

Выборы с несколькими победителями.
Выбор фильмов. Исследование правил
поиска победителей

Multiwinner voting rules study in movie selecting class

Multiwinner voting (формально)

Выборы (election): пара $E = (V, C)$

Множество избирателей: $V = \{v_1, \dots, v_n\}$

Множество кандидатов: $C = \{c_1, \dots, c_m\}$

Множество победителей: $W = \{w_1, \dots, w_k\} \subset C$

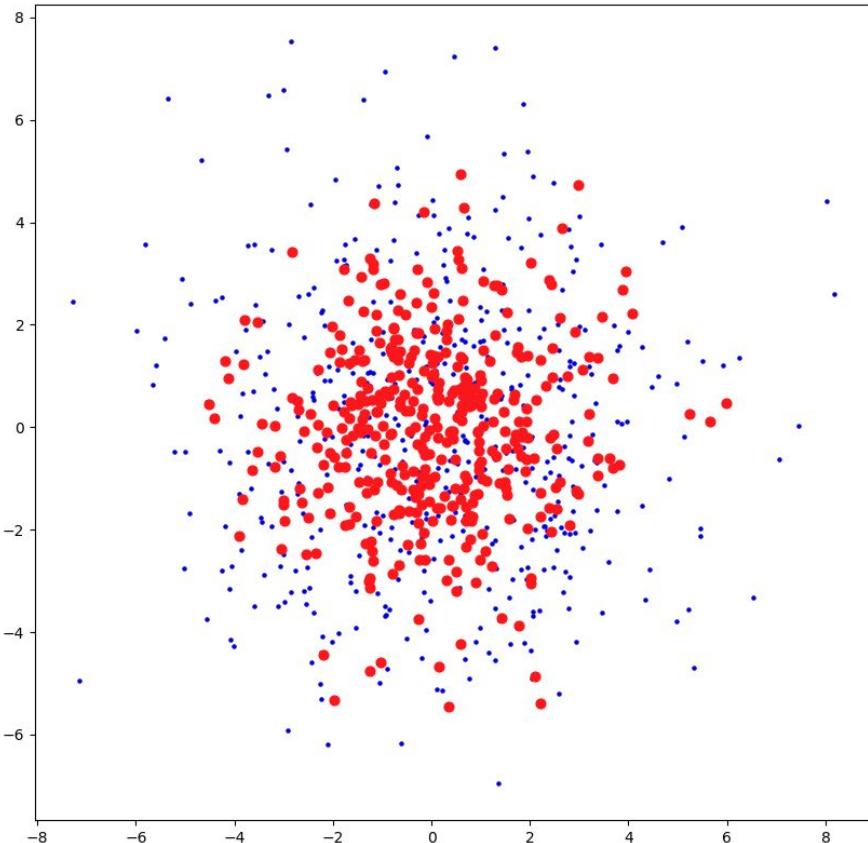
Правило поиска комитета: $R : (E, k) \rightarrow W \subset C, |W| = k$

Multiwinner voting (неформально)

Выборы *избирателями* среди множества кандидатов, k -подмножество которого объявляется победителями голосования

Multiwinner voting

Будем использовать визуализацию в двумерном пространстве, где каждому избирателю и кандидату соответствует определённая точка в этом пространстве. Измерения могут быть любые, в политике это может быть степень поддержки какого-то закона, для фильмов какой-то векторизованный признак, главное, что кандидат имеет точку зрения, выраженную в двух параметрах, и избиратель имеет свою точку зрения и проголосует в первую очередь за кандидата со схожей точкой зрения



Movie selection

- Пример: Выбор некоторого количества фильмов для полёта в самолёте, чтобы так или иначе удовлетворить желания всех пассажиров
- Пример: Выбор среди возможных точек для постройки к определённых зданий (пожарных участков) таким образом, чтобы как можно больше жителей имело данное строение поблизости
- Voter заинтересован, чтоб победил ближайший к нему кандидат
- Для вотера не имеют значения другие победители кроме ближайшего
- Каждый победитель может обслуживать любое число вотеров, для которых он является ближайшим, остальные вотеры не имеют для него значения

Почему победа нескольких близких кандидатов нежелательна для movie selection

В других классах, таких как shortlisting или parliamentary elections, если два кандидата близки, они скорее всего либо оба побеждают, либо оба нет.

В movie selection два близких кандидата работают примерно на одно и то же подмножество голосующих, каждому из которых будет вполне достаточно победы одного из них, второй их не сильно волнует, таким образом, при победе двух близких кандидатов один из них не сильно увеличивает общее удовлетворение избирателей, но занимает место в комитете

Алгоритмы подсчёта (Voting rules)

- Простые
 - SNTV
 - Bloc
 - k-Borda
- Сложные
 - STV
 - CC
 - Monroe
- Приближённые
 - Approximate Variants of Monroe and CC.
 - Optimization rule

Plurality score

Plurality score кандидата c - количество избирателей, поставивших c на первое место в своём списке

$$PS(c_i) = \sum_{j=1}^n [pos_{v_j}(c_i) = 1]$$

k-Approval score

K-approval score кандидата c - количество избирателей, поставивших c среди первых k позиций

$$AS_k(c_i) = \sum_{j=1}^n [pos_{v_j}(c_i) < k]$$

Borda score

Borda score кандидата c - сумма его позиций в списках всех избирателей

$$BS(c_i) = \sum_{j=1}^n (|C| - pos_{v_j})$$

$$BS(c_i) = \sum_{j=1}^n pos_{v_j}$$

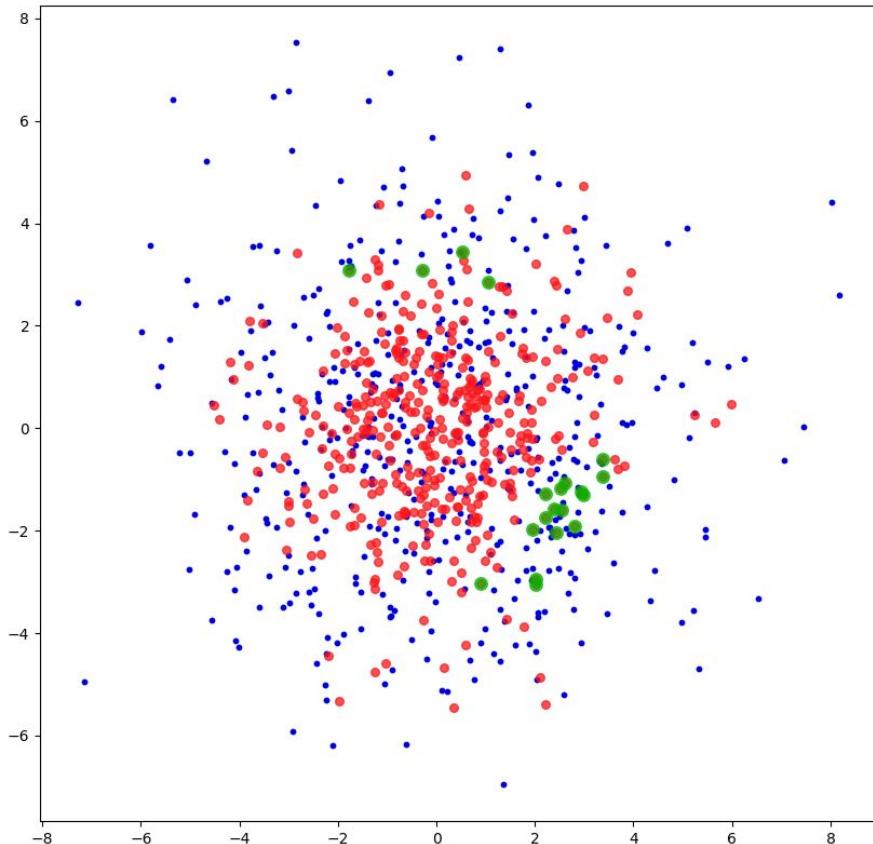
S-score

Взглянем на предыдущие методы подсчёта очков определённого кандидата так: за позицию в каждом списке даётся определённое число очков, зависящее от позиции, например, в случае Plurality score за первую позицию даётся 1 очко, за остальные 0. В случае k-approval score за первые k даётся по одному очку, за остальные - 0. В Borda score даётся $m - pos_{v_j}(c_i)$ очков (либо просто $pos_{v_j}(c_i)$, но тогда чем меньше - тем лучше).

Определим вектор $\mathbf{s} = \{s_1, \dots, s_m\}$, тогда введём некий универсальный метод, где s -score: $ss(c) = \sum_{i=0}^n s_{pos_{v_i}}(c)$, то есть за первое место в списке даём s_1 очков, за второе s_2 , и так далее

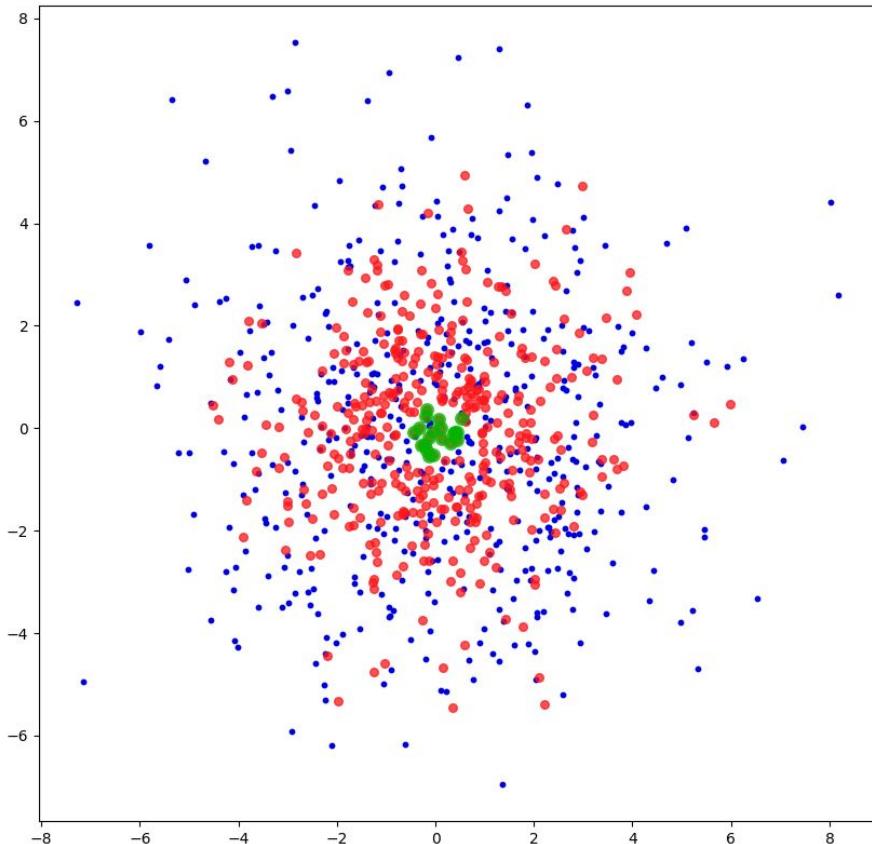
Bloc (k-approval)

Победителями объявляются k кандидатов с наибольшими значениями k-approval score



k-Borda

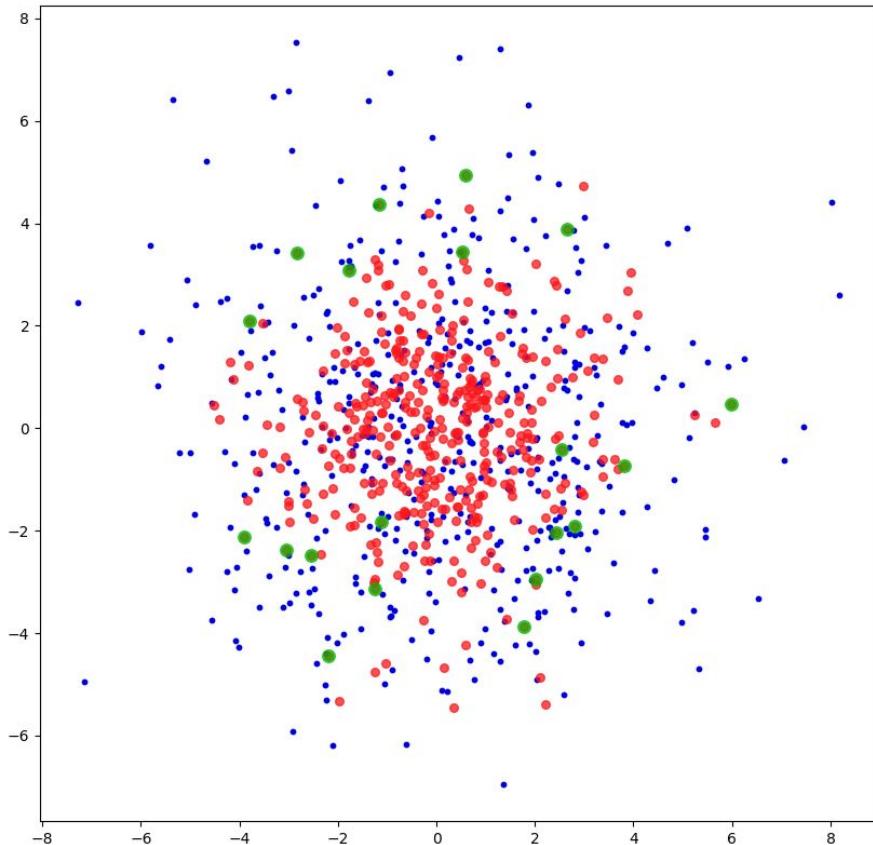
Победителями объявляются k кандидатов с наибольшими значениями k-Borda score



SNTV (k-plurality)

Single NonTransferable Vote

Победителями объявляются k кандидатов с наибольшими значениями plurality score



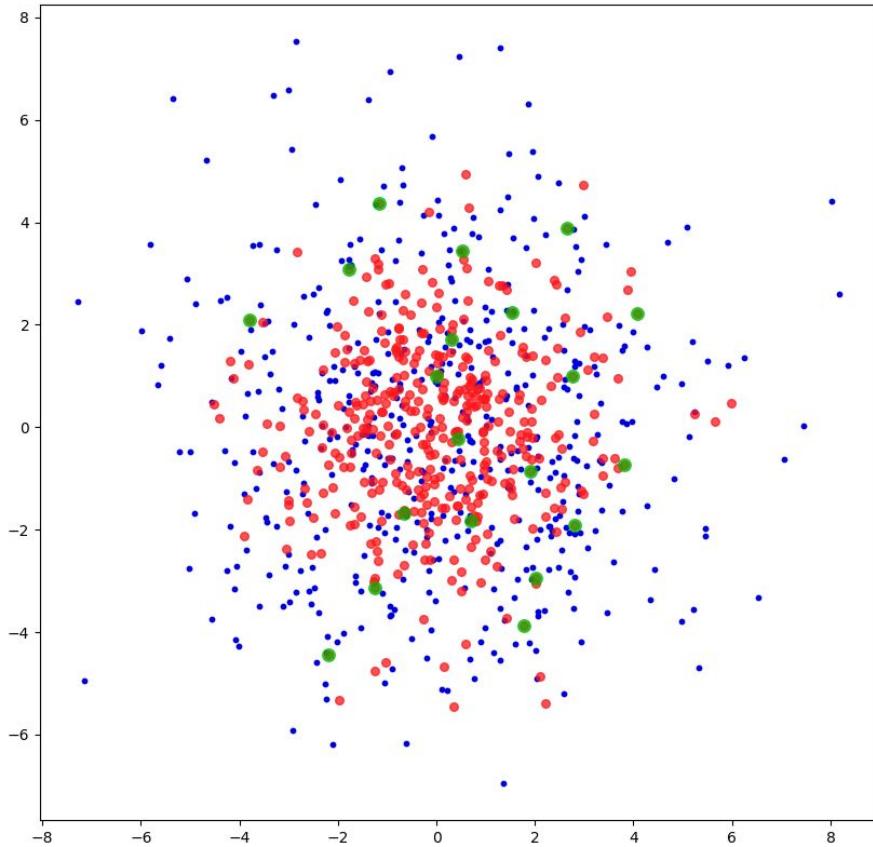
STV

Single Transferable Vote

Поэтапный алгоритм: Если существует кандидат с

$PS \geq q = \frac{|V|}{k+1} + 1$ (q - drop quota), то включаем c в комитет, удаляем q списков, где c первый и удаляем c из оставшихся списков

Если такого кандидата не существует, удаляем из всех списков кандидата с наименьшим PS



CC

CC approx

Monroe

Monroe approx

Пристальное рассмотрение

При не слишком глубоком анализе задачи становится понятно, что она имеет нечто общее с задачей дискретной оптимизации *Warehouse problem*, а значит задачу можно решить методами дискретной оптимизации

Discrete optimization

Введём некоторую меру удовлетворённости избирателей (или же меру качества работы алгоритма). Очевидно, что чем ближе к избирателю ближайший из выбранных кандидатов, тем он довольнее, остальные же победители особо не интересуют этого избирателя. В таком случае можно считать среднее расстояние (сумму расстояний)* от избирателя до ближайшего победителя:

$$Score(E) = \sum_{i=1}^n \min_j (||v_i - w_j||)$$

*среднее эквивалентно сумме с точностью до константы

Discrete optimization

Такая мера плоха тем, что сильно зависит от изначального расстояния между избирателями и кандидатами, то есть от плотности точек в пространстве может существенно меняться порядок данного значения, кроме того, любые неравномерности распределения также будут сильно влиять на значение. Поэтому можно изменить подход, и считать не абсолютное значение удовлетворения избирателя, а относительное, то есть отношение расстояния до ближайшего победителя к расстоянию до ближайшего кандидата:

$$Score(E) = \sum_{i=1}^n \frac{\min_j(||v_i - w_j||)}{\min_j(||v_i - c_j||)}$$

Discrete optimization

Таким образом, получили задачу дискретной оптимизации:

$$Object function : f = \sum_{i=1}^n \frac{\min_j(||v_i - w_j||)}{\min_j(||v_i - c_j||)}$$

$f \rightarrow \max$
subject to

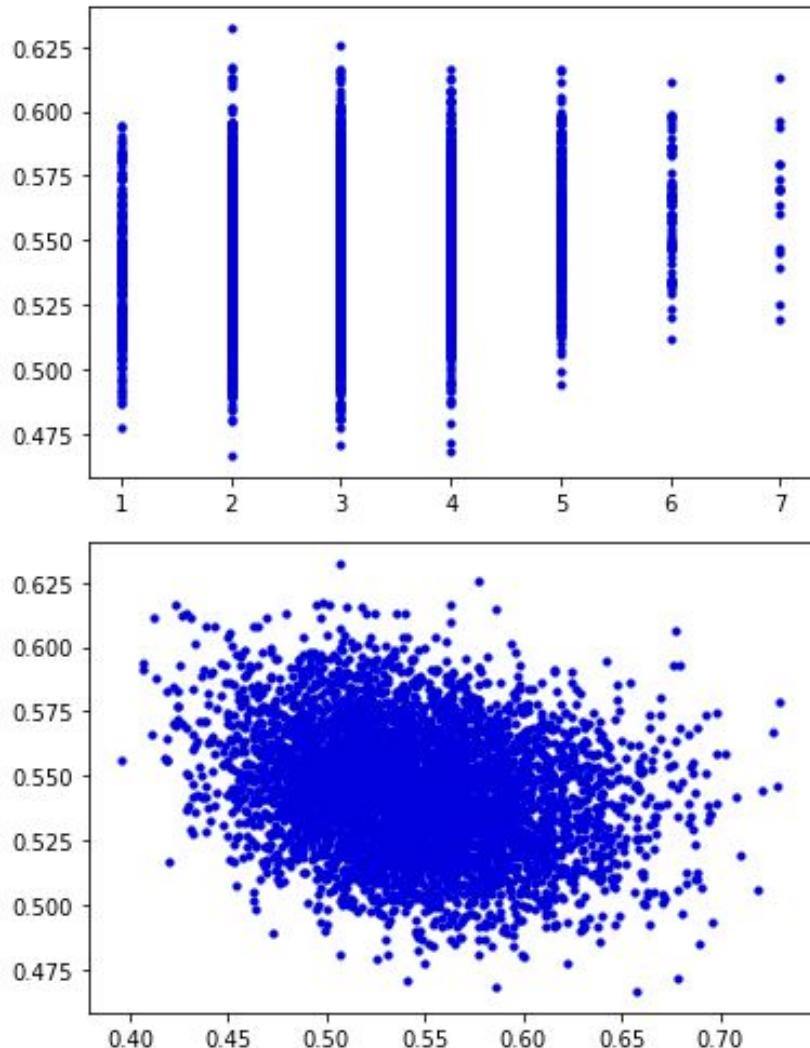
$$\begin{cases} \sum_{j=1}^m x_j = k, \\ x_j \in \{0, 1\}, \quad j = 1, \dots, m \end{cases}$$

Здесь вектор \mathbf{x} - decision variables, то есть при $x_j = 1$ кандидат c_j попадает в список победителей голосования, иначе - нет

Discrete optimization: pruning the search space

Очищение поискового пространства от
“плохих” кандидатов

- Удаление кандидатов с $PS = 0$
- Удаление кандидатов с $BS > 0.7|V||C|$
- Удаление кандидатов расположенных
ближе 0.75MeanDist к кандидату с
большим PS



Discrete optimization

В качестве первого метода возьмём local search или же neighbouring:

Сначала необходимо каким-то способом выбрать комитет

Далее для каждого победителя рассматриваем его соседей - невыбранных кандидатов в определённом радиусе вокруг, считаем для каждого из них, если передать место в комитете от текущего победителя текущему кандидату, таким образом находим в этом радиусе лучшего кандидата, увеличивающего скор наибольшим образом

Это должно работать, так как в классе movie selecting голосование довольно локальное: избирателя волнуют ближайший кандидат и ближайший победитель, если сменить внутри небольшого радиуса победителя, это не изменит скор у избирателей, находящихся настолько далеко, что ни один из рассматриваемых кандидатов для них не является ближайшим, при этом внутри своего радиуса можно неплохо изменить ситуацию и определить победителя более оптимально, остаётся одна небольшая проблема....

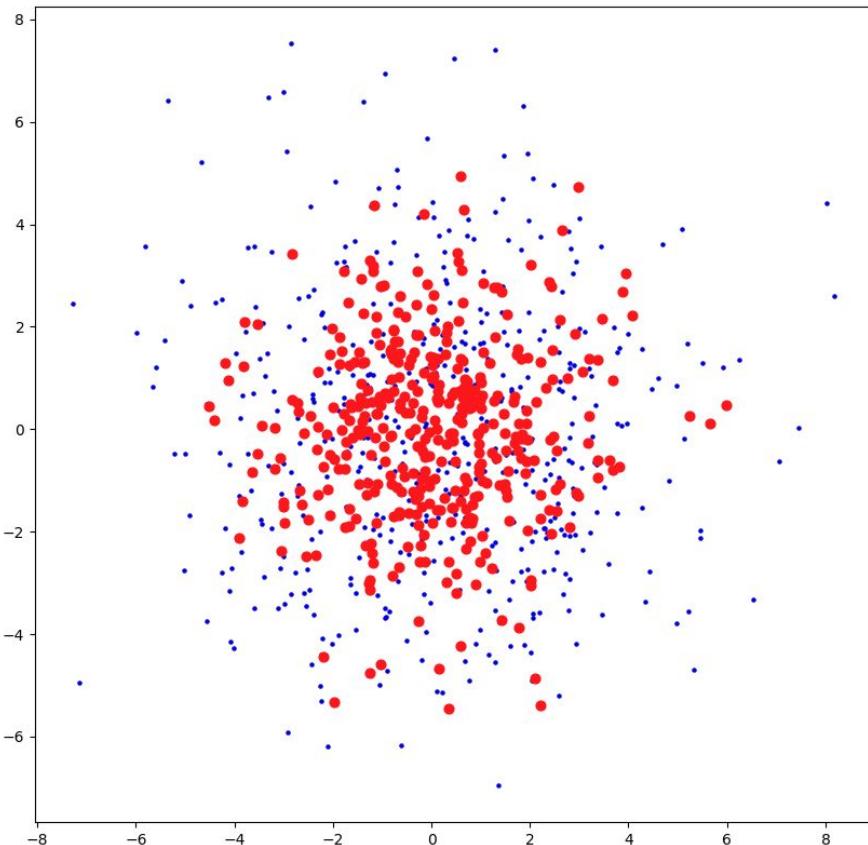
...как выбрать начальный комитет?

