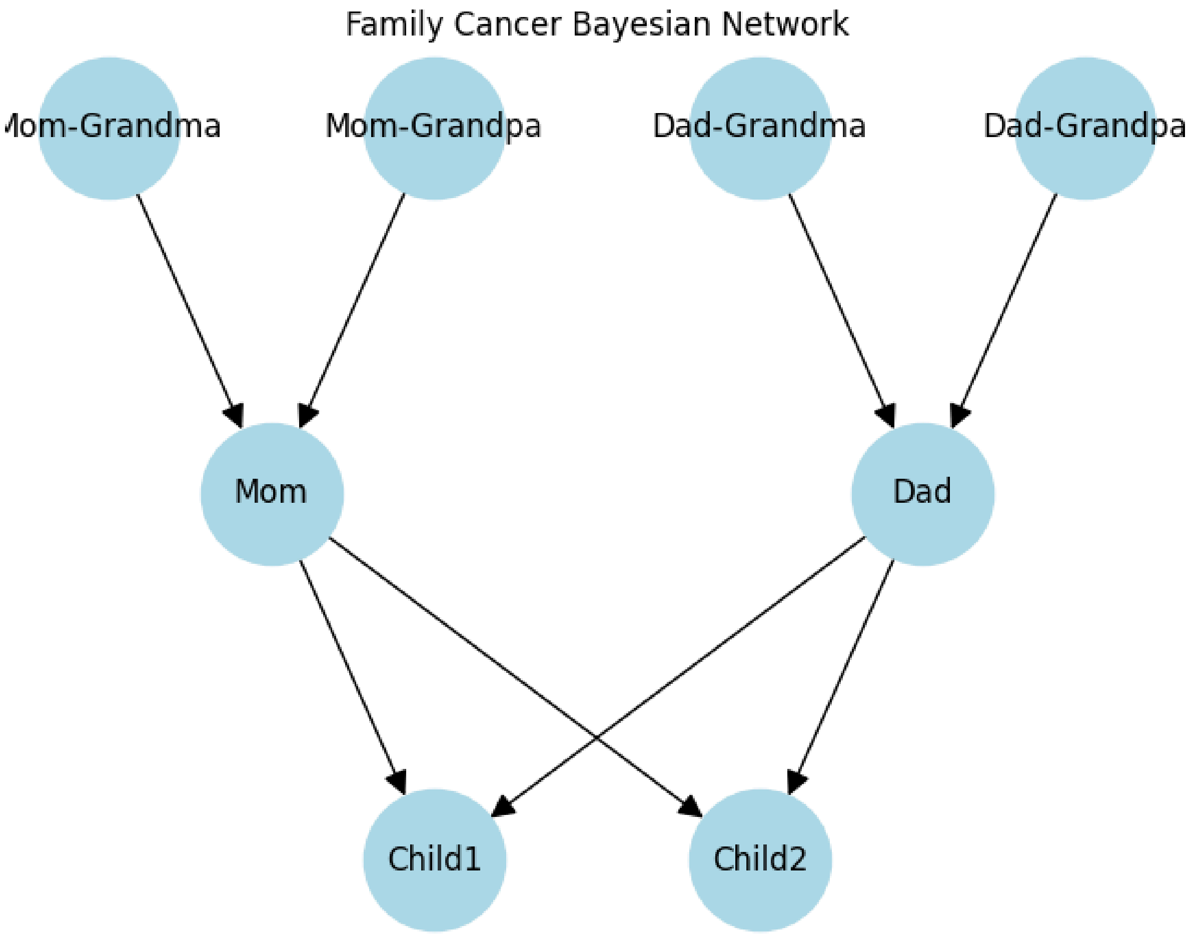
In [1]:
```python
# AA228/CS238: DMU

# BayesNet Family Cancer inference

# So far in this course you have learned a lot of theory but have not yet pu

# Lets use a single problem the entire way. Lets say we are datascientists a
# But how can we try to estimate this?

# To start, please install:
# !pip install pgmpy networkx matplotlib pydot
```

In [1]:
```python
# 2. Representation of a joint distribution with a BayesNet:

from pgmpy.models import BayesianNetwork
from pgmpy.factors.discrete import TabularCPD
import matplotlib.pyplot as plt
import networkx as nx
```

In [2]:
```python
# First lets represent the joint distribution as a directed graph below.

# Lets assume we have some data on the conditional probability of each havin

# We must write a probabilistic graphical model of a family that has a:
# mom-grandma, mom-grandpa, dad-grandma, dad-grandpa, mom, dad, child1, chil

# And we want to represent this as a graph with edges and nodes (per person)


# Define the Bayesian Network structure
model = BayesianNetwork([
    ('Mom-Grandma', 'Mom'),
    ('Mom-Grandpa', 'Mom'),
    ('Dad-Grandma', 'Dad'),
    ('Dad-Grandpa', 'Dad'),
    ('Mom', 'Child1'),
    ('Dad', 'Child1'),
    ('Mom', 'Child2'),
    ('Dad', 'Child2'),
])

# Define CPDs (Conditional Probability Distributions)
# Grandparents' prior probabilities of having cancer
prior_on_cancer = 0.5

cpd_MG = TabularCPD(variable='Mom-Grandma', variable_card=2, values=[[prior_
                                                                      [1-pric

cpd_MP = TabularCPD(variable='Mom-Grandpa', variable_card=2, values=[[prior_
                                                                      [1-pric

cpd_DG = TabularCPD(variable='Dad-Grandma', variable_card=2, values=[[prior_
                                                                      [1-pric
```

```
cpd_DP = TabularCPD(variable='Dad-Grandpa', variable_card=2, values=[[prior_
                                                                      [1-pric

# Lets explain this a bit, first, we name the node. Then we assign the varia

cancer_probability_given_both_parents_have_cancer = 0.9999
cancer_probability_given_one_parent_has_cancer = 0.7
cancer_probability_given_no_parent_has_cancer = 0.01
# Mom's probability of having cancer given her parents' cancer status
cpd_M = TabularCPD(variable='Mom', variable_card=2,
                   evidence=['Mom-Grandma', 'Mom-Grandpa'], evidence_card=[2
                   values=[
                       [1-cancer_probability_given_no_parent_has_cancer, 1-c
                       [cancer_probability_given_no_parent_has_cancer, cance
                   ])

# Now lets explain this a bit, first, we assign the variable cardinality (va
# Then we assign the conditional probability that we have cancer given what

# Dad's probability of having cancer given his parents' cancer status
cpd_D = TabularCPD(variable='Dad', variable_card=2,
                   evidence=['Dad-Grandma', 'Dad-Grandpa'], evidence_card=[2
                   values=[
                       [1-cancer_probability_given_no_parent_has_cancer, 1-c
                       [cancer_probability_given_no_parent_has_cancer, cance
                   ])

# Children's probability of having cancer given their parents' cancer status
cpd_C1 = TabularCPD(variable='Child1', variable_card=2,
                    evidence=['Mom', 'Dad'], evidence_card=[2, 2],
                   values=[
                       [1-cancer_probability_given_no_parent_has_cancer, 1-c
                       [cancer_probability_given_no_parent_has_cancer, cance
                   ])

cpd_C2 = TabularCPD(variable='Child2', variable_card=2,
                    evidence=['Mom', 'Dad'], evidence_card=[2, 2],
                   values=[
                       [1-cancer_probability_given_no_parent_has_cancer, 1-c
                       [cancer_probability_given_no_parent_has_cancer, cance
                   ])

# Add CPDs to the model
model.add_cpds(cpd_MG, cpd_MP, cpd_DG, cpd_DP, cpd_M, cpd_D, cpd_C1, cpd_C2)

# Check if the model is valid
assert model.check_model()

# Visualize the Bayesian Network using networkx

# Create a directed graph from the model
G = nx.DiGraph()

# Add nodes and edges from the Bayesian network
G.add_nodes_from(model.nodes())
G.add_edges_from(model.edges())
```

```
# Manually specify positions for a hierarchical layout
pos = {
    'Mom-Grandma': (1, 3),
    'Mom-Grandpa': (3, 3),
    'Dad-Grandma': (5, 3),
    'Dad-Grandpa': (7, 3),
    'Mom': (2, 2),
    'Dad': (6, 2),
    'Child1': (3, 1),
    'Child2': (5, 1),
}

# Draw the graph
nx.draw(G, pos, with_labels=True, node_size=3000, node_color='lightblue', ar
plt.title("Family Cancer Bayesian Network")
plt.show()
```



