Quantum Integer Programming
Solving Quadratic Knapsack problem

Shivaprasad U Hulyal (EP19B030) [a] and Neelkanth Rawat (PH20C026)[a]

[a]Department of Physics, Indian Institute of Technology Madras, Chennai
January 4, 2022

Abstract

The Quadratic Knapsack Problem involves maximising a quadratic objective function subject to a linear knapsack constraint. This problem manifests itself naturally in Operations Research, Combinatorics, Statistics. Problems like the Maximal Clique problem, which appear in the fields like DNA computing, Telecommunication, can also be formulated as an equivalent Quadratic Knapsack Problem. We reformulated the problem as an integer linear program and then solved it using (a) Commercial Solver: CBC, (b) formulating it as a standard QUBO and (c) GAMA. The QUBO was solved using solvers based on simulated and quantum annealing in the latter two approaches. For different instances of problems solved using these two methods, the optimum values and the time taken to obtain them were compared with those for CBC.

Keywords— Quadratic Knapsack, Quantum Annealing, GAMA

1 Introduction

The binary quadratic knapsack problem (QKP) introduced by Gallo et al.[5] is defined as follows: Consider a set of items where each of the i^{th} item has a positive integer weight w_i. In addition, we are given an $n \times n$ non-negative symmetric integer matrix $P = \{p_{ij}\}$, where p_{ii} is the profit achieved if the item i is selected and p_{ij} is a profit achieved if both items i and j are selected. One is interested in selecting a subset of these items such that the overall weight does not exceed a given knapsack capacity C, as well as the overall profit, gets maximised.
Let N denotes the total number of items. Then by introducing a binary variable x_i which takes value 1 if the item i is selected and 0 otherwise the problem can be mathematically formulated as[7]:

$$\begin{aligned} \max \quad & \sum_{i,j\in N, i<j} p_{ij}x_i x_j + \sum_{j\in N} p_{jj}x_j \\ \text{subject to} \quad & \sum_{j\in N} w_j x_j \leq C \\ & x_j \in \{0,1\}, j \in N \end{aligned} \tag{1}$$

QKP appears naturally in many disciplines like statistics, Operations Research. It is also interesting to note that QKP is a generalisation of the famous graph-theoretic problem of finding Maximal Clique[4] which also has some interesting applications in theoretical aspects of quantum computing[6].

2 ILP Formulation

2.1 Formulation-1

We introduce the variables y_{ij} defined for $i < j$ which is 1 iff $x_i = 1$ & $x_j = 1$[8]. The functional form of the variable y_{ij} can be mimicked by the following constraint equations.

$$\begin{aligned} y_{ij} &\leq x_i \\ y_{ij} &\leq x_j \\ x_i + x_j &\leq 1 + y_{ij} \end{aligned}$$

It then leads to the ILP Model:

$$\begin{aligned} \textbf{max} \quad & \sum_{i,j\in N, i<j} p_{ij}y_{ij} + \sum_{j\in N} p_{jj}x_j \\ \textbf{s.t.} \quad & \sum_{j\in N} w_j x_j \leq C \\ & y_{ij} \leq x_i; \quad i,j \in N, i<j \\ & y_{ij} \leq x_j; \quad i,j \in N, i<j \\ & x_i + x_j \leq 1 + y_{ij}; \quad i,j \in N; \quad i<j \\ & x_j, y_{ij} \in \{0,1\}; \quad i,j \in N, i<j \end{aligned} \tag{2}$$

To convert inequality constraints to equality constraints, we introduce slack variables s, s_2, s_3, s_4 into the constraints (one for each constraint) and re-write the problem as:

$$\begin{aligned} \textbf{max} \quad & \sum_{i,j\in N, i<j} p_{ij}y_{ij} + \sum_{j\in N} p_{jj}xj + 0^T\overrightarrow{s} \\ \textbf{s.t.} \quad & \sum_{j\in N} w_j x_j + s = C \\ & -x_i + y_{ij} + s_{2_{ij}} = 0; \quad i,j \in N, i<j \\ & -x_j + y_{ij} + s_{3_{ij}} = 0; \quad i,j \in N, i<j \\ & x_i + x_j - y_{ij} + s_{4_{ij}} = 1; \quad i,j \in N, i<j \\ & x_j, y_{ij} \in \{0,1\}; i,j \in N, i<j \\ & s_i \in \{0,1\}; s \in \{0,C\} \end{aligned} \tag{3}$$

2.2 Formulation-2

This involves solving C+1 problems corresponding to the different $\tilde{c}$

$$\begin{aligned} \textbf{max} \quad & \sum_{i,j\in N, i<j} p_{ij}y_{ij} + \sum_{j\in N} p_{jj}x_j \\ \textbf{s.t.} \quad & \sum_{j\in N} w_j x_j = \tilde{c} \end{aligned}$$

where $\tilde{c} \in \{0, 1, 2, \ldots, C\}$

$$\begin{array}{rcl} y_{ij} & \leq & x_i; \quad i,j \in N, i < j \\ y_{ij} & \leq & x_j; \quad i,j \in N, i < j \\ x_i + x_j & \leq & 1 + y_{ij}; \quad i,j \in N; \quad i < j \\ x_j, y_{ij} & \in & \{0,1\}; \quad i,j \in N, i < j \end{array} \tag{4}$$

$$\textbf{max} \quad \sum_{i,j \in N, i<j} p_{ij} y_{ij} + \sum_{j \in N} p_{jj} xj$$

$$\begin{array}{rcl} \textbf{s.t.} \quad \sum_{j \in N} w_j x_j & = & \tilde{c} \\ \tilde{c} & \in & \{0, 1, 2, \ldots, C\} \\ -x_i + y_{ij} + s_{2ij} & = & 0; \quad i,j \in N, i < j \\ -x_j + y_{ij} + s_{3ij} & = & 0; \quad i,j \in N, i < j \\ x_i + x_j - y_{ij} + s_{4ij} & = & 1; \quad i,j \in N, i < j \\ x_j, y_{ij} & \in & \{0,1\}; i,j \in N, i < j \\ s_i & \in & \{0,1\}; \end{array} \tag{5}$$

2.3 Matrix equation for constraints

For Formulation-1 By observing the patterns the coefficients of x_{ij} and y_{ij} make, the general form of the coefficient matrix A can be found as:

$$A = \begin{pmatrix} \overrightarrow{W}_{1\times N} & 0_{1\times d} & 0_{1\times d} & 0_{1\times d} & 0_{1\times d} & 1 \\ \mathrm{G}_{d\times N} & I_{d\times d} & I_{d\times d} & 0_{d\times d} & 0_{d\times d} & 0_{d\times 1} \\ \mathrm{R}_{d\times N} & I_{d\times d} & 0_{d\times d} & I_{d\times d} & 0_{d\times d} & 0_{d\times 1} \\ \mathrm{L}_{d\times N} & -I_{d\times d} & 0_{d\times d} & 0_{d\times d} & I_{d\times d} & 0_{d\times 1} \end{pmatrix}$$

where: d=$\frac{N(N-1)}{2}$

$$G = \begin{bmatrix} -1 & 0 & \ldots & 0 & 0 \\ \vdots & \vdots & \ldots & 0 & 0 \\ -1_N & 0_N & \ldots & 0 & 0 \\ 0 & -1 & \ldots & 0 & 0 \\ \vdots & \vdots & \ldots & 0 & 0 \\ \vdots & -1_{N-1} & \ldots & 0 & 0 \\ \vdots & \vdots & \ldots & 0 & 0 \\ 0 & 0 & \ldots & -1 & 0 \end{bmatrix}_{d\times N}$$

$$R = \begin{bmatrix} 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ \vdots & \vdots & 0 & \ddots & 0 \\ 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 \\ \vdots & \vdots & 0 & 0 & \ddots \\ 0 & 0 & 0 & 0 & -1 \end{bmatrix}_{d\times N}$$

$$L = -1 \times (G + R)$$

Note that in this case the last element of the variable vector

$$\mathbf{x} = (\overrightarrow{x}|\overrightarrow{y}|\overrightarrow{s_2}|\overrightarrow{s_3}|\overrightarrow{s_4}|s)^\top$$

is not a binary variable.