- Случайная выборка размера k из множества кандидатов

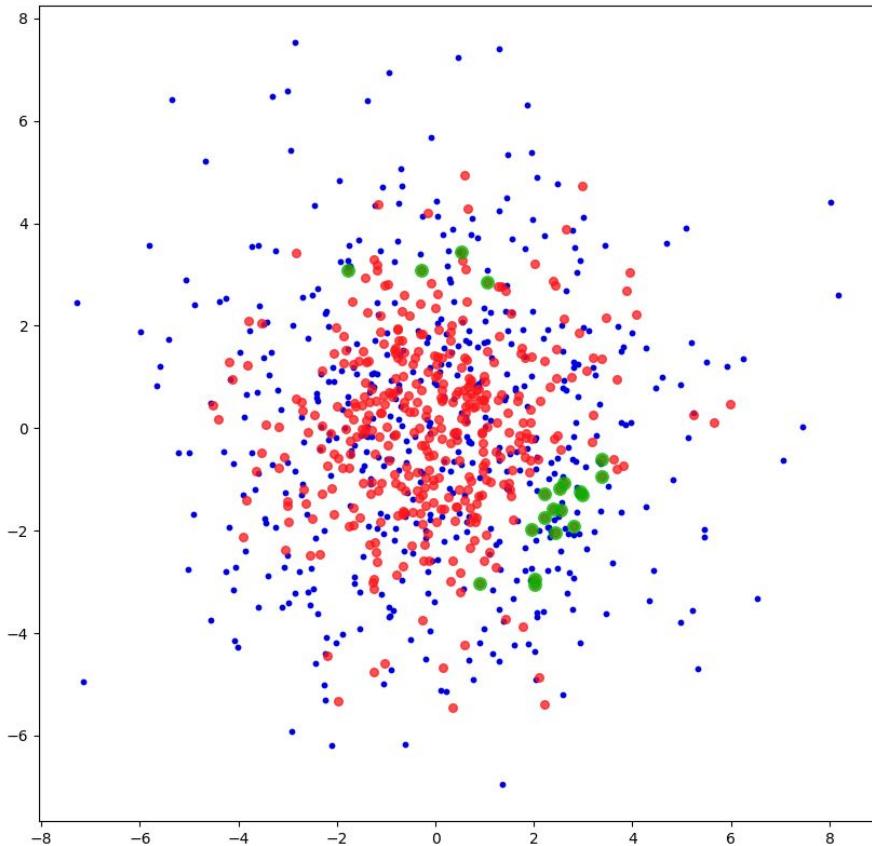
Свойства правил

```
1 Vamount = 500  
2 Camount = 400  
3 ComAmount = 20  
4 k = ComAmount
```

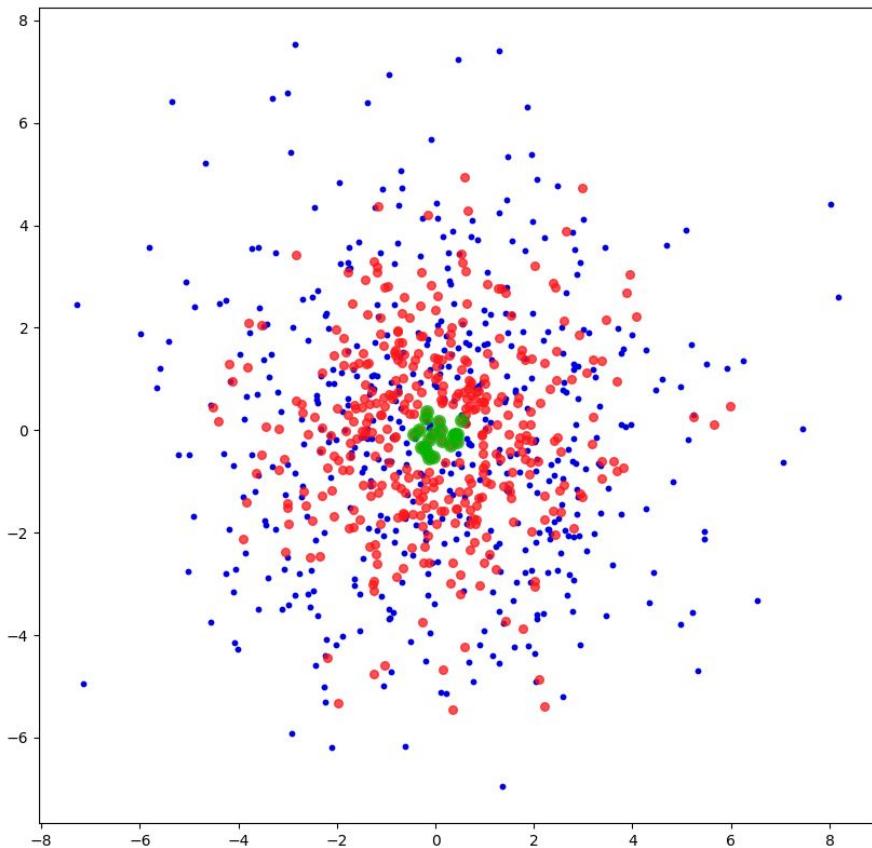
```
1 V = VotGen(Vamount, bounds = 7)  
2 C = CandGen(Camount, 'normal', bounds = 3)
```



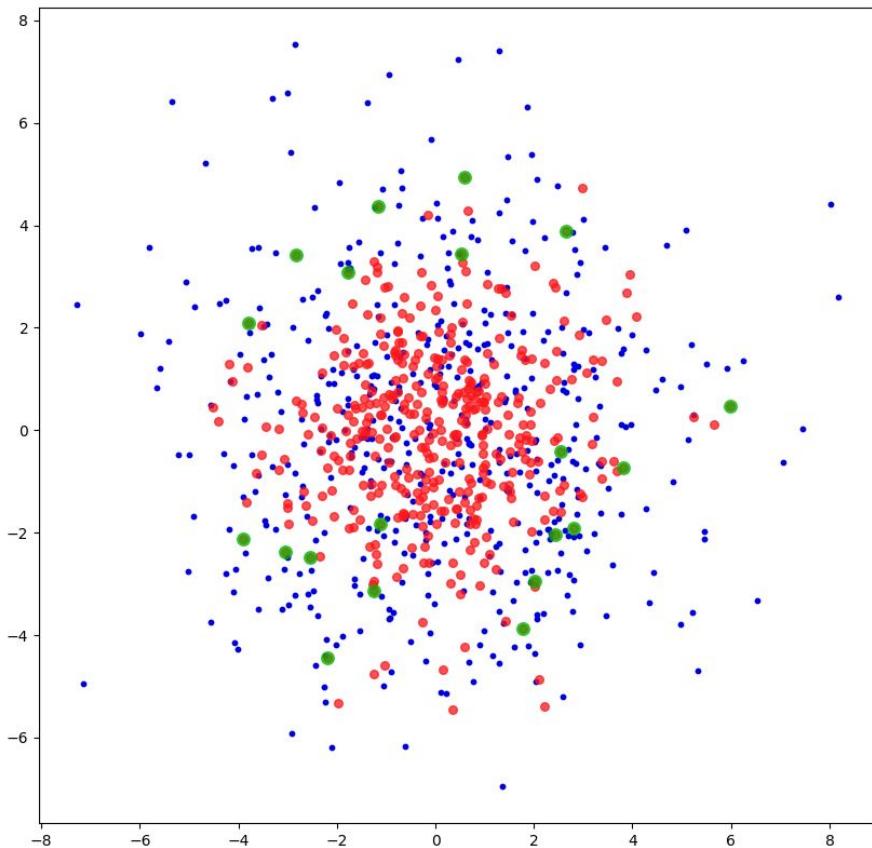
Score of k-Approval rule is 0.312968912153284
median Score of k-Approval rule is 0.1668916571772276



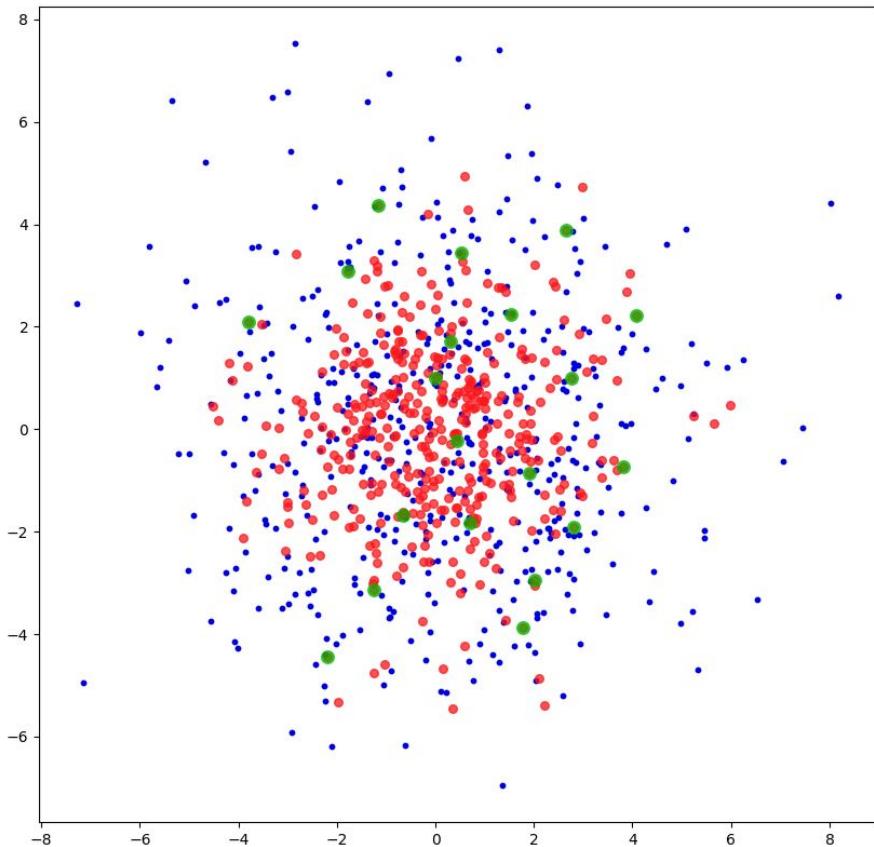
Score of k-Borda rule is 0.1680509745140211
median Score of k-Borda rule is 0.12568051402406705



Score of SNTV rule is 0.4548458441015673
median Score of SNTV rule is 0.25883505143915875



Score of STV rule is 0.47923452993148685
median Score of STV rule is 0.37966224530429926



Близкие кандидаты

Рассмотрим случай, когда существует пара кандидатов (c_1, c_2), расстояние между которыми мало относительно среднего расстояния между кандидатами. В таком случае в рейтингах большинства избирателей c_1 и c_2 будут располагаться рядом, посмотрим, как это отразится на победителях в разных правилах подсчёта:

- K-approval: так как здесь имеет значение, насколько часто кандидат попадает в первые k позиций, с большой вероятностью c_1 и c_2 будут либо вместе попадать в первые k позиций, либо не попадать, поэтому для этого правила большая вероятность того, что близкие кандидаты победят вместе (либо вместе не победят)
- k-Borda: здесь схожая с предыдущим правилом ситуация, так как имеют значения позиции кандидата в списках избирателей, а у близких кандидатов в большинстве списков будут близкие позиции
- SNTV: в этом правиле важны только первые позиции в списках избирателей, а близкие кандидаты будут конкурировать друг с другом за первые места у ближайших избирателей, значит с большой вероятностью ни c_1 , ни c_2 не войдут в комитет.
- STV: здесь больше шансов пройти для одного из двух близких кандидатов, так как один из них может быть удалён из-за малого PS, тогда все голоса перейдут ко второму как к ближайшему по позиции

Посмотрим некоторые примеры определения комитета

Distribution

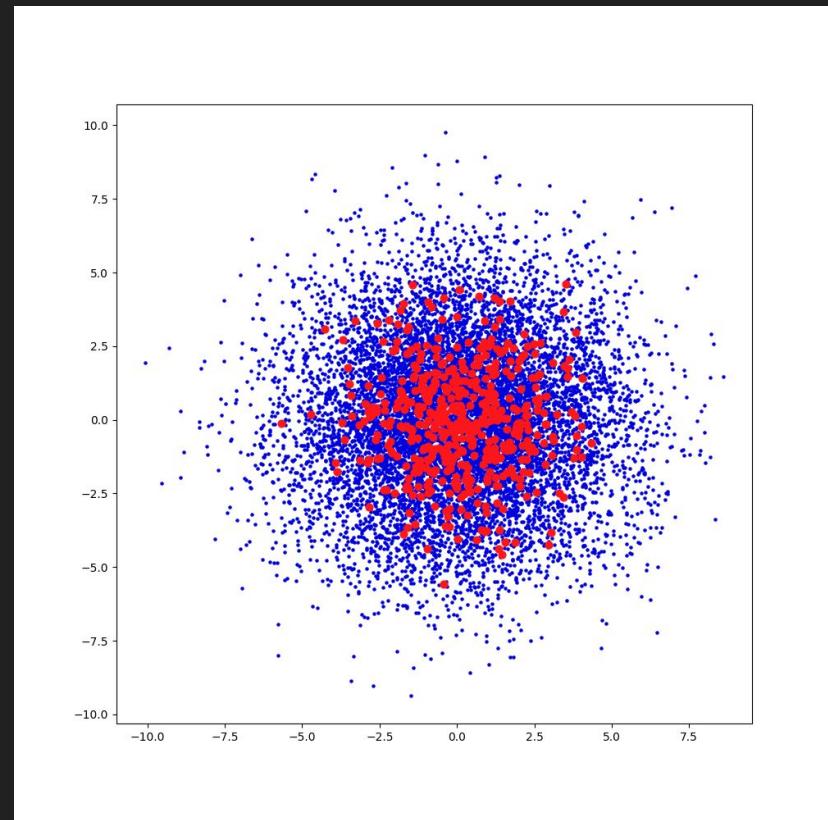
10000 voters

500 candidates

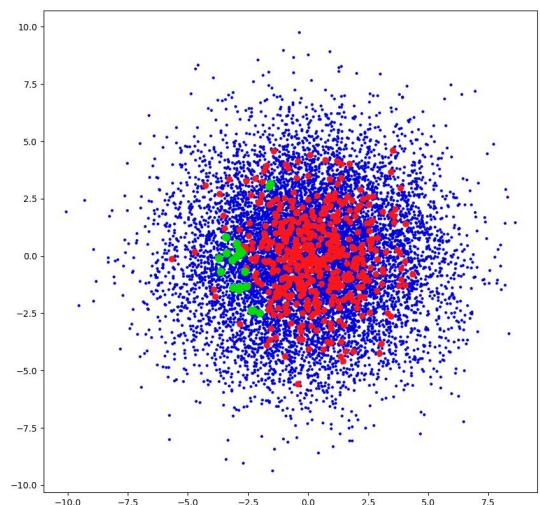
20 in committee

voters $\sim N(0, 7)$

candidates $\sim N(0, 3)$

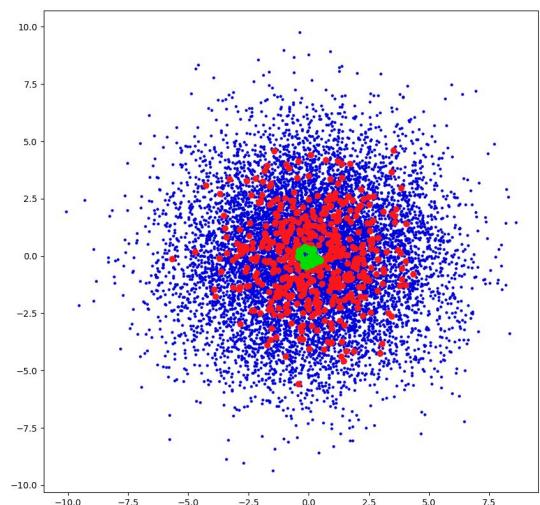


Bloc



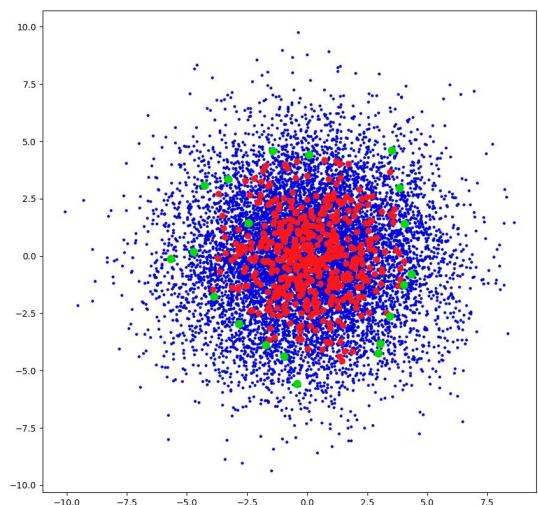
0.121

k-Borda



0.043

SNTV



0.298

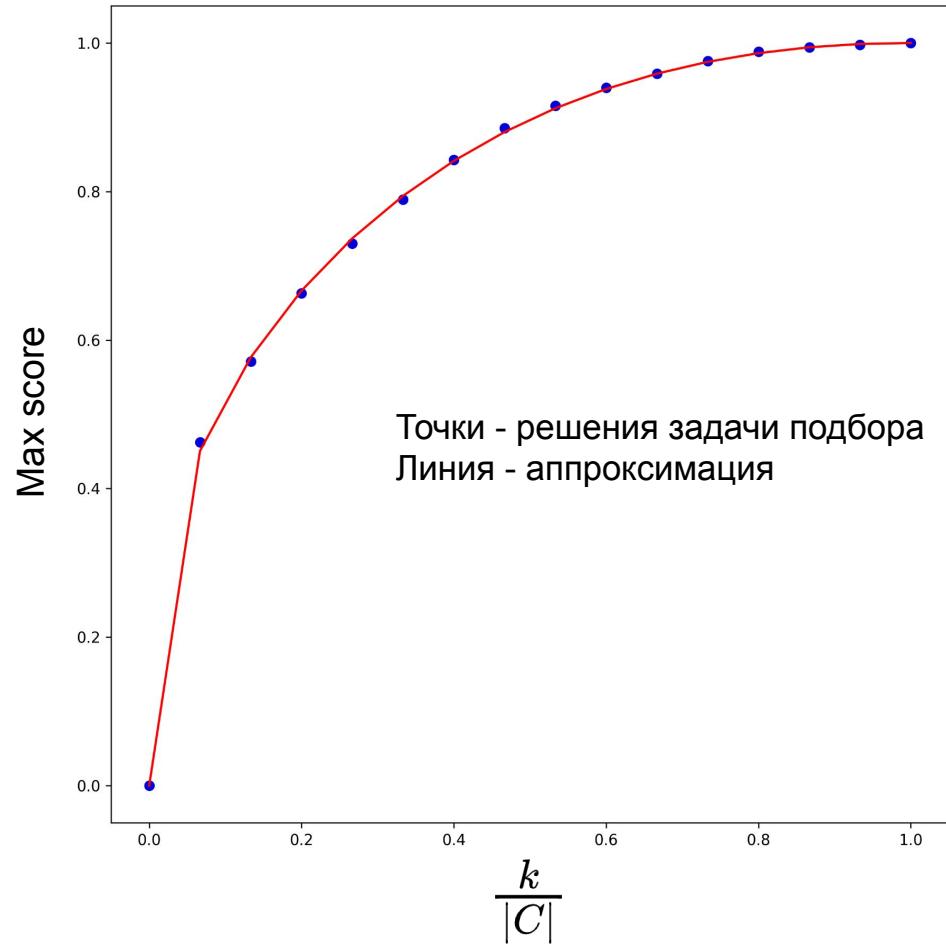
Зависимость максимально достижимого значения Score от отношения числа победителей к числу кандидатов

Кажется логичным, что чем больший процент кандидатов побеждает, тем выше должна быть удовлетворённость избирателей, ведь добавление каждого нового победителя увеличивает скор у избирателей, как для тех, у кого он был ближайшим кандидатом, так и для тех, чей ближайший победитель дальше нового члена комитета, но пока непонятен конкретный характер зависимости, ведь распределение случайно, случайно и каждое расстояние между избирателем и кандидатом, поэтому случайна и добавка к скору с добавлением нового победителя, не так ли?

Не совсем так....

Оказывается максимально доступный для конкретного распределения скор зависит от отношения $k/|C|$ всегда по конкретному закону

График надо менять



Этот закон....

$$y = x^{ax+b}$$

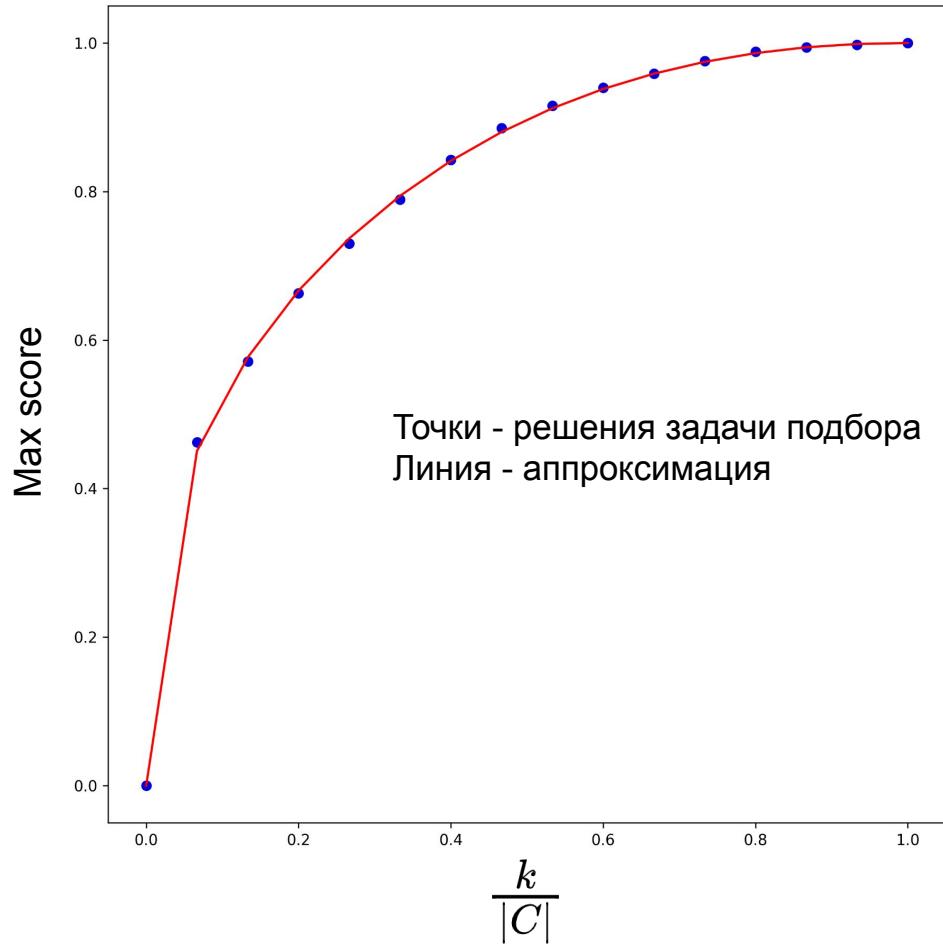
Где

$$a = -0.31719287263569135$$

$$b = 0.31549858529141006$$

В данном случае

График надо менять



Но...

Определить, от чего зависят a и b не получилось, это случайные параметры, меняющиеся в зависимости от выборки для одного и того же распределения

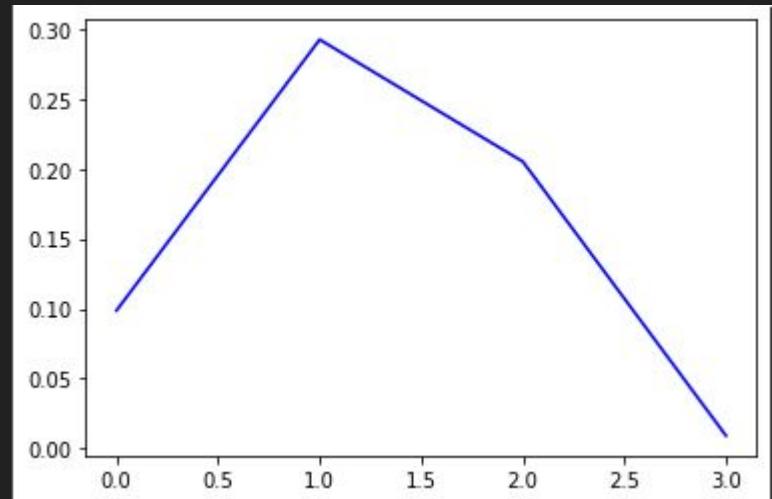
Но всё же a и b зависят от гиперпараметров, но неявным образом, так как остаются случайными

В первом случае в распределении кандидатов низкая плотность, а избирателей - высокая. (5, 50)

Во втором случае плотности равны. (5, 5)

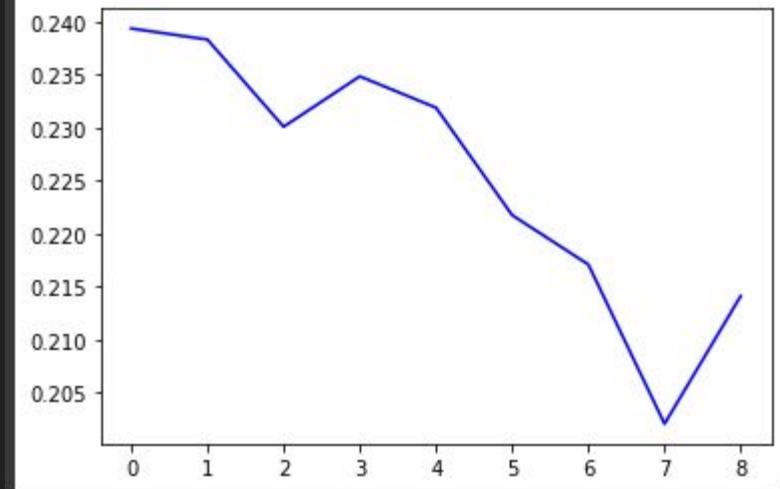
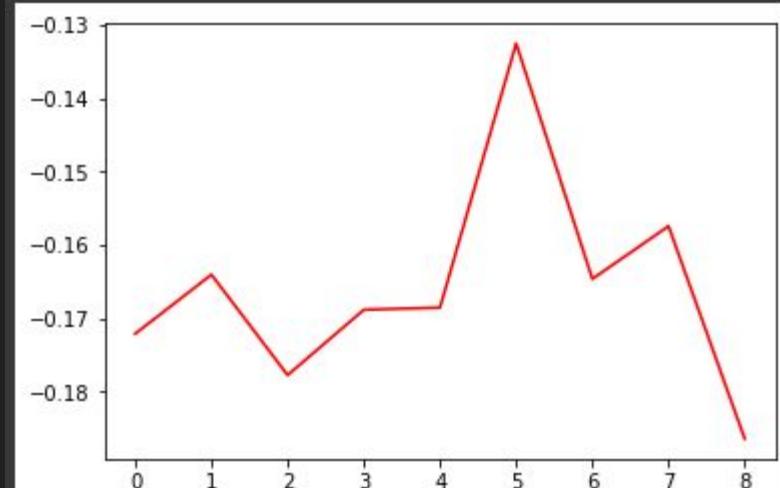
В третьем случае в распределении кандидатов высокая плотность, а избирателей - низкая. (50, 5)

В четвёртом как в первом ($\sqrt{50}$, 50)



Как А и Б зависят от добавления новых избирателей при неизменных остальных значениях

вроде немного убывает, поэтому и максимальный скор тоже убывает, что логично, но об этом позже



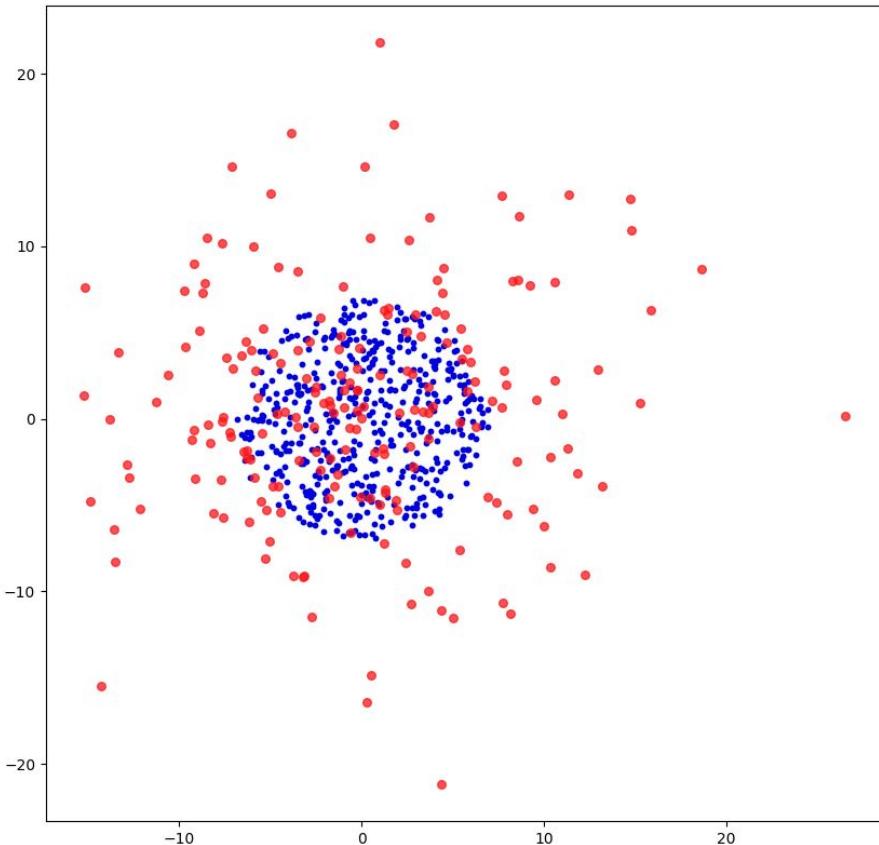
Распределение

$|V| = 500$

$|C| = 200$

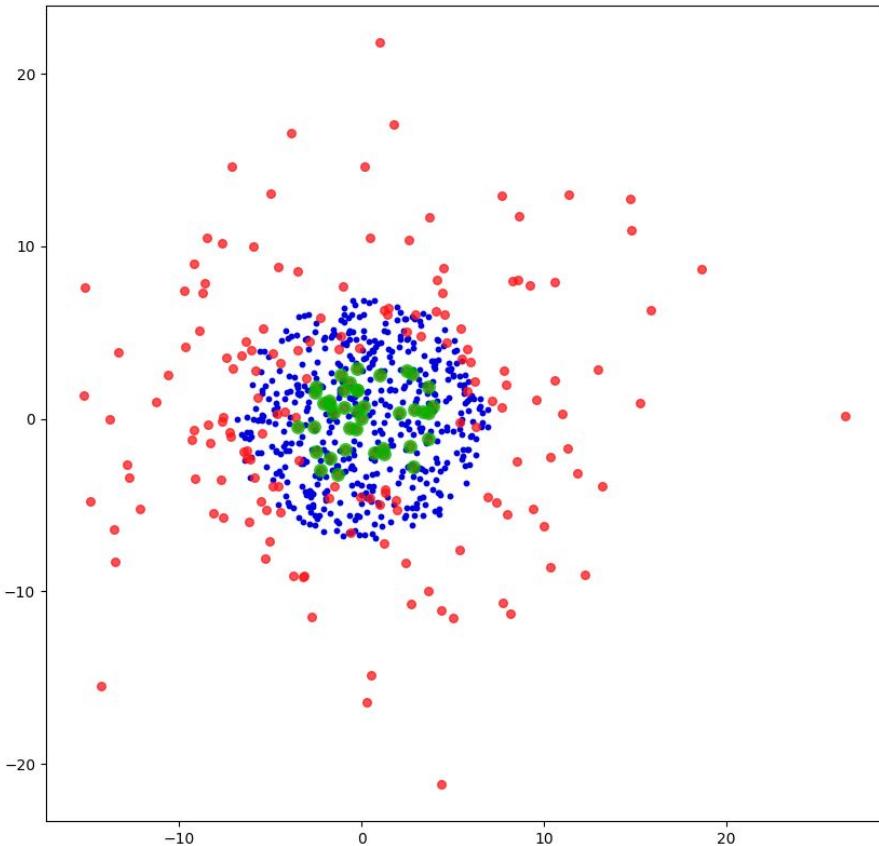
$K = 40$

uniform/normal 7/7



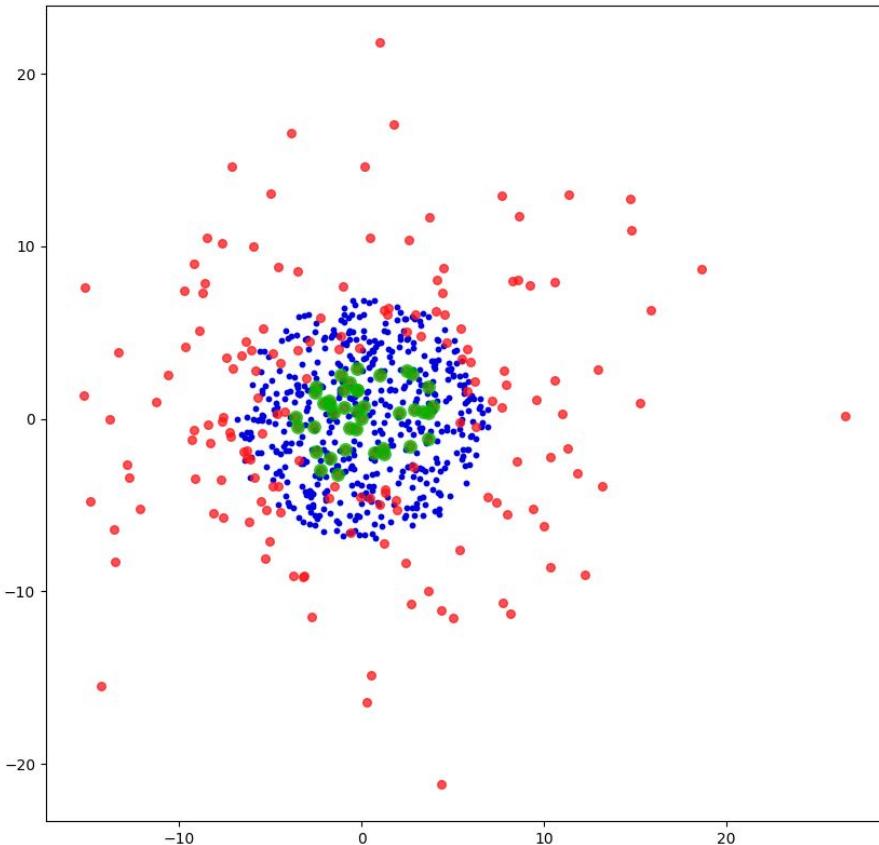
k-approval

Score = 0.6228511112408245



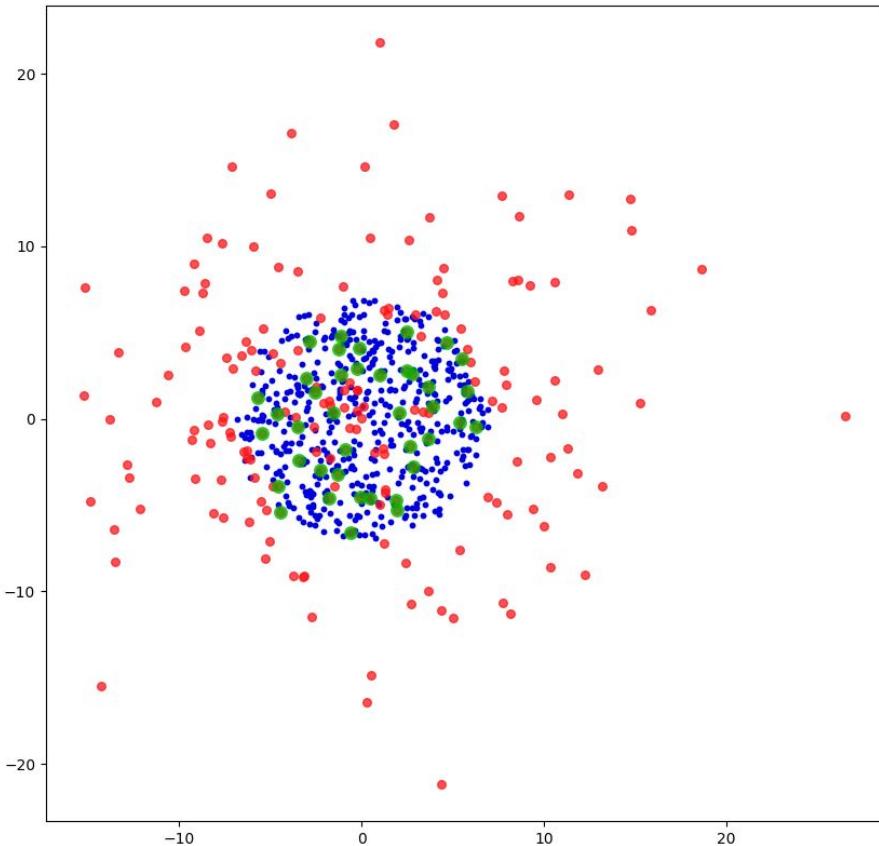
k-Borda

```
Score of k-Borda rule is  
0.599118966297716
```



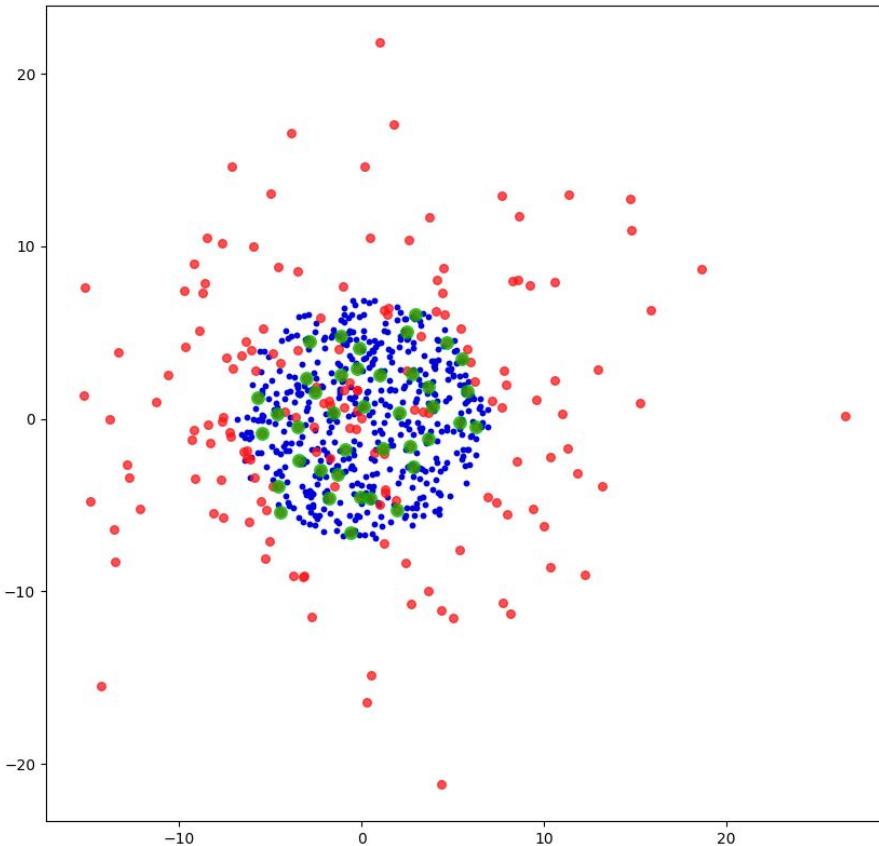
STV

```
Score of STV rule is  0.8789697356694398
```



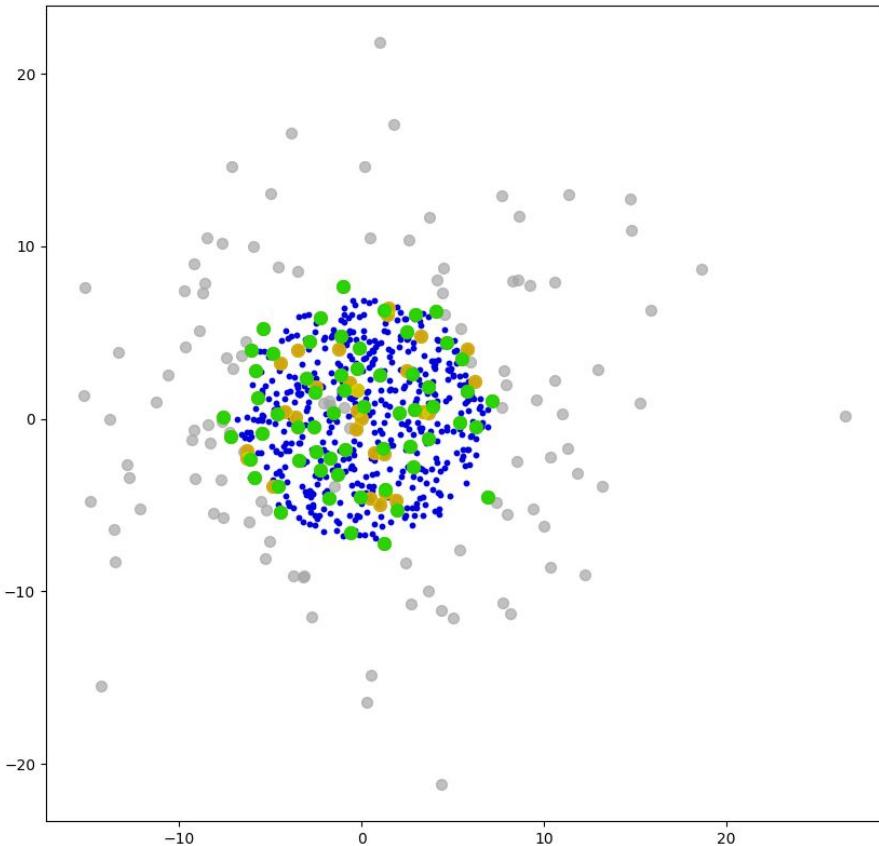
SNTV

Score of SNTV rule is 0.9030759081171648



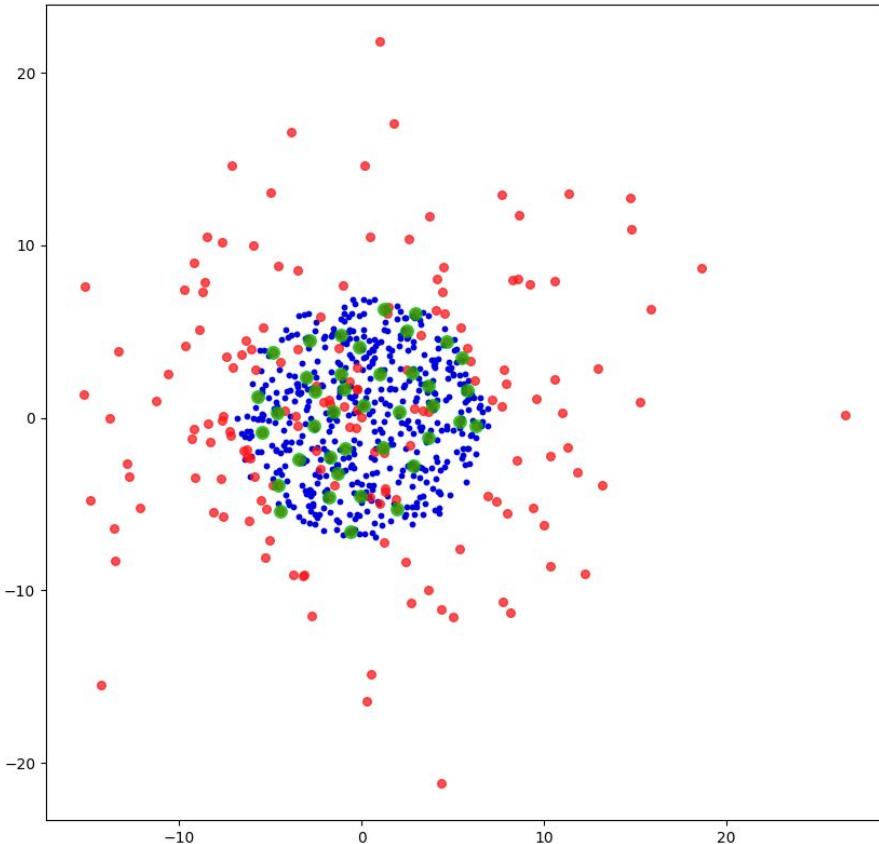
Pruning

```
Removed 111+0+29 = 140 Candidates
```



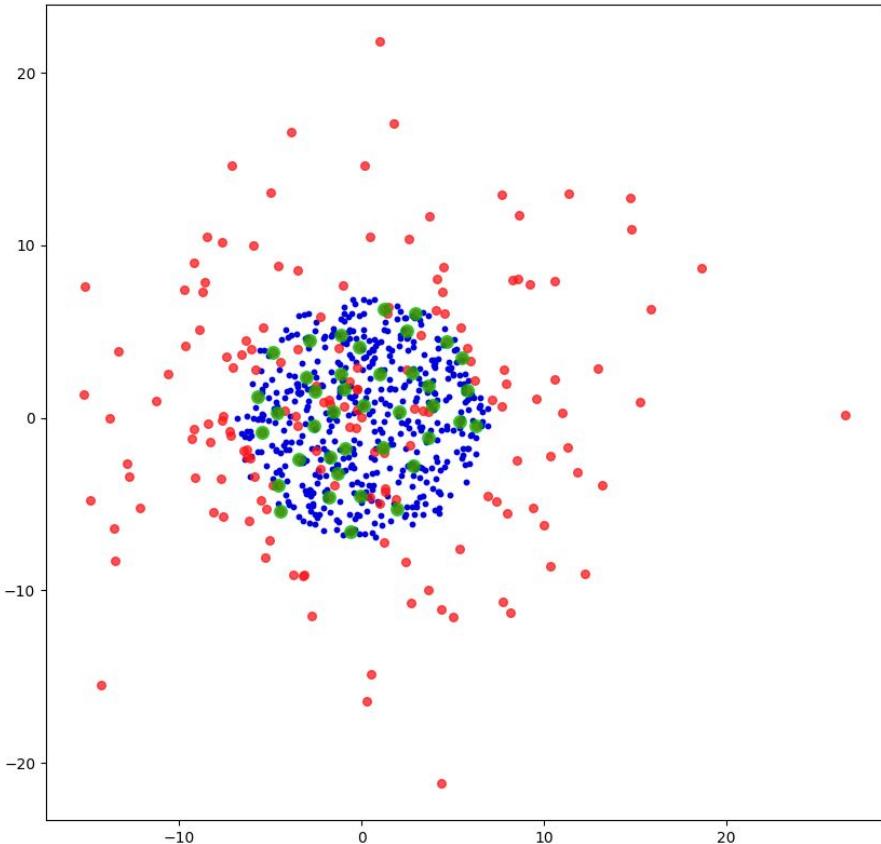
Optimization

Opt with random start and search space pruning



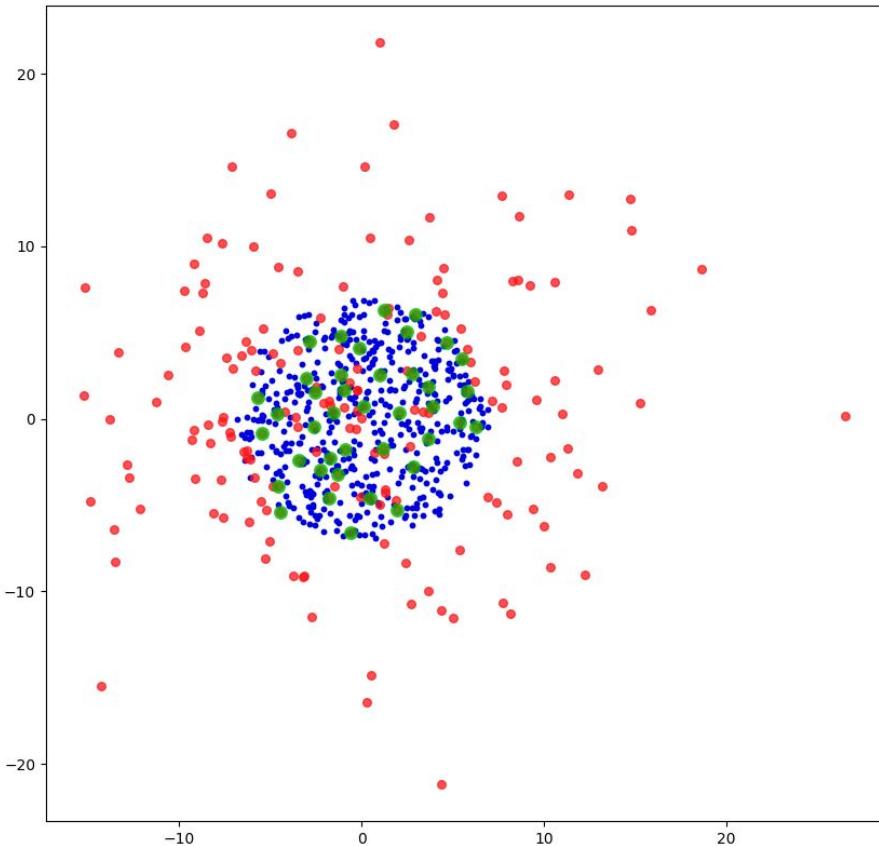
Optimization

Opt with SNTV start and search
space pruning

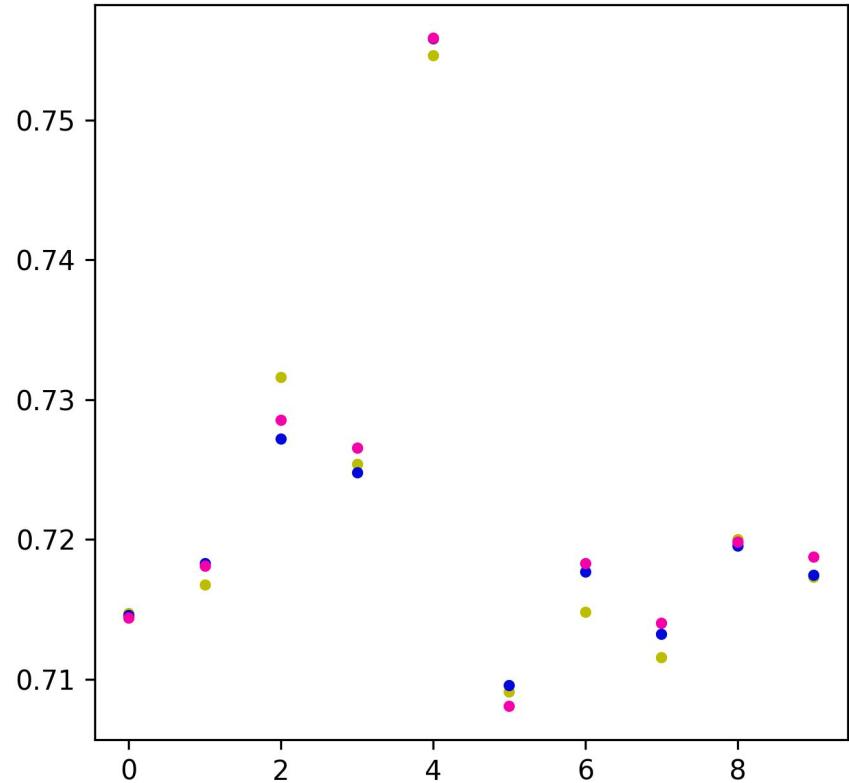
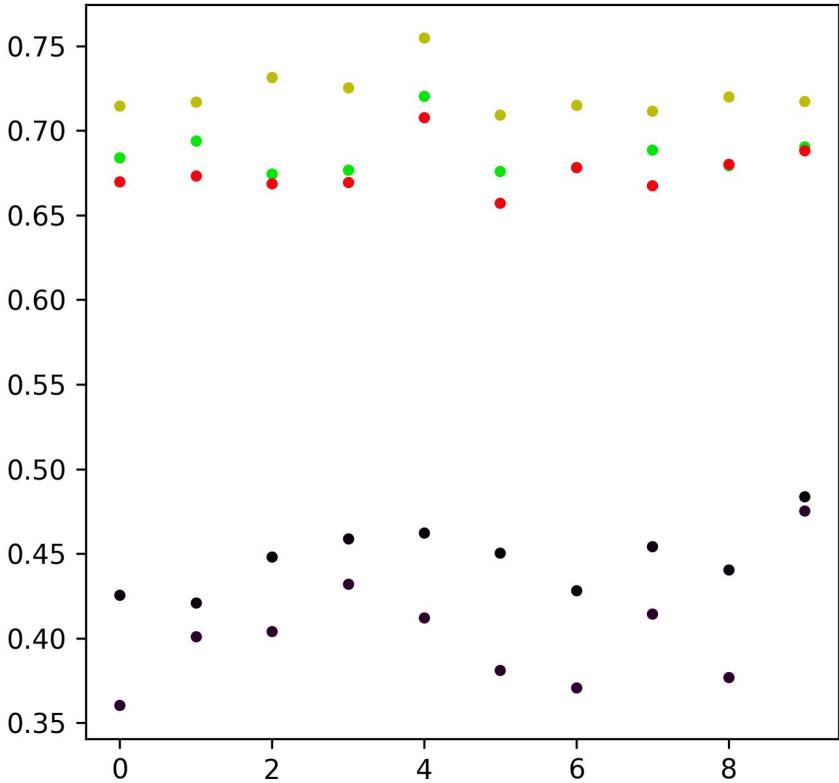


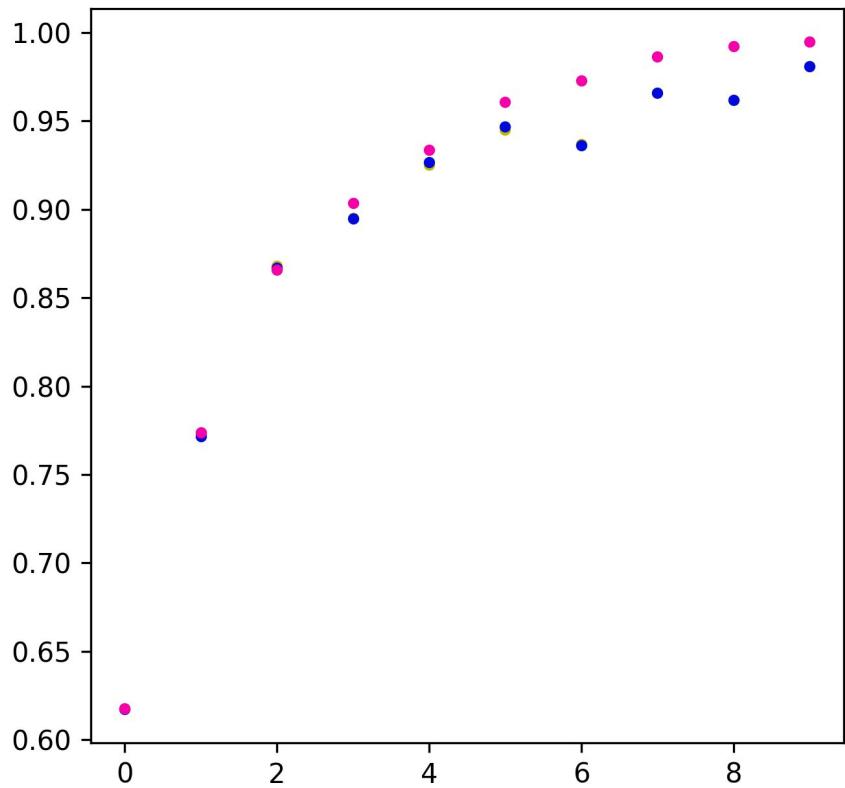
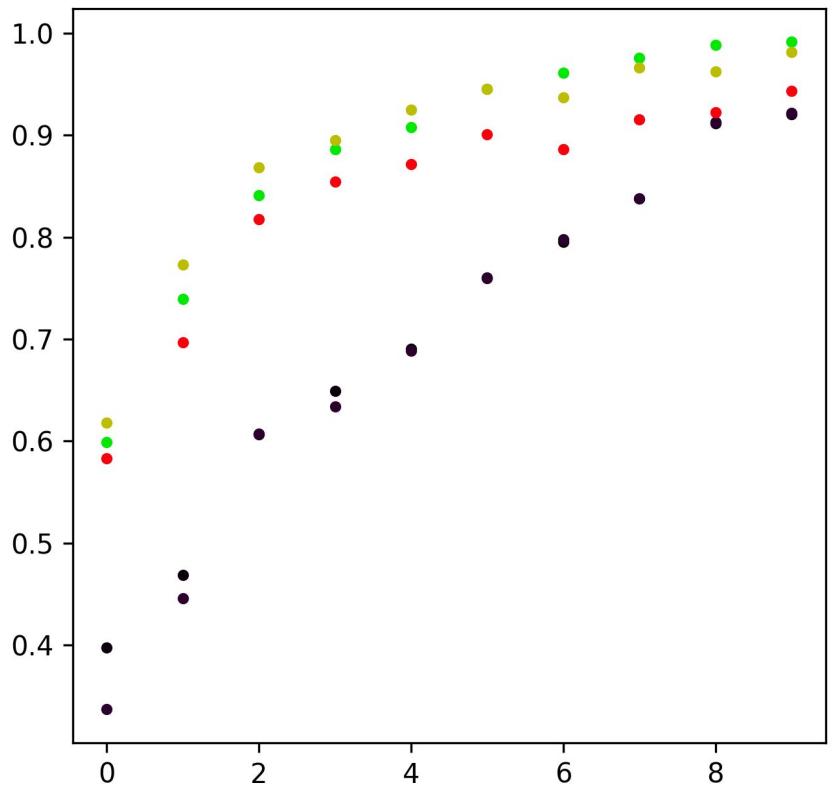
Optimization

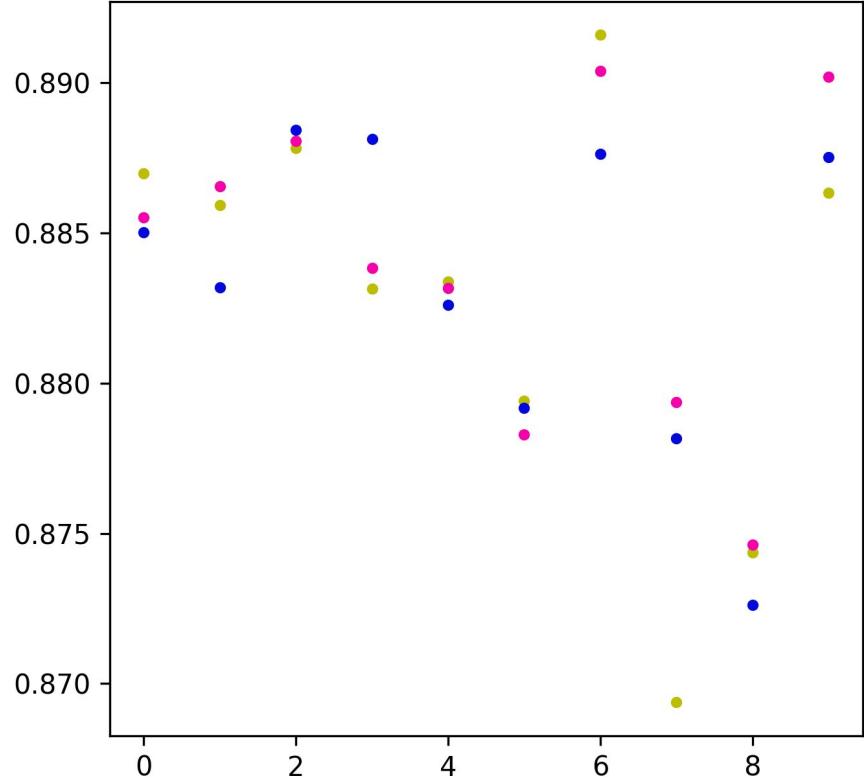
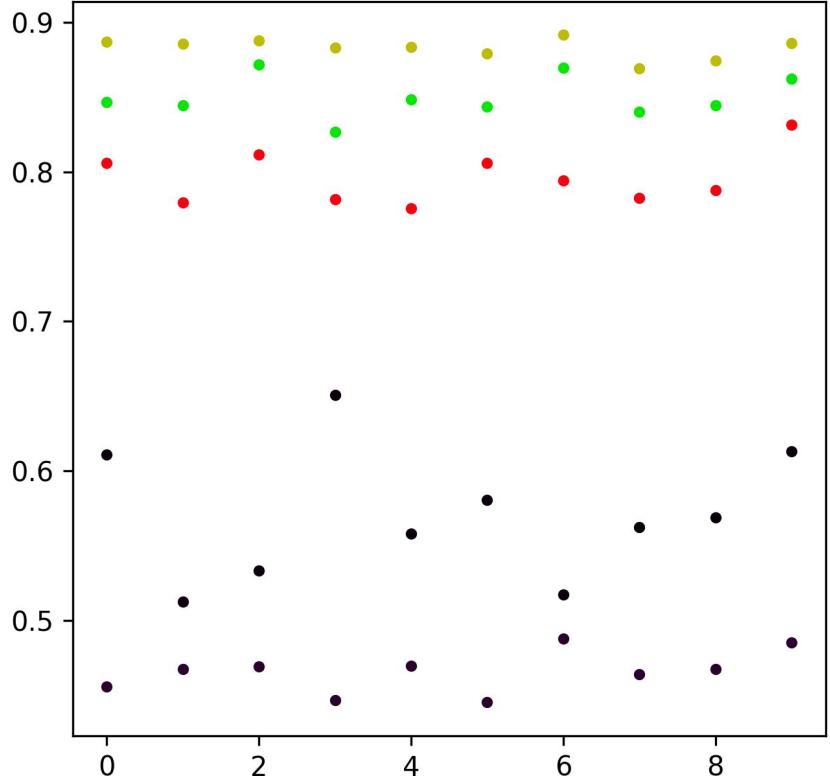
Opt with SNTV start

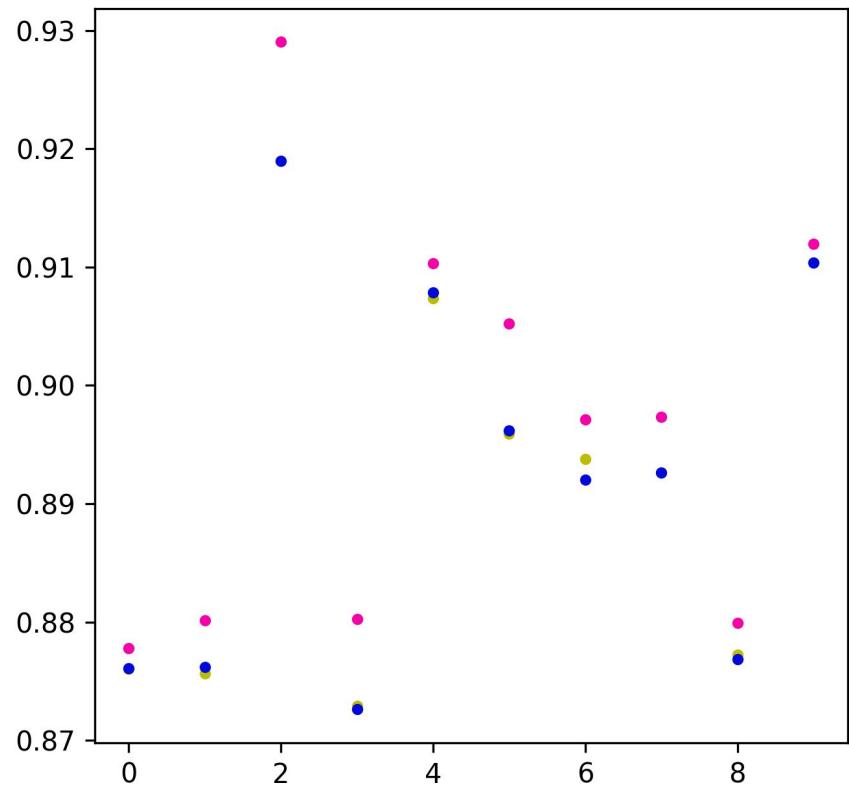
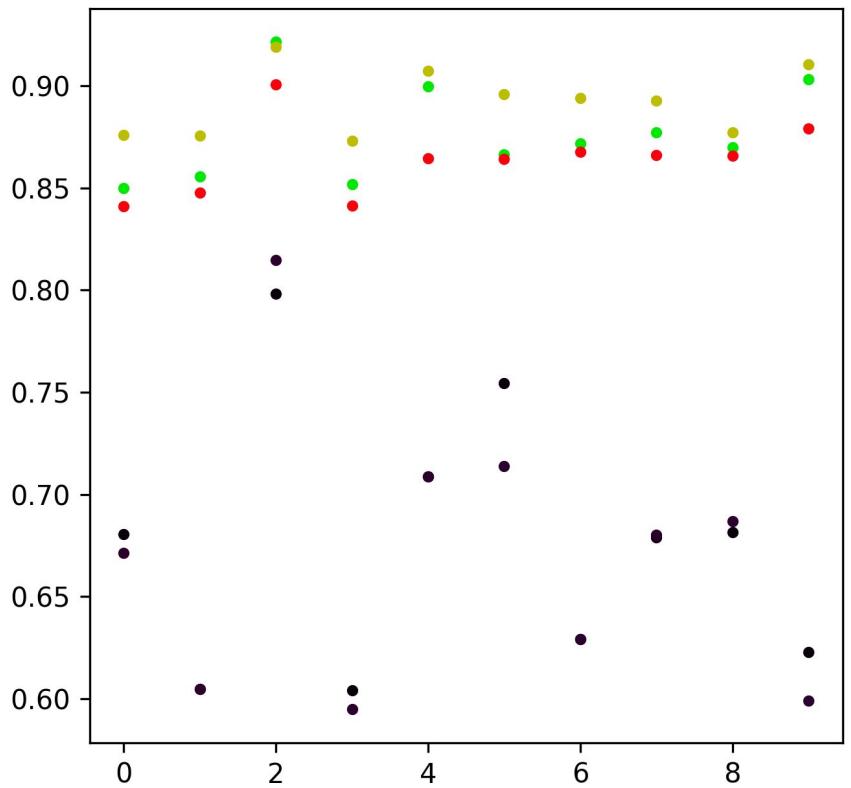


Жёлтый - OptRand, Синий - SNTV Opt, Сиреневый - SNTV full Opt, красный - STV, зелёный - SNTV, k-borda и k-approval чёрные(?)

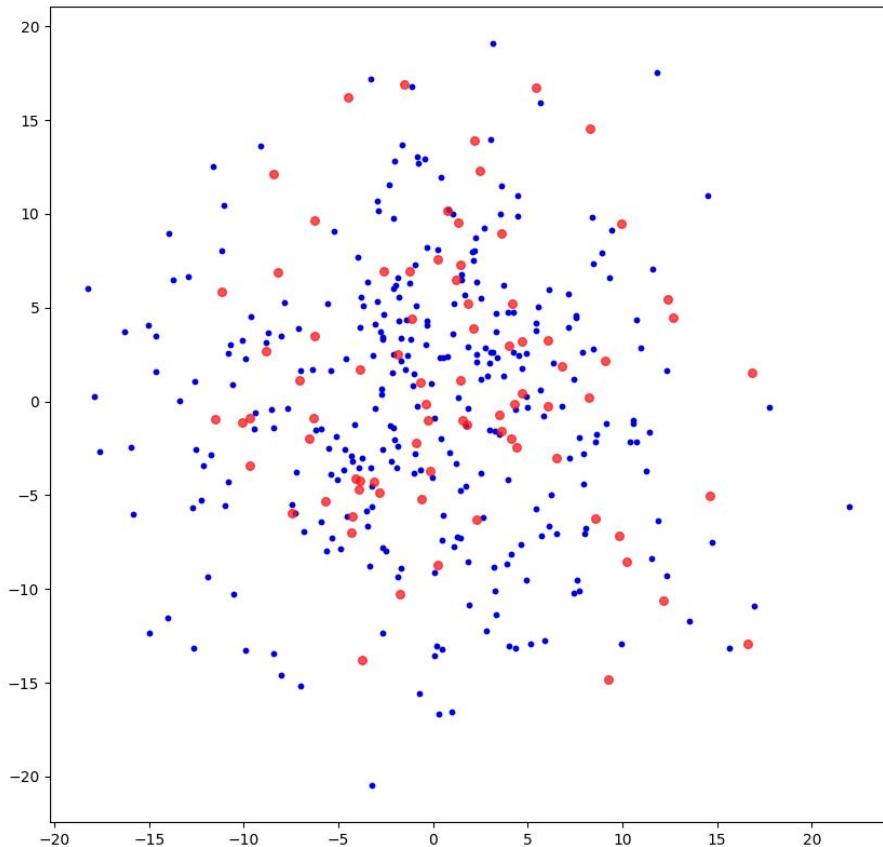








normal/normal, $D = 7$, $|V| = 300$, $|C| = 80$, $k = 20$



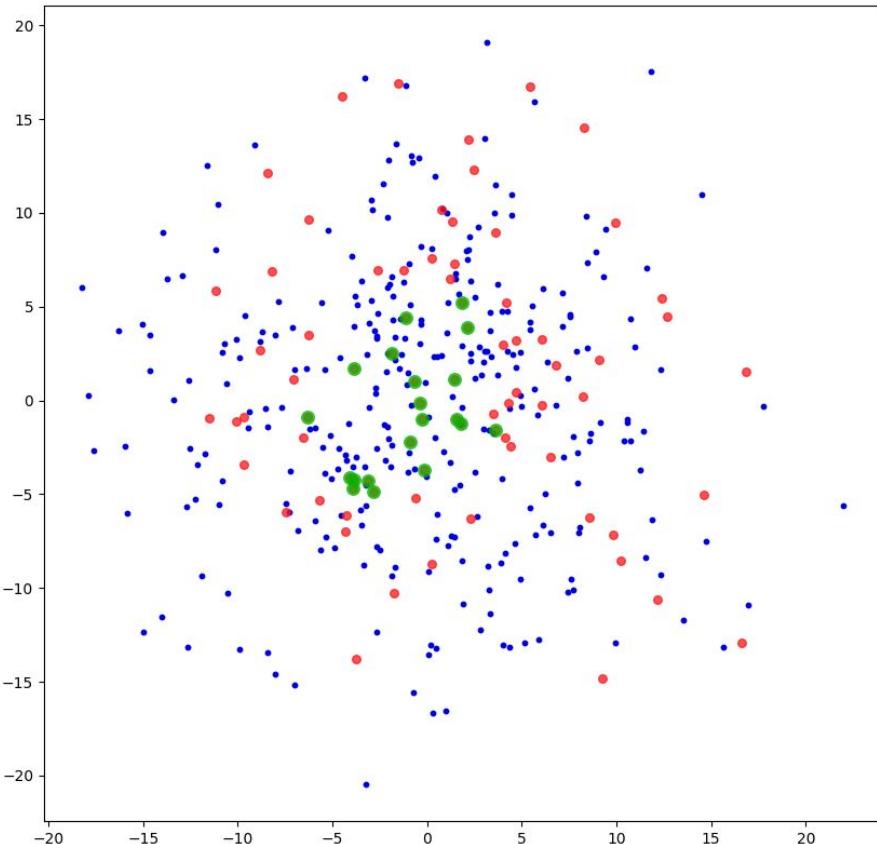
Bloc

```
Score of k-Approval rule is
```

```
0.5350270903365089
```

```
median Score of k-Approval rule is
```

```
0.4373379768256773
```



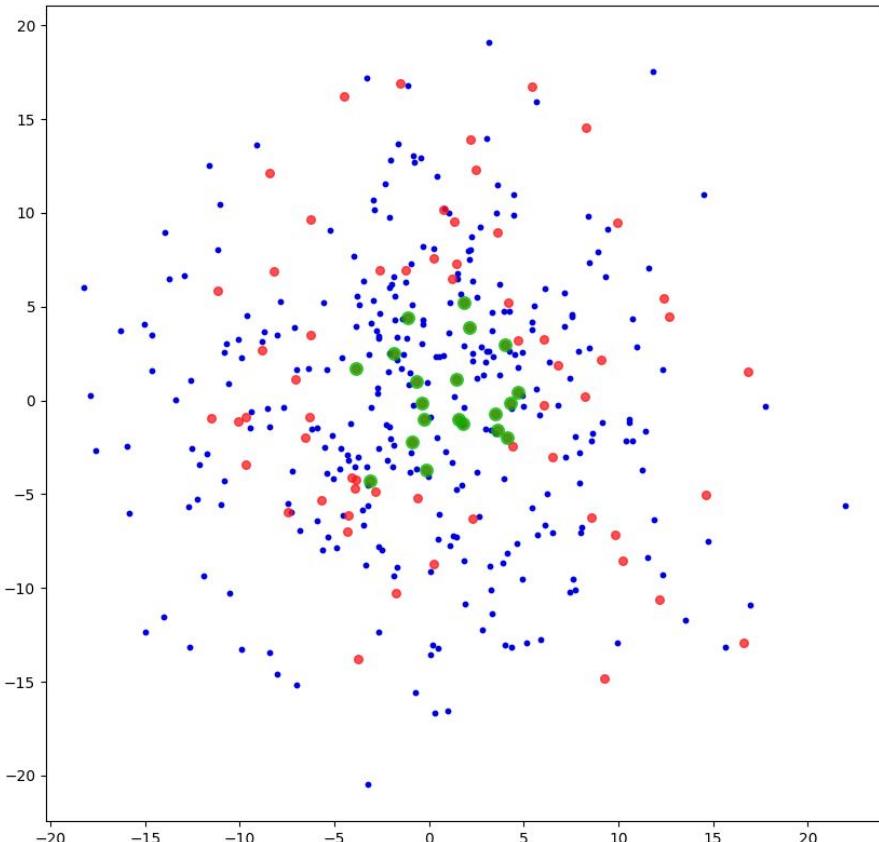
Borda

Score of k-Borda rule is

0.5359746199999916

median Score of k-Borda rule is

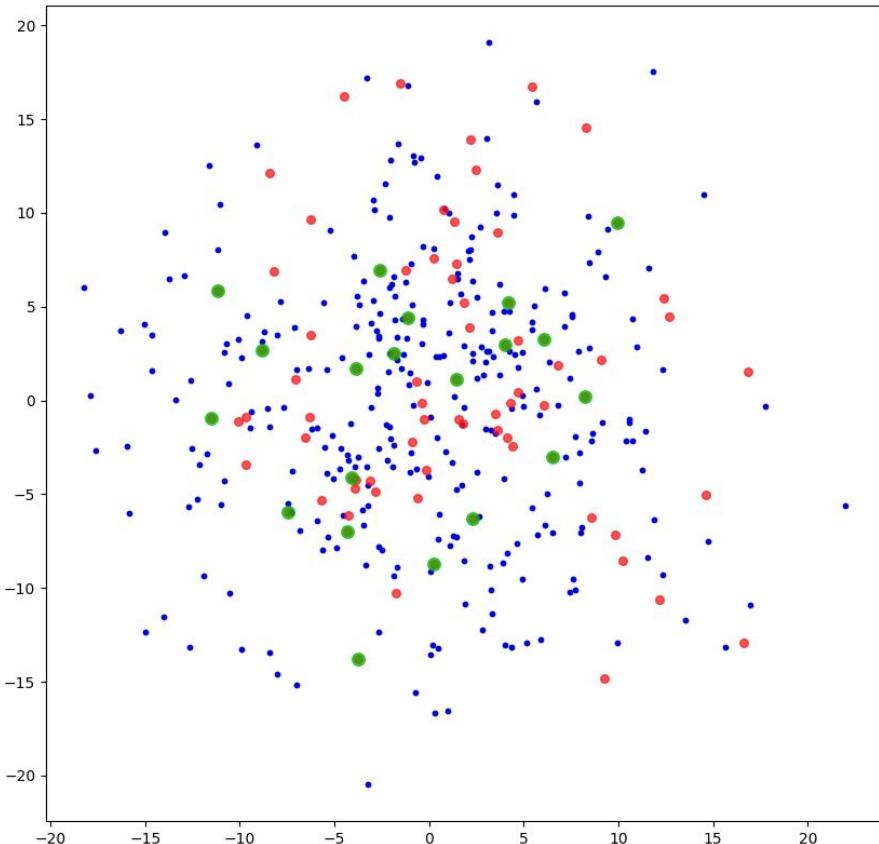
0.44499654094484564



SNTV

Score of SNTV rule is 0.7332613137563297

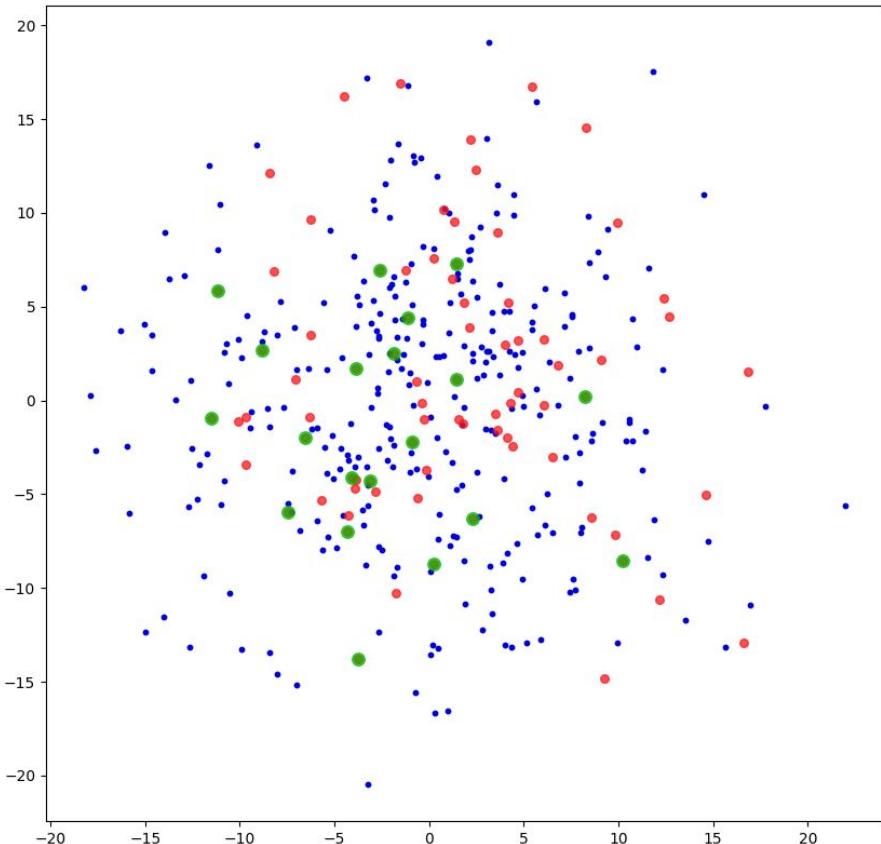
median Score of SNTV rule is 1.0



STV

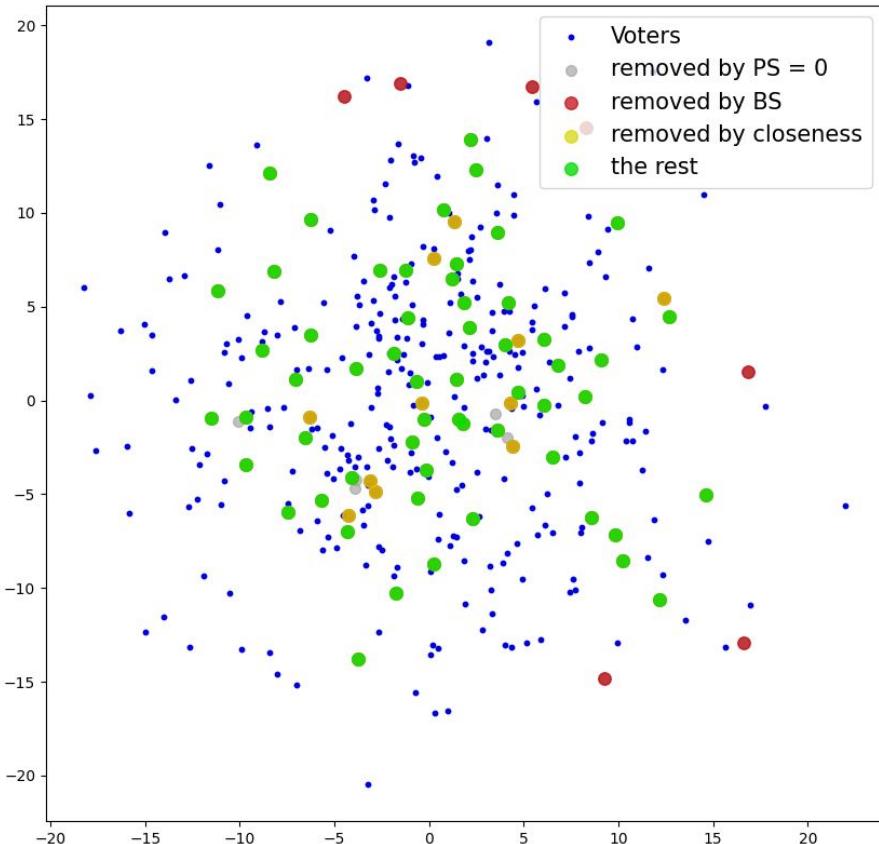
```
Score of STV rule is  0.7237201962427368
```

```
median Score of STV rule is  
0.9343724926165514
```



Pruning

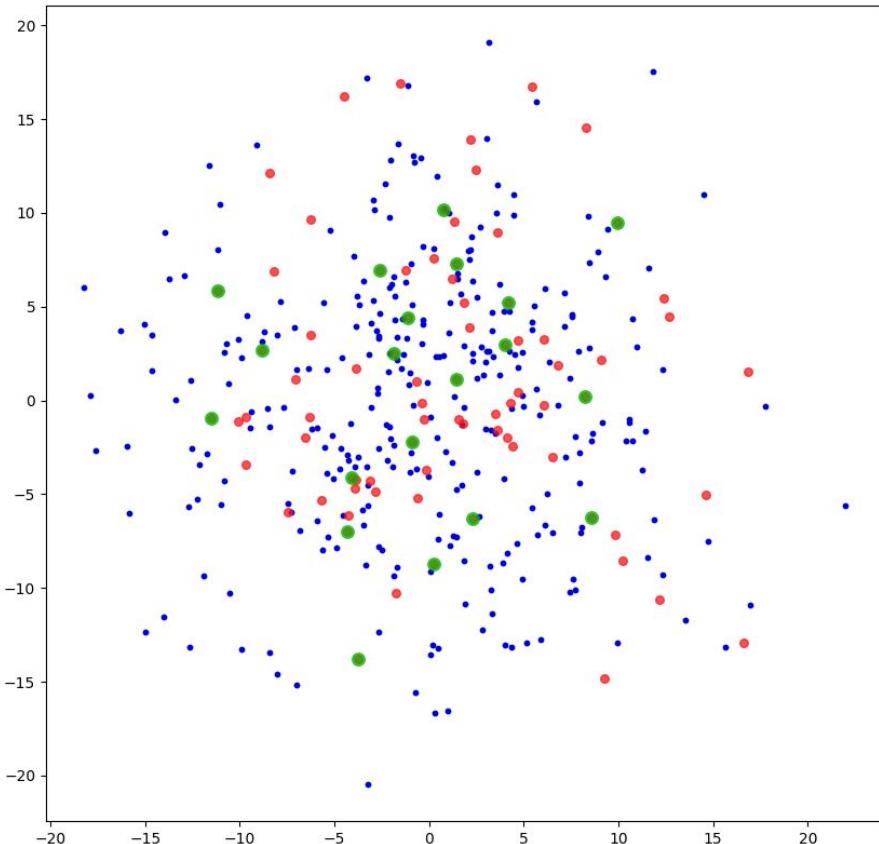
Removed 23



Random Optimization

```
Score of Optimization is  
0.7655166839929489
```

```
median Score of optimization is  
0.9877938820778647
```



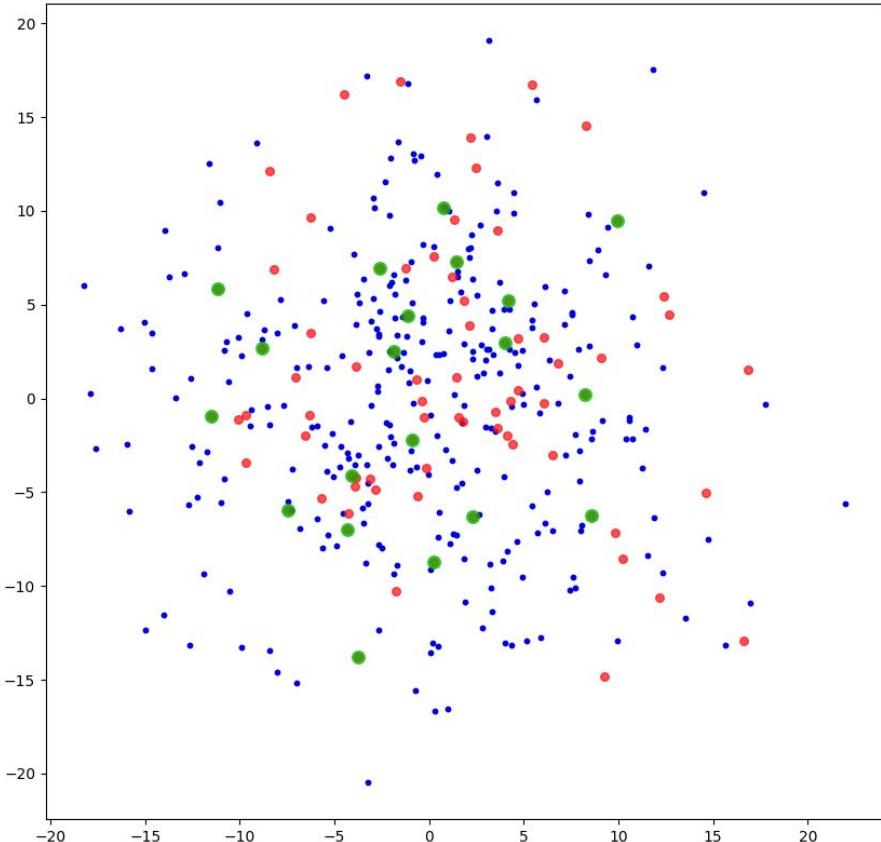
SNTV-opt (P)

```
Score of SNTV-opt (pruned) rule is
```

```
0.7659649386465276
```

```
median Score of SNTV-opt (pruned) rule is
```

```
0.9877938820778647
```



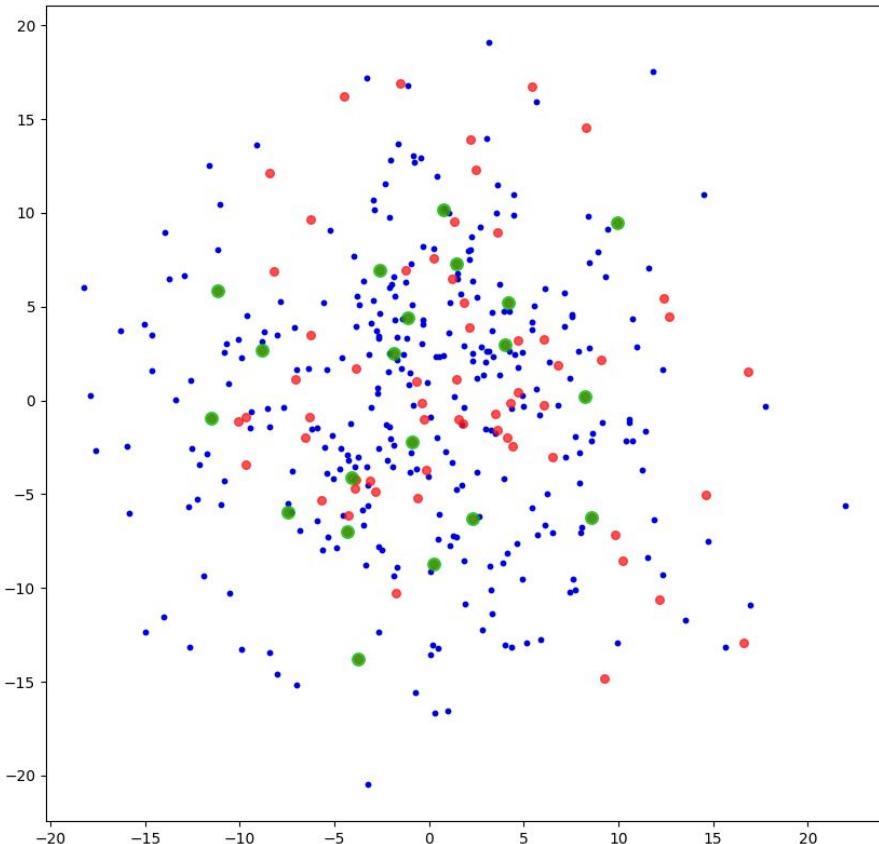
SNTV-opt

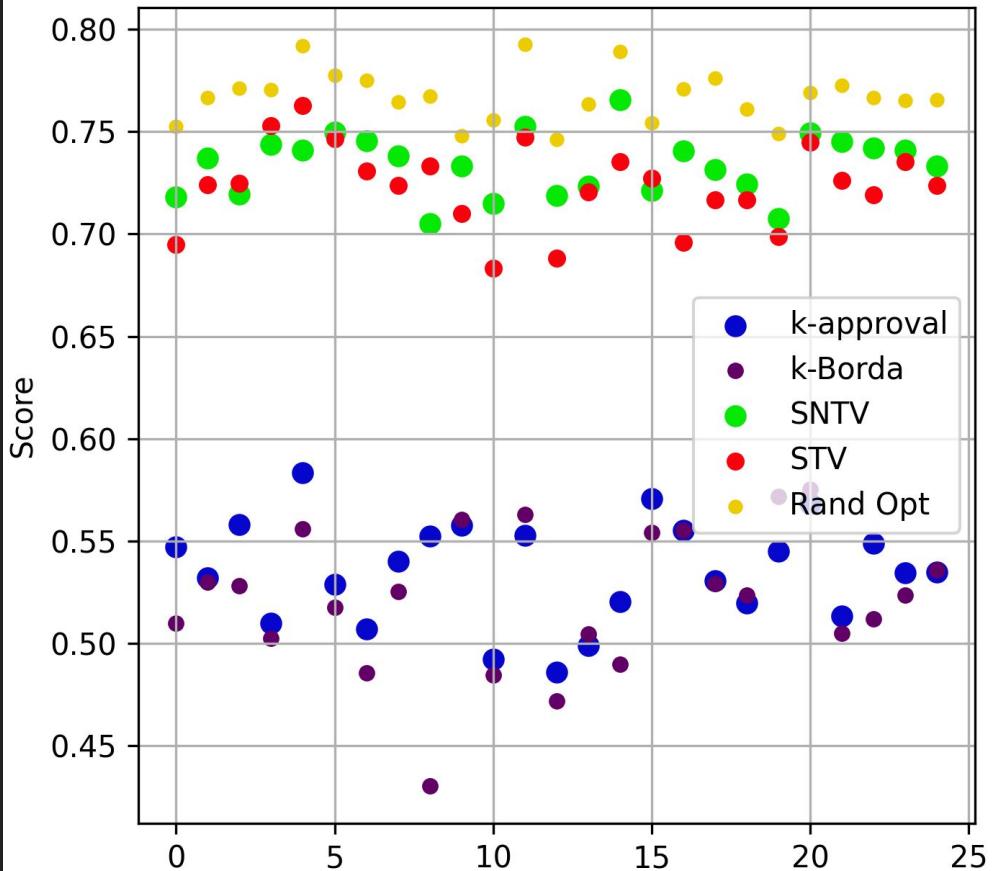
```
Score of SNTV-opt rule is
```

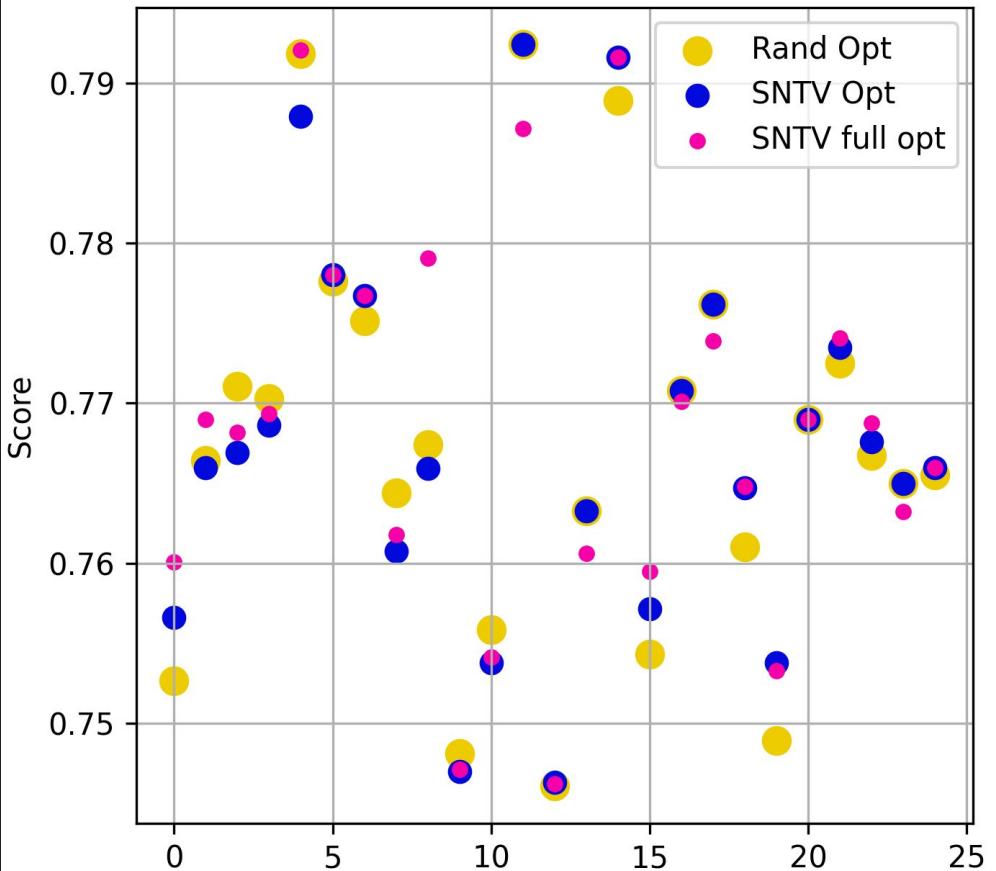
```
0.7659649386465276
```

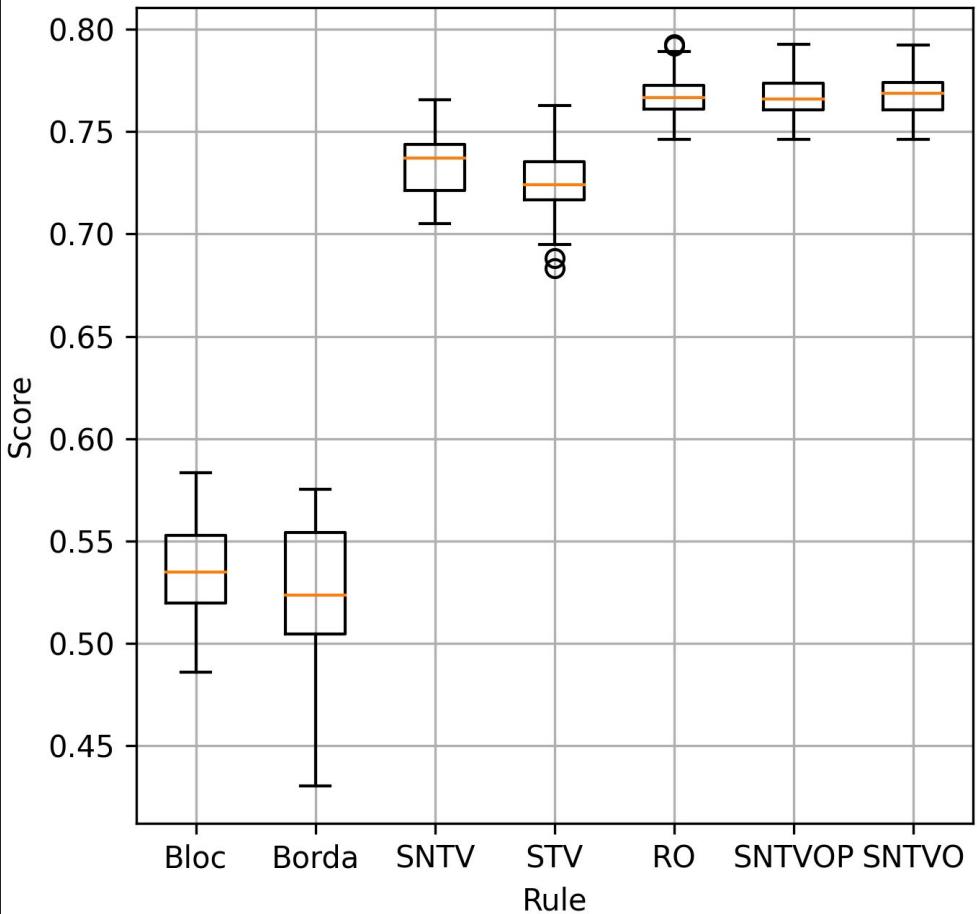
```
median Score of SNTV-opt rule is
```

```
0.9877938820778647
```

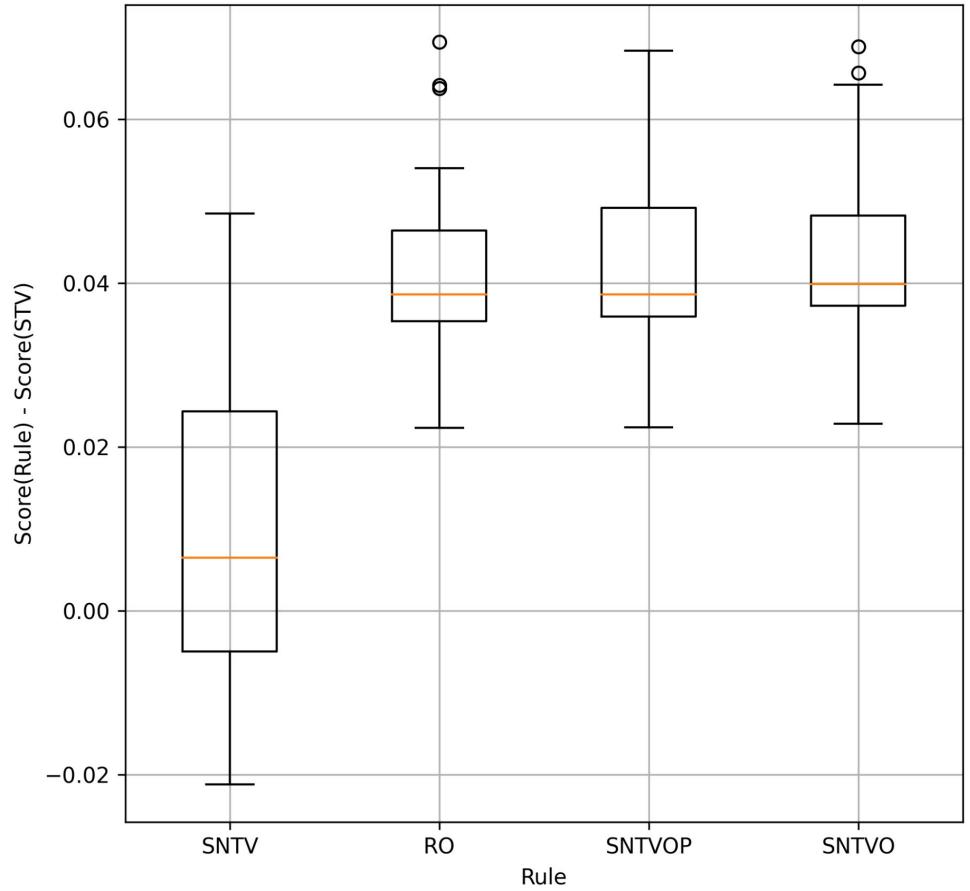








В среднем оптимизационные алгоритмы на 5-6% лучше, чем STV

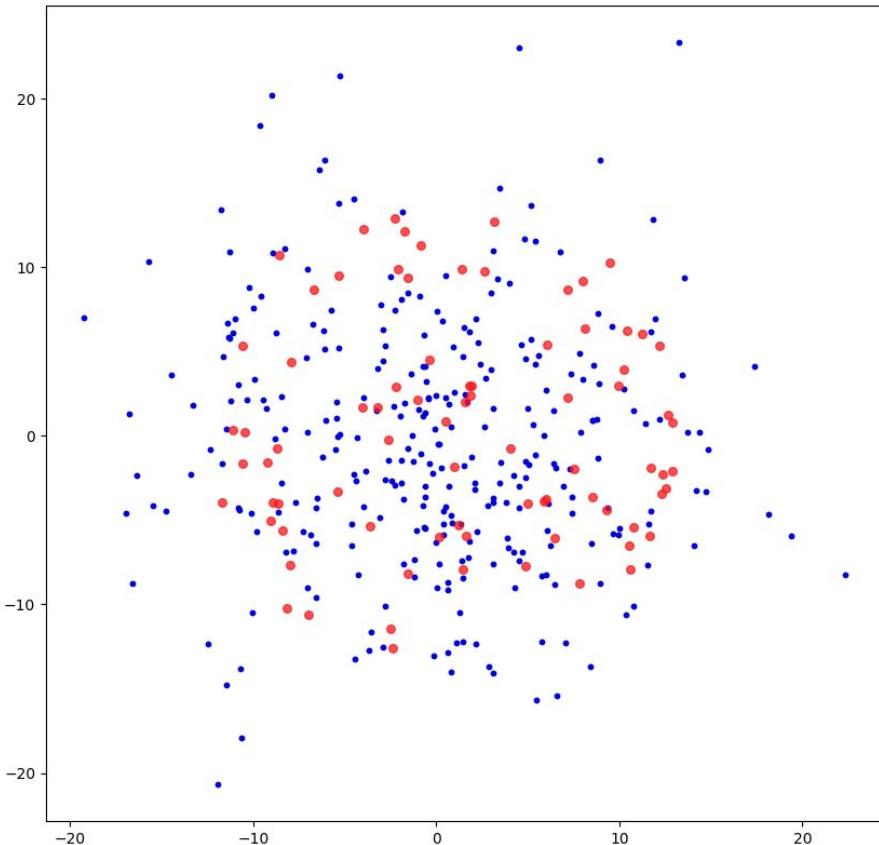


normal/uniform

$|V| = 300$

$|C| = 80$

$K = 20$



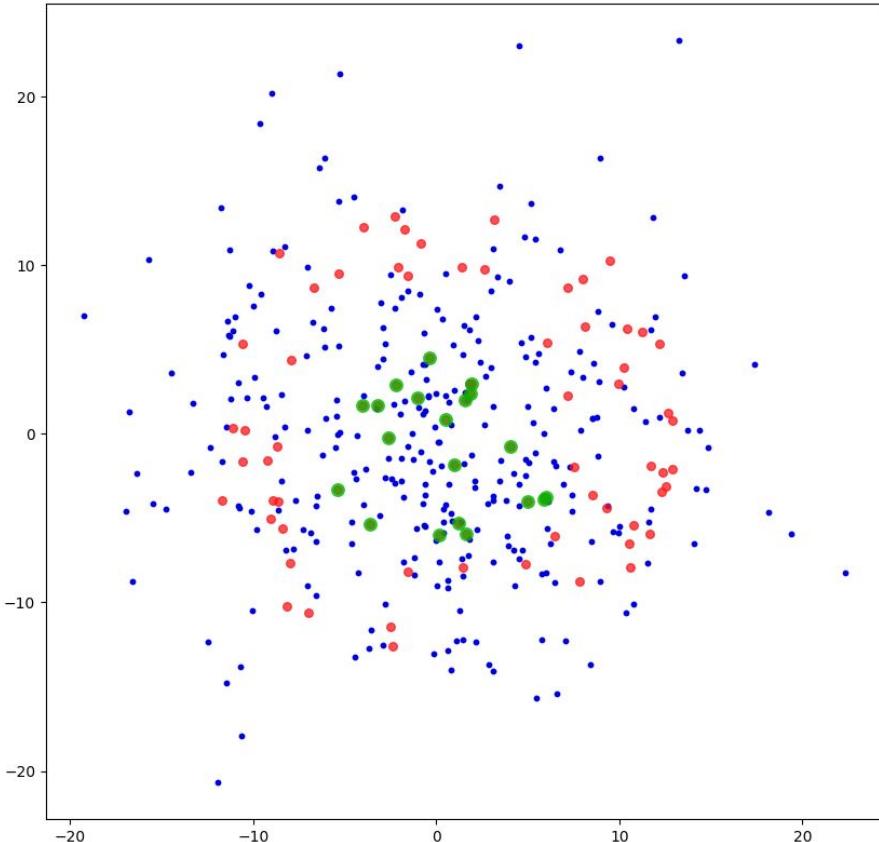
Bloc

```
Score of k-Approval rule is
```

```
0.5845522712331812
```

```
median Score of k-Approval rule is
```

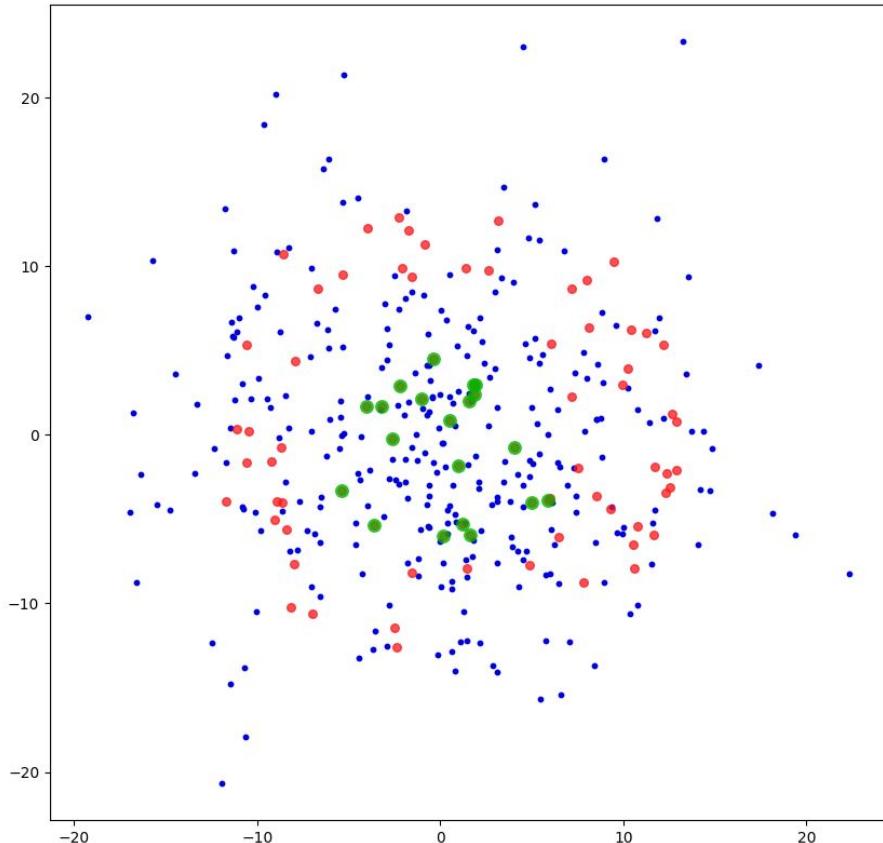
```
0.5029042069874181
```



Borda

```
Score of k-Borda rule is  
0.5836893454717038
```

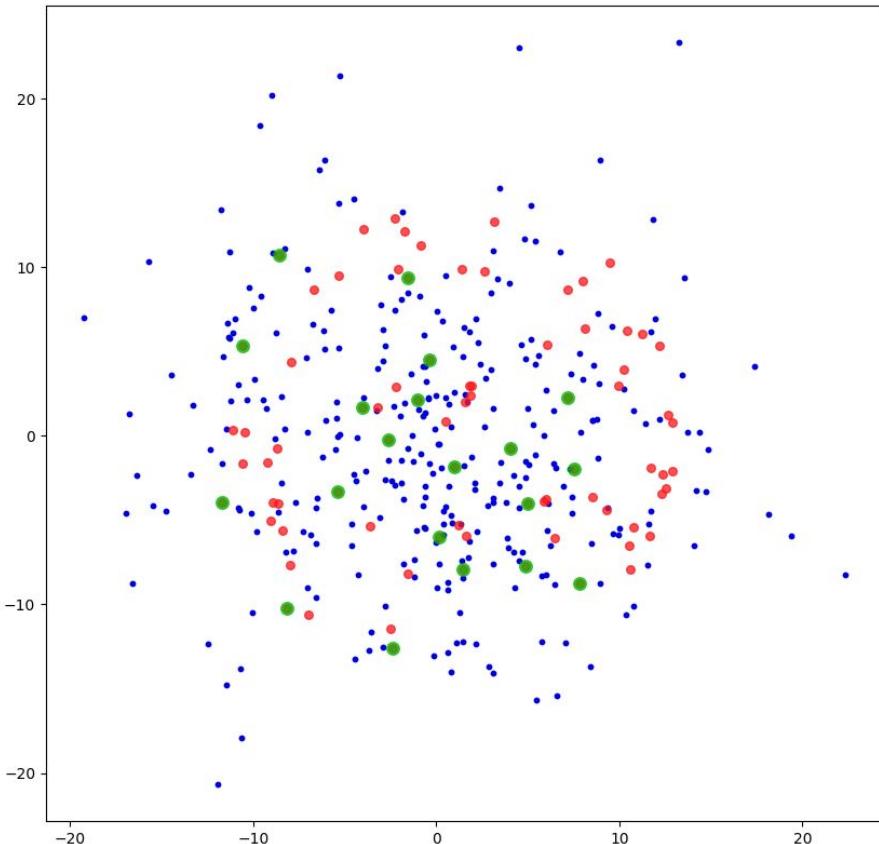
```
median Score of k-Borda rule is  
0.49380843019070086
```



SNTV

```
Score of SNTV rule is 0.751873769463003
```

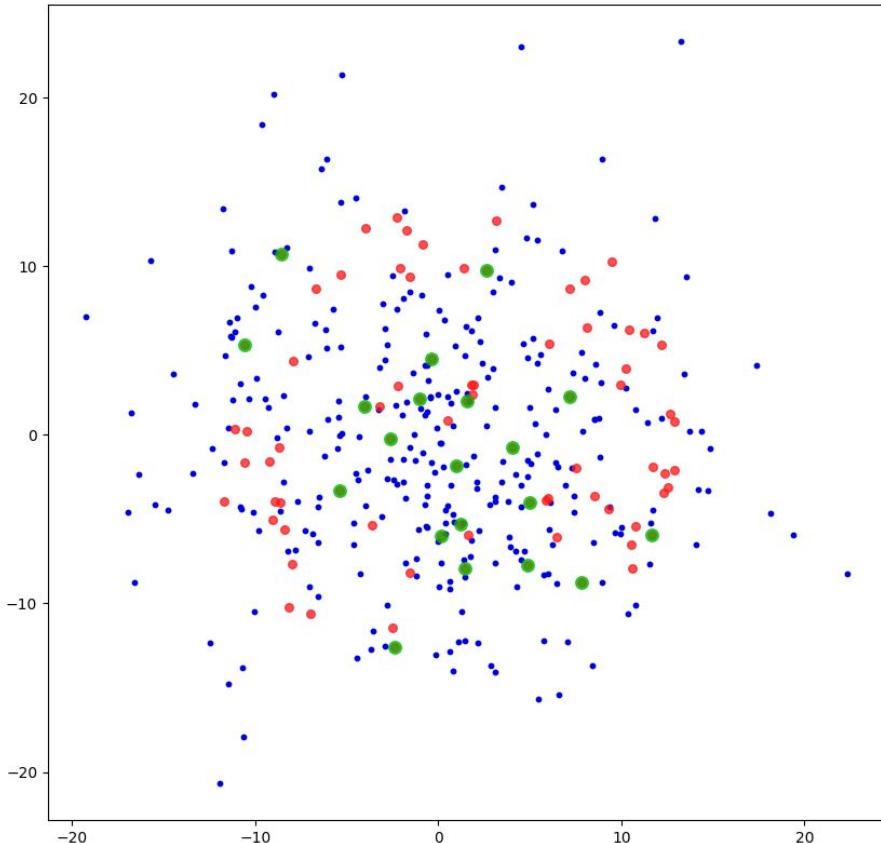
```
median Score of SNTV rule is 1.0
```



STV

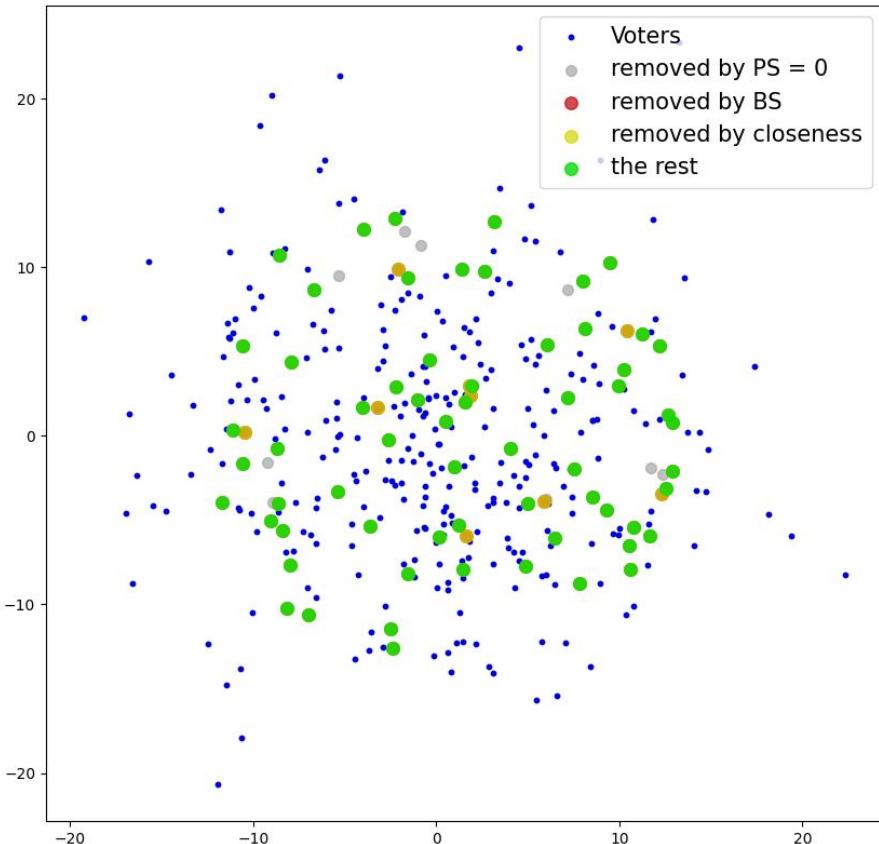
```
Score of STV rule is  0.7492368573346507
```

```
median Score of STV rule is  1.0
```



Pruning

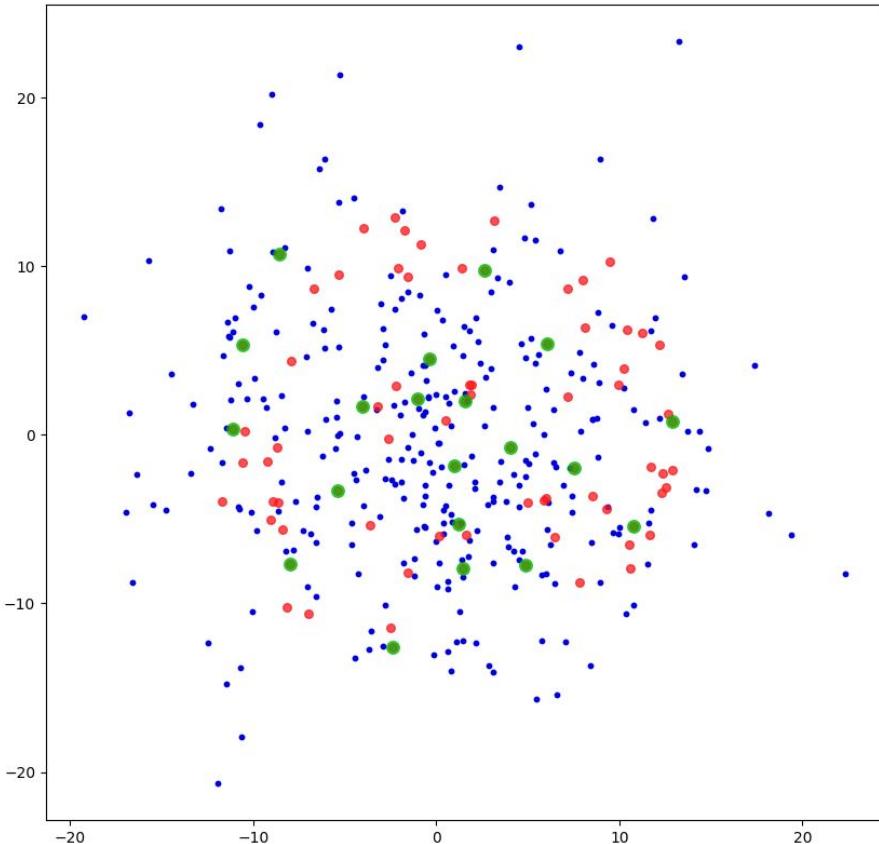
Removed 18



R-opt

```
Score of Optimization is  
0.7744302678233046
```

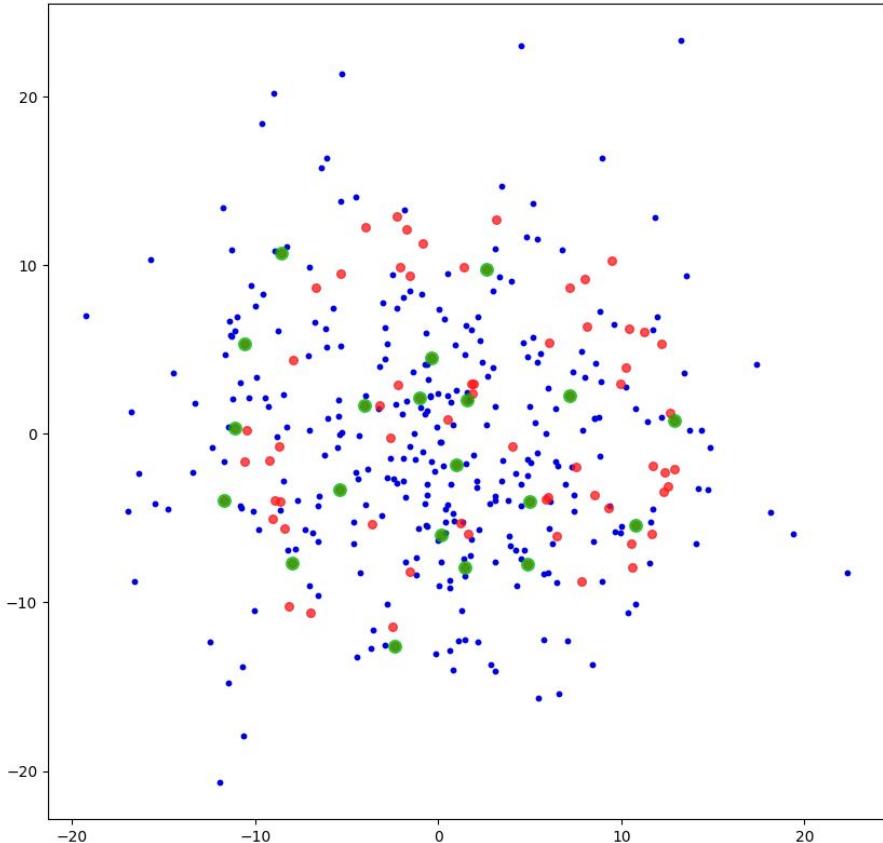
```
median Score of optimization is  
0.9055984533056153
```



SNTV-opt(P)

```
Score of SNTV-opt (prunned) rule is  
0.7781037957239083
```

```
median Score of SNTV-opt (prunned) rule is  
0.9293882175578596
```



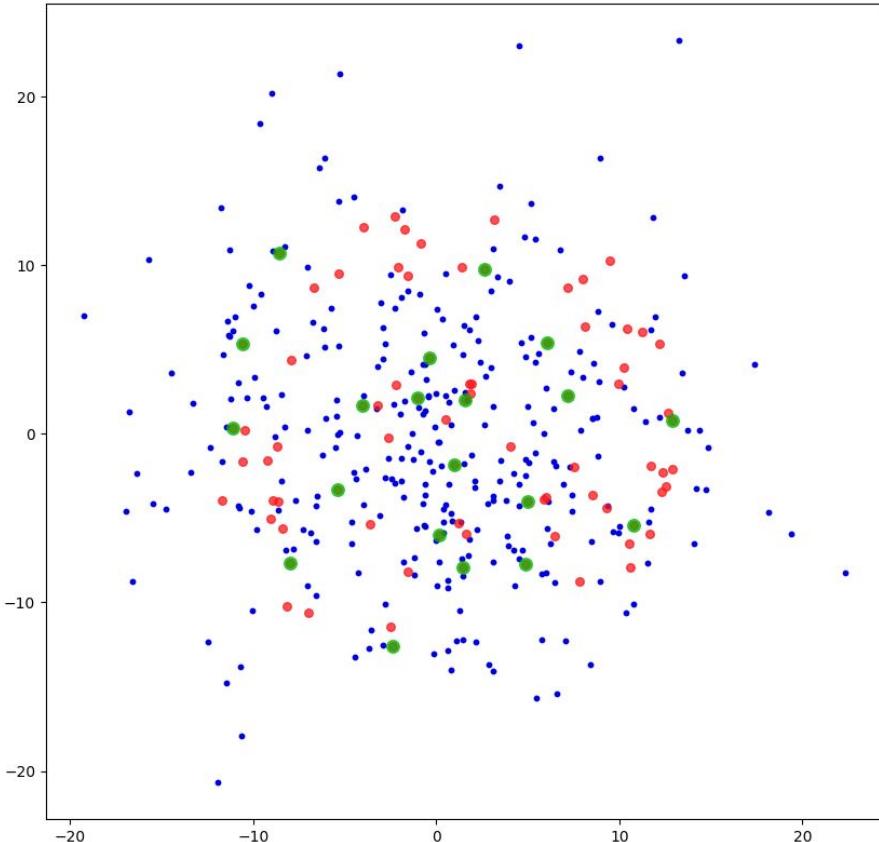
SNTV-opt

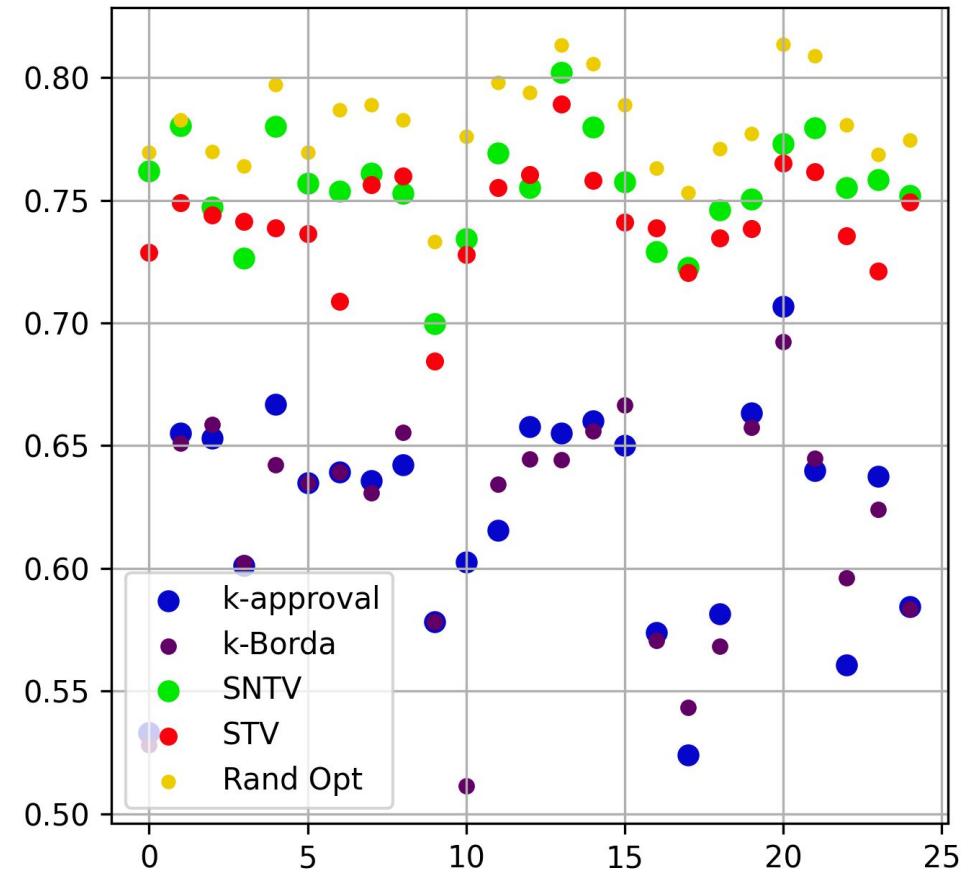
```
Score of SNTV-opt rule is
```

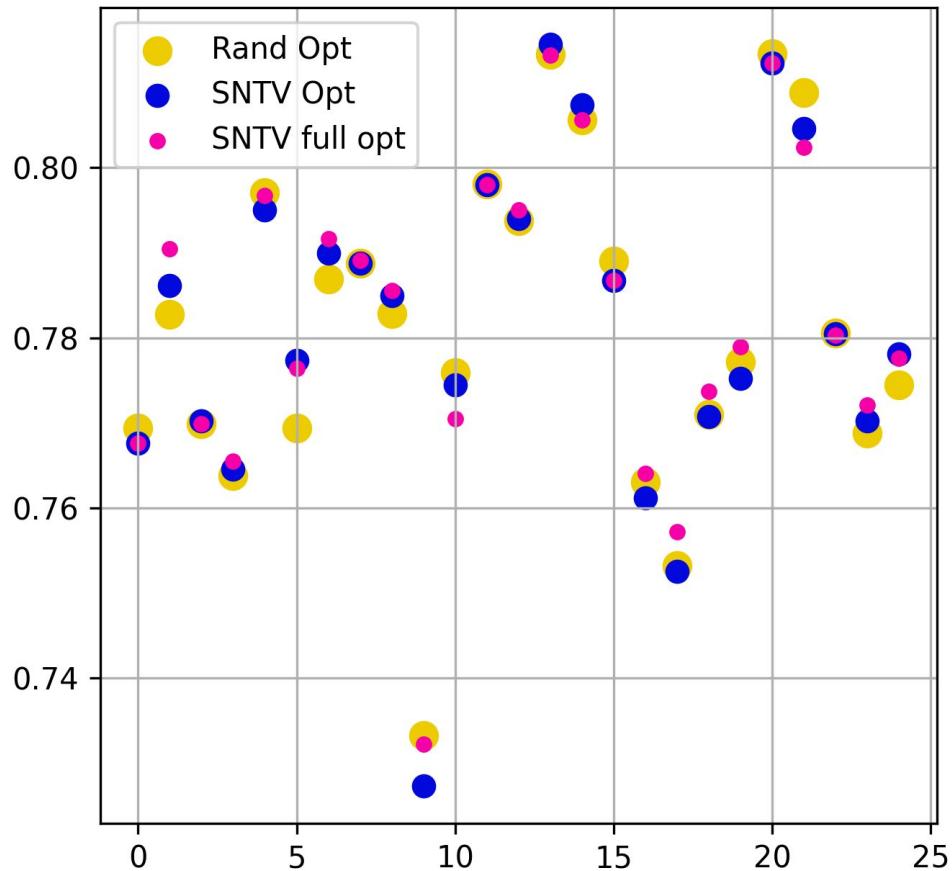
```
0.7775841644058993
```

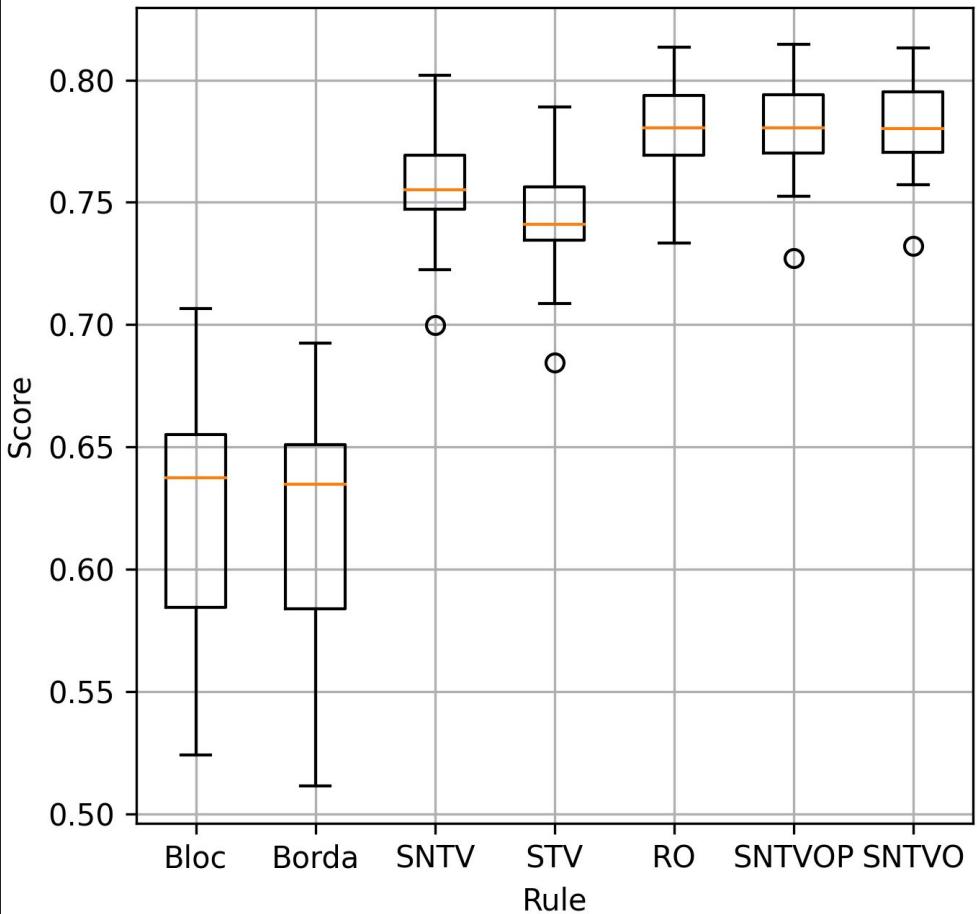
```
median Score of SNTV-opt rule is
```

```
0.9217865938909209
```

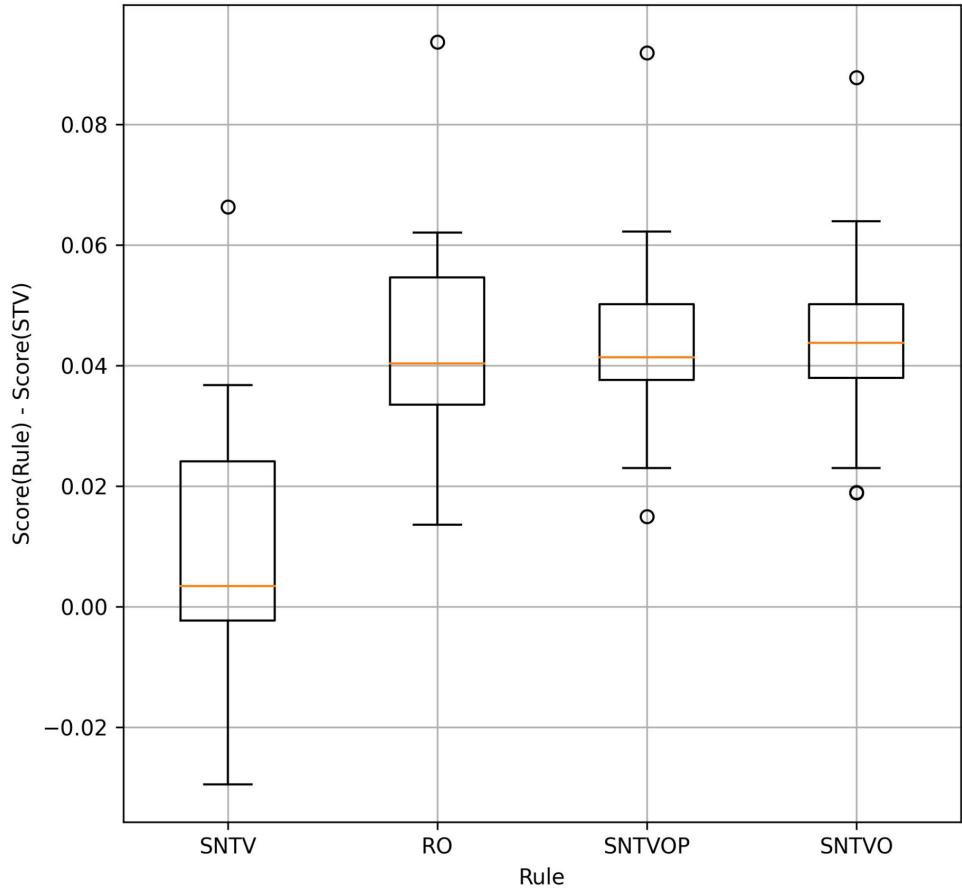








Для такого распределения
оптимизационные алгоритмы
также оказываются в среднем
на 5% лучше STV

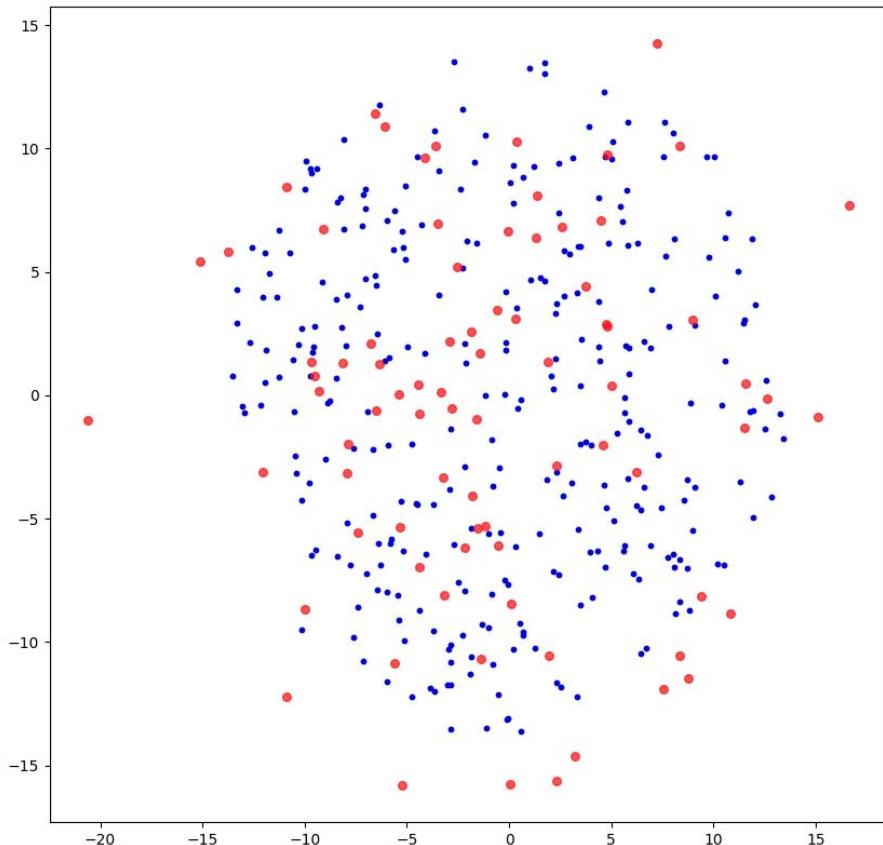


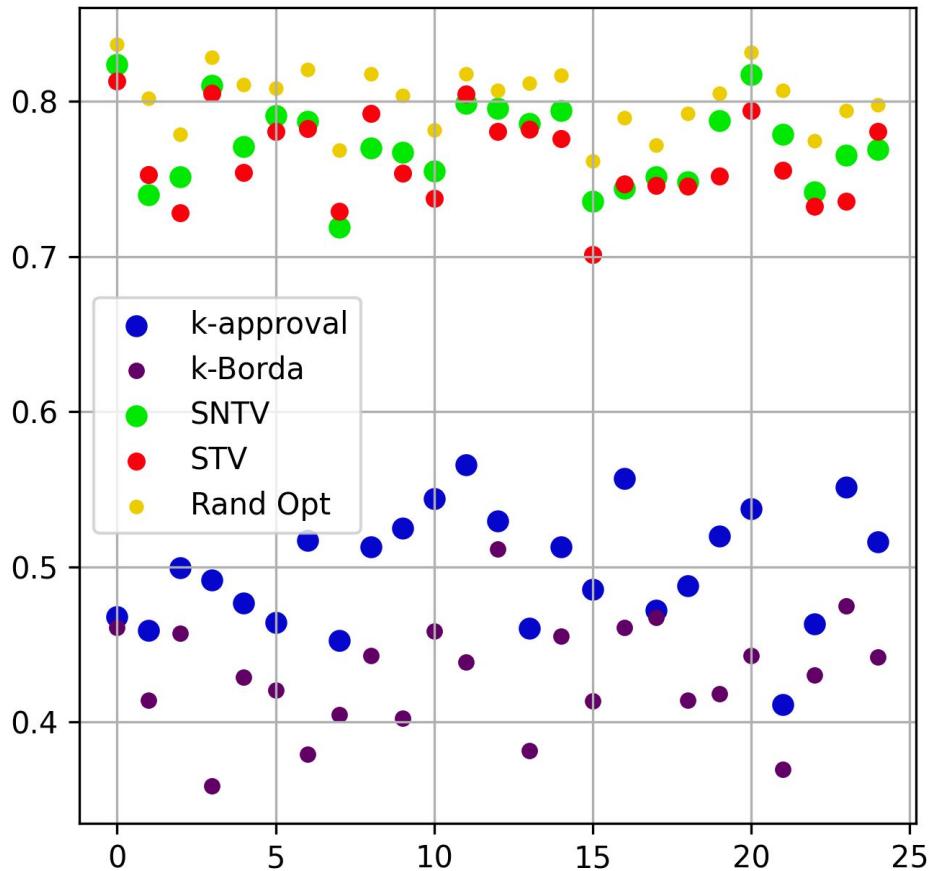
uniform/nomal

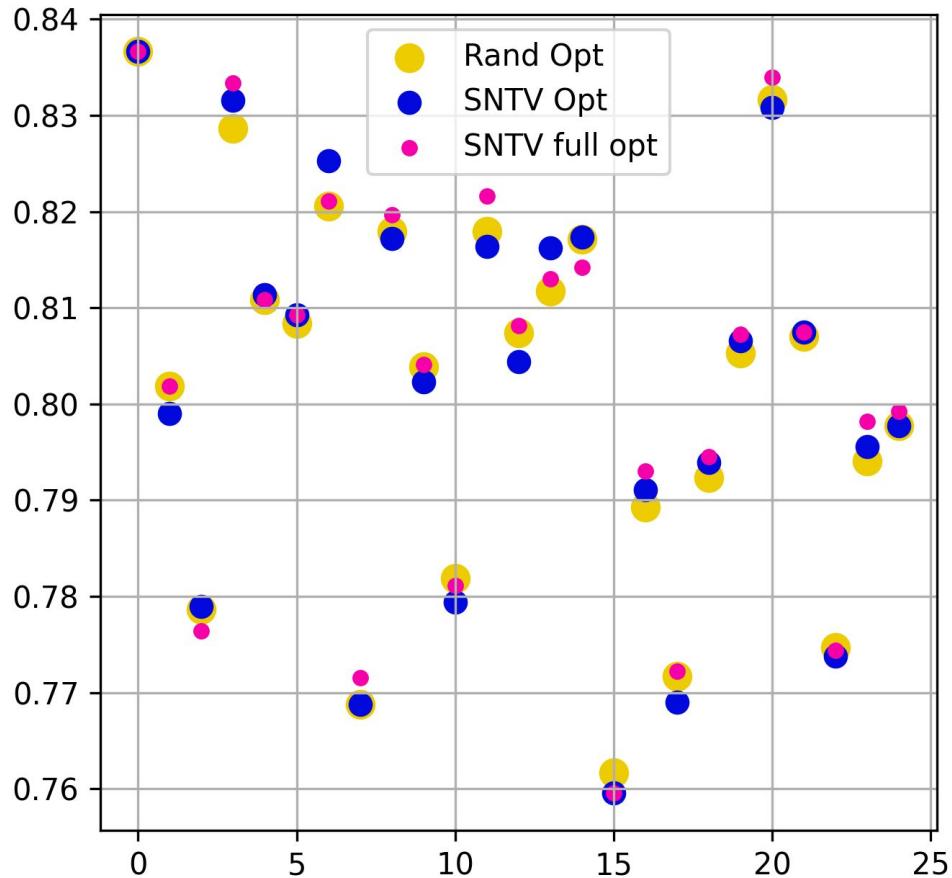
$|V| = 300$

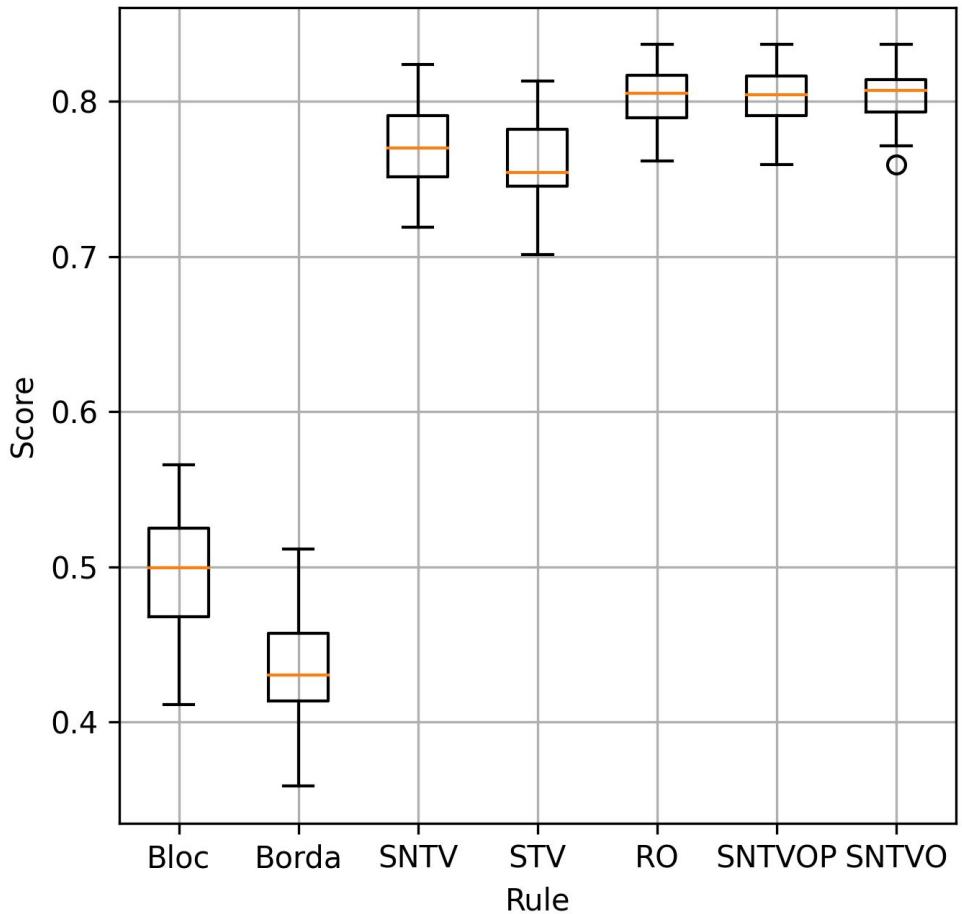
$|C| = 80$

$K = 20$

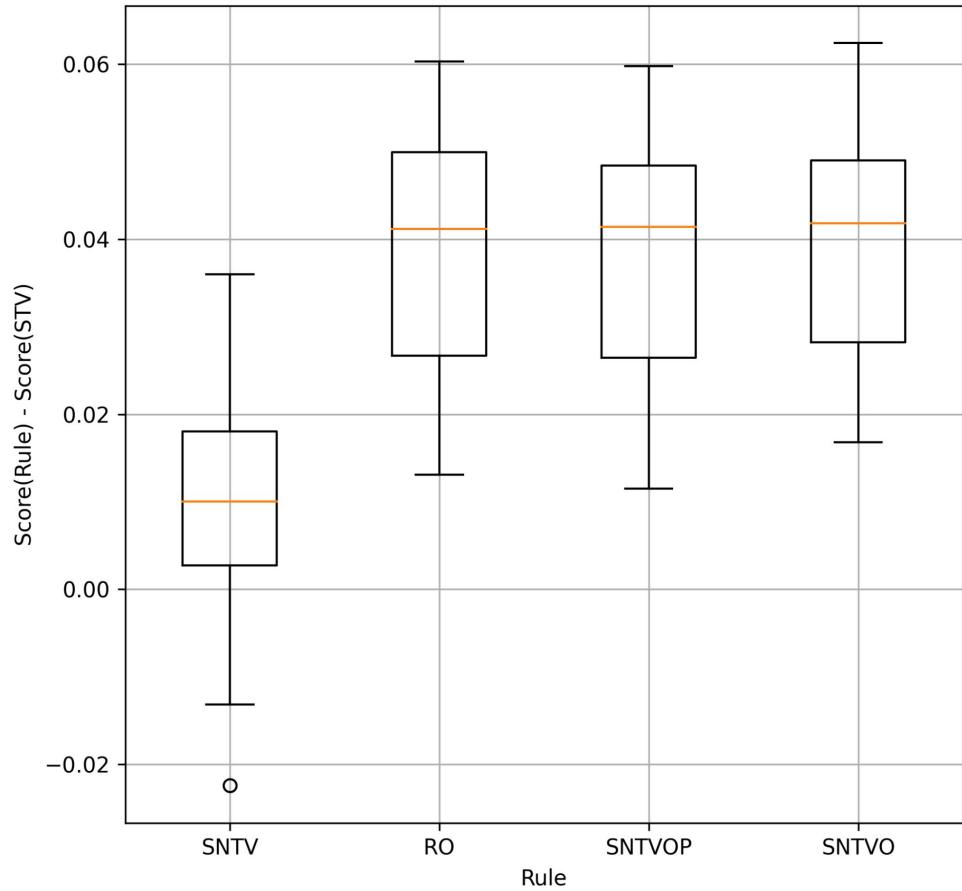








Для такого распределения
оптимизационные алгоритмы
также оказываются в среднем
на 5.6% лучше STV

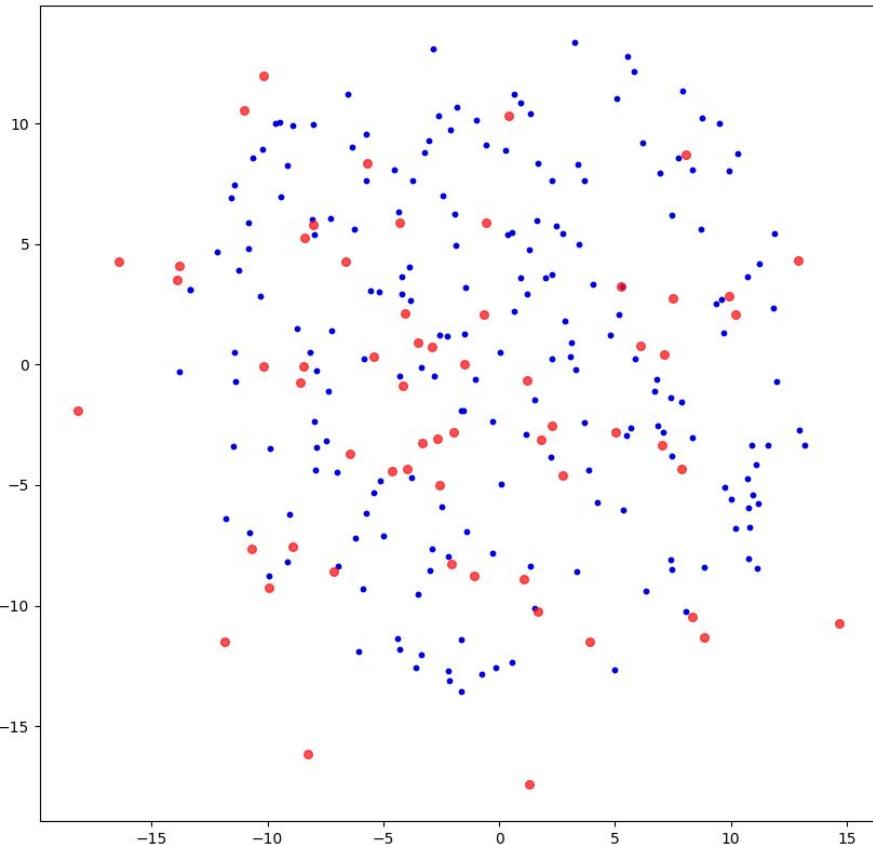


uniform/normal

$$|V| = 200$$

$$|C| = 60$$

$$K = 30$$



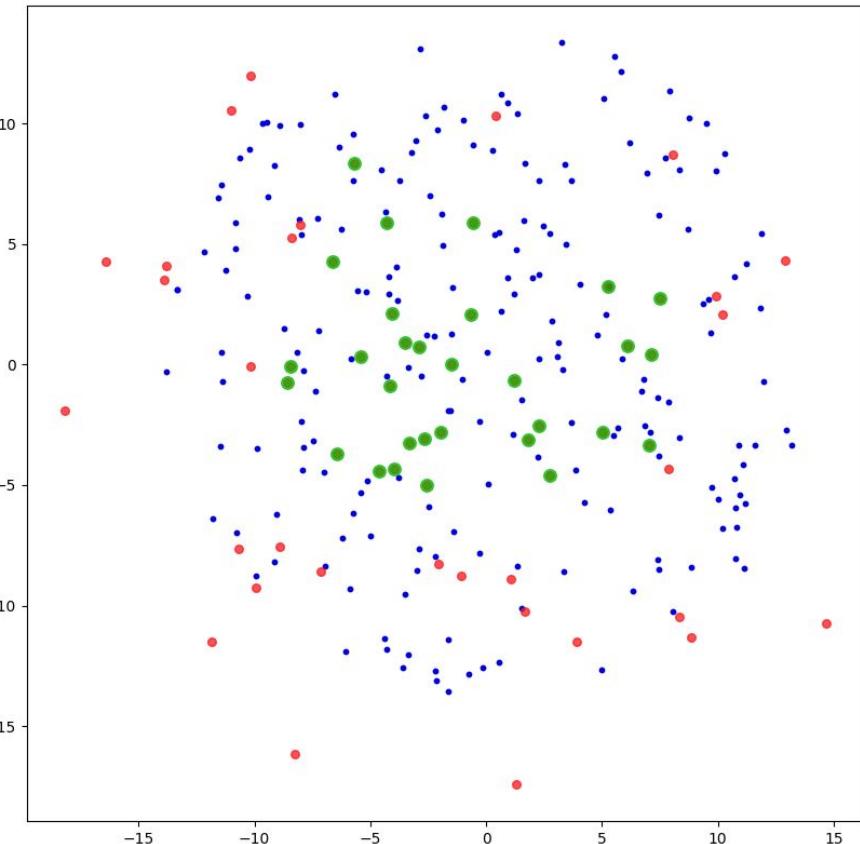
Bloc

```
Score of k-Approval rule is
```

```
0.7069660146667477
```

```
median Score of k-Approval rule is
```

```
0.8067498237908917
```