In [3]:
```
# 2. Inference:

# Next, now that we have our graph, lets ask the graph a question!
# Given observing that mom-grandma has cancer, determine the change in proba

from pgmpy.inference import VariableElimination

# Perform inference using Variable Elimination
infer = VariableElimination(model)
```

```
prior_prob = infer.query(variables=['Child1'], evidence={})
print("Prior Probability of Child1 having cancer with no observation:")
print(prior_prob)

posterior_prob = infer.query(variables=['Child1'], evidence={'Mom-Grandma':
print("\nPosterior Probability of Child1 having cancer given we observe Mom-
print(posterior_prob)

posterior_prob = infer.query(variables=['Child1'], evidence={'Mom-Grandma':
print("\nPosterior Probability of Child1 having cancer given we observe Mom-
print(posterior_prob)

posterior_prob = infer.query(variables=['Child1'], evidence={'Mom-Grandma':
print("\nPosterior Probability of Child1 having cancer given we observe all
print(posterior_prob)

posterior_prob = infer.query(variables=['Child1'], evidence={'Mom-Grandma':
print("\nPosterior Probability of Child1 having cancer given we observe all
print(posterior_prob)
```

```
Prior Probability of Child1 having cancer with no observation:
+-----------+---------------+
| Child1    |   phi(Child1) |
+===========+===============+
| Child1(0) |        0.4296 |
+-----------+---------------+
| Child1(1) |        0.5704 |
+-----------+---------------+

Posterior Probability of Child1 having cancer given we observe Mom-Grandma h
as cancer:
+-----------+---------------+
| Child1    |   phi(Child1) |
+===========+===============+
| Child1(0) |        0.3489 |
+-----------+---------------+
| Child1(1) |        0.6511 |
+-----------+---------------+

Posterior Probability of Child1 having cancer given we observe Mom-Grandma d
oes NOT have cancer:
+-----------+---------------+
| Child1    |   phi(Child1) |
+===========+===============+
| Child1(0) |        0.5103 |
+-----------+---------------+
| Child1(1) |        0.4897 |
+-----------+---------------+

Posterior Probability of Child1 having cancer given we observe all 4 grandpa
rents have cancer:
+-----------+---------------+
| Child1    |   phi(Child1) |
+===========+===============+
| Child1(0) |        0.3000 |
+-----------+---------------+
| Child1(1) |        0.7000 |
+-----------+---------------+

Posterior Probability of Child1 having cancer given we observe all 4 grandpa
rents don't have cancer:
+-----------+---------------+
| Child1    |   phi(Child1) |
+===========+===============+
| Child1(0) |        0.9763 |
+-----------+---------------+
| Child1(1) |        0.0237 |
+-----------+---------------+
```

In [4]:
```
# as you can see, our probability of Child1 having cancer can range drastica
```

In [5]:
```
# 3. Parameter Learning:

# Now, where did these conditional probability tables come from? We just mad
# But lets get some data on families around the world over the last 100 year
```

```python
# First lets generate some fake data.

import numpy as np
import pandas as pd
from pgmpy.models import BayesianNetwork
from pgmpy.estimators import MaximumLikelihoodEstimator


def create_data_set(num_families = 100,
                    true_cancer_probability_given_both_parents_have_cancer =
                    true_cancer_probability_given_one_parent_has_cancer = 0.
                    true_cancer_probability_given_no_parent_has_cancer = 0.0
                    true_prior_on_cancer = 0.5):

    data = []

    # Grandparents' prior probabilities of having cancer
    prob_MG = [1-true_prior_on_cancer, true_prior_on_cancer]  # [No Cancer,
    prob_MP = [1-true_prior_on_cancer, true_prior_on_cancer]
    prob_DG = [1-true_prior_on_cancer, true_prior_on_cancer]
    prob_DP = [1-true_prior_on_cancer, true_prior_on_cancer]

    # Mom's probability of having cancer given her parents' cancer status
    cpd_M_values = np.array(
        (
            [1-true_cancer_probability_given_no_parent_has_cancer, 1-true_c
        [true_cancer_probability_given_no_parent_has_cancer, true_cancer_pr

        )
    )

    # Dad's probability of having cancer given his parents' cancer status
    cpd_D_values = np.array(
        (
            [1-true_cancer_probability_given_no_parent_has_cancer, 1-true_c
        [true_cancer_probability_given_no_parent_has_cancer, true_cancer_pr

        )
    )

    # Children's probability of having cancer given their parents' cancer st
    cpd_C_values = np.array(
        (
            [1-true_cancer_probability_given_no_parent_has_cancer, 1-true_c
        [true_cancer_probability_given_no_parent_has_cancer, true_cancer_pr

        )
    )

    for _ in range(num_families):
        family = {}

        # Generate grandparents' cancer status
        family['Mom-Grandma'] = np.random.choice([0,1], p=prob_MG)
        family['Mom-Grandpa'] = np.random.choice([0,1], p=prob_MP)
        family['Dad-Grandma'] = np.random.choice([0,1], p=prob_DG)
        family['Dad-Grandpa'] = np.random.choice([0,1], p=prob_DP)

        # Mom's cancer status
        mg = family['Mom-Grandma']
        mp = family['Mom-Grandpa']
        index_M = mg * 2 + mp  # Convert binary states to index (00->0, 01->
        prob_M = cpd_M_values[:, index_M]
        family['Mom'] = np.random.choice([0,1], p=prob_M)

        # Dad's cancer status
        dg = family['Dad-Grandma']
        dp = family['Dad-Grandpa']
        index_D = dg * 2 + dp
        prob_D = cpd_D_values[:, index_D]
        family['Dad'] = np.random.choice([0,1], p=prob_D)

        # Child1's cancer status
        mom = family['Mom']
        dad = family['Dad']
        index_C = mom * 2 + dad
        prob_C1 = cpd_C_values[:, index_C]
        family['Child1'] = np.random.choice([0,1], p=prob_C1)

        # Child2's cancer status
        prob_C2 = cpd_C_values[:, index_C]  # Same as Child1
        family['Child2'] = np.random.choice([0,1], p=prob_C2)

        data.append(family)

    # Convert to pandas DataFrame
    df = pd.DataFrame(data)
    return df

example_dataset = create_data_set(num_families = 10)
print("Here is an example dataset of 10 families in the world, where 0=did r
example_dataset
```

```
Here is an example dataset of 10 families in the world, where 0=did not have
cancer, 1=did have cancer
```

