INF273 - Assignment 4

Elias Djupesland - edj001

April 2022

Call_7_Vehicle_3

	Average Objective	Best Objective	Improvement (%)	Running time(s)
Random Search	1601142	1406341	131	2
Local Search-1-insert	1398423	1134176	186	2
Local Search-2-exchange	1499984	1153343	181	2
Local Search-3-exchange	1167143	1134176	186	1
Simulated Annealing-1-insert	1174760	1134176	186	2
Simulated Annealing-2-exchange	1187829	1153343	181	2
Simulated Annealing-3-exchange	1150827	1134176	186	2
SA-new operators (equal weights)	1585184	1183864	174	11
SA-new operators (tuned weights)	1481538	1153343	181	8

Call_7_Vehicle_3 best found solution:
[4, 4, 7, 7, 0, 2, 2, 0, 1, 5, 5, 3, 3, 1, 0, 6, 6]

Call_18_Vehicle_5

	Average Objective	Best Objective	Improvement (%)	Running time(s)
Random Search	6548058	5957582	50	3
Local Search-1-insert	3653637	3126639	187	3
Local Search-2-exchange	4033065	3128697	186	3
Local Search-3-exchange	3860385	3300159	172	3
Simulated Annealing-1-insert	3574008	3160957	183	3
Simulated Annealing-2-exchange	3386457	2907556	208	3
Simulated Annealing-3-exchange	4054233	3587517	150	3
SA-new operators (equal weights)	4995638	4288752	109	20
SA-new operators (tuned weights)	4877541	4285997	109	17

Call_18_Vehicle_5 best found solution:
[4, 4, 15, 15, 17, 17, 13, 13, 0, 6, 6, 0, 8, 8, 18, 18, 12, 16, 16, 12, 0, 7, 7, 10, 10, 1, 1, 0, 9, 9, 5, 5, 14, 14, 0, 2, 3, 2, 11, 11, 3]

Call_35_Vehicle_7

	Average Objective	Best Objective	Improvement (%)	Running time(s)
Random Search	16351130	15256773	21	3
Local Search-1-insert	11166743	10298503	79	4
Local Search-2-exchange	9408228	8709613	111	4
Local Search-3-exchange	11151948	10074882	83	3
Simulated Annealing-1-insert	10845013	10207918	80	4
Simulated Annealing-2-exchange	9467266	8785585	109	4
Simulated Annealing-3-exchange	11311527	10579041	74	3
SA-new operators (equal weights)	13887164	12019326	53	30
SA-new operators (tuned weights)	13282516	12117022	52	29

Call_35_Vehicle_7 best found solution:
[34, 34, 27, 27, 32, 32, 20, 20, 0, 11, 11, 17, 21, 17, 21, 0, 16, 16, 30, 30, 18, 18, 29, 29, 0, 4, 4, 6, 6, 13, 13, 26, 26, 0, 12, 12, 31, 31, 0, 23, 23, 8, 1, 8, 1, 5, 5, 0, 14, 14, 35, 35, 10, 10, 0, 7, 3, 2, 33, 24, 25, 28, 9, 22, 2, 19, 15, 19, 28, 22, 24, 9, 15, 33, 25, 7, 3]

Call_80_Vehicle_20

	Average Objective	Best Objective	Improvement (%)	Running time(s)
Random Search	46770347	46770347	0	7
Local Search-1-insert	27472461	25677972	82	11
Local Search-2-exchange	26212503	24014045	95	11
Local Search-3-exchange	28633543	27489330	71	10
Simulated Annealing-1-insert	27324610	26142925	79	12
Simulated Annealing-2-exchange	25380285	24369039	92	12
Simulated Annealing-3-exchange	28952277	28006957	67	10
SA-new operators (equal weights)	26869322	25265099	85	144
SA-new operators (tuned weights)	26386896	24433914	91	157

Call_80_Vehicle_20 best found solution:
[32, 51, 51, 32, 0, 25, 25, 45, 45, 14, 14, 0, 54, 54, 69, 27, 69, 27, 0, 15, 15, 62, 17, 62, 17, 73, 73, 0, 61, 26, 61, 26, 0, 21, 21, 44, 44, 5, 5, 77, 77, 0, 8, 8, 0, 72, 72, 0, 63, 35, 63, 35, 0, 41, 41, 64, 64, 49, 49, 48, 48, 0, 1, 79, 1, 79, 0, 55, 55, 37, 36, 37, 36, 0, 46, 46, 0, 57, 57, 6, 6, 0, 4, 4, 34, 34, 67, 67, 31, 31, 47, 47, 0, 74, 42, 42, 74, 0, 53, 53, 13, 13, 75, 75, 0, 70, 68, 70, 68, 0, 9, 9, 58, 58, 52, 52, 0, 76, 76, 0, 40, 56, 80, 29, 65, 80, 11, 2, 22, 39, 78, 59, 59, 19, 38, 10, 19, 3, 29, 20, 66, 66, 50, 22, 43, 50, 28, 28, 7, 10, 39, 60, 23, 71, 7, 23, 33, 78, 60, 2, 24, 24, 71, 16, 43, 30, 56, 30, 65, 11, 18, 38, 20, 40, 33, 16, 18, 12, 3, 12]

Call_130_Vehicle_40

	Average Objective	Best Objective	Improvement (%)	Running time(s)
Random Search	76627567	76627567	0	13
Local Search-1-insert	44802308	43045851	78	24
Local Search-2-exchange	45165544	42406536	81	24
Local Search-3-exchange	48074838	45906170	69	19
Simulated Annealing-1-insert	45027593	42524438	80	20
Simulated Annealing-2-exchange	44680212	42777315	79	23
Simulated Annealing-3-exchange	47909850	46528570	65	20
SA-new operators (equal weights)	45807874	43515064	76	426
SA-new operators (tuned weights)	45652933	43025021	78	500

Call_130_Vehicle_40 best found solution:
[41, 41, 1, 50, 50, 1, 0, 126, 126, 22, 22, 0, 18, 91, 18, 91, 0, 15, 15, 53, 53, 0, 61, 61, 83, 83, 0, 32, 32, 82, 82, 0, 25, 25, 112, 112, 0, 102, 102, 130, 130, 0, 103, 103, 0, 118, 118, 30, 30, 0, 98, 98, 124, 109, 124, 109, 0, 21, 73, 21, 73, 0, 111, 81, 111, 81, 0, 20, 20, 0, 59, 51, 59, 51, 0, 106, 34, 106, 34, 0, 16, 16, 37, 48, 37, 48, 0, 67, 67, 68, 36, 68, 36, 0, 122, 101, 122, 101, 0, 99, 99, 10, 10, 0, 45, 45, 0, 19, 19, 128, 128, 0, 9, 9, 0, 75, 75, 0, 8, 8, 39, 39, 0, 119, 119, 0, 90, 90, 47, 47, 0, 29, 29, 0, 108, 108, 64, 64, 0, 52, 52, 0, 17, 17, 0, 88, 95, 95, 88, 0, 66, 66, 94, 94, 57, 57, 0, 23, 23, 7, 7, 0, 114, 117, 114, 117, 0, 13, 13, 0, 80, 80, 0, 43, 43, 0, 120, 120, 71, 71, 0, 38, 38, 0, 87, 78, 79, 129, 62, 76, 6, 69, 125, 46, 12, 105, 55, 58, 28, 4, 121, 60, 104, 58, 31, 110, 56, 107, 24, 24, 54, 28, 5, 27, 33, 70, 44, 129, 104, 97, 46, 69, 74, 12, 42, 49, 11, 14, 96, 123, 105, 65, 54, 72, 77, 89, 125, 27, 74, 2, 11, 31, 77, 65, 55, 35, 79, 97, 72, 49, 115, 84, 107, 44, 116, 113, 3, 14, 100, 60, 33, 85, 62, 93, 116, 76, 63, 56, 3, 127, 127, 123, 26, 93, 86, 92, 113, 96, 2, 115, 26, 42, 121, 84, 40, 87, 92, 70, 6, 35, 40, 110, 89, 5, 78, 85, 100, 63, 86, 4]

Call_300_Vehicle_90

	Average Objective	Best Objective	Improvement (%)	Running time(s)
Random Search	170784643	170784643	0	28
Local Search-1-insert	126656700	122426944	40	46
Local Search-2-exchange	134871638	131770840	29	43
Local Search-3-exchange	135157836	132363346	29	37
Simulated Annealing-1-insert	127721579	124025208	38	44
Simulated Annealing-2-exchange	136232183	133135640	28	43
Simulated Annealing-3-exchange	136056474	132914751	28	39
SA-new operators (equal weights)	124396172	120612282	42	1166
SA-new operators (tuned weights)	122141611	118985571	44	1865

Call_300_Vehicle_90 best found solution:
[0, 0, 0, 0, 0, 0, 0, 139, 139, 0, 70, 53, 53, 70, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 287, 287, 0, 219, 294, 294, 219, 0, 0, 0, 0, 0, 0, 77, 90, 90, 77, 0, 254, 89, 297, 254, 89, 297, 0, 0, 0, 68, 290, 68, 290, 0, 59, 59, 0, 0, 0, 104, 104, 0, 199, 173, 173, 199, 0, 28, 51, 28, 51, 0, 0, 212, 182, 275, 275, 212, 182, 0, 0, 34, 34, 0, 176, 44, 176, 44, 81, 81, 0, 101, 22, 101, 22, 0, 0, 0, 183, 183, 255, 255, 130, 130, 0, 253, 148, 148, 253, 0, 131, 127, 127, 131, 0, 94, 91, 94, 91, 0, 264, 189, 264, 189, 0, 300, 300, 12, 12, 0, 50, 187, 198, 187, 50, 198, 0, 222, 240, 222, 164, 240, 164, 0, 262, 226, 226, 262, 0, 5, 125, 125, 5, 0, 0, 268, 113, 268, 113, 0, 79, 14, 79, 14, 140, 140, 0, 117, 117, 0, 0, 280, 203, 42, 280, 42, 203, 0, 32, 293, 293, 27, 27, 32, 0, 185, 284, 165, 185, 165, 284, 0, 56, 56, 20, 20, 0, 105, 105, 41, 7, 41, 7, 0, 209, 209, 0, 3, 3, 0, 179, 103, 30, 179, 103, 30, 0, 115, 18, 18, 115, 0, 246, 246, 181, 73, 181, 73, 0, 167, 40, 167, 40, 0, 98, 98, 0, 210, 11, 210, 11, 0, 61, 74, 250, 61, 74, 250, 0, 220, 141, 120, 120, 141, 220, 0, 58, 102, 102, 58, 0, 118, 118, 55, 55, 0, 84, 17, 84, 17, 0, 0, 194, 235, 163, 194, 235, 163, 0, 114, 114, 108, 281, 281, 108, 0, 9, 95, 157, 95, 9, 157, 0, 69, 227, 146, 69, 146, 227, 0, 80, 80, 0, 270, 274, 286, 286, 274, 270, 0, 46, 46, 0, 0, 1, 213, 228, 228, 1, 213, 0, 62, 289, 62, 289, 0, 65, 288, 291, 65, 288, 291, 0, 37, 67, 145, 184, 159, 72, 87, 285, 24, 168, 223, 26, 54, 97, 38, 112, 200, 155, 47, 218, 116, 283, 216, 160, 208, 197, 193, 232, 279, 229, 128, 122, 269, 135, 282, 225, 110, 172, 112, 174, 265, 299, 159, 15, 261, 247, 170, 252, 169, 215, 299, 121, 178, 119, 21, 19, 238, 78, 168, 67, 6, 88, 111, 206, 156, 86, 247, 232, 82, 33, 295, 76, 136, 138, 82, 256, 169, 180, 205, 196, 47, 251, 273, 134, 29, 152, 282, 201, 211, 106, 10, 93, 144, 211, 296, 132, 23, 161, 277, 71, 166, 48, 154, 66, 155, 100, 180, 195, 124, 177, 109, 239, 49, 202, 92, 153, 119, 204, 242, 200, 71, 298, 129, 202, 245, 251, 158, 75, 170, 258, 64, 292, 93, 207, 8, 23, 151, 158, 174, 236, 142, 257, 230, 234, 263, 207, 35, 186, 142, 25, 248, 166, 35, 243, 133, 285, 271, 172, 206, 269, 36, 126, 83, 237, 100, 283, 143, 57, 99, 4, 107, 15, 76, 278, 191, 45, 16, 86, 111, 276, 175, 244, 97, 96, 49, 188, 249, 39, 217, 63, 296, 177, 109, 196, 45, 78, 154, 2, 192, 260, 29, 16, 245, 132, 31, 298, 192, 107, 215, 85, 263, 236, 204, 88, 143, 38, 66, 256, 36, 273, 48, 244, 137, 75, 195, 52, 224, 258, 126, 260, 54, 190, 4, 160, 6, 2, 135, 153, 124, 218, 134, 122, 31, 237, 37, 138, 279, 13, 205, 184, 106, 63, 243, 214, 241, 151, 265, 43, 248, 278, 231, 72, 137, 271, 217, 214, 33, 233, 123, 129, 171, 150, 25, 87, 231, 144, 272, 208, 57, 267, 21, 238, 193, 259, 123, 267, 150, 85, 223, 60, 257, 19, 201, 99, 239, 171, 52, 221, 229, 266, 162, 147, 252, 221, 292, 13, 64, 128, 10, 272, 156, 230, 188, 276, 161, 241, 259, 249, 149, 26, 116, 191, 190, 145, 234, 149, 24, 147, 266, 43, 92, 110, 175, 197, 136, 121, 233, 60, 152, 162, 8, 216, 277, 39, 224, 186, 242, 261, 178, 96, 225, 295, 133, 83]

Explanation of the operators

My operators did not perform as well as i wanted to. It might be because of not enough randomness, and that exchanging calls between two vehicles after that have been placed in a vehicle does not happen enough. I tried tuning the weights, but i did not see much better performance than setting them all to 1/3. The operators did however outperform the basic operators in the largest instance. Maybe they will perform better in combination with the others when the operators are weighted in ALNS.

dummy_to_best This operator chooses a random call from the dummy vehicle, and tries to put in in every vehicle and check cost. It places the call in the vehicle with the smallest cost. The idea is that calls are only placed in a vehicle that we know is good (at least somewhat good)

Pseudocode

```
def dummy_to_best(input_solution):
    if dummy == []:
        return input_solution

    dummy_call = select a random dummy call

    best_position = input_solution

    for vehicle in vehicles:
        placement = place dummy call in vehicle
        if feasibility(placement):
            placement_cost = cost_function(placement)
            if placement_cost < best_position:
                best_position = placement

    return best_position
```

route_shuffle The operator chooses a random vehicle and tests n_shuffles number of times a random shuffle of the calls to look for better cost. The idea is to test more combinations of call order.

Pseudocode

```
def route_shuffle(input_solution):
    random_vehicle = choose a random vehicle

    if vehicle is empty:
        return input_solution

    n_shuffles = 3

    best_shuffle = input_solution

    for i in range(n_shuffles):
        current = random shuffle vehicle calls

        if feasibility(current):
            current_cost = cost_function(current)
            if current_cost < best_shuffle:
                best_shuffle = current

return best_shuffle
```

two_exhange_capacity The operator is similar to two exchange, but it picks a vehicle with small cargo space as the removal vehicle, and a weighted pick of the largest cargo capacity vehicles as the insertion vehicle. The idea is to move cargo from small to larger vehicles.

Pseudocode

```
def two_exhange_capacity(initial_solution):

    biggest_capacity, smallest_capacity = split vehicle list in two,
    by cargo capacity


    if all smallest vehicles are empty:
        return initial_solution

    frm = smallest_capacity[0]
    for vehicle in smallest_capacity:
        if vehicle not empty :
            frm = vehicle
            break

    # Weighted pick of largest capacity vehicles
    weights = [1/n for n in 2...len(biggest_capacity)+2)]
    to = choice(biggest_capacity, weights=weights)

    Swap a random call frm - to
    Shuffle to vehicle

    return new_solution
```