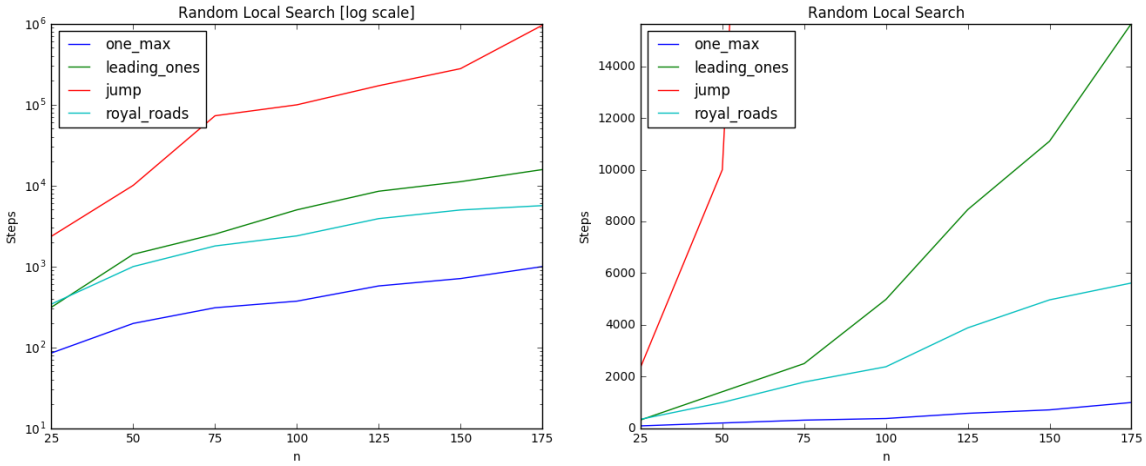
Analysis

1.) Random Local Search



The algorithm takes much more time for "jump" than for each other problem.

In comparison to the other algorithms it always needs longer for "leading_ones" than "royal_roads".

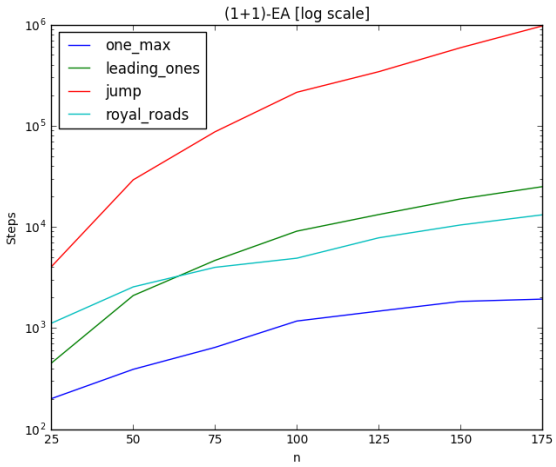
"one_max" is always executed as the fastest one. This might be because there is only one local (and also global) maximum and the steps are smaller. In contrast to that "royal_roads" might return the same value for much more combinations because the bits are splitted into groups. "jump" is the hardest problem because the space around the global maximum always returns the same value.

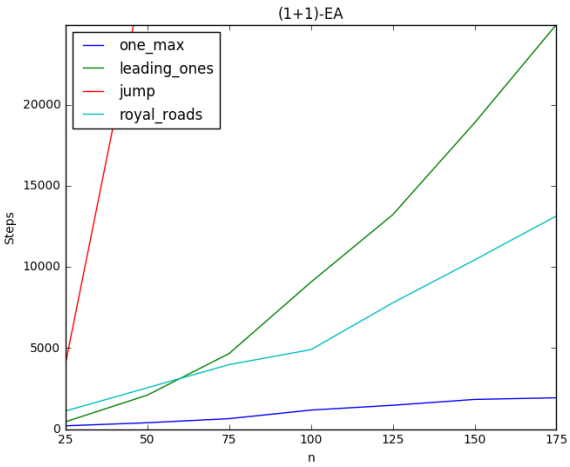
Unfortunately we couldn't evaluate our "bin_val" implementation because it took too long (no results for $n = 25$ after 24 hours). This was probably caused by the bad complexity of our implementation. All values are defined as integer arrays instead of bit arrays. Instead of requiring 25 bits for $n = 25$ we need $25 * sizeof(int)$ bits, which might be much larger.

Approximated complexity (in O-Notation):

- one_max: $\mathcal{O}(n)$
- leading_ones: $\mathcal{O}(n^c)$
- jump: $\mathcal{O}(c^n)$
- royal_roads: $\mathcal{O}(n)$

2.) Evolutionary Algorithm - (1+1)-EA



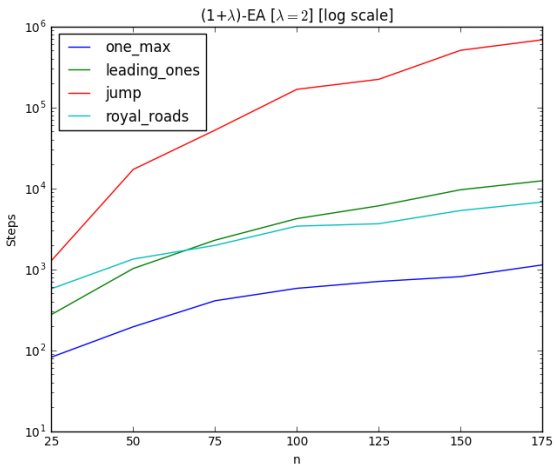


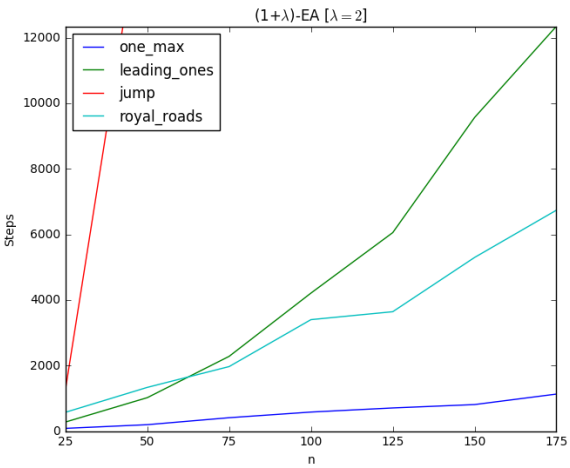
This algorithm takes just as long as RLS for all problems.

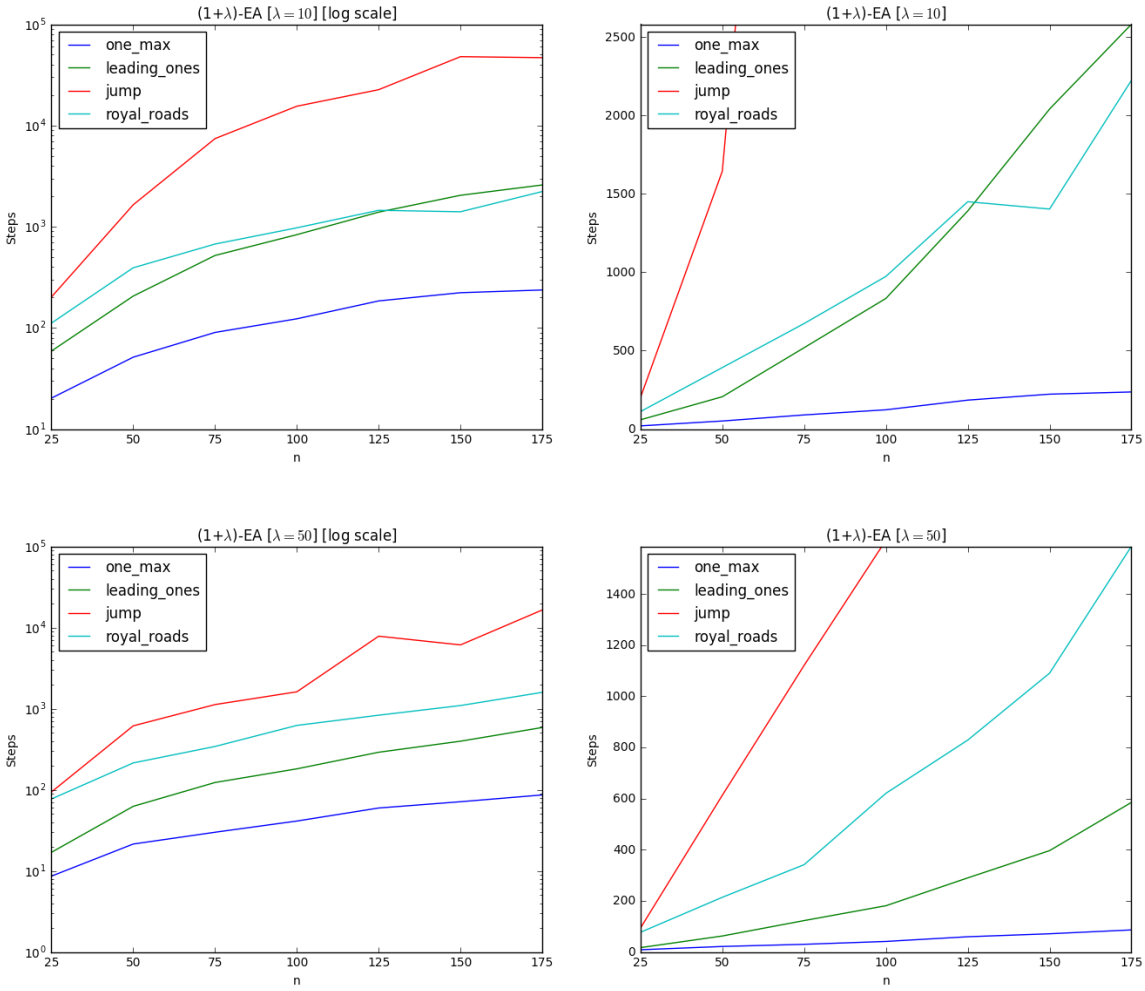
Approximated complexity (in O-Notation):

- one_max: $\mathcal{O}(n)$
- leading_ones: $\mathcal{O}(n^c)$
- jump: $\mathcal{O}(n^c)$
- royal_roads: $\mathcal{O}(n^c)$

3.) Evolutionary Algorithm - (1+λ)-EA







Again the algorithm takes much more time for "jump" than for each other problem - independent of λ.

Interestingly the costs sometimes decrease between two values n_i and n_{i+1} This might be due to the randomness of finding the solution.

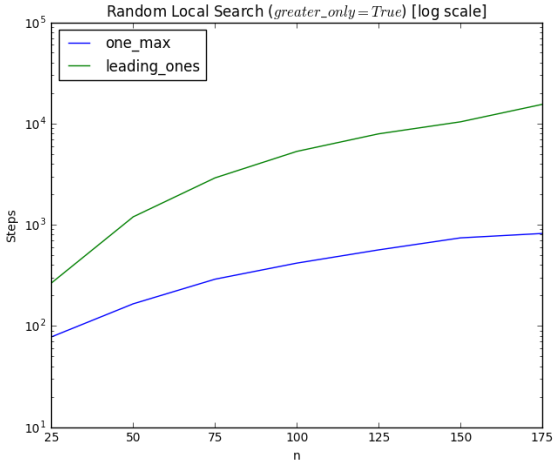
With an increasing λ the algorithm becomes much faster (by factor ~10) for all problems. Furthermore the costs of "leading_ones" shrink faster than the costs of "royal_roads" with an increasing λ.

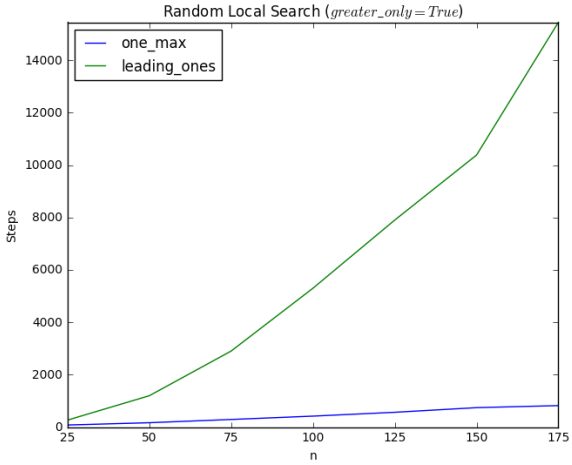
In comparison to the other algorithms this algorithm with $\lambda = 50$ is the only one with a relative good performance for "jump".

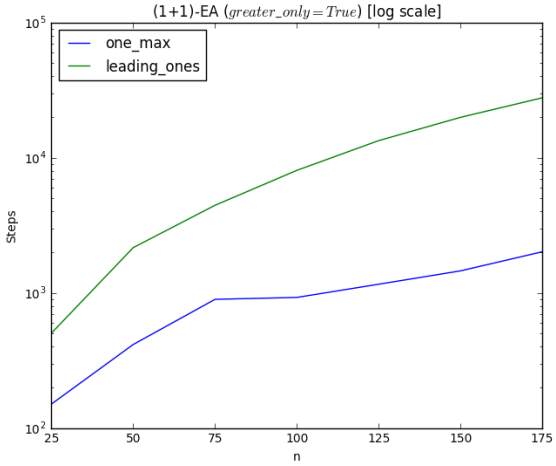
Approximated complexity (in O-Notation):

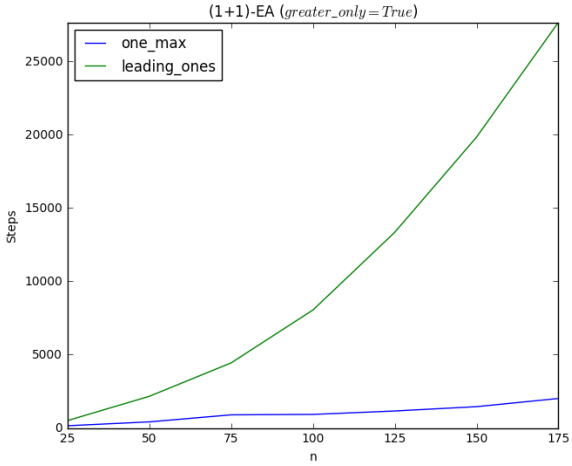
- one_max: $\mathcal{O}(n)$
- leading_ones: $\mathcal{O}(n^c)$
- jump: $\mathcal{O}(n^c)$
- royal_roads: $\mathcal{O}(n^c)$

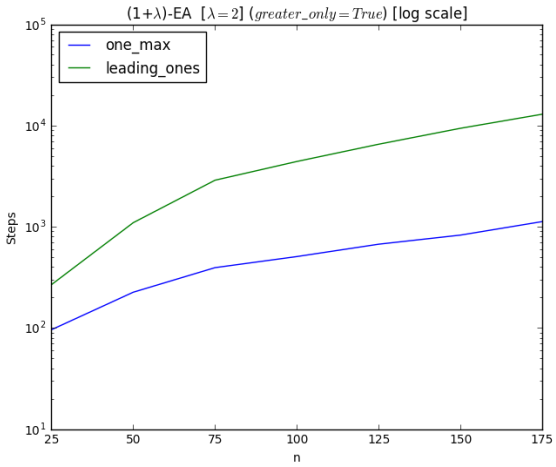
Results with $greater\_only = True$

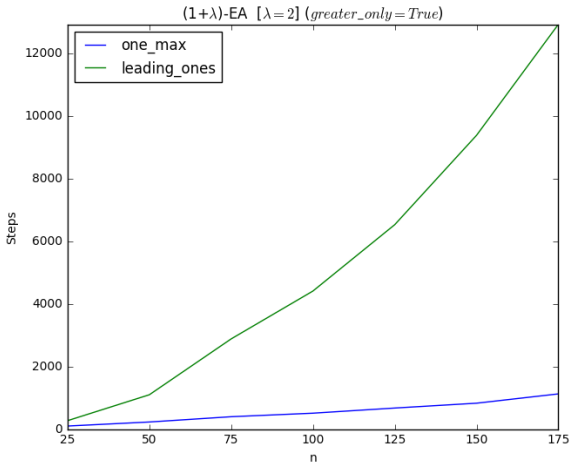


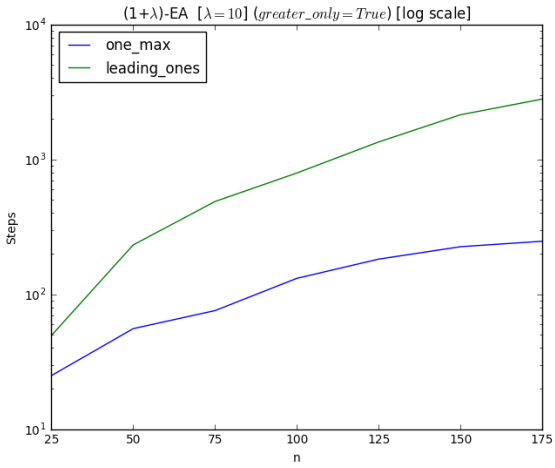


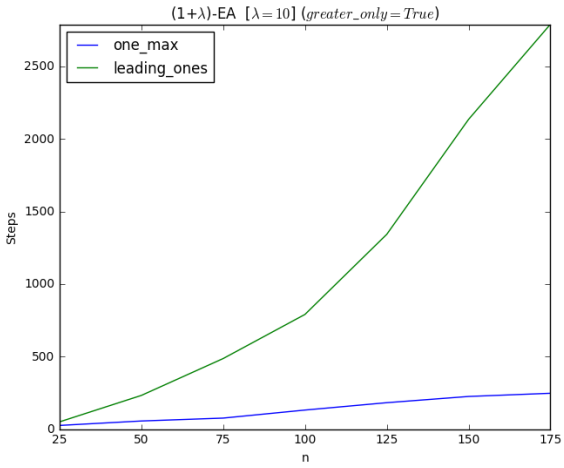


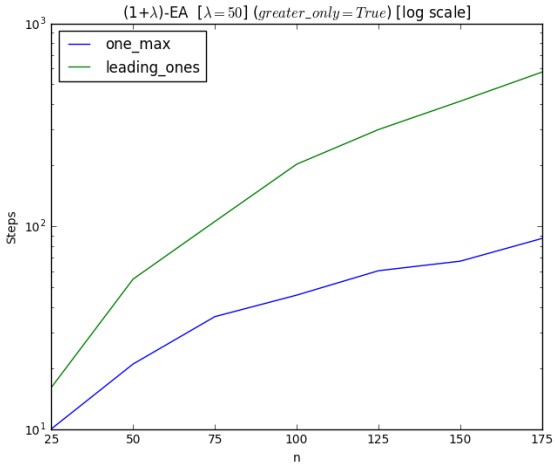


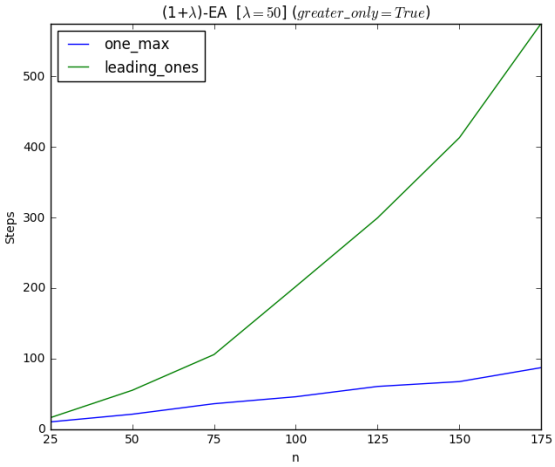












With having this little change in our code, almost all algorithms couldn't find the solution for three out of all five problems. Even just for $n = 25$ we couldn't find the optimal value within a few hours.

By looking at the results we got, we can see that the performance of the algorithms is the same for "one_max" & "leading_ones" - regardless of setting $greater\_only$ to $True$ or $False$.