Out[5]:

	Mom-Grandma	Mom-Grandpa	Dad-Grandma	Dad-Grandpa	Mom	Dad	Child1	Child2
0	0	1	0	1	0	1	0	1
1	1	0	0	0	1	0	0	1
2	1	0	1	0	1	1	1	1
3	0	1	1	1	1	1	1	1
4	1	0	1	0	1	1	1	1
5	1	0	0	0	0	0	0	0
6	0	0	1	1	0	1	1	1
7	0	1	1	0	0	1	0	0
8	1	1	1	1	1	1	1	1
9	1	0	0	0	0	0	0	0

In [6]:

```python
# Now lets use this data to estimate these values with MLE:
model.fit(example_dataset, estimator=MaximumLikelihoodEstimator)

# Get the estimated CPDs (conditional probability distributions)
estimated_cpds = model.get_cpds()

# Print the estimated CPDs
for cpd in estimated_cpds:
    print(cpd)
    print("\n")
```

```
WARNING:pgmpy:Replacing existing CPD for Mom-Grandma
WARNING:pgmpy:Replacing existing CPD for Mom
WARNING:pgmpy:Replacing existing CPD for Mom-Grandpa
WARNING:pgmpy:Replacing existing CPD for Dad-Grandma
WARNING:pgmpy:Replacing existing CPD for Dad
WARNING:pgmpy:Replacing existing CPD for Dad-Grandpa
WARNING:pgmpy:Replacing existing CPD for Child1
WARNING:pgmpy:Replacing existing CPD for Child2
```

```
+----------------+-----+
| Mom-Grandma(0) | 0.4 |
+----------------+-----+
| Mom-Grandma(1) | 0.6 |
+----------------+-----+


+----------------+-----+
| Mom-Grandpa(0) | 0.6 |
+----------------+-----+
| Mom-Grandpa(1) | 0.4 |
+----------------+-----+


+----------------+-----+
| Dad-Grandma(0) | 0.4 |
+----------------+-----+
| Dad-Grandma(1) | 0.6 |
+----------------+-----+


+----------------+-----+
| Dad-Grandpa(0) | 0.6 |
+----------------+-----+
| Dad-Grandpa(1) | 0.4 |
+----------------+-----+


+-------------+----------------+-----+----------------+----------------+
| Mom-Grandma | Mom-Grandma(0) | ... | Mom-Grandma(1) | Mom-Grandma(1) |
+-------------+----------------+-----+----------------+----------------+
| Mom-Grandpa | Mom-Grandpa(0) | ... | Mom-Grandpa(0) | Mom-Grandpa(1) |
+-------------+----------------+-----+----------------+----------------+
| Mom(0)      | 1.0            | ... | 0.4            | 0.0            |
+-------------+----------------+-----+----------------+----------------+
| Mom(1)      | 0.0            | ... | 0.6            | 1.0            |
+-------------+----------------+-----+----------------+----------------+


+-------------+----------------+-----+----------------+----------------+
| Dad-Grandma | Dad-Grandma(0) | ... | Dad-Grandma(1) | Dad-Grandma(1) |
+-------------+----------------+-----+----------------+----------------+
| Dad-Grandpa | Dad-Grandpa(0) | ... | Dad-Grandpa(0) | Dad-Grandpa(1) |
+-------------+----------------+-----+----------------+----------------+
| Dad(0)      | 1.0            | ... | 0.0            | 0.0            |
+-------------+----------------+-----+----------------+----------------+
| Dad(1)      | 0.0            | ... | 1.0            | 1.0            |
+-------------+----------------+-----+----------------+----------------+


+-----------+--------+--------+--------------------+--------+
| Dad       | Dad(0) | Dad(0) | Dad(1)             | Dad(1) |
+-----------+--------+--------+--------------------+--------+
| Mom       | Mom(0) | Mom(1) | Mom(0)             | Mom(1) |
+-----------+--------+--------+--------------------+--------+
| Child1(0) | 1.0    | 1.0    | 0.6666666666666666 | 0.0    |
+-----------+--------+--------+--------------------+--------+
| Child1(1) | 0.0    | 0.0    | 0.3333333333333333 | 1.0    |
+-----------+--------+--------+--------------------+--------+


+-----------+--------+--------+--------------------+--------+
| Dad       | Dad(0) | Dad(0) | Dad(1)             | Dad(1) |
+-----------+--------+--------+--------------------+--------+
| Mom       | Mom(0) | Mom(1) | Mom(0)             | Mom(1) |
+-----------+--------+--------+--------------------+--------+
| Child2(0) | 1.0    | 0.0    | 0.3333333333333333 | 0.0    |
+-----------+--------+--------+--------------------+--------+
| Child2(1) | 0.0    | 1.0    | 0.6666666666666666 | 1.0    |
+-----------+--------+--------+--------------------+--------+
```