Formulation-2 The matrix A for this case remains the same as Formulation-1, with the last column deleted.

$$A = \begin{pmatrix} \overrightarrow{W}_{1\times N} & 0_{1\times d} & 0_{1\times d} & 0_{1\times d} & 0_{1\times d} \\ \mathrm{G}_{d\times N} & I_{d\times d} & I_{d\times d} & 0_{d\times d} & 0_{d\times d} \\ \mathrm{R}_{d\times N} & I_{d\times d} & 0_{d\times d} & I_{d\times d} & 0_{d\times d} \\ \mathrm{L}_{d\times N} & -I_{d\times d} & 0_{d\times d} & 0_{d\times d} & I_{d\times d} \end{pmatrix}_{(3d+1)\times(N+4d)}$$

$$\mathbf{x} = (\overrightarrow{x}|\overrightarrow{y}|\overrightarrow{s_2}|\overrightarrow{s_3}|\overrightarrow{s_4})^\top$$

2.4 RHS vector of the constraint equation

Formulation-1 The RHS Vector of the constraint equation takes the form:

$$b^T = (C|0_{1\times 2d}|1_{1\times d})_{1\times(3d+1)}$$

Formulation-2 For this case, one is required to solve constraint equation for all $\tilde{c} \leq C$ separately, where the RHS vector of constraint equation is:

$$b_{\tilde{c}}^T = (\tilde{c}|0_{1\times 2d}|1_{1\times d})_{1\times(3d+1)}$$

3 Methodology

3.1 QUBO formulation

Binarisation of variables: We need to binarize our variables before formulating the problem as QUBO. Only the last variable s needs to be encoded. We use the binary encoding scheme:

$$\begin{aligned} s &= 2^0 s_{10} + 2^1 s_{11} + 2^2 s_{12} + ... + 2^{\alpha-1} s_{1\alpha-1} \\ &= (2^0, 2^1, 2^2, ..., 2^{\alpha-1})^T \overrightarrow{s_1} \end{aligned}$$

where $\alpha = \lceil log_2(C+1) \rceil$ is the number of binary digits required to encode.
Now the coefficient matrix ($\tilde{A}$) of the constraint equations becomes:

$$\begin{pmatrix} \overrightarrow{W}_{1\times N} & 0_{1\times d} & 0_{1\times d} & 0_{1\times d} & 0_{1\times d} & 2^0 & ... & 2^{\alpha-1} \\ \mathrm{G}_{d\times N} & I_{d\times d} & I_{d\times d} & 0_{d\times d} & 0_{d\times d} & 0_{d\times 1} & ... & 0_{d\times 1} \\ \mathrm{R}_{d\times N} & I_{d\times d} & 0_{d\times d} & I_{d\times d} & 0_{d\times d} & 0_{d\times 1} & ... & 0_{d\times 1} \\ \mathrm{L}_{d\times N} & -I_{d\times d} & 0_{d\times d} & 0_{d\times d} & I_{d\times d} & 0_{d\times 1} & ... & 0_{d\times 1} \end{pmatrix}$$

The ILP formulation of QKP in binary variables is:

$$\min_{\tilde{\mathbf{x}}} -(\mathbf{c}^\top \tilde{\mathbf{x}})$$

where

$$\mathbf{c}^\top = (p_{11}, \ldots p_{NN}, p_{12}, \ldots, p_{1N}, p_{23}, \ldots, p_{2N}, \ldots, p_{(N-1)N})$$

and $\tilde{\mathbf{x}} = (\overrightarrow{x}|\overrightarrow{y}|\overrightarrow{s_2}|\overrightarrow{s_3}|\overrightarrow{s_4}|\overrightarrow{s_1})$

$$s.t. \tilde{\mathbf{A}}\tilde{\mathbf{x}} = \mathbf{b}$$

$$\tilde{\mathbf{x}}_\mathbf{i} \in \{0,1\}$$

We unconstrain the problem as follows[2]:

$$\min_{\tilde{\mathbf{x}}} -\mathbf{c}^\top \tilde{\mathbf{x}} + \rho(\tilde{\mathbf{A}}\tilde{\mathbf{x}} - \mathbf{b})^\top (\tilde{\mathbf{A}}\tilde{\mathbf{x}} - \mathbf{b}) \tag{6}$$

$$\tilde{\mathbf{x}}_\mathbf{i} \in \{0,1\}$$

where ρ is the penalty value.
Exploiting the fact that $\tilde{x}^2 = \tilde{x}$ for $\tilde{x} \in \{0,1\}$, we can make the linear terms appear in the diagonal of the Q matrix.

$$\rho(\tilde{\mathbf{A}}\tilde{\mathbf{x}}-\mathbf{b})^\top(\tilde{\mathbf{A}}\tilde{\mathbf{x}}-\mathbf{b}) = \rho(\tilde{\mathbf{x}}^\top(\tilde{\mathbf{A}}^\top\tilde{\mathbf{A}})\tilde{\mathbf{x}} - 2\mathbf{b}^\top\tilde{\mathbf{A}}\tilde{\mathbf{x}} + \mathbf{b}^\top\mathbf{b})$$

So the required QUBO is:

$$\min_{\tilde{\mathbf{x}}} -\mathbf{c}^\top\tilde{\mathbf{x}} + \rho(\tilde{\mathbf{x}}^\top(\tilde{\mathbf{A}}^\top\tilde{\mathbf{A}})\tilde{\mathbf{x}} - 2\mathbf{b}^\top\tilde{\mathbf{A}}\tilde{\mathbf{x}} + \mathbf{b}^\top\mathbf{b}) \quad (7)$$

$$\tilde{\mathbf{x}}_\mathbf{i} \in \{0,1\}$$

3.2 Testing QUBO

We adopted the following procedure proposed by Gallo, Hammer, and Simone and referenced by Caprara et.al; in 1999: [3]

- Profit matrix p_{ij} of different densities (number of non-zero elements) where the non-zero elements were uniformly distributed in the interval [0,100]
- Weight w_j in the constraint function was uniformly distributed in [1, 50].
- The capacity C was randomly selected from [50, Σw_j].

3.3 GAMA

GAMA is a novel quantum-classical algorithm to solve linear and non-linear integer programs by evaluating the graver basis which forms an optimal test set. In general calculating Graver bases is exponentially hard on classical computers due to which they are not used for solving practical problems on classical commercial solvers. With a quantum annealer, however, it may be a viable approach to use them.[1]
The Graver bases $G(A)$ of the matrix A is a minimal set of elements of the kernel $ker_Z A$ with respect to the partial ordering $x \sqsubseteq y$ when $x_i y_i \geq 0$ (the two vectors x and y lie on the same orthand) and $|x_i| \leq |y_i|$.
Solving an ILP with GAMA essentially involves following three steps:

1. Calculation of feasible solution(s)($\overrightarrow{x_f}$) for constraint equation $\mathbf{Ax} = \mathbf{b}, \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}$ by reformulating this matrix equation as a QUBO:

$$\min_{\mathbf{X}} \mathbf{X}^\top\mathbf{Q_B}\mathbf{X} \quad (8)$$

$$\mathbf{X_i} \in \{0,1\}^{nk} \quad (9)$$

where: $Q_B = \mathbf{E}^\top\mathbf{Q_I}\mathbf{E} + 2diag[(\mathbf{L}^\top\mathbf{Q_I} - \mathbf{b}^\top\mathbf{A})\mathbf{E}]$ (10)

$$\mathbf{Q_I} = \mathbf{A}^\top\mathbf{A} \quad (11)$$

where matrices $\mathbf{E}$ and $\mathbf{L}$ encodes the integer variable vector $\mathbf{x}$ in terms of binary variables $\mathbf{X_i}$ as: $\mathbf{x}$=$\mathbf{L}$+$\mathbf{EX}$.

For Formulation 1:

$\mathbf{L} = \mathbf{0}$ and

$$\mathbf{E} = \begin{pmatrix} I_{(N+4d)\times(N+4d)} & 0_{(N+4d)\times(\lceil log_2(C+1)\rceil)} \\ 0_{1\times(N+4d)} & D_{1\times(\lceil log_2(C+1)\rceil)} \end{pmatrix}$$

where

$$D = \left(2^0, 2^1, 2^2, ..., 2^{\lceil log_2(C+1)\rceil - 1}\right)$$

For Formulation 2:

$\mathbf{L} = \mathbf{0}$ and $\mathbf{E} = \mathbf{I}$; Hence

$$Q_B = \mathbf{A}^\top\mathbf{A} + 2diag[-(\mathbf{b}_{\tilde{\mathbf{c}}}^\top\mathbf{A})] \quad (12)$$

In this case we solve $C+1$ such QUBOs, one for each $\tilde{c} \leq C$

2. Calculation of Graver-bases ($G(A)$) of the coefficient matrix A. Note that if $\overrightarrow{x_g} \in G(A)$ then $A\overrightarrow{x_g} = 0$. We found graver elements of matrix A as follows:

   Formulation-1: Since the difference of any two feasible solutions x_{1f} & x_{2f} of the above equation satisfy $A(x_{1f} - x_{2f}) = 0$. We obtained an over-complete Graver basis by taking the difference of every two solution vectors of $Ax = b$.

   Formulation-2: For this case, the method is similar to formulation-1 with the only difference being that now one can only take difference between feasible solutions for the constraint equations $Ax = b_{\tilde{c}}$ for a particular $\tilde{c}$ as the RHS vector $b_{\tilde{c}}$ is different for different $\tilde{c}$s.

3. Augmentation step: Note that if $\overrightarrow{x_f}$ is a feasible solution of constraint equations then so is $\overrightarrow{x_f} + \overrightarrow{x_g}$ as $A(\overrightarrow{x_f} + \overrightarrow{x_g}) = A(\overrightarrow{x_f}) + A(\overrightarrow{x_g}) = b + 0 = b$. In this way, one can move from one feasible solution to other without leaving the lattice of feasible solution.

   Augmentation Schemes one can use[2]:

   (a) Augmentation method 1: In this method the objective function is evaluated at $\{\overrightarrow{x_f} + \overrightarrow{x_g}\}$ for all $\overrightarrow{x_g} \in G(A)$ and the element of $G(A)$ for which the value of objective function is minimum is chosen for augmentation.

   (b) Augmentation method 2: In this method, one begins to evaluate objective function at $\{\overrightarrow{x_f} + \overrightarrow{x_g}\}$, and chooses that first encountered element of $G(A)$ for which there is some improvement in the objective function's value.

   Hence by choosing an appropriate augmentation method, one can arrive at the feasible solution, which also minimises the cost function.

3.4 Testing GAMA

We adopted the same procedure as we did for testing the results of QUBO with the only difference being that this time we also considered following 4 different capacities values for a given N and density:

1. Capacity 1 $\in [50, \Sigma w_j + 1]$
2. Capacity 2 $\in [(0.25 \times \Sigma w_j) + 1, (0.40 \times \Sigma w_j) + 1]$
3. Capacity 3 $= int(0.5(\Sigma w_j))$
4. Capacity 4 $\in [(0.80 \times \Sigma w_j), (0.95 \times \Sigma w_j) + 1]$

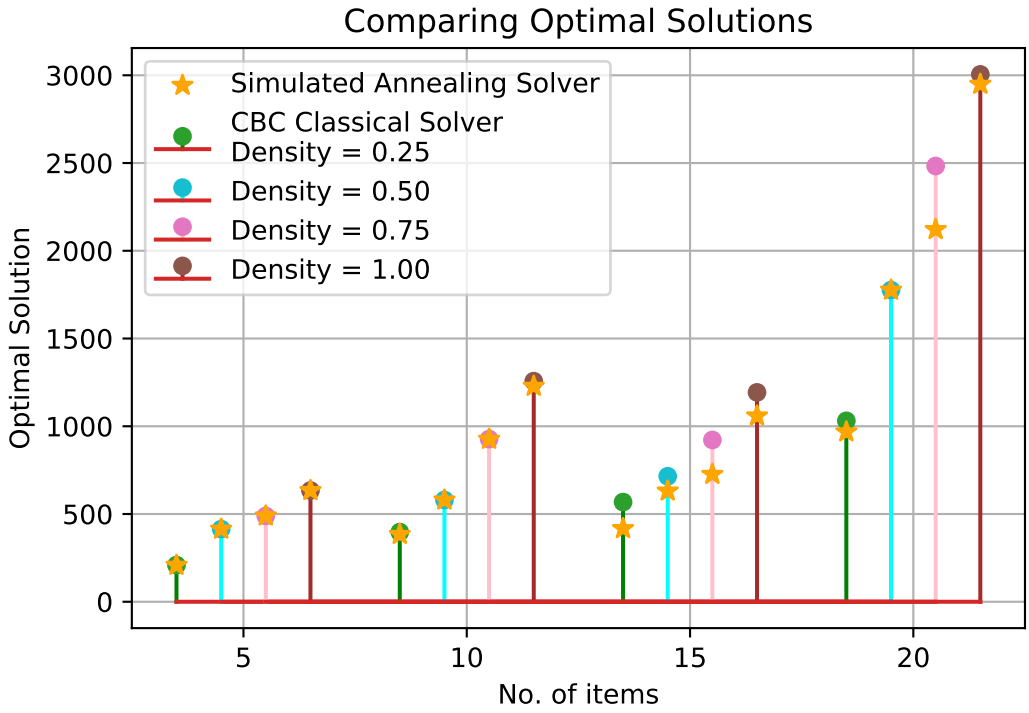
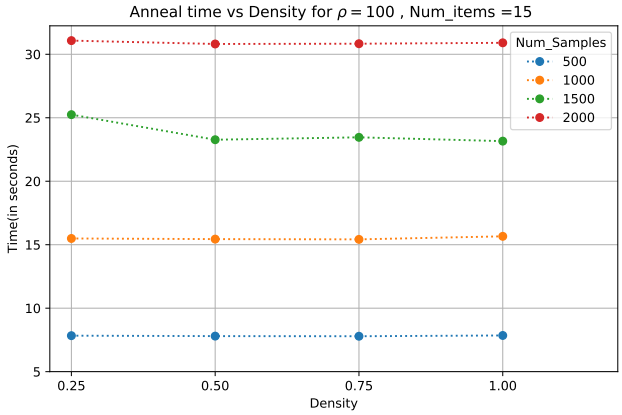
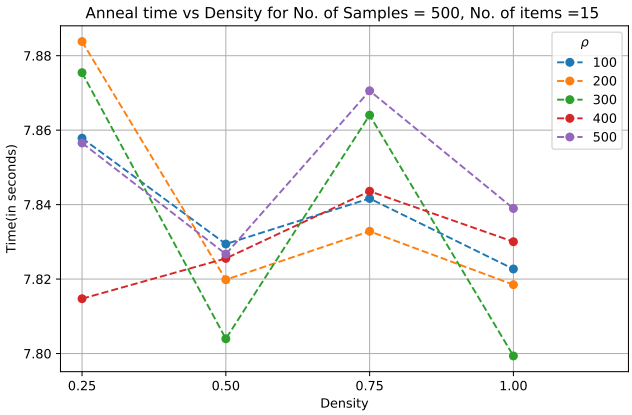


Figure 1: Comparison of optimum values of the objective function obtained from QUBO equivalent version solved by simulated annealing with those obtained from classical solver:CBC.

