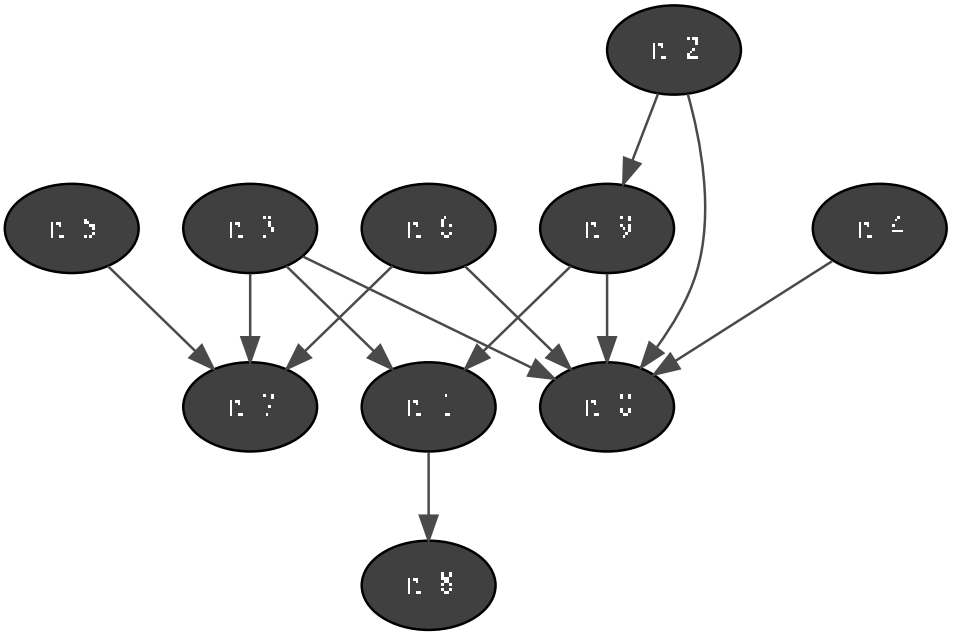
In [1]:
```
import pyAgrum as gum
import pyAgrum.lib.notebook as gnb
import pyAgrum.lib.image as gimg
from pyAgrum.lib.bn_vs_bn import GraphicalBNComparator
```

Générer un BN aléatoirement

In [2]:
```
generator=gum.BNGenerator()
bn=generator.generate(n_nodes = 10, n_arcs = 12, n_modmax = 4)
bn
```

Out[2]:



Générer une base de données CSV à partir d'un BN

In [3]:
```
gum.generateCSV(bn,name_out="test.csv", n=1000, show_progress=False, with_labels=False
```

Out[3]:
```
-13978.79939850938
```

Calculer un G2 ou un Chi2 à partir d'une base de données CSV

In [4]:
```
lea=gum.BNLearner("test.csv")
print(lea.chi2("n_1","n_5",["n_6","n_3"])) # return stats,p-value
print(lea.chi2("n_5","n_4",["n_0"])) # return stats,p-value
```

```
(49.48069493073561, 0.06662642953182055)
(28.13433676631615, 0.25441450308878927)
```

In [5]:
```
print(lea.G2("n_1","n_5",["n_6","n_3"])) # return stats,p-value
print(lea.G2("n_5","n_4",["n_0"])) # return stats,p-value
```

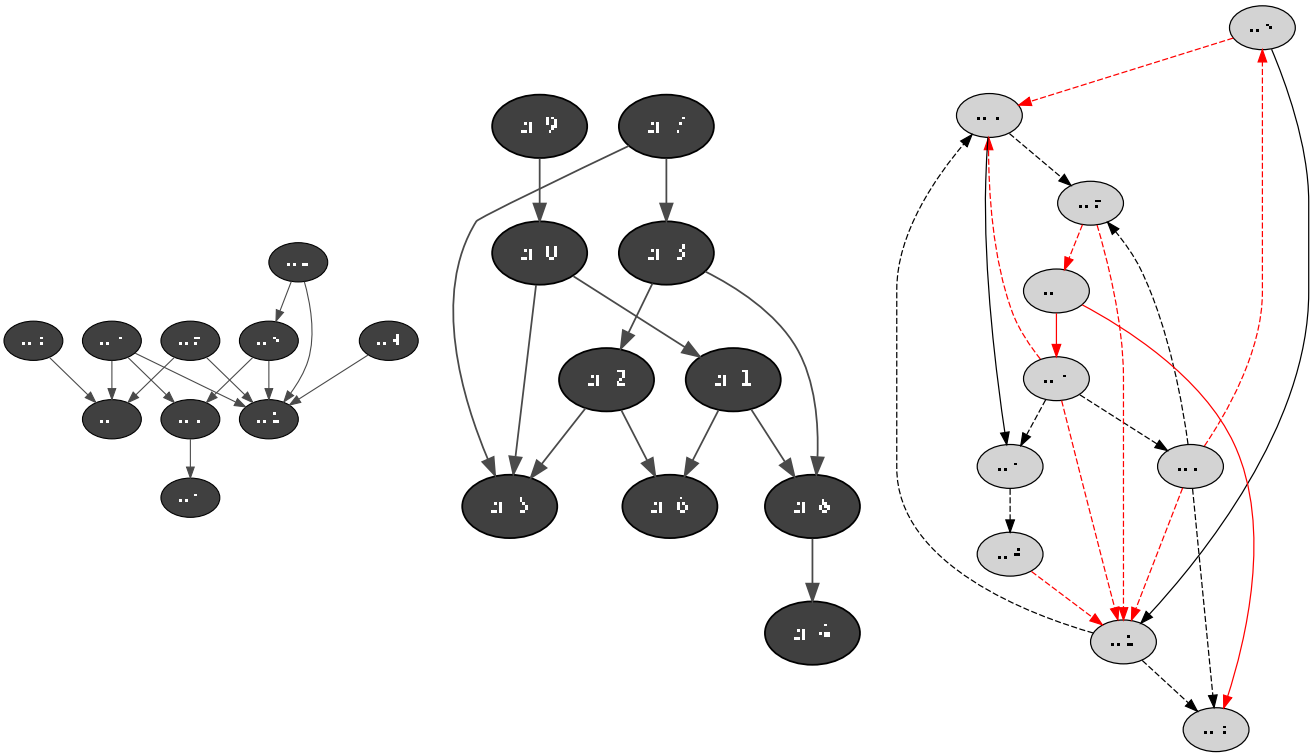
```
(51.52818831799166, 0.045106931285787966)
(30.443757250436605, 0.17047630089646942)
```

Comparer des structures de BN

In [18]:

```
bn2=generator.generate(n_nodes = 10, n_arcs = 12, n_modmax = 4)
c=GraphicalBNComparator(bn,bn2)

help(c.dotDiff)
gnb.sideBySide(bn,bn2,gnb.getGraph(c.dotDiff()))
```

```
Help on method dotDiff in module pyAgrum.lib.bn_vs_bn:

dotDiff() method of pyAgrum.lib.bn_vs_bn.GraphicalBNComparator instance
    Return a pydotplus graph that compares the arcs of _bn1 (reference) with those of s
elf._bn2.
    full black line: the arc is common for both
    full red line: the arc is common but inverted in _bn2
    dotted black line: the arc is added in _bn2
    dotted red line: the arc is removed in _bn2

    Warning
    -------
    if pydotplus is not installed, this function just returns None

    Returns
    -------
    pydotplus.Dot
      the result dot graph or None if pydotplus can not be imported
```



In [17]:

```
help(c.hamming)
print(f"hamming : {c.hamming()}")
```

```
Help on method hamming in module pyAgrum.lib.bn_vs_bn:

hamming() method of pyAgrum.lib.bn_vs_bn.GraphicalBNComparator instance
    Compute hamming and structural hamming distance

    Hamming distance is the difference of edges comparing the 2 skeletons, and Structur
al Hamming difference is the
    difference comparing the cpdags,    including the arcs' orientation.

    Returns
    -------
    dict[double,double]
      A dictionnary containing 'hamming','structural hamming'
```

hamming : {'hamming': 14, 'structural hamming': 17}

In [16]:
```
help(c.skeletonScores)
print(f"scores : {c.skeletonScores()}")
```

```
Help on method skeletonScores in module pyAgrum.lib.bn_vs_bn:

skeletonScores() method of pyAgrum.lib.bn_vs_bn.GraphicalBNComparator instance
    Compute Precision, Recall, F-score for skeletons of self._bn2 compared to self._bn1

    precision and recall are computed considering BN1 as the reference

    Fscor is 2*(recall* precision)/(recall+precision) and is the weighted average of Pr
ecision and Recall.

    dist2opt=square root of (1-precision)^2+(1-recall)^2 and represents the euclidian d
istance to the ideal point (precision=1, recall=1)

    Returns
    -------
    dict[str,double]
      A dictionnary containing 'precision', 'recall', 'fscore', 'dist2opt' and so on.

scores : {'count': {'tp': 5, 'tn': 26, 'fp': 7, 'fn': 7}, 'recall': 0.4166666666666667,
'precision': 0.4166666666666667, 'fscore': 0.4166666666666667, 'dist2opt': 0.8249579113
843053}
```

In [ ]:
```

```