In [7]:
```
# now lets go back and do inference again with these new CPDs
```

In [8]:
```
# 2. Inference:

# Next, lets ask the graph a question!
# Given observing that mom-grandma has cancer, determine the change in proba

from pgmpy.inference import VariableElimination

# Perform inference using Variable Elimination
infer = VariableElimination(model)

prior_prob = infer.query(variables=['Child1'], evidence={})
print("Prior Probability of Child1 having cancer with no observation:")
print(prior_prob)

posterior_prob = infer.query(variables=['Child1'], evidence={'Mom-Grandma':
print("\nPosterior Probability of Child1 having cancer given we observe Mom-
print(posterior_prob)

posterior_prob = infer.query(variables=['Child1'], evidence={'Mom-Grandma':
print("\nPosterior Probability of Child1 having cancer given we observe Mom-
print(posterior_prob)

posterior_prob = infer.query(variables=['Child1'], evidence={'Mom-Grandma':
print("\nPosterior Probability of Child1 having cancer given we observe all
print(posterior_prob)

posterior_prob = infer.query(variables=['Child1'], evidence={'Mom-Grandma':
print("\nPosterior Probability of Child1 having cancer given we observe all
print(posterior_prob)
```

```
Prior Probability of Child1 having cancer with no observation:
+-----------+---------------+
| Child1    |   phi(Child1) |
+===========+===============+
| Child1(0) |        0.4886 |
+-----------+---------------+
| Child1(1) |        0.5114 |
+-----------+---------------+

Posterior Probability of Child1 having cancer given we observe Mom-Grandma h
as cancer:
+-----------+---------------+
| Child1    |   phi(Child1) |
+===========+===============+
| Child1(0) |        0.3616 |
+-----------+---------------+
| Child1(1) |        0.6384 |
+-----------+---------------+

Posterior Probability of Child1 having cancer given we observe Mom-Grandma d
oes NOT have cancer:
+-----------+---------------+
| Child1    |   phi(Child1) |
+===========+===============+
| Child1(0) |        0.6791 |
+-----------+---------------+
| Child1(1) |        0.3209 |
+-----------+---------------+

Posterior Probability of Child1 having cancer given we observe all 4 grandpa
rents have cancer:
+-----------+---------------+
| Child1    |   phi(Child1) |
+===========+===============+
| Child1(0) |        0.0000 |
+-----------+---------------+
| Child1(1) |        1.0000 |
+-----------+---------------+

Posterior Probability of Child1 having cancer given we observe all 4 grandpa
rents don't have cancer:
+-----------+---------------+
| Child1    |   phi(Child1) |
+===========+===============+
| Child1(0) |        1.0000 |
+-----------+---------------+
| Child1(1) |        0.0000 |
+-----------+---------------+
```

In [9]:
```
# Very different! As you can see the probability of the child getting cancer
# Now this is because we have used an MLE estimate which is not great when r
# Given that we have only seen 10 families, perhaps we are overestimating ho
```

In [10]:
```
# Since we do not have much data, lets try to assign a "prior" on our parame

from pgmpy.estimators import BayesianEstimator
```

```python
# Now lets use this data to estimate these values with MLE:
model.fit(example_dataset, estimator=BayesianEstimator)

# Get the estimated CPDs (conditional probability distributions)
estimated_cpds = model.get_cpds()

# Print the estimated CPDs
for cpd in estimated_cpds:
    print(cpd)
    print("\n")
```

```
WARNING:pgmpy:Replacing existing CPD for Mom-Grandma
WARNING:pgmpy:Replacing existing CPD for Mom
WARNING:pgmpy:Replacing existing CPD for Mom-Grandpa
WARNING:pgmpy:Replacing existing CPD for Dad-Grandma
WARNING:pgmpy:Replacing existing CPD for Dad
WARNING:pgmpy:Replacing existing CPD for Dad-Grandpa
WARNING:pgmpy:Replacing existing CPD for Child1
WARNING:pgmpy:Replacing existing CPD for Child2
```

```
+----------------+----------+
| Mom-Grandma(0) | 0.433333 |
+----------------+----------+
| Mom-Grandma(1) | 0.566667 |
+----------------+----------+


+----------------+----------+
| Mom-Grandpa(0) | 0.566667 |
+----------------+----------+
| Mom-Grandpa(1) | 0.433333 |
+----------------+----------+


+----------------+----------+
| Dad-Grandma(0) | 0.433333 |
+----------------+----------+
| Dad-Grandma(1) | 0.566667 |
+----------------+----------+


+----------------+----------+
| Dad-Grandpa(0) | 0.566667 |
+----------------+----------+
| Dad-Grandpa(1) | 0.433333 |
+----------------+----------+


+-------------+--------------------+-----+--------------------+
| Mom-Grandma | Mom-Grandma(0)     | ... | Mom-Grandma(1)     |
+-------------+--------------------+-----+--------------------+
| Mom-Grandpa | Mom-Grandpa(0)     | ... | Mom-Grandpa(1)     |
+-------------+--------------------+-----+--------------------+
| Mom(0)      | 0.7222222222222222 | ... | 0.2777777777777778 |
+-------------+--------------------+-----+--------------------+
| Mom(1)      | 0.2777777777777778 | ... | 0.7222222222222222 |
+-------------+--------------------+-----+--------------------+


+-------------+---------------------+-----+---------------------+
| Dad-Grandma | Dad-Grandma(0)      | ... | Dad-Grandma(1)      |
+-------------+---------------------+-----+---------------------+
| Dad-Grandpa | Dad-Grandpa(0)      | ... | Dad-Grandpa(1)      |
+-------------+---------------------+-----+---------------------+
| Dad(0)      | 0.8529411764705882  | ... | 0.14705882352941177 |
+-------------+---------------------+-----+---------------------+
| Dad(1)      | 0.14705882352941177 | ... | 0.8529411764705882  |
+-------------+---------------------+-----+---------------------+


+-----------+---------------------+-----+---------------------+
| Dad       | Dad(0)              | ... | Dad(1)              |
+-----------+---------------------+-----+---------------------+
| Mom       | Mom(0)              | ... | Mom(1)              |
+-----------+---------------------+-----+---------------------+
| Child1(0) | 0.8076923076923077  | ... | 0.11904761904761904 |
+-----------+---------------------+-----+---------------------+
| Child1(1) | 0.19230769230769232 | ... | 0.8809523809523809  |
+-----------+---------------------+-----+---------------------+


+-----------+---------------------+-----+---------------------+
| Dad       | Dad(0)              | ... | Dad(1)              |
+-----------+---------------------+-----+---------------------+
| Mom       | Mom(0)              | ... | Mom(1)              |
+-----------+---------------------+-----+---------------------+
| Child2(0) | 0.8076923076923077  | ... | 0.11904761904761904 |
+-----------+---------------------+-----+---------------------+
| Child2(1) | 0.19230769230769232 | ... | 0.8809523809523809  |
+-----------+---------------------+-----+---------------------+
```