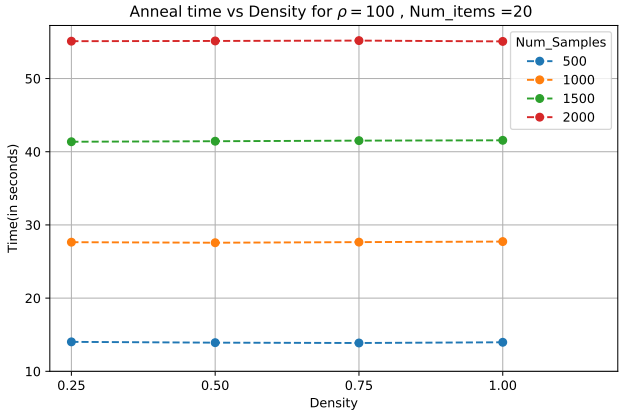
4 Time taken for computation of Optimal Solution from QUBO solved by annealing



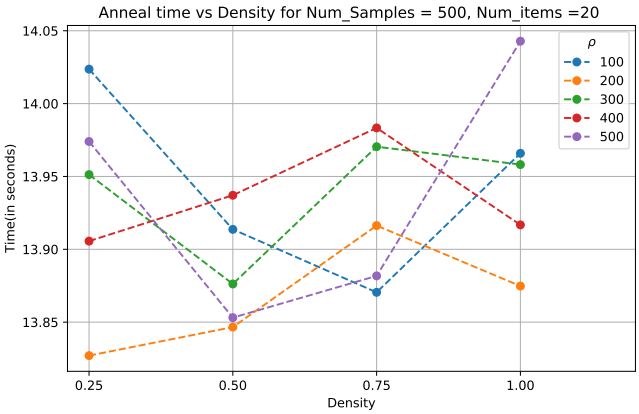
(a) N = 15 and $\rho = 100$



(b) N = 15 and No. of samples = 500

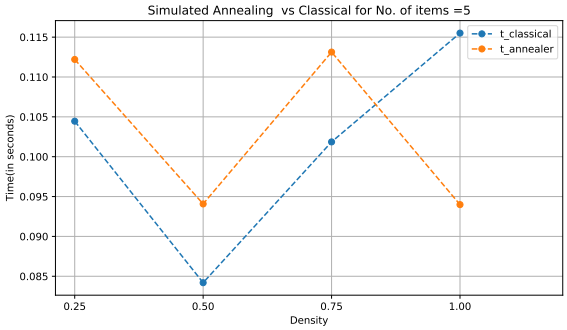


(c) N = 20 and $\rho = 100$

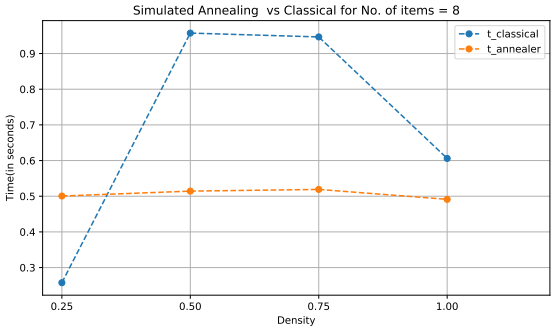


(d) N = 20 and No. of samples = 500

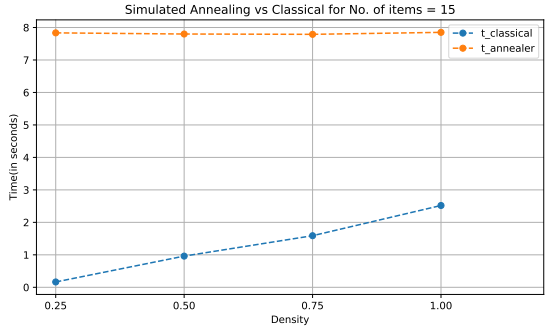
Figure 2: Comparing time taken by simulated annealing for Profit Matrix p_{ij} of different densities for (1) fixed number of items - N, and penalty factor - ρ (a,c) and (2) for a fixed number of samples (b,d)



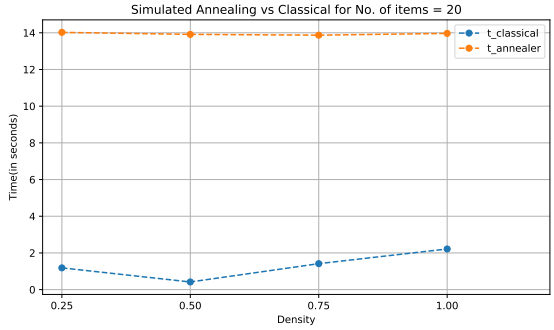
(a) No. of items = 5



(b) No. of items = 10



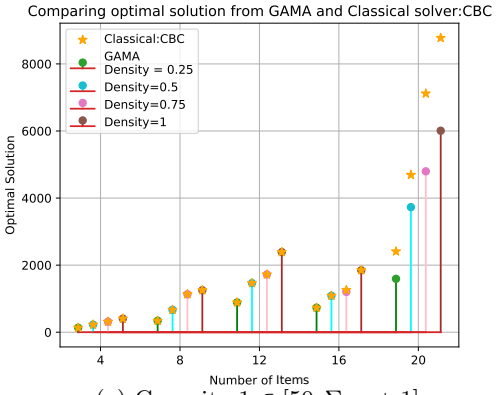
(c) No. of items = 15



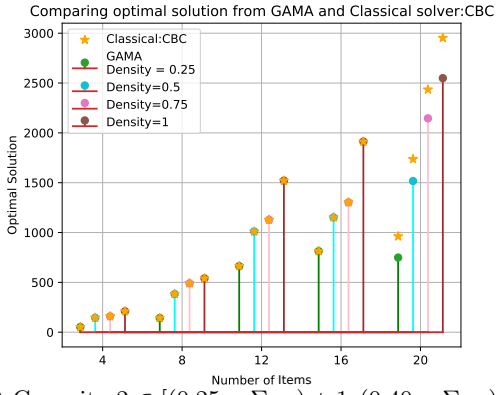
(d) No. of items = 20

Figure 3: Comparing time taken by simulated annealing for Profit Matrix p_{ij} of different densities for different N with CBC Classical Solver

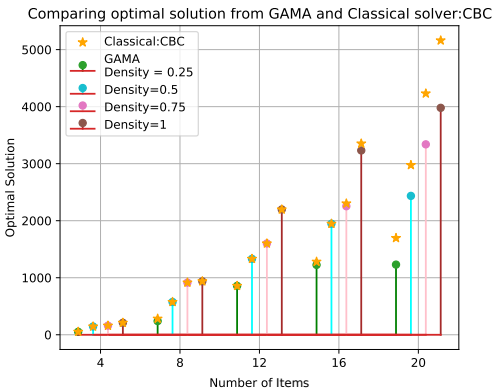
5 Formulation 1: Optimal Objective function values from GAMA



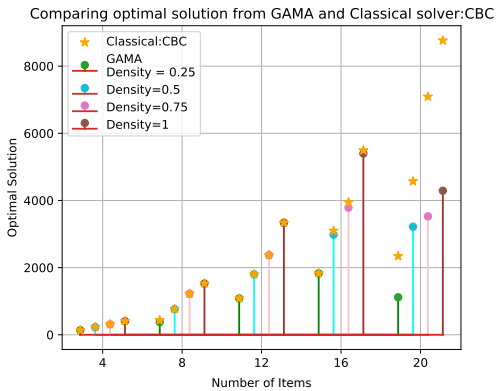
(a) Capacity 1 $\in [50, \Sigma w_j + 1]$



(b) Capacity 2 $\in [(0.25 \times \Sigma w_j) + 1, (0.40 \times \Sigma w_j) + 1]$

