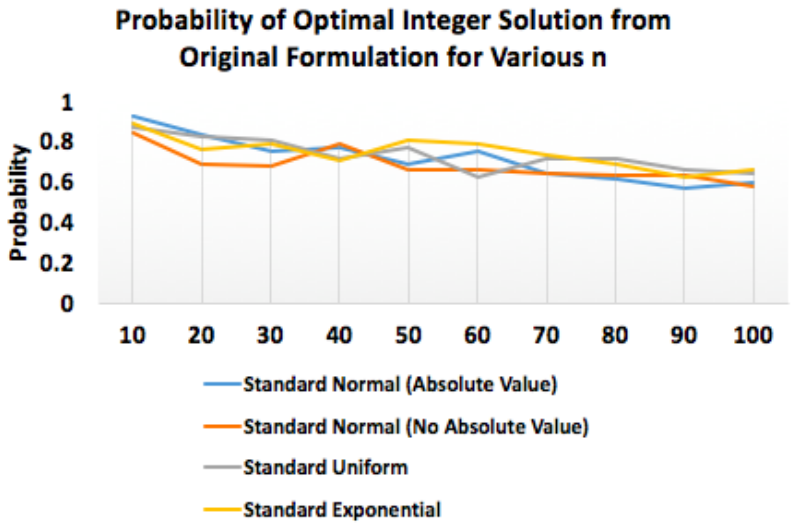

Figure 16: Probability of Optimal Integer Solution from Original Formulation for Various n and Various Data

The next experiment was to see if and how the performance of the model would change by changing the mean of the standard normal distribution. The resulting performances are shown in Figures 18 and 19.

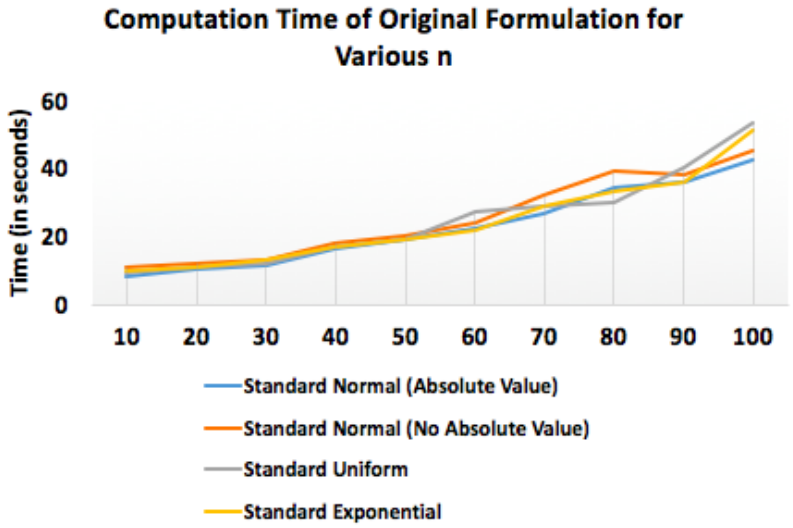

Figure 17: Computation Time of Original Formulation for Various n and Various Data (100 Trials)

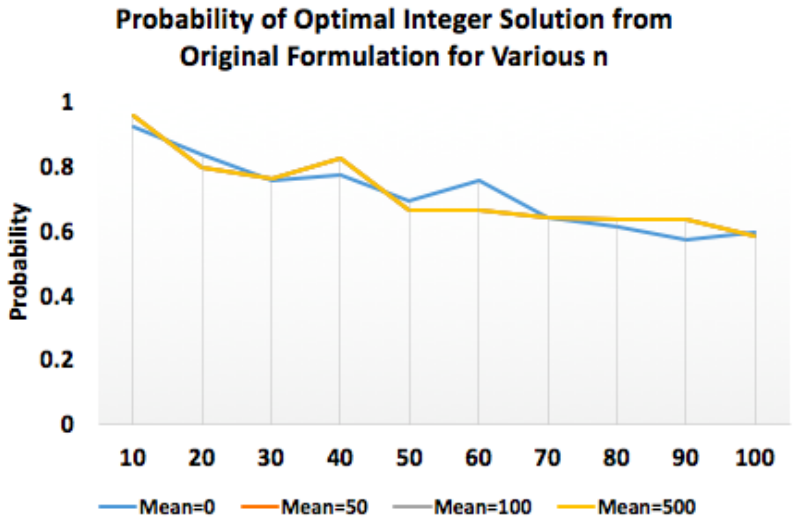

Figure 18: Probability of Optimal Integer Solution from Original Formulation for Various n and Various Data

2

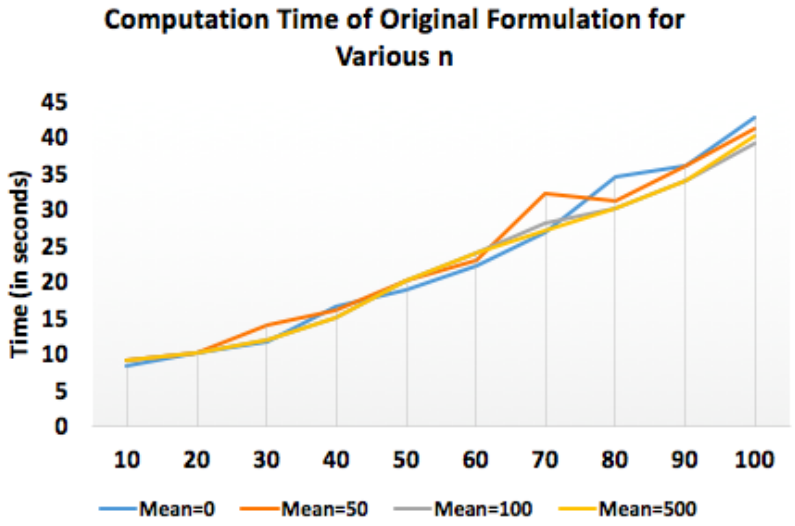

Figure 19: Computation Time of Original Formulation for Various n and Various Data (100 Trials)

References

1. Van Rantwijk, Joris. "Maximum Weighted Matching." Joris_VR, 7 Apr. 2013, jorisvr.nl/article/maximum-matching.

2. "Complete Graph." Wikipedia, Wikimedia Foundation, 9 Mar. 2020, en.wikipedia.org/wiki/Complete_graph.