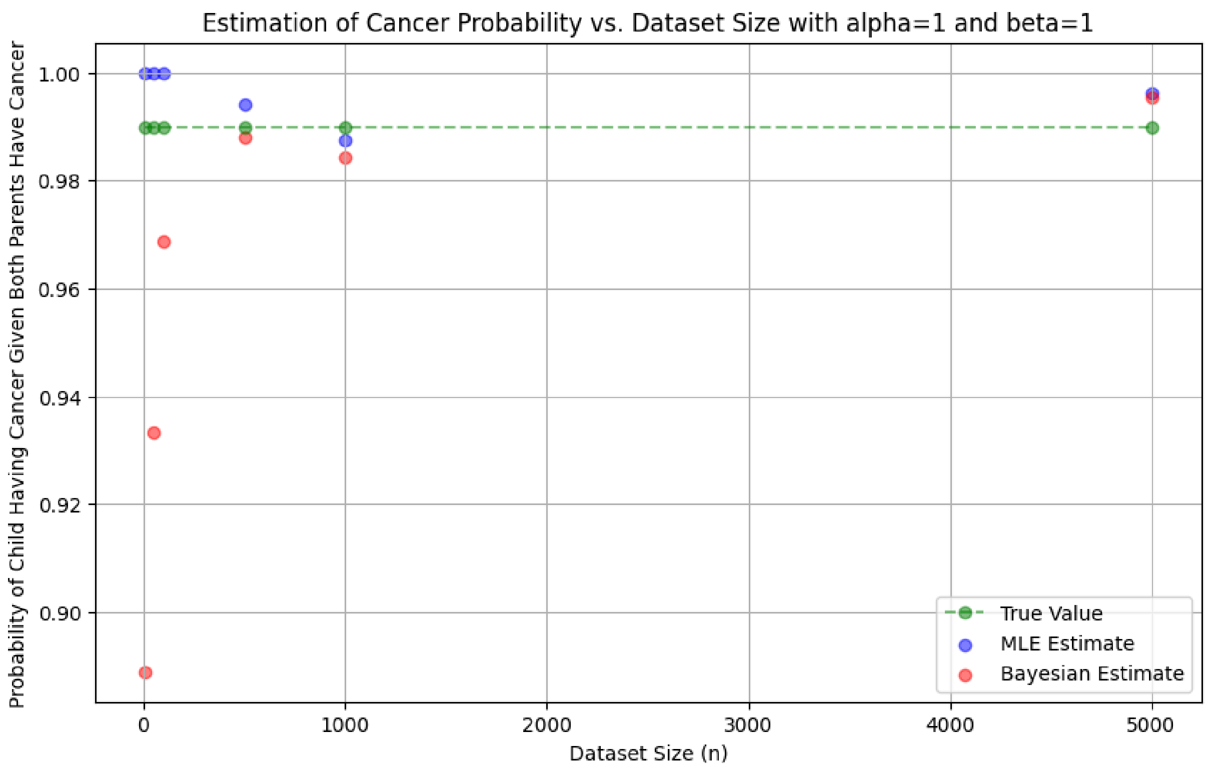
In [11]:

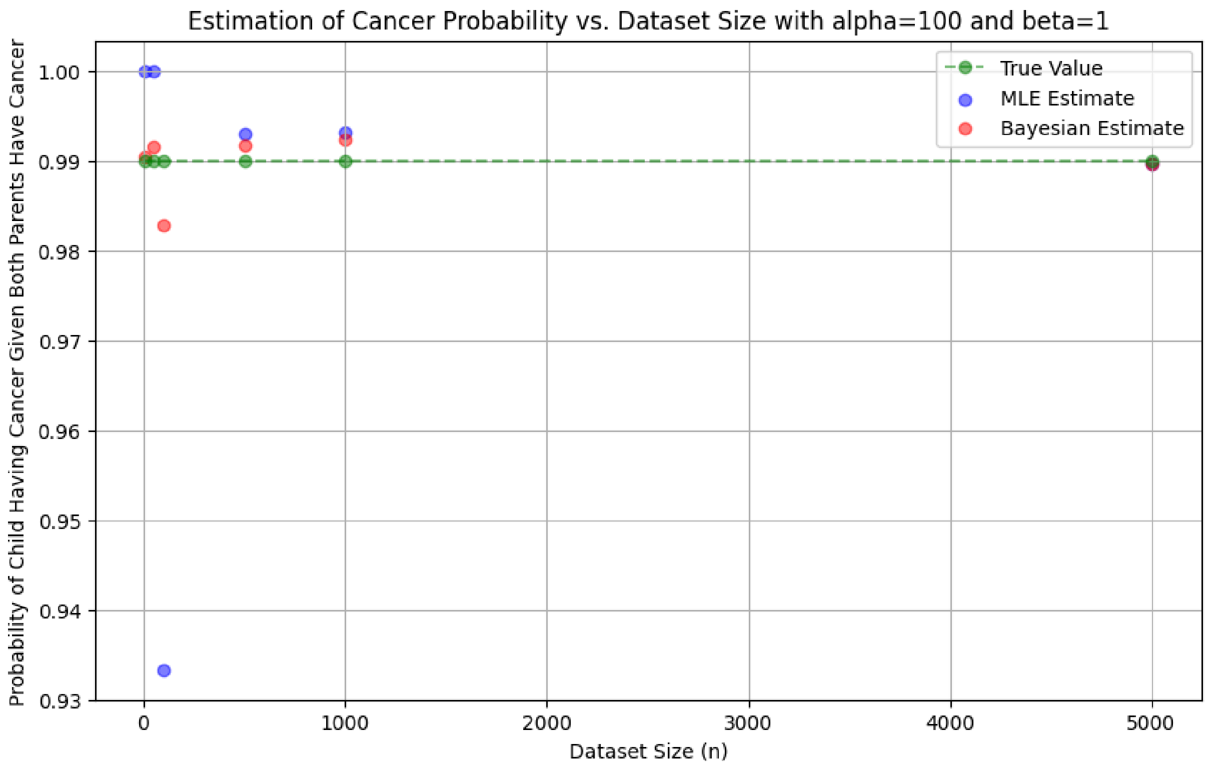
```
# Very different! Go back and see how different these values are. Now lets c

from pgmpy.inference import VariableElimination

# Perform inference using Variable Elimination
infer = VariableElimination(model)

prior_prob = infer.query(variables=['Child1'], evidence={})
print("Prior Probability of Child1 having cancer with no observation:")
print(prior_prob)

posterior_prob = infer.query(variables=['Child1'], evidence={'Mom-Grandma':
print("\nPosterior Probability of Child1 having cancer given we observe Mom-
print(posterior_prob)

posterior_prob = infer.query(variables=['Child1'], evidence={'Mom-Grandma':
print("\nPosterior Probability of Child1 having cancer given we observe Mom-
print(posterior_prob)

posterior_prob = infer.query(variables=['Child1'], evidence={'Mom-Grandma':
print("\nPosterior Probability of Child1 having cancer given we observe all
print(posterior_prob)

posterior_prob = infer.query(variables=['Child1'], evidence={'Mom-Grandma':
print("\nPosterior Probability of Child1 having cancer given we observe all
print(posterior_prob)
```