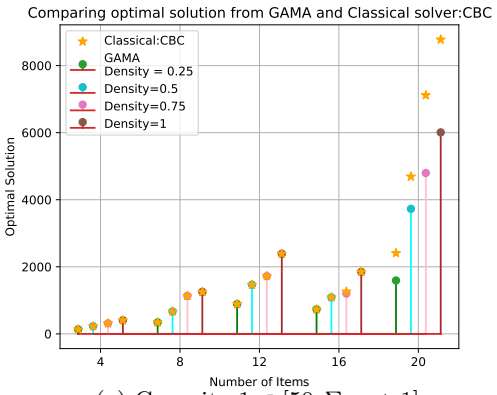


(c) Capacity 3 = $int(0.5(\Sigma w_j))$

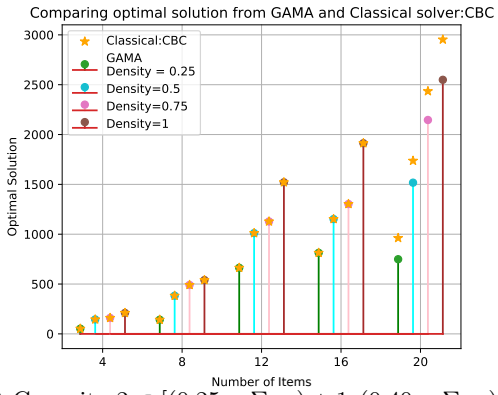


(d) Capacity 4 $\in [(0.80 \times \Sigma w_j), (0.95 \times \Sigma w_j) + 1]$

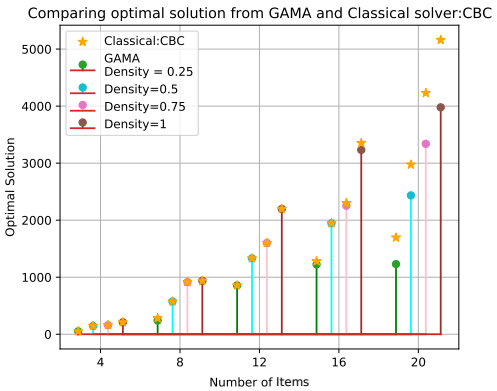
Figure 4: Comparing optimal solution obtained from GAMA with one obtained from the Classical solver CBC for different Number of items and densities for various capacities when Augmentation Method-1 was used.



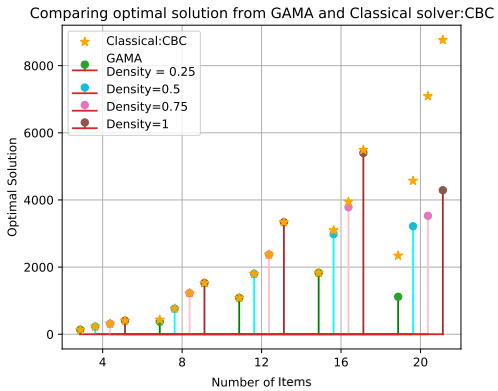
(a) Capacity 1 $\in [50, \Sigma w_j + 1]$



(b) Capacity 2 $\in [(0.25 \times \Sigma w_j) + 1, (0.40 \times \Sigma w_j) + 1]$



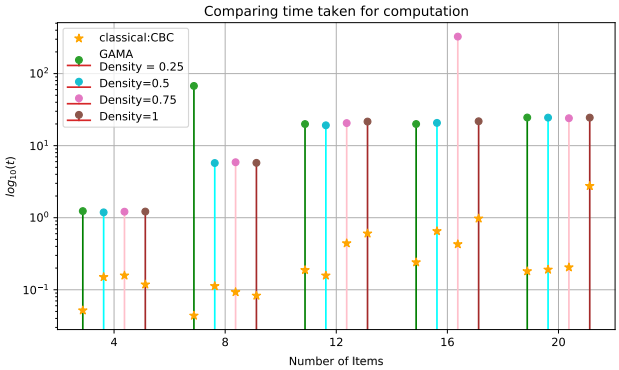
(c) Capacity 3 = $int(0.5(\Sigma w_j))$



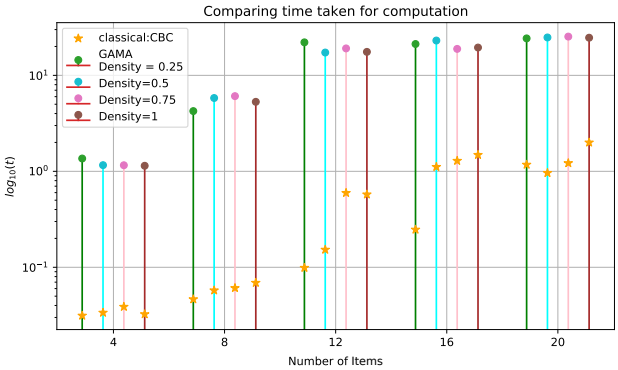
(d) Capacity 4 $\in [(0.80 \times \Sigma w_j), (0.95 \times \Sigma w_j) + 1]$

Figure 5: Comparing optimal solution obtained from GAMA with one obtained from the Classical solver CBC for different Number of Items and densities for various capacities when Augmentation Method-2 was used

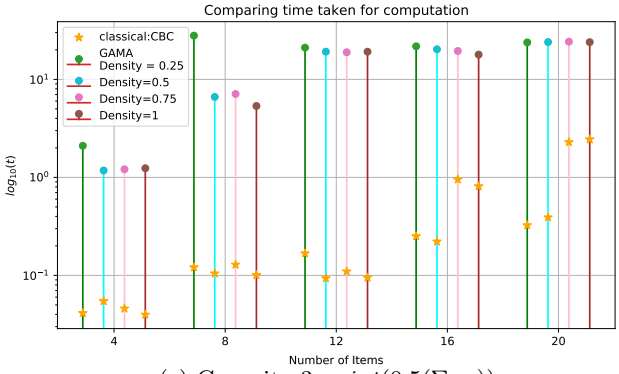
6 Formulation 1: Time taken for computation of Optimal Solution from GAMA



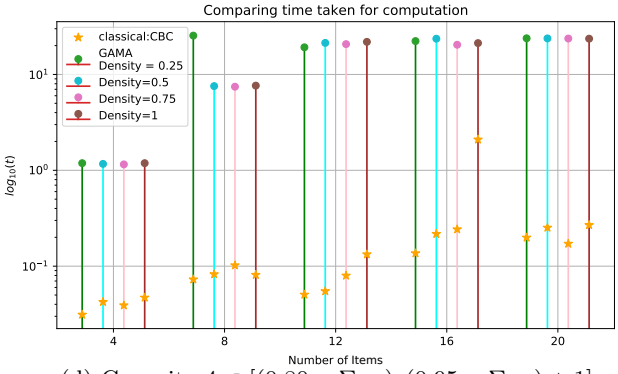
(a) Capacity 1 $\in [50, \Sigma w_j + 1]$



(b) Capacity 2 $\in [(0.25 \times \Sigma w_j) + 1, (0.40 \times \Sigma w_j) + 1]$

