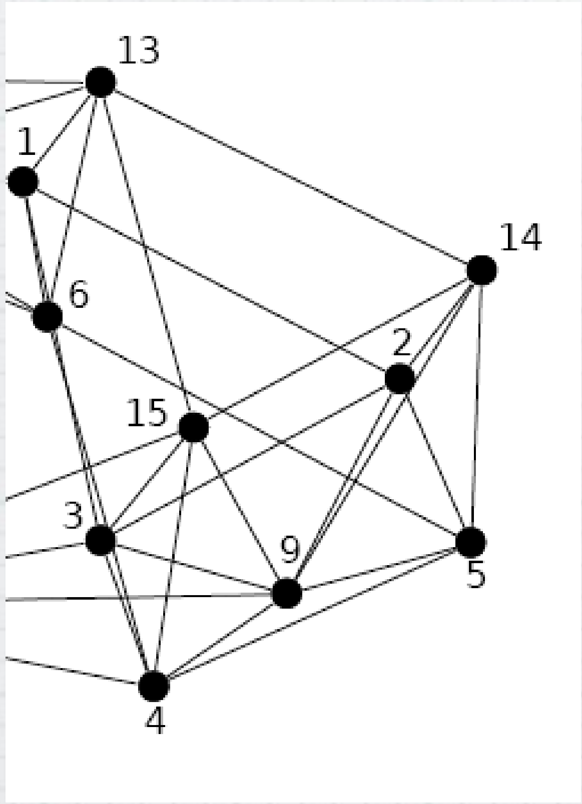
1,2,3...QAP!

Matteo Fischetti
(joint work with Michele Monaci and Domenico Salvagnin)
DEI University of Padova

IPDU, Newcastle, July 2011

QAP definition

- complete directed graph $G=(V,A)$ and $n=|V|$ **facilities** to be assigned to its **nodes**
- **distance** from node i to node j is b_{ij}
- required **flow** from facility u to facility v is a_{uv}
- decision var.s: $x_{iu}=1$ iff facility u is assigned to node i, $=0$ *othw.*



$$\min \sum_i \sum_u \sum_j \sum_v a_{uv}\, b_{ij}\, x_{iu}\, x_{jv}$$

$$\sum_i x_{iu} = 1 \quad \forall u$$

$$\sum_u x_{iu} = 1 \quad \forall i$$

$$x_{iu} \in \{0,1\} \quad \forall i,u$$

ESC instances in 3 steps

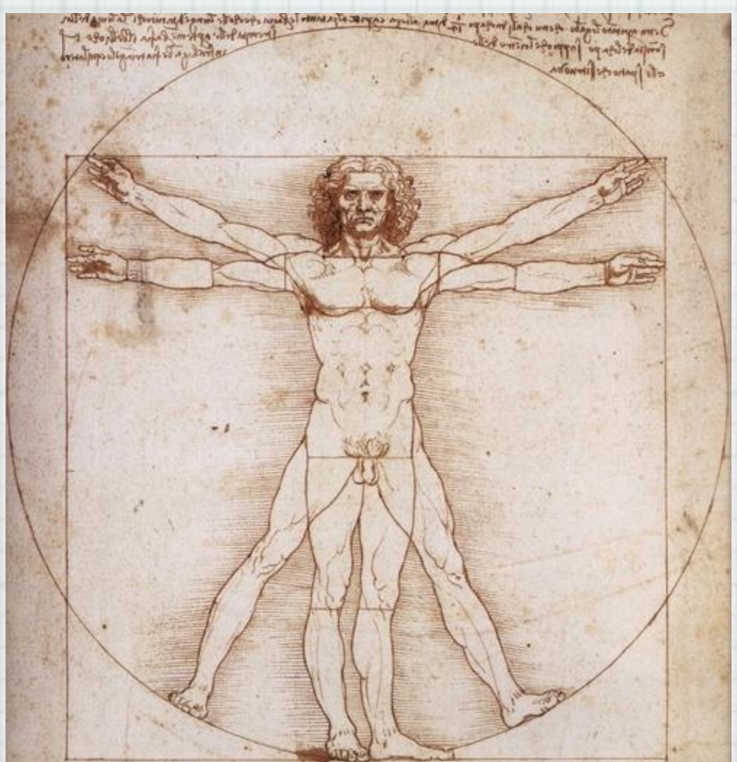
B. Eschermann and H.J. Wunderlich [EsWu:90]

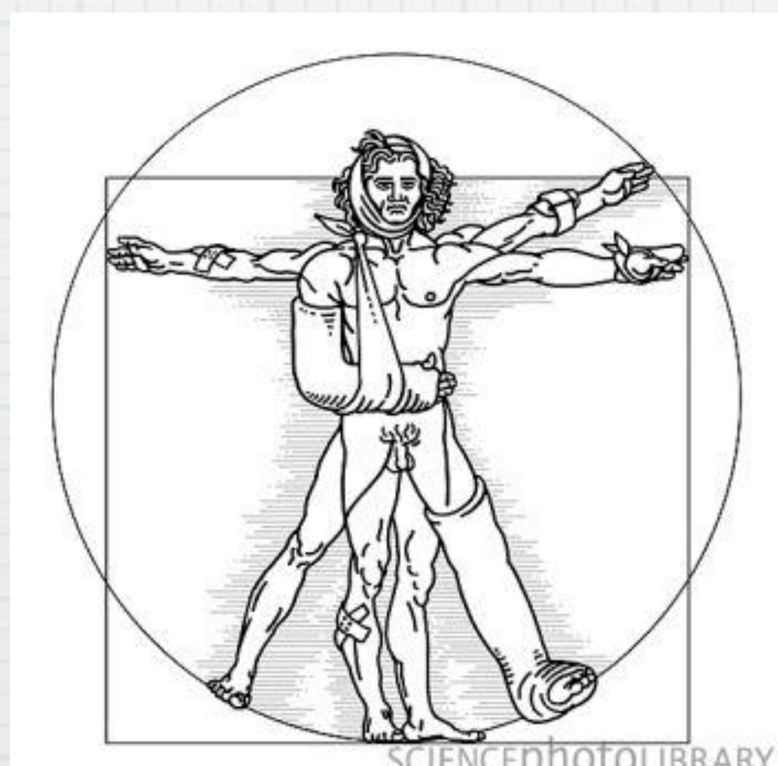
These examples stem from an application in computer science, from the testing of self-testable sequential circuits. are due to [ClPe:94] (*n*=16) and [BrClMaPe:96] (*n*=32).

	name	n	feas.sol.	permutation/bound	gap
	Esc16a	16	68 (OPT)	(2,14,10,16,5,3,7,8,4,6,12,11,15,13,9,1)	
	Esc16b	16	292 (OPT)	(6,3,7,5,13,1,15,2,4,11,9,14,10,12,8,16)	
	Esc16c	16	160 (OPT)	(11,14,10,16,12,8,9,3,13,6,5,7,15,2,1,4)	
	Esc16d	16	16 (OPT)	(14,2,12,5,6,16,8,10,3,9,13,7,11,15,4,1)	
	Esc16e	16	28 (OPT)	(16,7,8,15,9,12,14,10,11,2,6,5,13,4,3,1)	
	Esc16f	16	0 (OPT)	(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16)	
	Esc16g	16	26 (OPT)	(8,11,9,12,15,16,14,10,7,6,2,5,13,4,3,1)	
	Esc16h	16	996 (OPT)	(13,9,10,15,3,11,4,16,12,7,8,5,6,2,1,14)	
	Esc16i	16	14 (OPT)	(13,9,11,3,7,5,6,2,1,15,4,14,12,10,8,16)	
	Esc16j	16	8 (OPT)	(8,3,16,14,2,12,10,6,9,5,13,11,4,7,15,1)	
*	Esc32a	32	130 (Ro-TS)	103 (L&P)	20.77 %
*	Esc32b	32	168 (Ro-TS)	132 (L&P)	21.43 %
*	Esc32c	32	642 (SIM-1)	616 (L&P)	4.05 %
*	Esc32d	32	200 (Ro-TS)	191 (L&P)	4.50 %
	Esc32e	32	2 (OPT)	(1,2,5,6,8,16,13,19,9,32,7,22,24,20,4,12,3, 17,29,21,11,25,27,18,30,31,23,28,14,15,26,10)	
	Esc32g	32	6 (OPT)	(14,15,16,12,11,26,30,10,25,8,29,22,31,28, 13,1,19,9,17,32,24,18,4,2,20,5,21,3,7,6,23,27)	
*	Esc32h	32	438 (Ro-TS)	424 (L&P)	3.20 %
	Esc64a	64	116 (SIM-1)	98 (SDP1)	15.52 %
	Esc128	128	64 (GRASP)	2 (GLB)	96.86 %

Step 1: don’t break my symmetries!

* Many QAP instances are highly symmetrical (certain facility/node permutations do not affect solution cost nor feasibility)
* Symmetry is typically viewed as a useful feature in mathematics, but…
* … it tricks enumeration (equivalent sol.s visited again and again)
* Usual recipe in discrete optimization: **break it!**
* Instead, we propose a new way to exploit it to **reduce** problem size and complexity

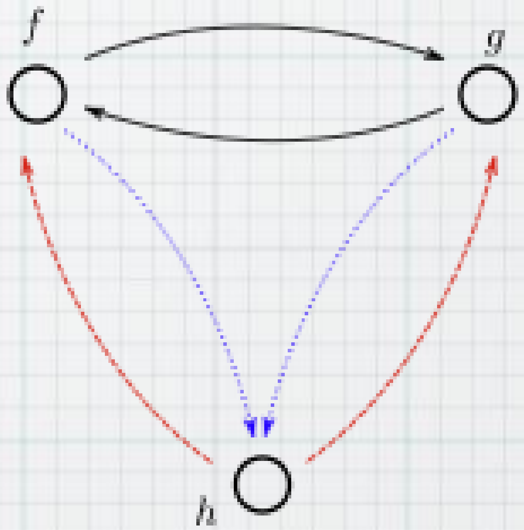




Clone definition

Assume wlog $b_{ii} = 0$ for all nodes *i*.

Two facilities f and g are **clones** iff:



- $a_{fg} = a_{gf}$
- $a_{fh} = a_{gh}$ for all h <> f,g
- $a_{hf} = a_{hg}$ for all h <> f,g

Equivalence relation that **partitions** the set of facilities into clone clusters

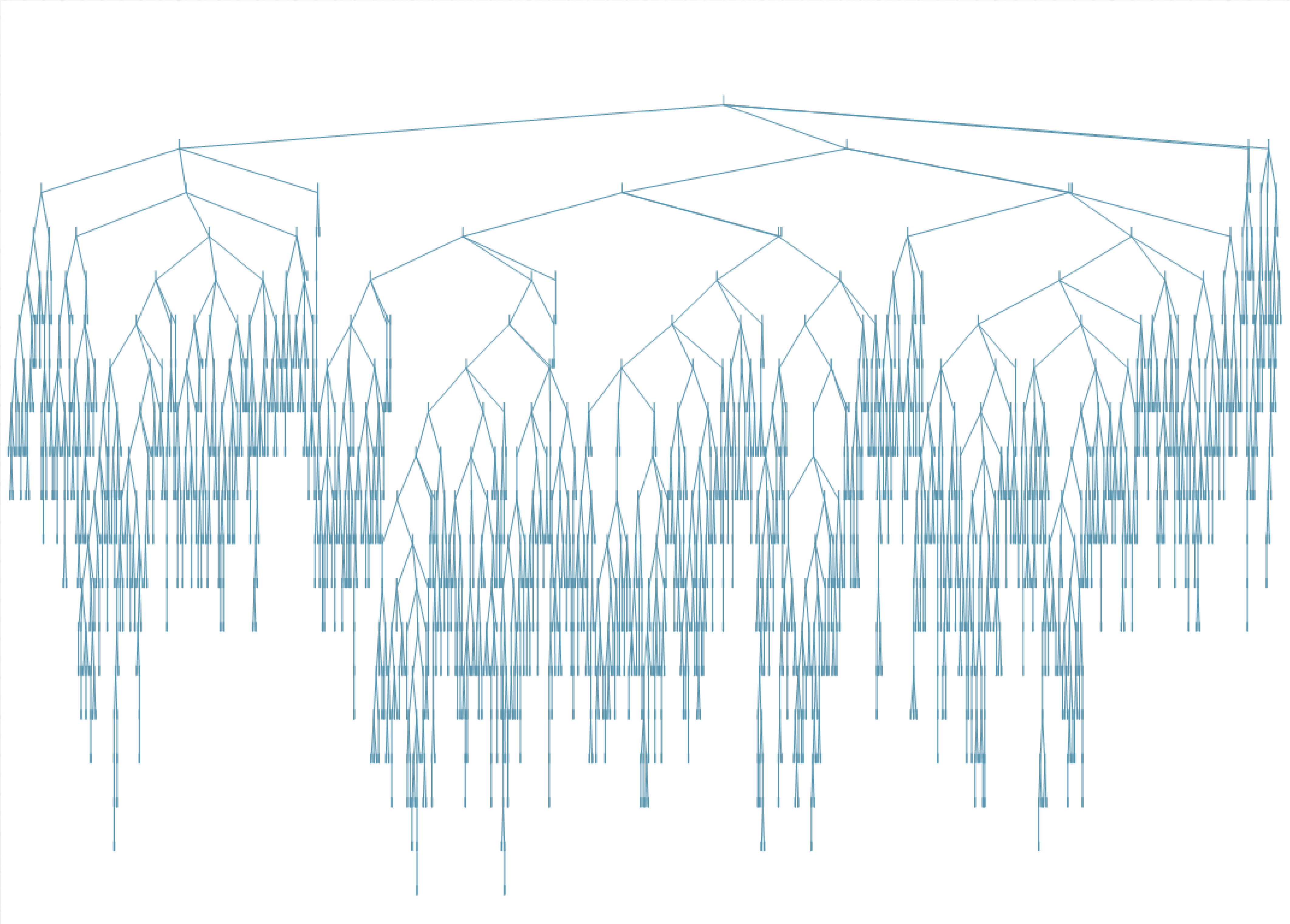
Shrinking clones

- All esc instances have such clones (not just “isolated” ones…)
- We shrink them and update the model accordingly

esc32a	32 x 32	32 x 26
esc32b	32 x 32	32 x 25
esc32c	32 x 32	32 x 10
esc32d	32 x 32	32 x 13
esc32h	32 x 32	32 x 14
esc64a	64 x 64	64 x 15
esc128	128 x 128	128 x 21

We are removing a huge amount of symmetry: any formulation is hopeless otherwise!!!

Step 2: B&C design



Main ingredients

- carefully chosen MILP formulation
- locally valid cut separation based on Gilmore-Lawler bounds
- custom QAP-specific branching strategy
- custom symmetry detection on matrix b and aggressive orbital branching

Step 2.1: choosing the model

Introducing variables $w_{iu} = (\sum_j \sum_v a_{uv} b_{ij} x_{jv}) x_{iu}$

we get the basic Kaufman-Broeckx (KB) MILP model

$$\min \sum_i \sum_u w_{iu}$$

$$\sum_i x_{iu} = |C_u| \quad \forall u$$

$$\sum_u x_{iu} = 1 \quad \forall i$$

$$\sum_j \sum_v a_{uv} b_{ij} x_{jv} \leq w_{iu} + M_{iu}(1 - x_{iu}) \quad \forall i, u$$

$$w_{iu} \geq 0 \quad \forall i, u$$

Step 2.1: handy MILP

The Kb model is tiny and fast ... but its bound is really bad (always zero at the root)

However we can improve it through the following family of inequalities (Xia and Yuan, 2006)

$$w_{iu} \geq minAP_{iu} \; x_{iu} \quad \forall i, u$$

where $minAP_{iu}$ is the Gilmore-Lawler term computed by solving a linear assignment problem with $x_{iu} = 1$

This family of cuts strengthen the KB model a lot → we separate local versions of them throughout the B&C tree, by using a fast separation procedure

Step 2.2: branching

- A good branching order is **crucial** for the B&C
- Default strategies are NOT particularly effective on these instances
- Basic idea → we want to branch first on the variables that have a **larger range** of objective values for the possible assignments
- We define the **branching priority** for x_{iu} as

$$(maxAP_{iu} - minAP_{iu})(n+1)^2 + u(n+1) + i$$

Step 2.3: orbital branching

* Clone shrinking takes care of (most of) symmetry on matrix a → what about matrix b?
* On esc instances, also matrix b contains symmetries (but not of clone type) → resort to **orbital branching** (Ostrowski et al., 2011)
* We compute the appropriate symmetry group **directly on matrix b** (faster than considering the whole model)
* we could have used *nauty*, but we exploited the particular structure of esc instances and implemented an ad-hoc procedure

Step 2.3: matrix b structure

* b_{ij} = HammingDistance(i-1,j-1)-1

* Two operations on binary string preserve the Hamming distance:

0011 → (bit flip) → 0000
1001 → 1010

0011 → (bit permutation) → 1001
1001 → 1100

* fix facility 11…1 form the beginning → no bit flips left

* compute orbits and stabilizers from explicit list of bit permutations!

Cplex 12.2 interactive mode
(8 threads, Intel Xeon 3.2Ghz, 16GB ram)

Instance	n	m	OPT	time (s)	#nodes
esc16a	16	9	68	0.35	4,133
esc16b	16	7	292	3.07	71,075
esc16c	16	12	160	130.98	2,652,014
esc16d	16	12	16	0.51	10,796
esc16e	16	8	28	0.05	421
esc16f	16	1	0	0.00	0
esc16g	16	9	26	0.04	450
esc16h	16	5	996	0.23	4,967
esc16i	16	10	14	0.18	3,216
esc16j	16	7	8	0.03	114
esc32c*	32	10	642	9,643.82	81,650,962
esc32d*	32	13	200	2,973.26	12,757,770
esc32e	32	6	2	0.04	70
esc32g	32	7	6	0.06	597
esc64a*	64	15	116	509.87	1,206,370
tai64c	64	2	1,855,928	18,250.40	1,216,074,081

B&C results

esc32c	616	642	642	1156s
esc32d	191	200	200	473s
esc64a	98	116	116	84s

IBM Cplex 12.2 on Intel Xeon 3.2GHz - 16GB RAM - 8 threads

3 instances solved in half an hour!

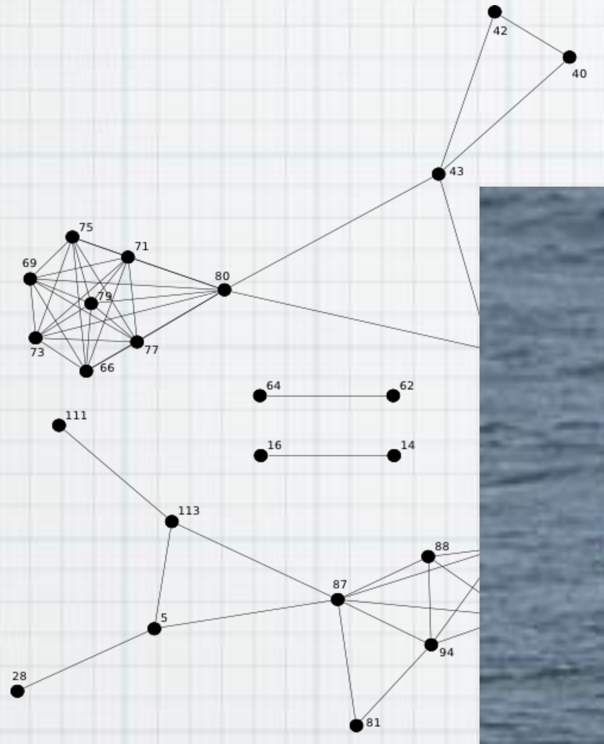
A closer look at esc64a

unshrunken	any	hopeless	+∞
shrunken	cplex default	>3.600	>8.000.000
shrunken	cplex tweaked	966	1.750.000
shrunken	cplex twk + ORD	577	1.300.000
shrunken	our B&C	84	142.000

IBM Cplex 12.2 on Intel Xeon 3.2GHz - 16GB RAM - 8 threads

Similar results are obtained on the other esc instances

Whale watching (esc128)



Esc128	128	64	(GRASP)	2	(GLB)	96.86 %

Step 3: flow splitting

* Split matrix a as $a = a_1 + a_2$, with $a_1, a_2 \geq 0$
* Solve $QAP(a_1,b)$ and $QAP(a_2,b)$ separately
* Lower bound property:

$$opt(QAP(a_1, b)) + opt(QAP(a_2, b)) \leq opt(QAP(a, b))$$

* full equivalence if we impose the two solutions coincide (equality) → variable splitting model
* just a relaxation otherwise (lower bound)

Two better than one?

- the two models are still QAPs of the **same size** as before → why should we want to do this?

- two main reasons:

1. the final bound after a **fixed amount of enumeration** on a weaker model might be much better than that based on a stronger model (strange but true!)

2. if the two QAPs have **a simpler structure** they might be much easier to solve than the original instance (in particular, we can actually add symmetry to the model!)

How to split the flow matrix?

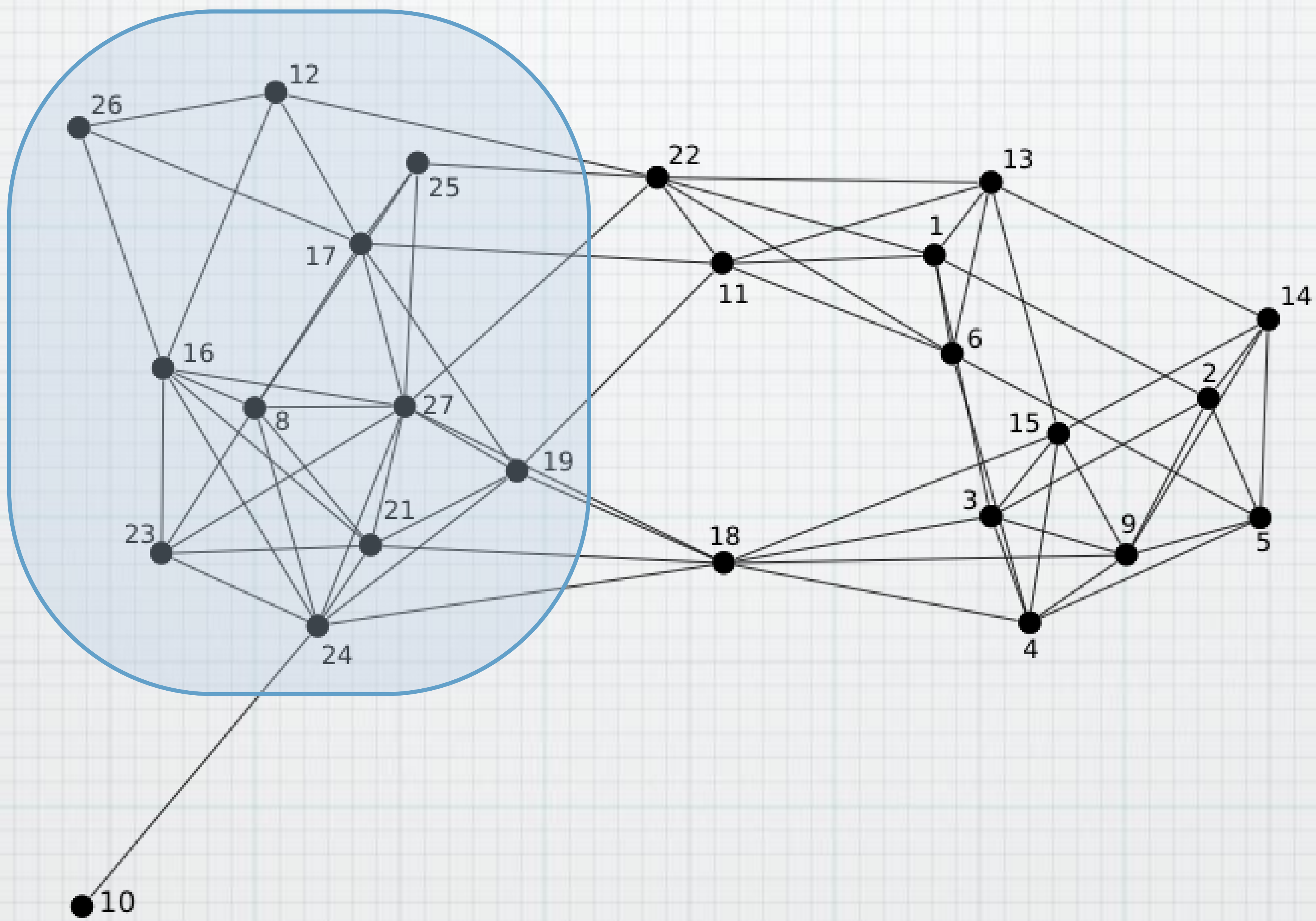
* two (independent) strategies

1. select a **subset of facilities** and zero out all distances in their clique → good strategy when there are (almost) disconnected components in flow support

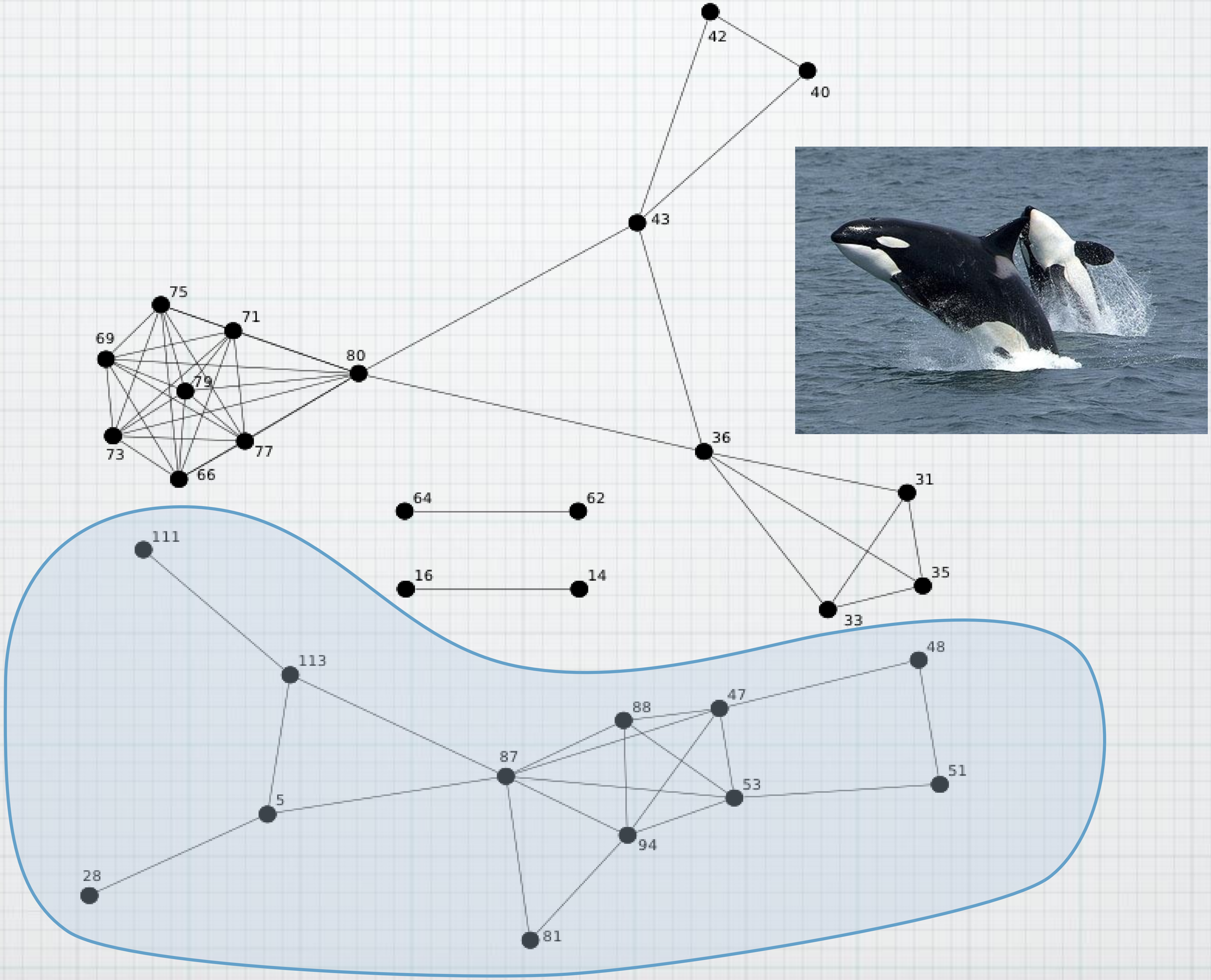
2. define a_1 as clip(a,[0,1]) and $a_2 = a - a_1$ → improve cost "uniformity"

3. can be applied sequentially to get a "longer" split chain $a = a_1 + a_2 + \ldots a_k$

Flow splitting for esc32a ...



… and for the big whale (esc128)



Flow splitting results

esc32a	130	68+60 = 128	6 + 45 = 51s
esc32h	438	340+98 = 438	4 + 7795 = 7799s
esc128	64	48+16 = 64	2 + 7 = 9s (!!!)

IBM Cplex 12.2 on Intel Xeon 3.2GHz - 16GB RAM - 8 threads

2 more instances solved and 1 much improved bound!

Conclusions & Future work

* We could solve unsolved esc instances in a surprisingly short amount of time, including esc128 (the largest QAPLIB instance ever solved)

TODO list

* develop a B&B algorithm using a variable-splitting model based on flow splitting
* try other QAP classes
* generalize to other classes of difficult MI(N)LPs → **Orbital shrinking** (F.-Liberti, 2011)

Thank you