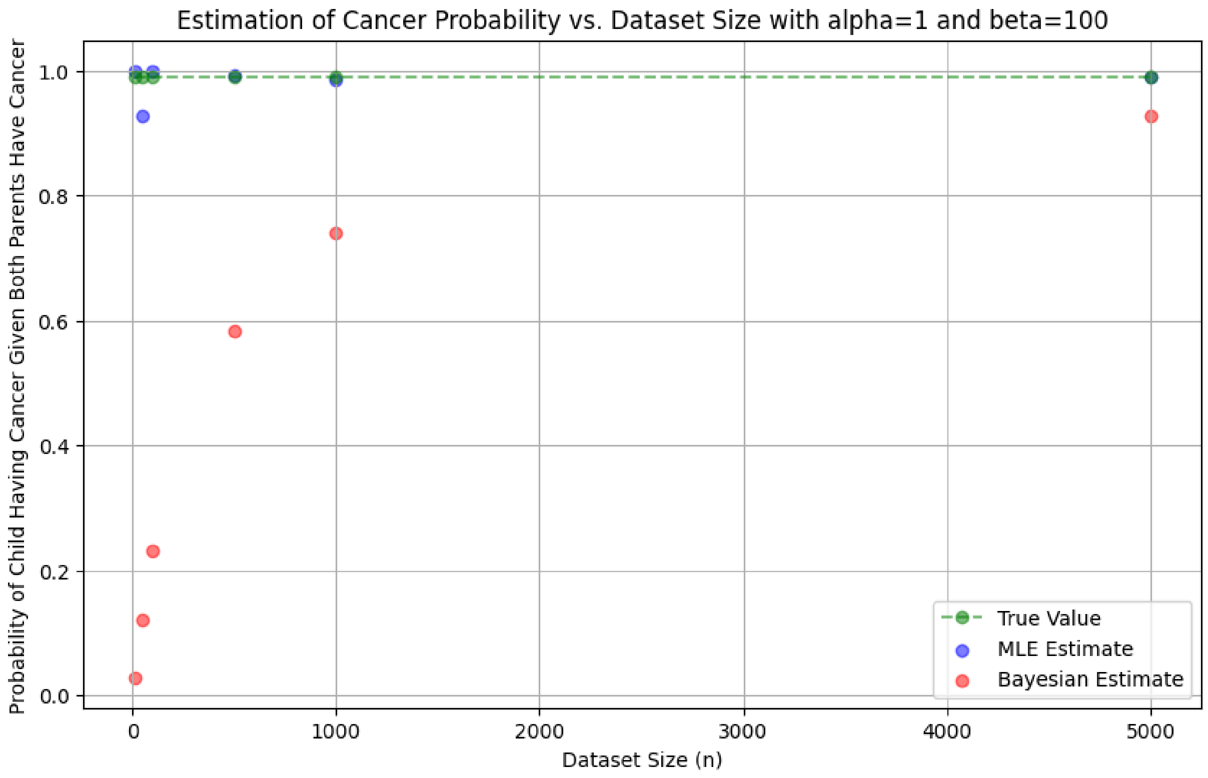
```
Prior Probability of Child1 having cancer with no observation:
+-----------+---------------+
| Child1    |   phi(Child1) |
+===========+===============+
| Child1(0) |        0.5039 |
+-----------+---------------+
| Child1(1) |        0.4961 |
+-----------+---------------+

Posterior Probability of Child1 having cancer given we observe Mom-Grandma h
as cancer:
+-----------+---------------+
| Child1    |   phi(Child1) |
+===========+===============+
| Child1(0) |        0.4547 |
+-----------+---------------+
| Child1(1) |        0.5453 |
+-----------+---------------+

Posterior Probability of Child1 having cancer given we observe Mom-Grandma d
oes NOT have cancer:
+-----------+---------------+
| Child1    |   phi(Child1) |
+===========+===============+
| Child1(0) |        0.5681 |
+-----------+---------------+
| Child1(1) |        0.4319 |
+-----------+---------------+

Posterior Probability of Child1 having cancer given we observe all 4 grandpa
rents have cancer:
+-----------+---------------+
| Child1    |   phi(Child1) |
+===========+===============+
| Child1(0) |        0.3294 |
+-----------+---------------+
| Child1(1) |        0.6706 |
+-----------+---------------+

Posterior Probability of Child1 having cancer given we observe all 4 grandpa
rents don't have cancer:
+-----------+---------------+
| Child1    |   phi(Child1) |
+===========+===============+
| Child1(0) |        0.7391 |
+-----------+---------------+
| Child1(1) |        0.2609 |
+-----------+---------------+
```

In [12]:
```
# now those estimates are much more reasonable right?
```

In [13]:
```
# But who is right or wrong? When should I use MLE vs. Bayesian Learning??

# As you can see, our choice to use MLE vs. Bayesian Learning should be base
# Intuitively we can probably guess that if both of your parents have cancer
# Maybe I subjectively believe that its really high, like 99%, so I will gue
# But maybe you believe the opposite! And you think its close to 0 still.. a
# Who is right? The frequentist (mle), bayesian me (with Beta(100,1), or bay

# NOTE: in this example our "true" true_cancer_probability_given_both_parent

# Below we observe different sizes of datasets and how close our estimate is

# True parameters
true_cancer_probability_given_both_parents_have_cancer = 0.99
true_cancer_probability_given_one_parent_has_cancer = 0.6
true_cancer_probability_given_no_parent_has_cancer = 0.01
true_prior_on_cancer = 0.5

# Dataset sizes to test
n_list = [10, 50, 100, 500, 1000, 5000]

# Lists to store estimates


beta_distributions = [(1,1), (100,1), (1,100)]

for (a0,b0) in beta_distributions:
    # Prior parameters for Bayesian estimation (Beta distribution)
    n_values = []
    mle_estimates = []
    bayes_estimates = []
    for n in n_list:
        # Generate data
        df = create_data_set(
            num_families=n,
            true_cancer_probability_given_both_parents_have_cancer=true_canc
            true_cancer_probability_given_one_parent_has_cancer=true_cancer_
            true_cancer_probability_given_no_parent_has_cancer=true_cancer_p
            true_prior_on_cancer=true_prior_on_cancer
        )

        # Filter data where both parents have cancer
        both_parents_have_cancer = df[(df['Mom'] == 1) & (df['Dad'] == 1)]
        total_trials = len(both_parents_have_cancer)

        if total_trials > 0:
            successes = both_parents_have_cancer['Child1'].sum()  # Number o
            failures = total_trials - successes

            # MLE estimate
            mle_estimate = successes / total_trials

            # Bayesian estimate
            a_post = a0 + successes
            b_post = b0 + failures
            bayes_estimate = a_post / (a_post + b_post)

            n_values.append(n)
            mle_estimates.append(mle_estimate)
            bayes_estimates.append(bayes_estimate)
        else:
            print(f"For n = {n}, no instances where both parents have cancer
    # Plotting the estimates
    plt.figure(figsize=(10, 6))

    # Plot the true value
    plt.plot(n_values, [true_cancer_probability_given_both_parents_have_canc
             label='True Value', marker='o', linestyle='--', color='green',

    # Plot the MLE estimates
    plt.scatter(n_values, mle_estimates, label='MLE Estimate', marker='o', l

    # Plot the Bayesian estimates
    plt.scatter(n_values, bayes_estimates, label='Bayesian Estimate', marker

    plt.xlabel('Dataset Size (n)')
    plt.ylabel('Probability of Child Having Cancer Given Both Parents Have C
    plt.title(f'Estimation of Cancer Probability vs. Dataset Size with alpha
    plt.legend()
    plt.grid(True)
    plt.show()
```