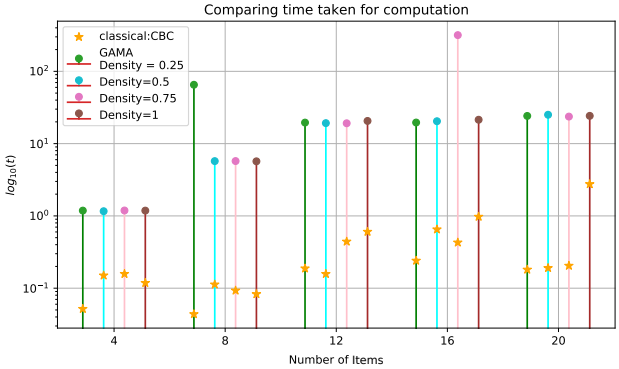


(c) Capacity 3 $= int(0.5(\Sigma w_j))$

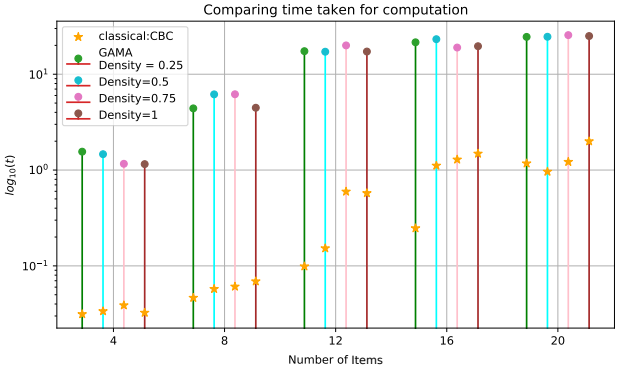


(d) Capacity 4 $\in [(0.80 \times \Sigma w_j), (0.95 \times \Sigma w_j) + 1]$

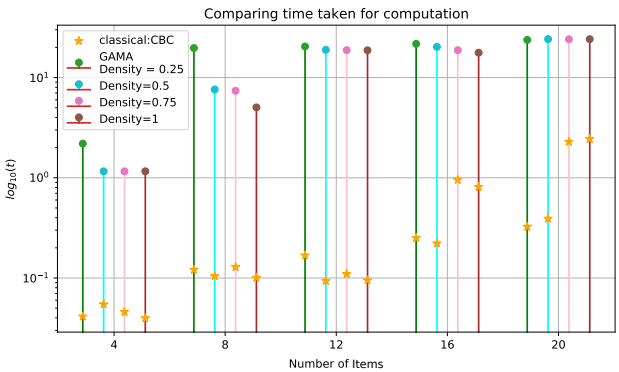
Figure 6: Comparing time taken to get optimal solution from GAMA when augmentation method 1 was used with that for Classical solver CBC for different Number of Items and densities for four different capacity ranges.



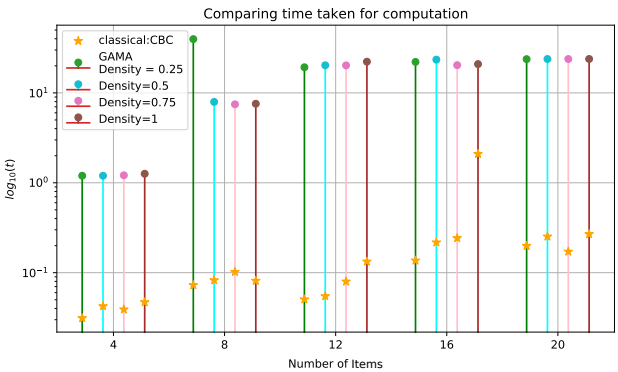
(a) Capacity 1 $\in [50, \Sigma w_j + 1]$



(b) Capacity 2 $\in [(0.25 \times \Sigma w_j) + 1, (0.40 \times \Sigma w_j) + 1]$



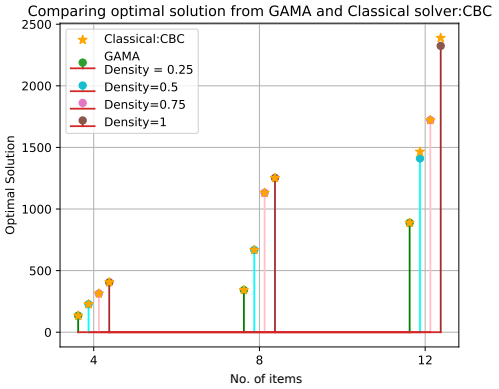
(c) Capacity 3 $= int(0.5(\Sigma w_j))$



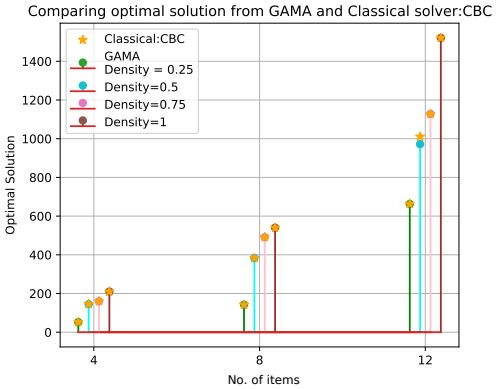
(d) Capacity 4 $\in [(0.80 \times \Sigma w_j), (0.95 \times \Sigma w_j) + 1]$

Figure 7: Comparing time taken to get optimal solution from GAMA when augmentation method 2 was used with that for Classical solver CBC for different Number of Items and densities for four different capacity ranges.

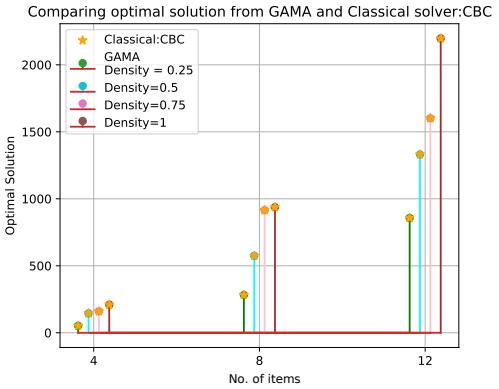
7 Formulation 2: Optimal Objective function values from GAMA



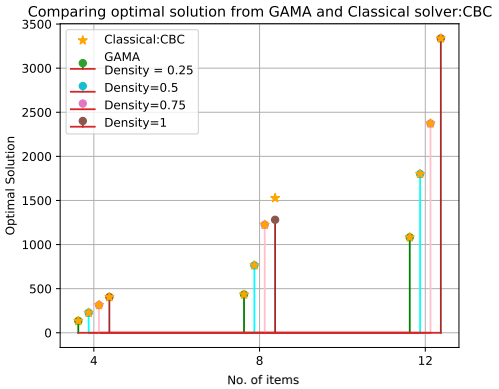
(a) Capacity 1 $\in [50, \Sigma w_j + 1]$



(b) Capacity 2 $\in [(0.25 \times \Sigma w_j) + 1, (0.40 \times \Sigma w_j) + 1]$

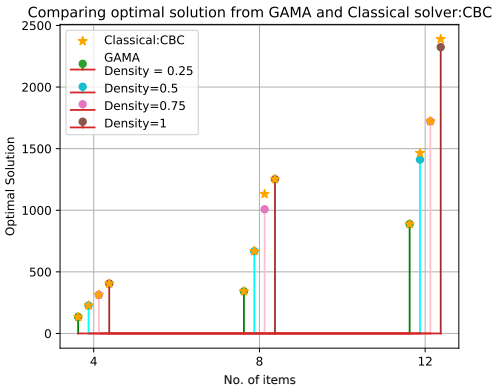


(c) Capacity 3 $= int(0.5(\Sigma w_j))$

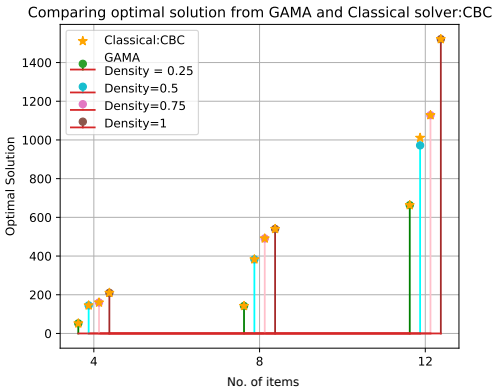


(d) Capacity 4 $\in [(0.80 \times \Sigma w_j), (0.95 \times \Sigma w_j) + 1]$

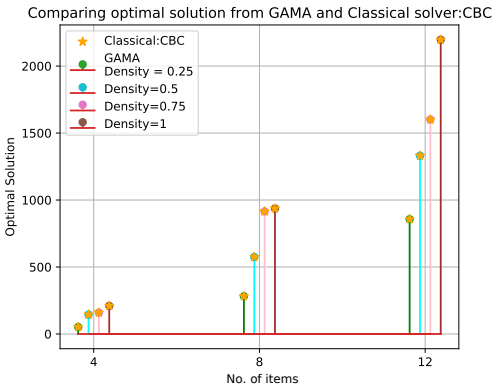
Figure 8: Comparing optimal solution obtained from GAMA with one obtained from the Classical solver CBC for different Number of Items and densities for various capacities when Augmentation Method-1 was used



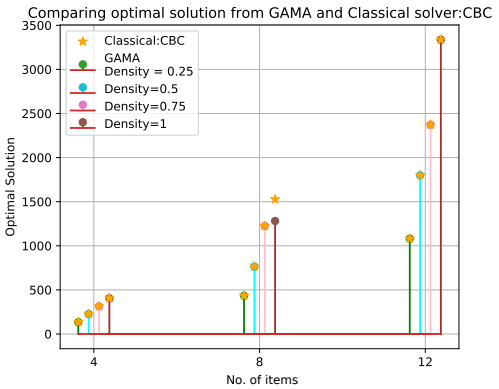
(a) Capacity 1 $\in [50, \Sigma w_j + 1]$



(b) Capacity 2 $\in [(0.25 \times \Sigma w_j) + 1, (0.40 \times \Sigma w_j) + 1]$



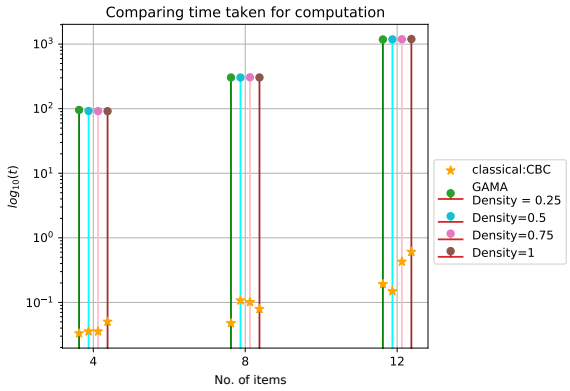
(c) Capacity 3 $= int(0.5(\Sigma w_j))$



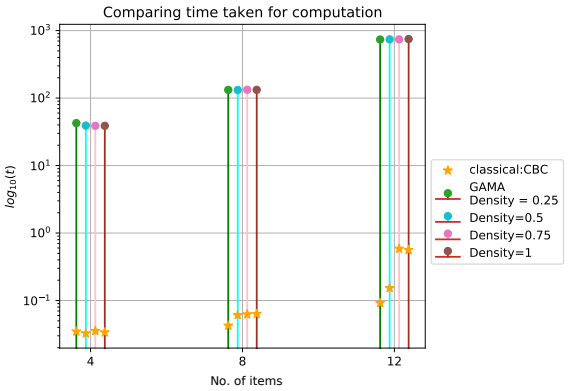
(d) Capacity 4 $\in [(0.80 \times \Sigma w_j), (0.95 \times \Sigma w_j) + 1]$

Figure 9: Comparing optimal solution obtained from GAMA with one obtained from the Classical solver CBC for different Number of Items and densities for various capacities when Augmentation Method-2 was used

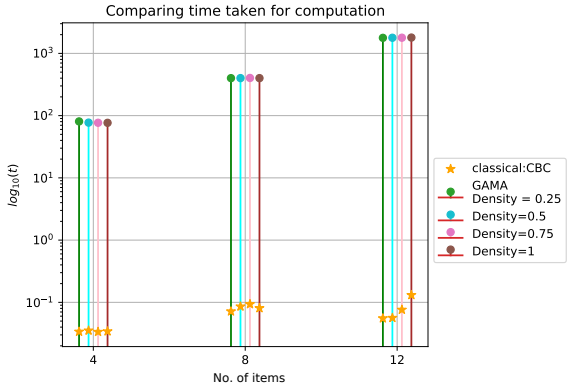
8 Formulation 2: Time taken for computation of Optimal Solution from GAMA



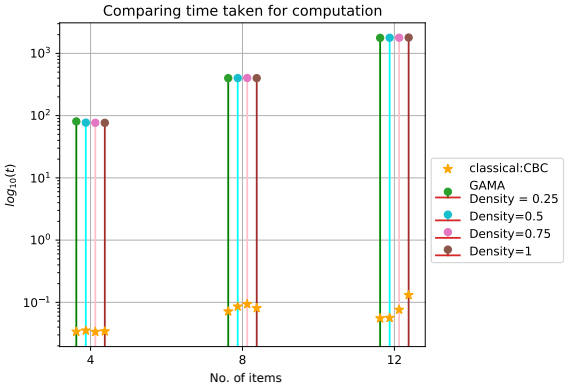
(a) Capacity 1 $\in [50, \Sigma w_j + 1]$



(b) Capacity 2 $\in [(0.25 \times \Sigma w_j) + 1, (0.40 \times \Sigma w_j) + 1]$

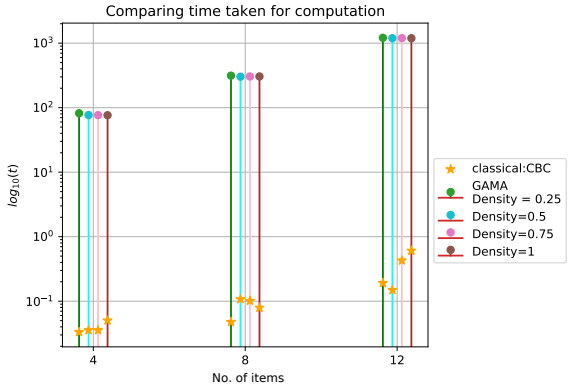


(c) Capacity 3 = $int(0.5(\Sigma w_j))$

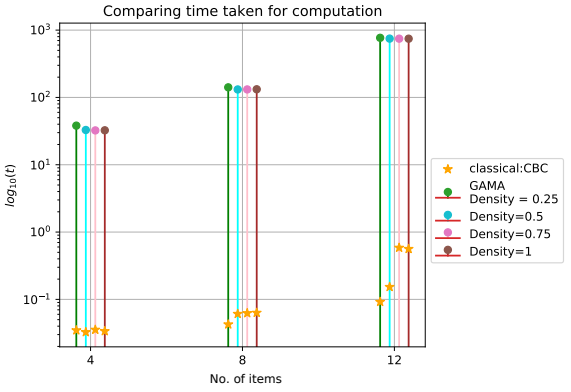


(d) Capacity 4 $\in [(0.80 \times \Sigma w_j), (0.95 \times \Sigma w_j) + 1]$

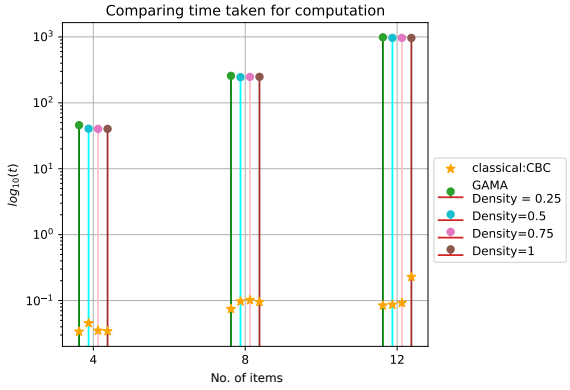
Figure 10: Comparing time taken to get optimal solution from GAMA when augmentation method 1 was used with that for Classical solver CBC for different Number of Items and densities for four different capacity ranges.



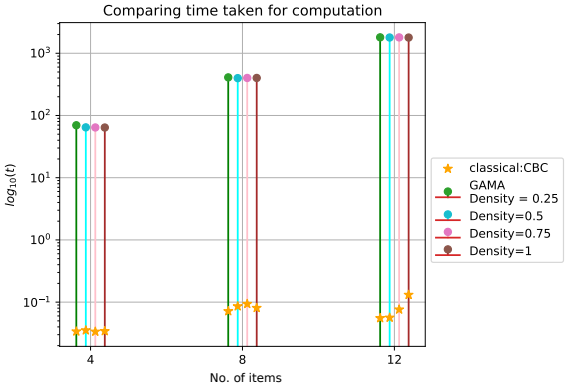
(a) Capacity 1 $\in [50, \Sigma w_j + 1]$



(b) Capacity 2 $\in [(0.25 \times \Sigma w_j) + 1, (0.40 \times \Sigma w_j) + 1]$



(c) Capacity 3 = $int(0.5(\Sigma w_j))$



(d) Capacity 4 $\in [(0.80 \times \Sigma w_j), (0.95 \times \Sigma w_j) + 1]$

Figure 11: Comparing time taken to get optimal solution from GAMA when augmentation method 2 was used with that for Classical solver CBC for different Number of items and densities for four different capacity ranges.

9 Results

9.1 ILP formulation of QKP solved as a QUBO by Simulated Annealing

1. For lower values of N, the optimum value of profit obtained from QUBO solved by annealing matches well with those obtained from the commercial solver CBC (fig. 1). However, for higher value of N, the results do not match with CBC's. The size of A matrix is quadratically dependent on N. As N increases the QUBO being fed to Simulating Annealing becomes large, hence becomes more susceptible to errors.

2. For a given N and penalty value, the anneal time was more or less the same for all densities and for a fixed number of samples (fig. 2a, 2c). This is because the QUBO depends linearly on $c^\top$ (where density decides the number of non zero elements of $c^\top$) and quadratically on A. The A matrix depends only on N even for different densities. So the complexity of QUBO is largely decided by N. Hence the time taken stays flat across different densities.

3. For a given N and fixed number of samples, for most of the penalty values, we observed a trend that as we increase the density of the profit matrix, anneal time first decreases then increases. (fig. 2b, 2d)

4. For (N,density) = (5, 1), (10, 0.5), (10, 0.75) and (10, 1.0) (fig. 3a and 3b) we observed that simulated annealing takes lesser time than CBC solver. For the rest of the cases, CBC solver solves the problem faster (fig. 3).

9.2 GAMA formulation-1:

1. For lower values of N (4,8,12 and 16), for all four capacity ranges considered, the optimum value of profit obtained from GAMA for both the augmentation method agrees well with the one obtained from the commercial solver CBC (fig. 4 and 5)

2. The classical CBC solver solves for $len(\overrightarrow{x})$ number of variables whereas the number of variables QUBO of GAMA solves for is $len(\overrightarrow{x}|\overrightarrow{y}|\overrightarrow{s_2}|\overrightarrow{s_3}|\overrightarrow{s_4}|\overrightarrow{s_1})$. The $\overrightarrow{s_2}, \overrightarrow{s_3}, \overrightarrow{s_4}$ are slack variables and $\overrightarrow{s_1}$ arise due to binarisation. Hence, there is a considerable increase in the time taken by GAMA in comparison to the CBC solver. (fig. 6 and 7)

3. For a given N and capacity, GAMA takes almost the same time for profit matrix of different densities, whereas this is not the case for the classical solver CBC. (fig. 6 and 7) This is highly desirable in industrial applications where you want your program to solve within a certain time frame, rather than wait for days without knowing if a solution would even exist as in case for CBC Solver.

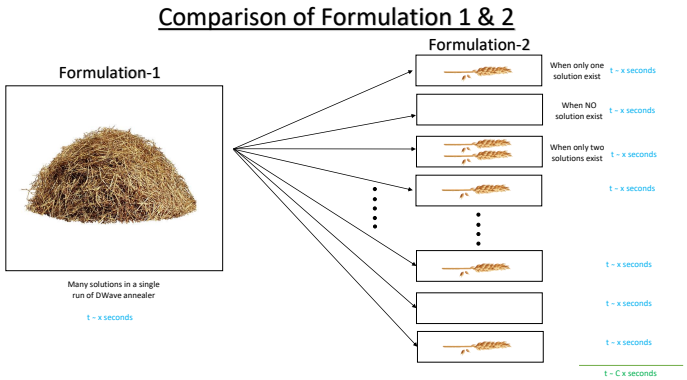


Figure 12: Comparison of solutions (analogy to hay) for Formulation-1 and 2. In Formulation-1 we obtain many solutions in one run which let's say takes x seconds. Whereas in Formulation-2 we need to solve the same problem C (total capacity) number of times where each take x seconds. In Formulation-2 the program redundantly takes x seconds to solve a problem for those $\tilde{c}$ for which no solution exists.

9.3 GAMA formulation-2:

1. In this, for a given C, there definitely are some $\tilde{c}$ for which no feasible solution exists but one usually does not know them a priori. Time spent in solving QUBO to get feasible solutions for such $\tilde{c}$ is the major issue with this approach.

2. There is a possibility of having some problem instances (i.e. a particular Profit matrix, weight vector and knapsack capacity) for which there is only a single feasible solutions for $\tilde{c}$ for which solution exists. In such a case, one cannot find graver vectors for augmentation purpose by taking difference. However in such a case one can still find feasible solution by evaluating profit function at the solutions so obtained and choose the one for which the profit value is maximum.

10 Conclusion

We reformulated the Quadratic Knapsack Problem as an ILP problem and solved it (a) by formulating it as a standard QUBO and (b) GAMA, and for different instances of the problem, the optimum values and the time taken to obtain them were compared with those for commercial solver: CBC. These two approaches involved solving some QUBO, which we tried to solve using both simulated annealing solver and quantum annealing. Although the results from the simulated annealing solver were good for most of the instances that we considered, this was not the case for the ones we obtained from quantum annealing.

11 Acknowledgement

We thank IIT-Madras and CMU for offering this course on classical and quantum approaches to solve optimisation problems. The project was the most exciting component of this course as for solving this problem of quadratic knapsack, almost all of the concepts that we learned in this course were used, hence giving us quite a hands-on experience on application of quantum computing.

References

[1] H. Alghassi, R. Dridi, and S. Tayur. Graver Bases via Quantum Annealing with Application to Non-Linear Integer Programs. pages 1–39, feb 2019. ISSN 2331-8422. URL http://arxiv.org/abs/1902.04215.

[2] D. E. Bernal, S. Tayur, and D. Venturelli. Quantum Integer Programming (QuIP) 47-779: Lecture Notes. 2020. URL http://arxiv.org/abs/2012.11382.

[3] A. Caprara, D. Pisinger, and P. Toth. Exact Solution of the Quadratic Knapsack Problem. *INFORMS Journal on Computing*, 11(2):125–137, 1999. ISSN 1091-9856. doi: 10.1287/ijoc.11.2.125. URL https://pubsonline.informs.org/doi/pdf/10.1287/ijoc.11.2.125.

[4] G. Chapuis, H. Djidjev, G. Hahn, and G. Rizk. Finding Maximum Cliques on the D-Wave Quantum Annealer. *Journal of Signal Processing Systems*, 91(3-4):363–377, mar 2019. ISSN 1939-8018. doi: 10.1007/s11265-018-1357-8. URL http://link.springer.com/10.1007/s11265-018-1357-8.

[5] G. Gallo, P. L. Hammer, and B. Simeone. *Quadratic knapsack problems*, pages 132–149. Springer Berlin Heidelberg, Berlin, Heidelberg, 1980. ISBN 978-3-642-00802-3. doi: 10.1007/BFb0120892. URL https://doi.org/10.1007/BFb0120892.

[6] A. R. Haverly. A comparison of quantum algorithms for the maximum clique problem. URL https://scholarworks.rit.edu/cgi/viewcontent.cgi?article=11892&context=theses.

[7] H. Kellerer, U. Pferschy, and D. Pisinger. The Quadratic Knapsack Problem. In *Knapsack Problems*, volume 12, pages 349–388. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004. doi: 10.1007/978-3-540-24777-7_12. URL http://link.springer.com/10.1007/978-3-540-24777-7_12.

[8] D. Pisinger. The quadratic knapsack problem—a survey. *Discrete Applied Mathematics*, 155(5):623–648, mar 2007. ISSN 0166218X. doi: 10.1016/j.dam.2006.08.007. URL https://linkinghub.elsevier.com/retrieve/pii/S0166218X06003878.