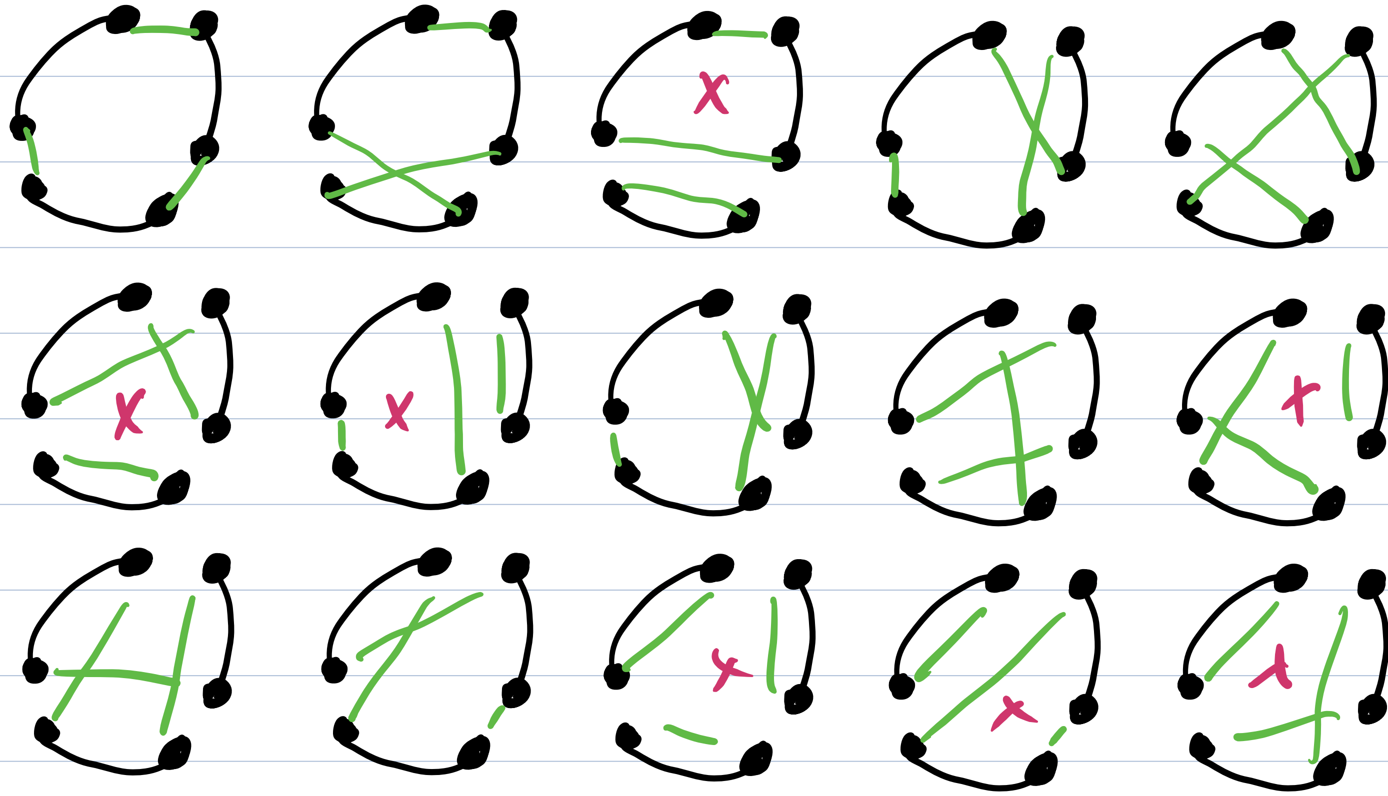
Example 2.35 **Many neighborhoods in a real-life application.** This example illustrates the use of many neighborhoods ($k = 9$) to solve a scheduling problem [26]. Many oil wells in onshore fields rely on artificial lift methods. Maintenance services such as cleaning, stimulation, and others are essential to these wells. They are performed by workover rigs. Workover rigs are available in a limited number with respect to the number of wells demanding service. The decision which workover rig should be sent to perform some maintenance services is based on factors such as the well production, the current location of the workover rig in relation to the demanding well, and the type of service to be performed. The problem of scheduling workover rigs consists in finding the best schedule S^* for the available m workover rigs, so as to minimize the production loss associated with the wells awaiting service. A schedule is represented by an ordered set of wells serviced by workover rigs. A variable neighborhood search metaheuristic has been proposed for this problem using the following neighborhoods in the shaking procedure:

1. Swapping of routes (SS) where the wells associated with two workover rigs are exchanged.
2. Swapping of wells from the same workover rig (SWSW) where two wells serviced by the same workover rig are exchanged.
3. Swapping of wells from different workover rigs (SWDW) where two wells affected by two different workover rigs are exchanged.
4. Add/drop (AD) where a well affected by a workover rig is reassigned to any position of the schedule of another workover rig.
5. Two applications of the SWSW transformation.
6. Two applications of the SWDW transformation.
7. Three applications of the SWDW transformation.
8. Successive application of two AD transformations.
9. Successive application of three AD transformations.



Topic 18 – Variations on Local Search

Wednesday, May 4

Announcements:

→ HW 5 due the last day of class Mon, May 9 11:59pm

→ Final will be takehome, due Mon, May 16, 11:59pm

#1: Variable Neighborhood Search

__Idea:__ Define different kinds of tweaks (different neighborhoods)

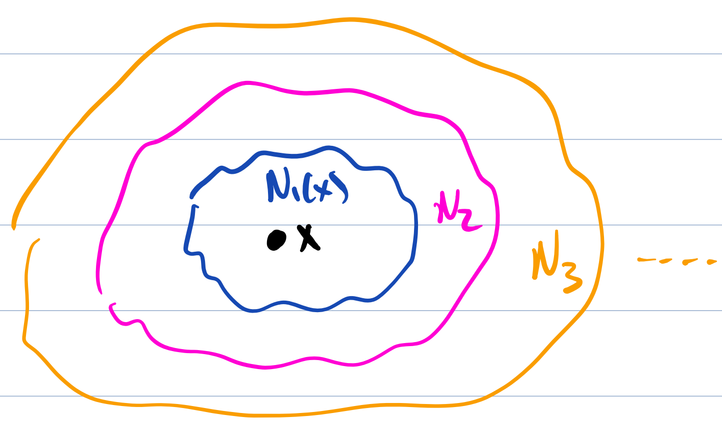
$N_1(x),\ N_2(x),\ N_3(x),\ \ldots,\ N_d(x)$

$\xrightarrow{\hspace{6cm}}$

more dramatic tweaks

Often the case:

$$N_1(x) \subseteq N_2(x) \subseteq N_3(x) \subseteq \cdots \subseteq N_d(x)$$



Procedure:

We have a current solution x

Pick $s \in N_1(x)$ randomly (s could be better or worse!)

Perform H-C on s to s'. (H-C = N_1 tweaks or something else)

If s' is better than x:

Set $x = s'$ and start over

Else:

Pick $s \in N_2(x)$ randomly

Perform H-C on s to get s'

If s' is better than x:

set $x = s'$ and start over

Else:

Pick $s \in N_3(x)$ ------ and so on

⋱

So we need:

* A neighborhood structure
* A H-C tweak (could re-use N_1 or could use something else)

Ex: Knapsack

$N_k(x)$ = pick k random items to remove

H-C = add 1 item back in randomly (if it doesn't violate constr.)

OR

H-C = add a random item and maybe remove an item as well

You could also try a certain nbhd multiple times before moving up to the next ones.

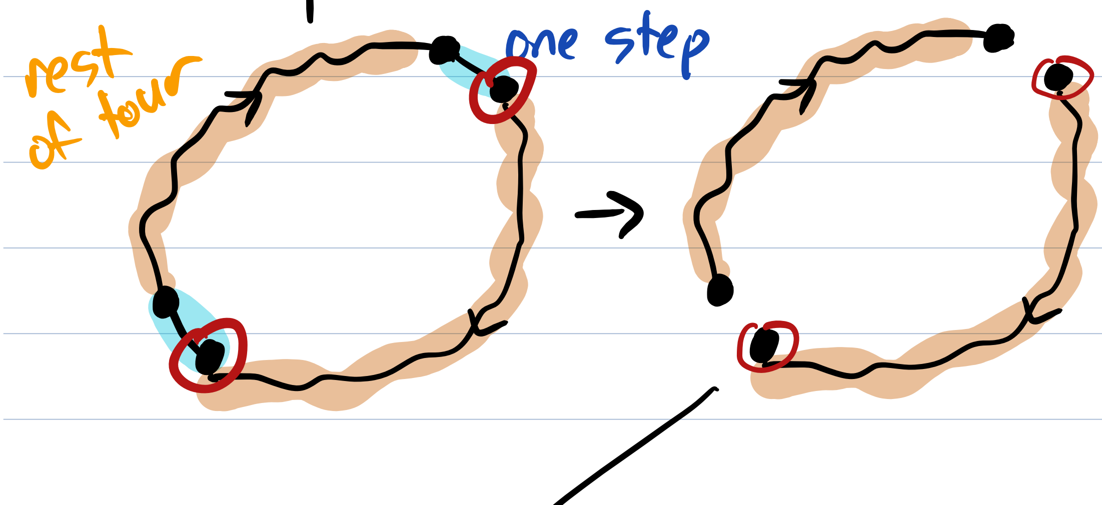
Ex: From "Metaheuristics: From Design to Implementation", by Talbi:

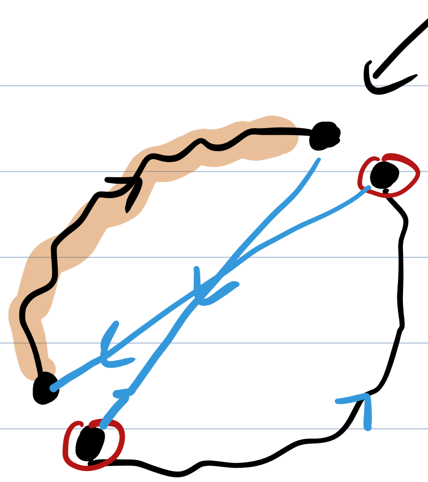
Example 2.35 **Many neighborhoods in a real-life application.** This example illustrates the use of many neighborhoods ($k = 9$) to solve a scheduling problem [26]. Many oil wells in onshore fields rely on artificial lift methods. Maintenance services such as cleaning, stimulation, and others are essential to these wells. They are performed by workover rigs. Workover rigs are available in a limited number with respect to the number of wells demanding service. The decision which workover rig should be sent to perform some maintenance services is based on factors such as the well production, the current location of the workover rig in relation to the demanding well, and the type of service to be performed. The problem of scheduling workover rigs consists in finding the best schedule S^* for the available m workover rigs, so as to minimize the production loss associated with the wells awaiting service. A schedule is represented by an ordered set of wells serviced by workover rigs. A variable neighborhood search metaheuristic has been proposed for this problem using the following neighborhoods in the shaking procedure:

1. Swapping of routes (SS) where the wells associated with two workover rigs are exchanged.
2. Swapping of wells from the same workover rig (SWSW) where two wells serviced by the same workover rig are exchanged.
3. Swapping of wells from different workover rigs (SWDW) where two wells affected by two different workover rigs are exchanged.
4. Add/drop (AD) where a well affected by a workover rig is reassigned to any position of the schedule of another workover rig.
5. Two applications of the SWSW transformation.
6. Two applications of the SWDW transformation.
7. Three applications of the SWDW transformation.
8. Successive application of two AD transformations.
9. Successive application of three AD transformations.

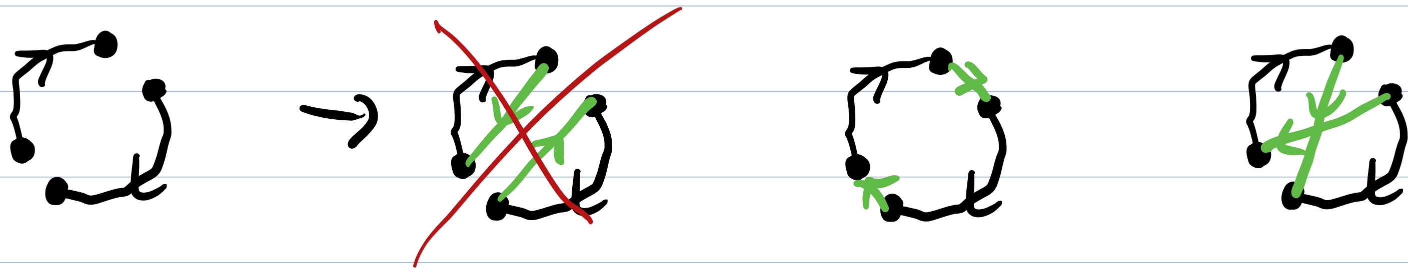
Ex: TSP

Previous tweak: "pick two cities, reverse the path in between"





Another way to (kind of) phrase this:
"delete two random edges, then reconnect the tour in the cheapest way that is still one big cycle."



not valid

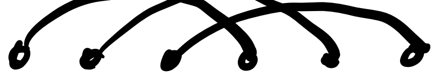
orig tour

reverse a block

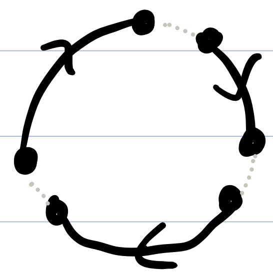
Another name for this is 2-opt.

\# of edges deleted

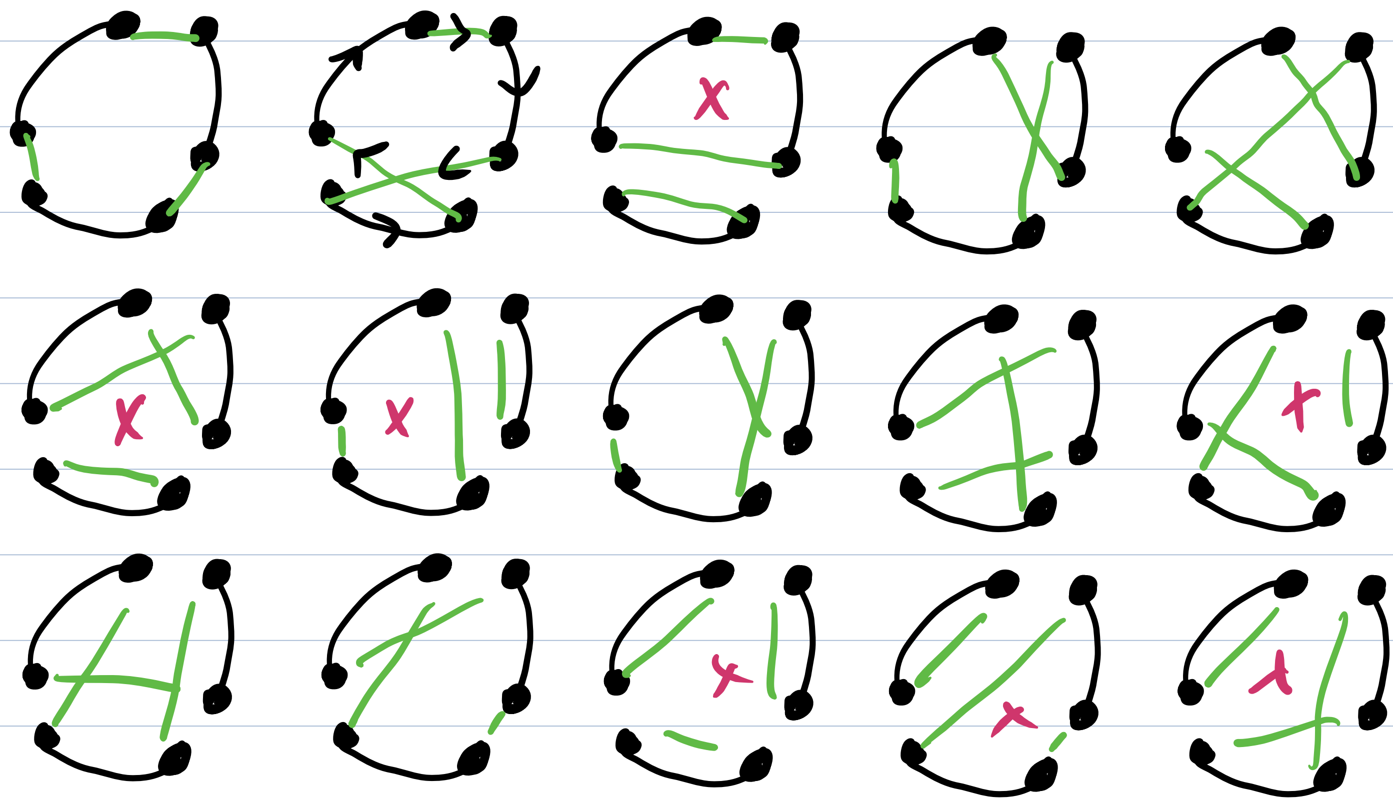
More generally, k-opt:
Delete k random edges and reconnect in the cheapest valid way possible.



$k=3$

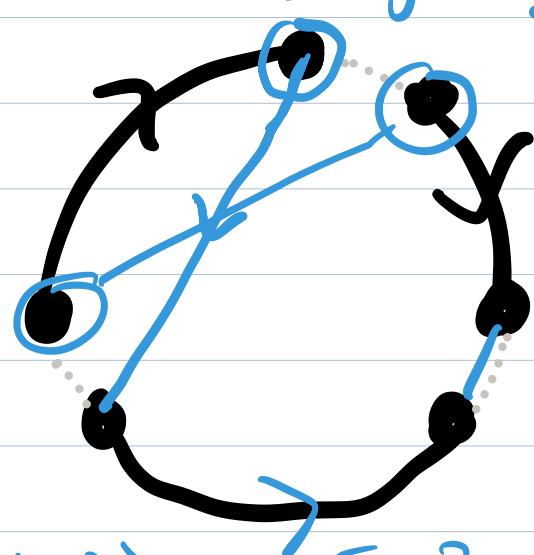


15 ways to reconnect



Why 15?

5 ways · 3 ways · 1 way
= 15



$\binom{6}{2}$

$\frac{(2k)!}{2^k \cdot k!}$

//

$(2k-1)(2k-3)\cdots 5\cdot 3\cdot 1 = (2k-1)!!$

8 dots: $7 \cdot 5 \cdot 3 \cdot 1 = 105$

$\binom{8}{2} = 28$

15 total reconnections, only 8 valid.

For things like H-C and SA, 3-opt is considered the best.

<u>Variant:</u>

Pick neighborhoods
$N_1(x), N_2(x), \ldots, N_d(x)$
Pick #s
$n_1, n_2, \ldots, n_d$

```
while True:
    for n_1 attempts:
        Pick s ∈ N_1(x)
        If s is better than x:
            x=s
        Repeat
    for n_2 attempts:
        Pick s ∈ N_2(x)
```

If s is better than x:

x = s

Repeat

Repeat for n_3 / N_3 and so on.