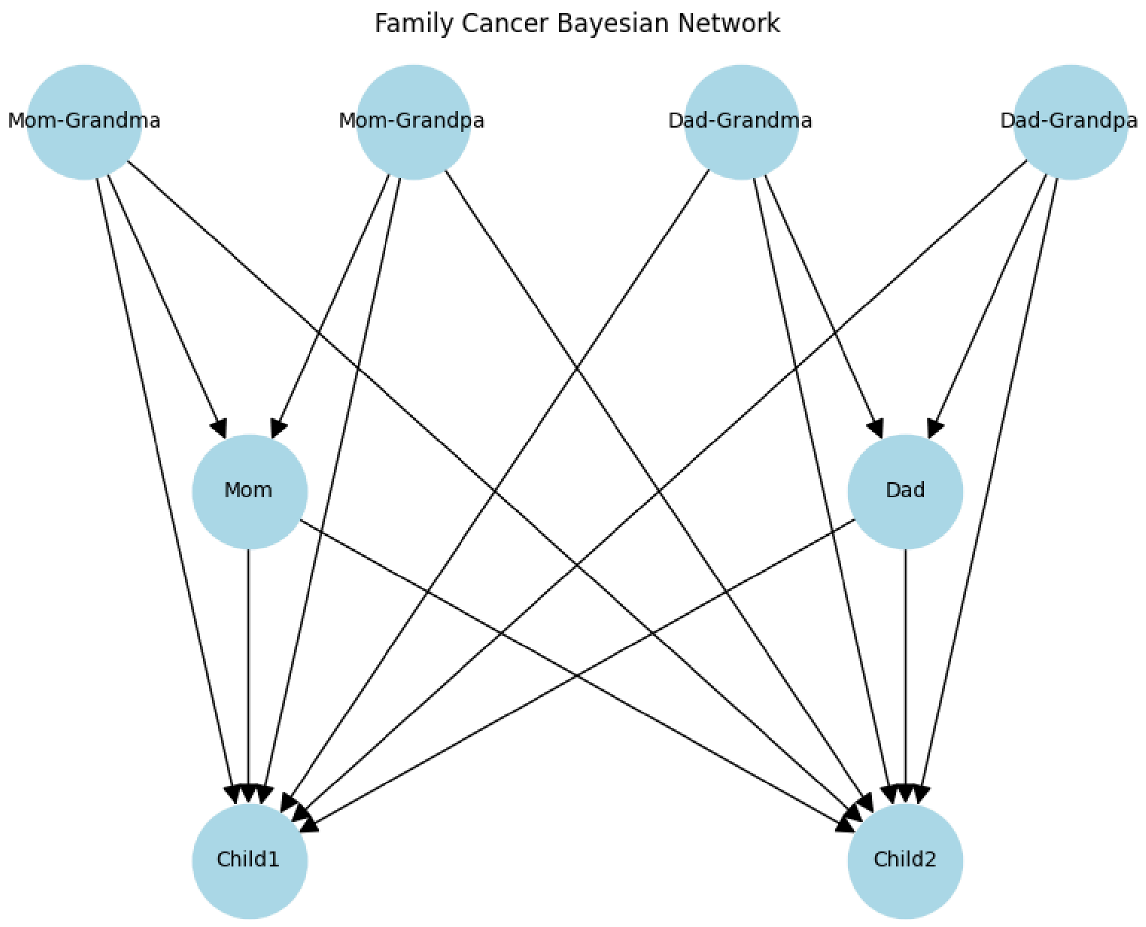






In [14]:
```
# Now lets discuss each of these 3 plots.

# The first we see that the MLE was way closer than the Beta(1,1) in the beg

# Second, the Beta(100,1) was much closer than the MLE estimate. This is bec

# Last, the Beta(1,100) was way off, so it actually hurt us compared to the
# So what is the lesson? If you have huge n and not much of a prior belief c
```

In [20]:
```python
# 5. Structure Learning

# Now lets say we read a paper by a famous doctor who believes there may be
# And so, us datascientists, want to let the data tell us if this is more li
# How can we let the data tell us which graph is more likely given enough da
# Well, we could model this by adding an edge from the grandparents to the c
# And then we can compute the likelihood of that graph (G2) vs. the original

# First, lets visualize G2 so you can see what I am talking about.

import matplotlib.pyplot as plt
import networkx as nx

# Create the graph
G2 = nx.DiGraph()

# Add nodes for grandparents, parents, and children
nodes = [
    'Mom-Grandma', 'Mom-Grandpa', 'Dad-Grandma', 'Dad-Grandpa',
    'Mom', 'Dad', 'Child1', 'Child2'
]
G2.add_nodes_from(nodes)

# Add edges from grandparents to parents
G2.add_edges_from([
    ('Mom-Grandma', 'Mom'), ('Mom-Grandpa', 'Mom'),
    ('Dad-Grandma', 'Dad'), ('Dad-Grandpa', 'Dad')
])

# Add edges from parents to children
G2.add_edges_from([
    ('Mom', 'Child1'), ('Mom', 'Child2'),
    ('Dad', 'Child1'), ('Dad', 'Child2')
])

# Add edges directly from grandparents to children
G2.add_edges_from([
    ('Mom-Grandma', 'Child1'), ('Mom-Grandma', 'Child2'),
    ('Mom-Grandpa', 'Child1'), ('Mom-Grandpa', 'Child2'),
    ('Dad-Grandma', 'Child1'), ('Dad-Grandma', 'Child2'),
    ('Dad-Grandpa', 'Child1'), ('Dad-Grandpa', 'Child2')
])

# Define the positions of the nodes
pos = {
    'Mom-Grandma': (1, 3),
    'Mom-Grandpa': (3, 3),
    'Dad-Grandma': (5, 3),
    'Dad-Grandpa': (7, 3),
    'Mom': (2, 2),
    'Dad': (6, 2),
    'Child1': (2, 1),
    'Child2': (6, 1),
}

model_G2 = BayesianNetwork(list(G2.edges()))

model_G2.add_cpds(cpd_MG, cpd_MP, cpd_DG, cpd_DP, cpd_M, cpd_D, cpd_C1, cpd_

# Draw the graph with arrows from grandparents to children
plt.figure(figsize=(8, 6))
nx.draw(
    G2, pos, with_labels=True, node_size=3000, node_color='lightblue',
    arrowsize=20, font_size=10, font_color='black'
)
plt.title("Family Cancer Bayesian Network")
plt.show()
```



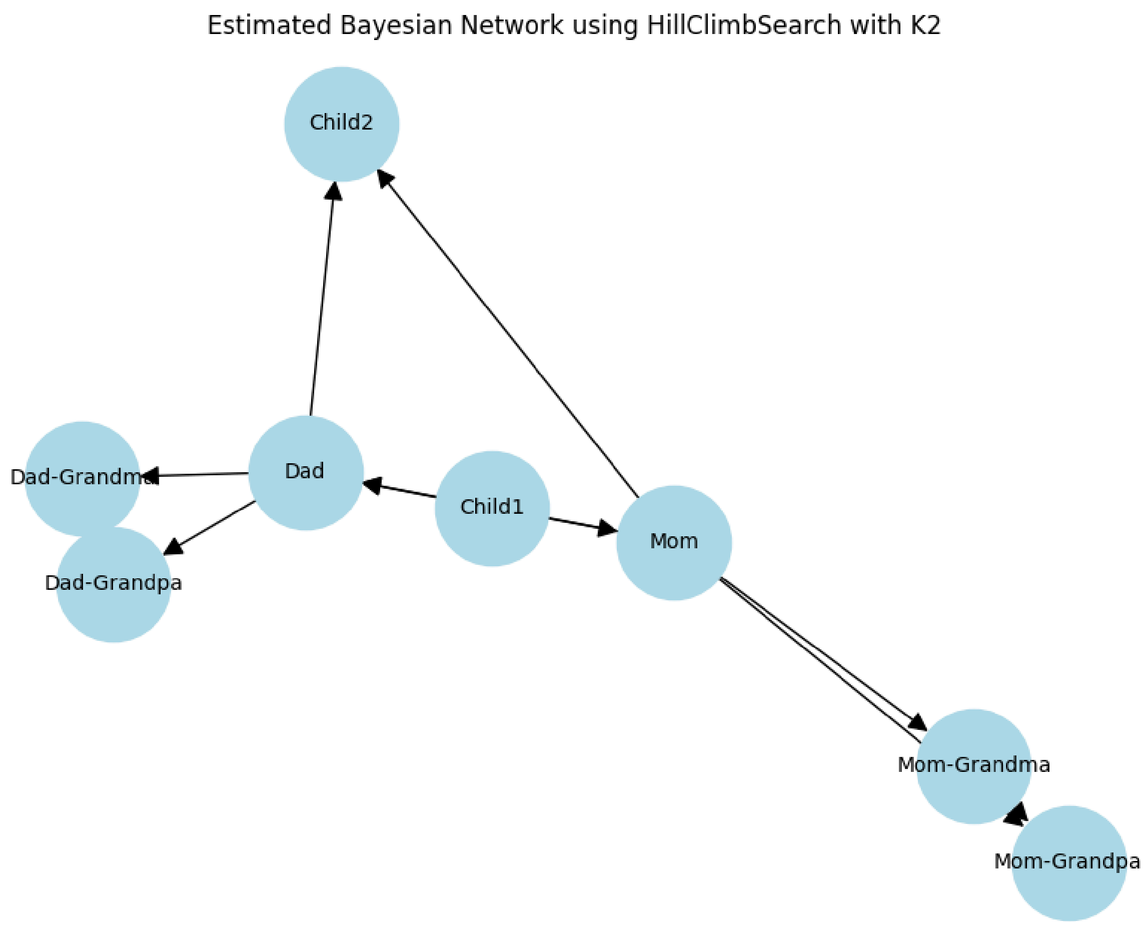
In [21]:
```
# Now lets calculate the Bayesian Score of G2 and G1 and lets see given the

# We will use the Bayesian Information Criterion (BIC) (BIC=log(P(data | moc
from pgmpy.estimators import BicScore

# Calculate BIC Scores
bic = BicScore(df)
score_G1 = bic.score(model)
score_G2 = bic.score(model_G2)

print(f"BIC Score of G1: {score_G1}")
print(f"BIC Score of G2: {score_G2}")
```

```python
# Compare the scores
if score_G1 > score_G2:
    print("G1 is more probable.")
else:
    print("G2 is more probable.")
```

```
BIC Score of G1: -21238.55236189877
BIC Score of G2: -21695.190708234622
G1 is more probable.
```

In [26]:
```python
# So now we have answered the question! Granted we generated the data so we

# Lets take this to the next level! Lets say we want to let the data TELL us

# Maybe there is some strange dependence between the parents! Where if one p

# But just like before it all depends on our priors and the size of n for ho

from pgmpy.estimators import K2Score, HillClimbSearch

# Dataset sizes to test
n=5000
df = create_data_set(num_families=n)

# Use HillClimbSearch to estimate the most probable graph
hc = HillClimbSearch(df)
estimated_model = hc.estimate(scoring_method='k2score', )

# Print the edges of the estimated model
print("Estimated Graph Structure:")
print(estimated_model.edges())

# Visualize the estimated graph using networkx
G_estimated = nx.DiGraph(estimated_model.edges())
pos = nx.spring_layout(G_estimated, seed=42)

plt.figure(figsize=(8, 6))
nx.draw(
    G_estimated, pos, with_labels=True, node_size=3000, node_color='lightblu
    arrowsize=20, font_size=10, font_color='black'
)
plt.title("Estimated Bayesian Network using HillClimbSearch with K2")
plt.show()
```

```
  0%|          | 0/1000000 [00:00<?, ?it/s]
Estimated Graph Structure:
[('Mom-Grandma', 'Mom-Grandpa'), ('Dad-Grandpa', 'Dad-Grandma'), ('Mom', 'Mo
m-Grandma'), ('Mom', 'Mom-Grandpa'), ('Mom', 'Child2'), ('Mom', 'Dad'), ('Da
d', 'Dad-Grandpa'), ('Dad', 'Dad-Grandma'), ('Dad', 'Child2'), ('Child1', 'M
om'), ('Child1', 'Dad')]
```



In [ ]:
```
# 6. Simple Decisions

# Now lets assume that you are a doctor, and you have the unfortunate respor
# Chemo is a cancer treatment that uses drugs to kill cancer cells, stop the
# If the patient has a high probability of having cancer, this may be their

# So lets use our knowledge of Decision Graphs and Maximum Expected Utility
```

In [ ]: