In [62]:

```
import sys
import numpy as np

from gamspy import (
    Container,Set,Alias,Parameter,Variable,Equation,Model,Problem,Sense,Opti
    Domain,Number,Sum,Product,Smax,Smin,Ord,Card,SpecialValues,
)
import gamspy.math as gpm

options = Options(relative_optimality_gap=0, equation_listing_limit=100, var
m = Container(options=options)

f = Set(m,'freq',records=range(1,21))
i = Set(m,'speaker',records=range(1,201))
j = Alias(m,'j',i)

maxfreqres = Parameter(m,'maxfreqres',records=1500)
spec = Parameter(m,'spec',description='specification limit',records=7000)
response = Parameter(m,'response',domain=[i,f])
response[i,f] = gpm.uniform(0,maxfreqres)
```

In [78]:

```
# Here is the set of arcs

coefs = Parameter(m,'coefs',domain=[i,j])
coefs[i,j] = Sum(f, gpm.abs(response[i, f] - response[j, f]))
# display(coefs.pivot())

good_match = Set(m, 'good_match', domain=[i, j])
good_match[i, j] = (coefs[i, j] <= 7000) & (i.ord < j.ord)
display(good_match.pivot())

x = Variable(m,"x",type="binary",domain=[i,j],description="flow")

obj = Sum(good_match, x[good_match])


not_matching = Equation(m, 'not_matching', domain=[i])
not_matching[i] = (Sum(j.where[good_match[i, j]], x[i, j]) +
                   Sum(j.where[good_match[j, i]], x[j, i])) <= 1


maxmatch = Model(m,
    name="maxmatch",
    equations=m.getEquations(),
    problem='MIP',
    sense=Sense.MAX,
    objective= obj
)
```

```
/home/samjenkins2001/CS524/venv/lib/python3.10/site-packages/gams/transfer/s
yms/_mixins/pivot.py:121: FutureWarning: Downcasting object dtype arrays on
.fillna, .ffill, .bfill is deprecated and will change in a future version. C
all result.infer_objects(copy=False) instead. To opt-in to the future behavi
or, set `pd.set_option('future.no_silent_downcasting', True)`
  df.fillna(fill_value, inplace=True)
```

	7	8	13	14	15	18	23	24	26	28	...	190	191
1	False	False	False	False	True	False	False	False	False	False	...	False	False
2	False	False	False	False	False	False	True	False	False	False	...	False	False
3	True	False	False	False	False	False	False	False	False	False	...	False	False
4	True	True	False	False	False	False	False	True	False	False	...	False	False
5	False	False	False	False	True	False	False	True	True	False	...	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...
176	False	False	False	False	False	False	False	False	False	False	...	False	False
177	False	False	False	False	False	False	False	False	False	False	...	False	False
182	False	False	False	False	False	False	False	False	False	False	...	False	False
184	False	False	False	False	False	False	False	False	False	False	...	False	False
189	False	False	False	False	False	False	False	False	False	False	...	False	False

152 rows × 159 columns

In [75]:

```
maxmatch.solve()
display(good_match.records)
print(f"Number of matched speakers = {maxmatch.objective_value}")
```

	speaker	j	element_text
0	1	15	
1	1	131	
2	2	23	
3	2	40	
4	2	68	
...	...	...	...
524	176	192	
525	177	196	
526	182	186	
527	184	193	
528	189	192	

529 rows × 3 columns

```
Number of matched speakers = 97.0
```

In [65]:

```
# Now we do the second part
bad_freq = Parameter(m, 'bad_freq', domain=[i, j, f])
bad_freq[i, j, f] = gpm.abs(response[i, f] - response[j, f]) >= 500

bad_freq_count = Sum(f, bad_freq[i, j, f])

good_match[i, j].where[bad_freq_count[i, j] > 3] = False
# Put your code here

not_matching_2 = Equation(m, 'not_matching_2', domain=[i])
not_matching_2[i] = (Sum(j.where[good_match[i, j]], x[i, j]) +
                   Sum(j.where[good_match[j, i]], x[j, i])) <= 1


maxmatch = Model(m,
    name="maxmatch",
    equations=m.getEquations(),
    problem='MIP',
    sense=Sense.MAX,
    objective= obj
)
```

In [66]:

```
maxmatch.solve()
display(good_match.records)
print(f"Number of matched speakers = {maxmatch.objective_value}")
```

	speaker	j	element_text
0	2	125	
1	4	7	
2	4	166	
3	5	15	
4	5	24	
...	...	...	...
124	162	181	
125	164	186	
126	169	194	
127	176	192	
128	184	193	

129 rows × 3 columns

```
Number of matched speakers = 54.0
```

In [ ]: