Call 7 Vehicle 3				
	Average objective	Best objective	Improvement (%)	Running time (seconds)
Random Search	1977054.4	1851768.0	43.65	0.1
Local Search	1531375.5	1477429.0	55.04	0.12
Simulated Annealing (old)	1531375.5	1477429.0	55.04	0.13
Simulated Annealing (with new operators)	1677178.6	1477429.0	55.04	0.41

Call 18 Vehicle 5				
	Average objective	Best objective	Improvement (%)	Running time (seconds)
Random Search	7039433.7	5792619.0	33.89	0.16
Local Search	3894102.7	3482990.0	60.24	0.22
Simulated Annealing (old)	3866492.3	3727829.0	57.45	0.23
Simulated Annealing (with new operators)	5157061.3	4439699.0	49.33	1.18

Call 35 Vehicle 7				
	Average objective	Best objective	Improvement (%)	Running time (seconds)
Random Search	18114779.4	16248192.0	11.32	0.24
Local Search	9846322.7	8821467.0	51.85	0.33
Simulated Annealing (old)	10438207.0	9545304.0	47.9	0.33
Simulated Annealing (with new operators)	12325463.9	10794120.0	41.09	3.47

Call 80 Vehicle 20				
	Average objective	Best objective	Improvement (%)	Running time (seconds)
Random Search	42211425.0	42211425.0	0.0	0.56
Local Search	20918229.2	19810961.0	53.07	0.81
Simulated Annealing (old)	20235612.8	19259583.0	54.37	0.83
Simulated Annealing (with new operators)	24180668.5	23258470.0	44.9	16.67

Call 130 Vehicle 40				
	Average objective	Best objective	Improvement (%)	Running time (seconds)
Random Search	75446687.0	75446687.0	0.0	0.95
Local Search	33429275.1	32125570.0	57.42	1.45
Simulated Annealing (old)	33219287.4	31618987.0	58.09	1.48
Simulated Annealing (with new operators)	35593256.8	34100207.0	54.8	37.26

Best solutions found with Simulated Annealing (with new operators)

Best solution found for Call 7 Vehicle 3

[1, 1, 0, 7, 7, 3, 3, 0, 5, 5, 6, 6, 0, 2, 2, 4, 4]

Best solution found for Call 18 Vehicle 5

[4, 4, 7, 7, 0, 15, 15, 3, 3, 0, 11, 11, 10, 10, 9, 9, 2, 2, 0, 12, 12, 14, 14, 0, 6, 6, 8, 8, 0, 1, 1, 5, 5, 13, 13, 16, 16, 17, 17, 18, 18]

Best solution found for Call 35 Vehicle 7

[2, 2, 0, 34, 34, 10, 10, 33, 33, 0, 25, 25, 30, 30, 9, 9, 0, 19, 19, 27, 27, 31, 31, 0, 6, 6, 16, 16, 15, 15, 26, 26, 0, 32, 32, 4, 4, 0, 28, 28, 13, 13, 29, 29, 1, 1, 0, 3, 3, 5, 5, 7, 7, 8, 8, 11, 11, 12, 12, 14, 14, 17, 17, 18, 18, 20, 20, 21, 21, 22, 22, 23, 23, 24, 24, 35, 35]

Best solution found for Call 80 Vehicle 20

[22, 22, 6, 6, 0, 62, 62, 5, 5, 38, 38, 0, 55, 55, 43, 43, 0, 47, 47, 19, 19, 0, 4, 4, 54, 54, 27, 27, 41, 41, 0, 33, 33, 29, 29, 75, 75, 0, 13, 13, 0, 26, 26, 12, 12, 70, 70, 35, 35, 0, 74, 74, 40, 40, 64, 64, 0, 48, 48, 24, 24, 58, 58, 0, 50, 50, 42, 42, 78, 78, 0, 34, 34, 2, 2, 0, 3, 3, 76, 76, 0, 20, 20, 63, 63, 31, 31, 8, 8, 0, 69, 69, 77, 77, 56, 56, 15, 15, 0, 46, 46, 18, 18, 0, 49, 49, 44, 44, 1, 1, 0, 45, 45, 0, 16, 16, 10, 10, 0, 66, 66, 32, 32, 25, 25, 0, 7, 7, 9, 9, 11, 11, 14, 14, 17, 17, 21, 21, 23, 23, 28, 28, 30, 30, 36, 36, 37, 37, 39, 39, 51, 51, 52, 52, 53, 53, 57, 57, 59, 59, 60, 60, 61, 61, 65, 65, 67, 67, 68, 68, 71, 71, 72, 72, 73, 73, 79, 79, 80, 80]

Best solution found for Call 130 Vehicle 40

[96, 96, 13, 13, 129, 129, 0, 84, 84, 128, 128, 5, 5, 0, 107, 107, 41, 41, 53, 53, 0, 7, 7, 116, 116, 75, 75, 0, 26, 26, 126, 126, 19, 19, 0, 37, 37, 102, 102, 2, 2, 29, 29, 0, 33, 33, 59, 59, 14, 14, 0, 68, 68, 78, 78, 0, 21, 21, 117, 117, 127, 127, 24, 24, 0, 50, 50, 109, 109, 0, 58, 58, 74, 74, 36, 36, 0, 114, 114, 92, 92, 40, 40, 0, 94, 94, 55, 55, 1, 1, 0, 108, 108, 31, 31, 0, 71, 71, 27, 27, 106, 106, 0, 103, 103, 28, 28, 61, 61, 20, 20, 0, 73, 73, 6, 6, 23, 23, 0, 104, 104, 9, 9, 0, 43, 43, 42, 42, 0, 47, 47, 39, 39, 124, 124, 0, 123, 123, 95, 95, 70, 70, 0, 25, 25, 99, 99, 0, 64, 64, 51, 51, 0, 101, 101, 56, 56, 86, 86, 0, 49, 49, 10, 10, 0, 69, 69, 11, 11, 0, 118, 118, 15, 15, 0, 76, 76, 110, 110, 113, 113, 0, 32, 32, 38, 38, 0, 88, 88, 0, 3, 3, 0, 77, 77, 125, 125, 8, 8, 0, 18, 18, 81, 81, 0, 115, 115, 45, 45, 0, 63, 63, 22, 22, 4, 4, 0, 65, 65, 87, 87, 0, 121, 121, 85, 85, 0, 57, 57, 44, 44, 79, 79, 0, 17, 17, 46, 46, 0, 16, 16, 130, 130, 0, 12, 12, 30, 30, 34, 34, 35, 35, 48, 48, 52, 52, 54, 54, 60, 60, 62, 62, 66, 66, 67, 67, 72, 72, 80, 80, 82, 82, 83, 83, 89, 89, 90, 90, 91, 91, 93, 93, 97, 97, 98, 98, 100, 100, 105, 105, 111, 111, 112, 112, 119, 119, 120, 120, 122, 122]

Transport all operator

Not transporting a call is the most expensive option for a call.
thus this operator aims to distribute all calls to a valid vehicle.
All not transported calls are sorted ascending according to number of vehicles that can pickup/deliver it,
to prevent a call that has many valid vehicles to steal a spot from a call with few valid vehicles.
The operator will try to find a feasible vehicle insert for a call.

Pseudo code:

```
==================================
calls = sort call asc by number of valid vehicles

for call in calls
   vehicles = valid vehicles for call
   for vehicle in vehicles
      insert call in vehicle
      if new solution is feasible
         return new solution

return original solution
==================================
```

Early search this operator is fast, because there are many feasible solutions in the neighbourhood.
Late search it is very slow, because it has to check all combinations of
not transported calls and valid vehicles and will probably return the original solution.

I think this operator leads to diversification,
because it moves a call from one vehicle (dummy) to another vehicle

Potential for improvement:
Sort vehicles by cost of transporting call

Reinsert most expensive operator

This operator aims to reduce objective of solution
by moving a call from one of the three most expensive vehicles to one of the three least expensive vehicles
To prevent getting stuck, the operator:

1. shuffles the order of the three most expensive vehicles before iterating,
2. shuffles the order of the the calls in that vehicle
3. shuffles the order of the three least expensive valid vehicles

Pseudo code:

```
==================================
fromVehicles = three most expensive vehicles
shuffle fromVehicles

for fromVehicle in fromVehicle
   calls = calls in fromVehicle
   shuffle calls

   for call in calls
      toVehicles = three least expensive vehicles
      shuffle toVehicles

      for toVehicle in toVehicles
         insert call in toVehicle
         if new solution is feasible
            return new solution
==================================
```

Early search this operator is very fast, but very ineffective,
because it will move calls random between vehicles, and not necessarily result in a better solution.
Late game the operator is time expensive, but can reduce objective of solution a lot.

This operator leads to diversification of the solution,
because it moves a call from one vehicle to another.

Potential for improvement:
Triple for loop yuck <|:^)
Reduce the randomness of the operator.
Improve choice of call to relocate and insert vehicle

Brute force vehicle operator

Transport all and reinsert from most expensive inserts call in the front/back of the vehicle.
This operator aims to find a feasible permutation of a vehicle with a lower cost than the original solution.
Because finding permutations is extremely time expensive, this operator is limited to changing vehicles with 2 to 4 calls

Pseudo code:

==========
vehicle = random vehicle with 2 to 4 calls

check objective for all permutations of calls in vehicle

return solution with best objective
==========

Early search will this operator often return the original solution,
because there are no vehicles with two or more calls or the selected vehicle is already optimal.
Late search this operator is time expensive, but always finds the best solution for a vehicle.

This operator leads to intensification of the solution,
because it aims to improve a part of the solution.

Potential for improvement:
Find a better way to pick which vehicle to brute force,
for example by choosing from a queue of last changed vehicles.
Operator is limited to vehicles with few calls