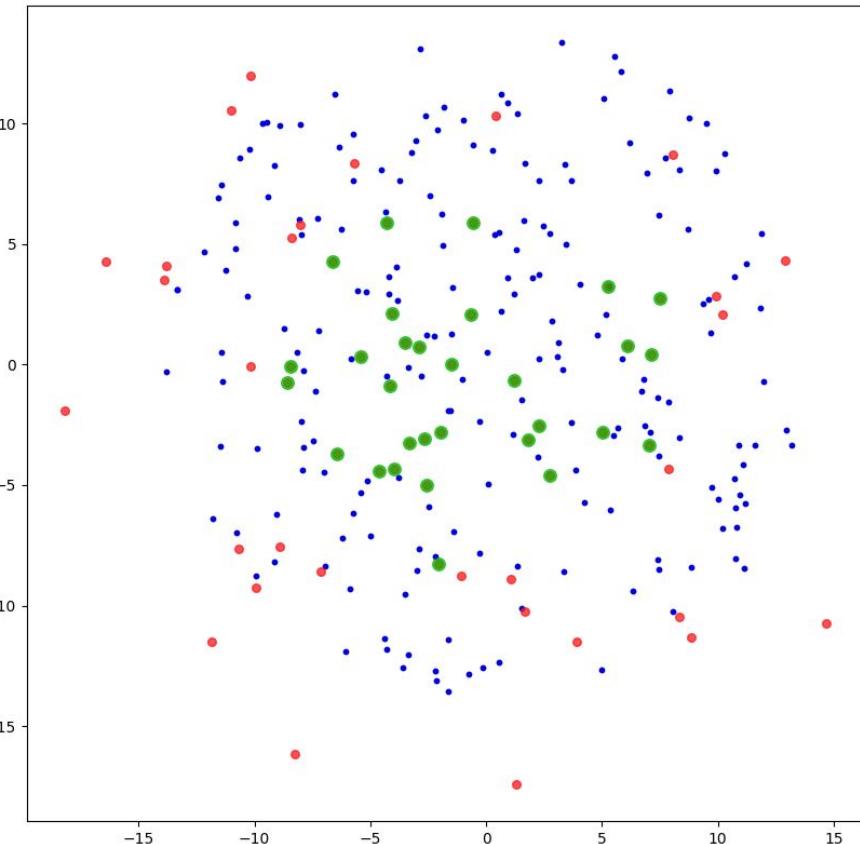
Borda

Score of k-Borda rule is

0.7192691795547481

median Score of k-Borda rule is

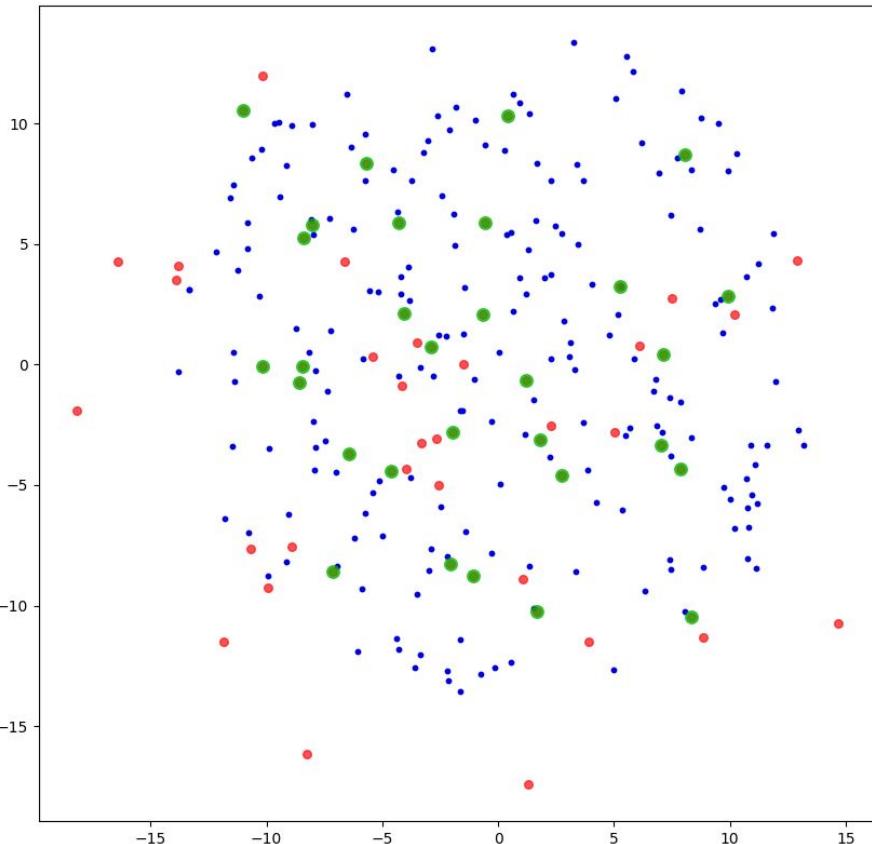
0.8635756856722894



SNTV

Score of SNTV rule is 0.9253329927095829

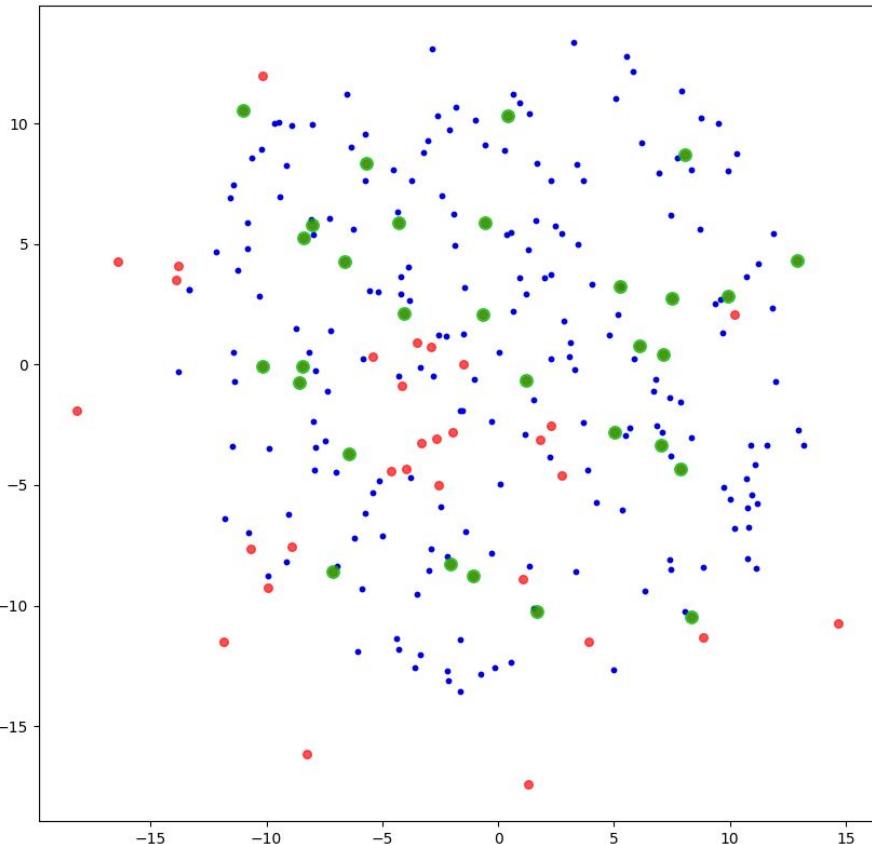
median Score of SNTV rule is 1.0



STV

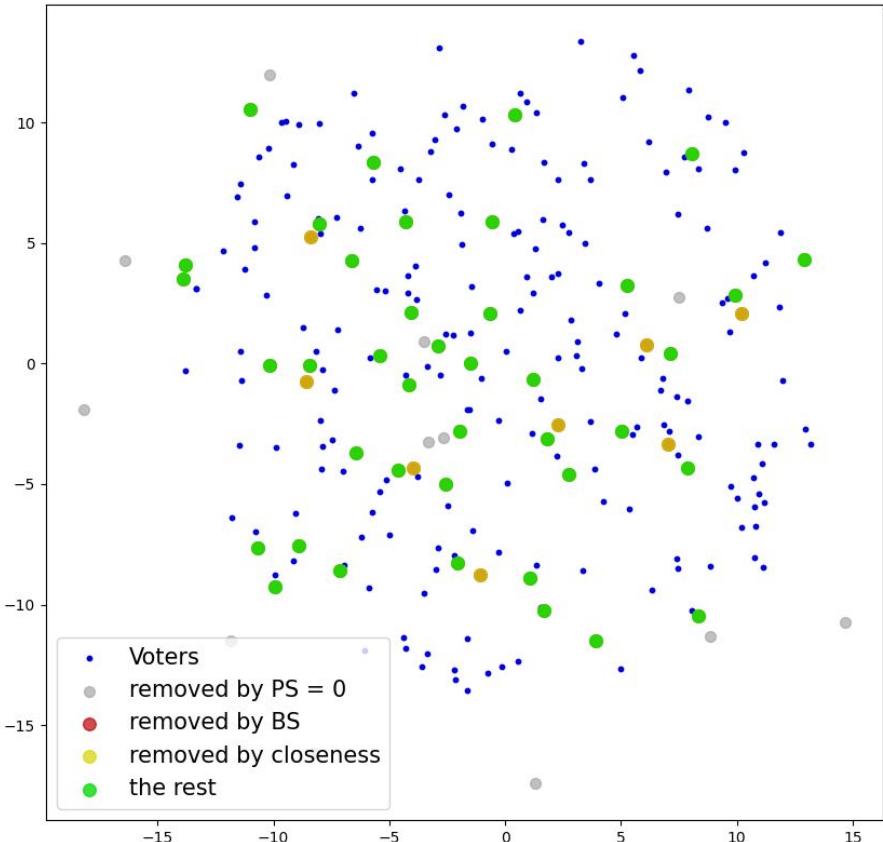
```
Score of STV rule is  0.9009125579966704
```

```
median Score of STV rule is  1.0
```



Pruning

Removed 20

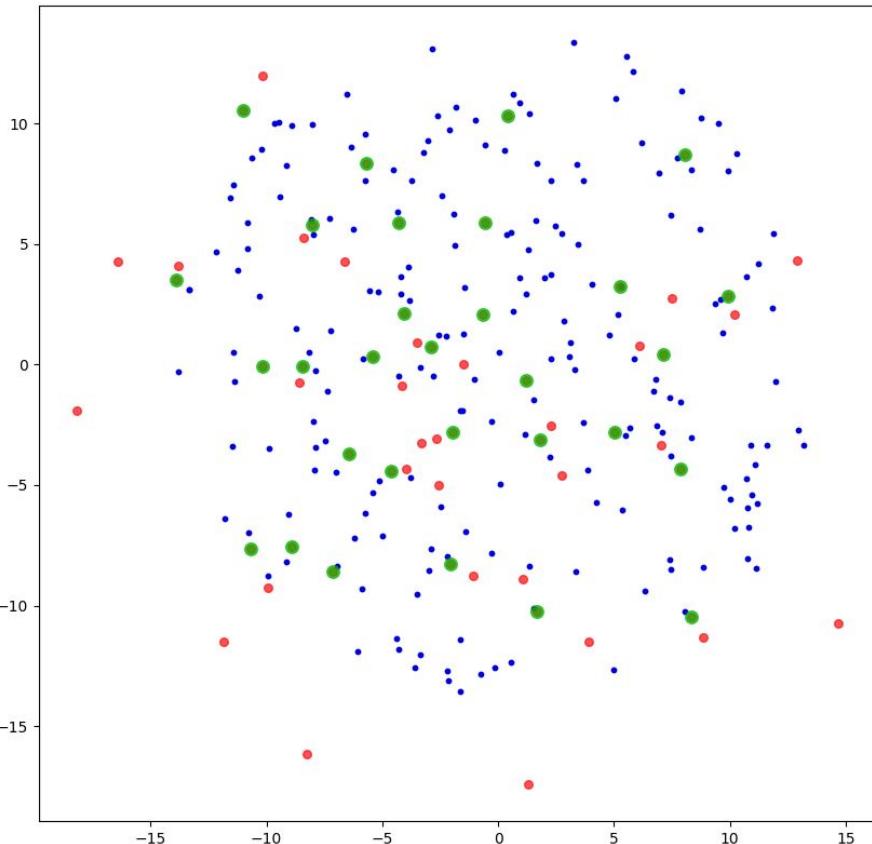


R-opt

Score of Optimization is

0.9451170006155162

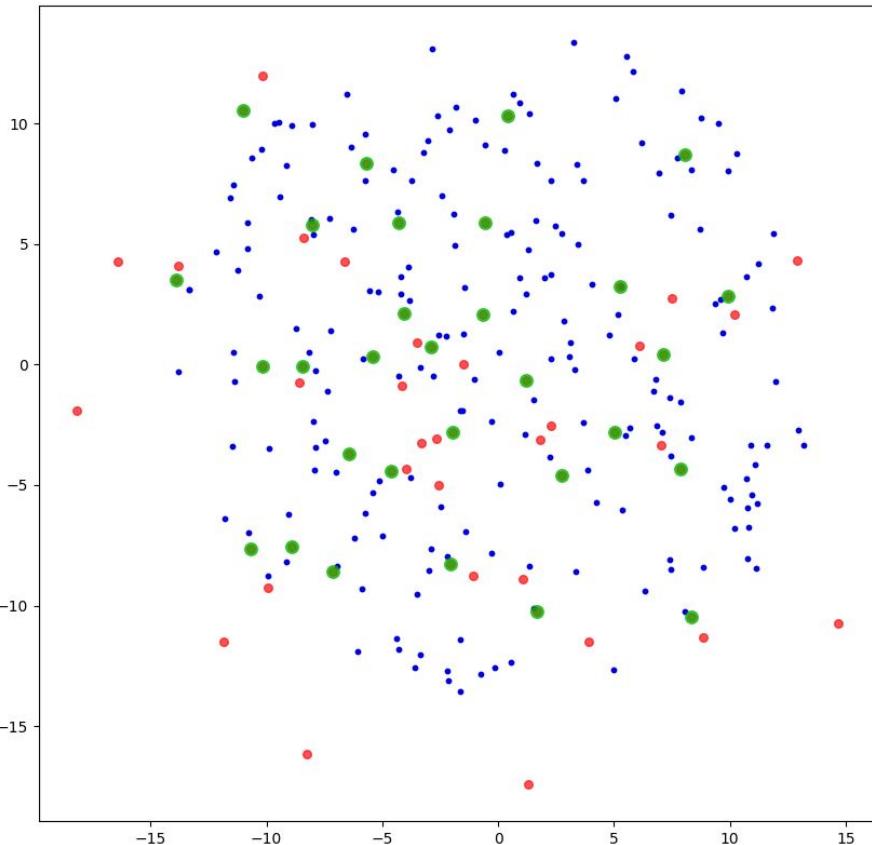
median Score of optimization is 1.0



SNTV-opt(P)

```
Score of SNTV-opt (prunned) rule is  
0.9449205623055994
```

```
median Score of SNTV-opt (prunned) rule is  
1.0
```

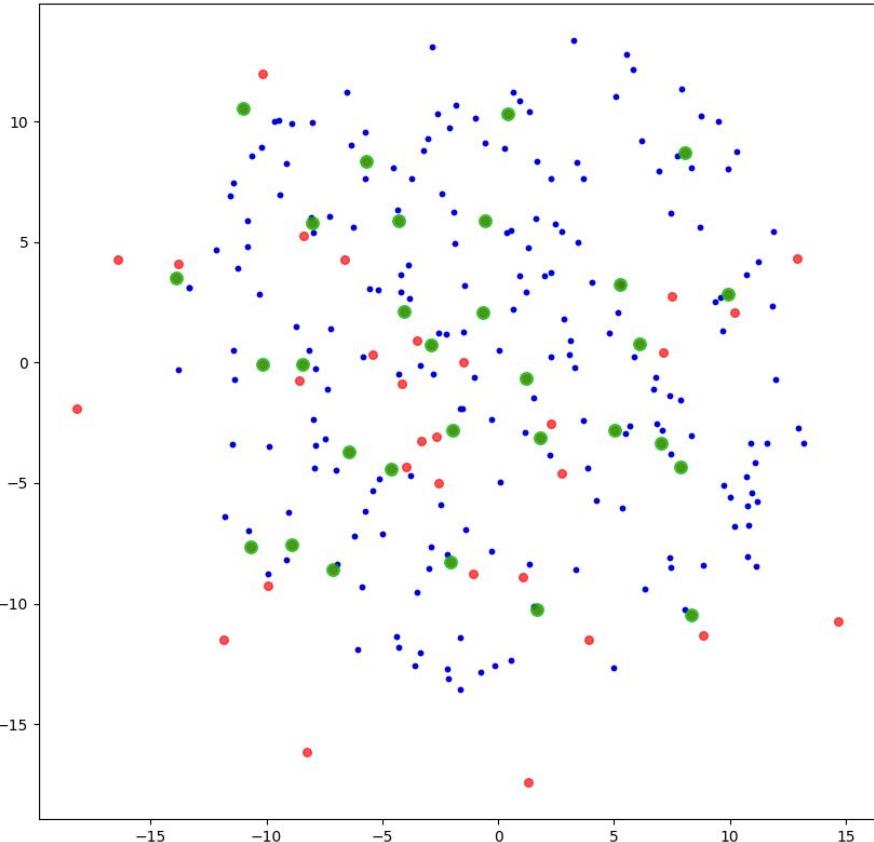


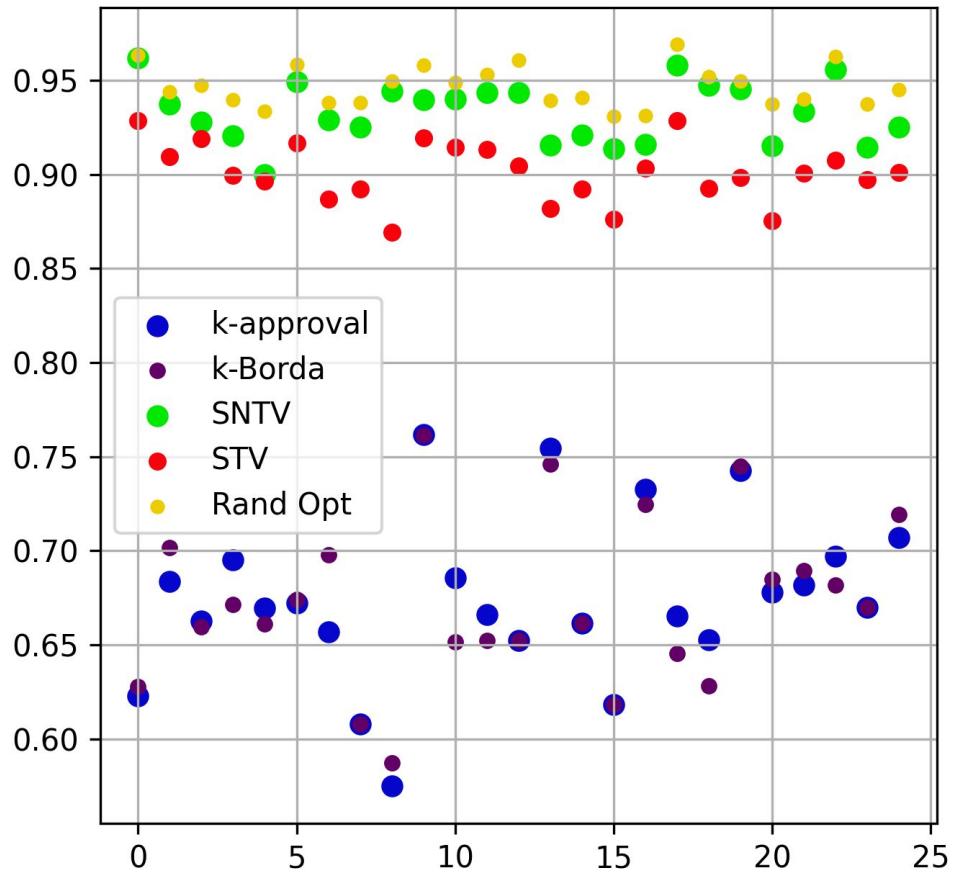
SNTV-opt

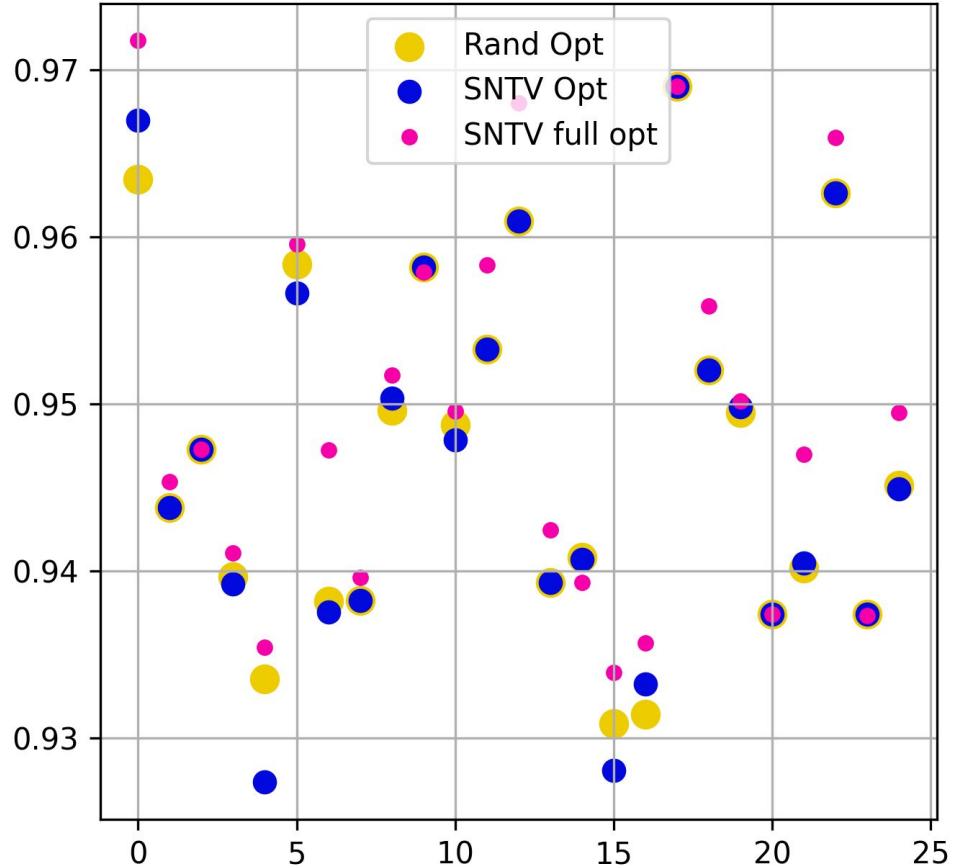
```
Score of SNTV-opt rule is
```

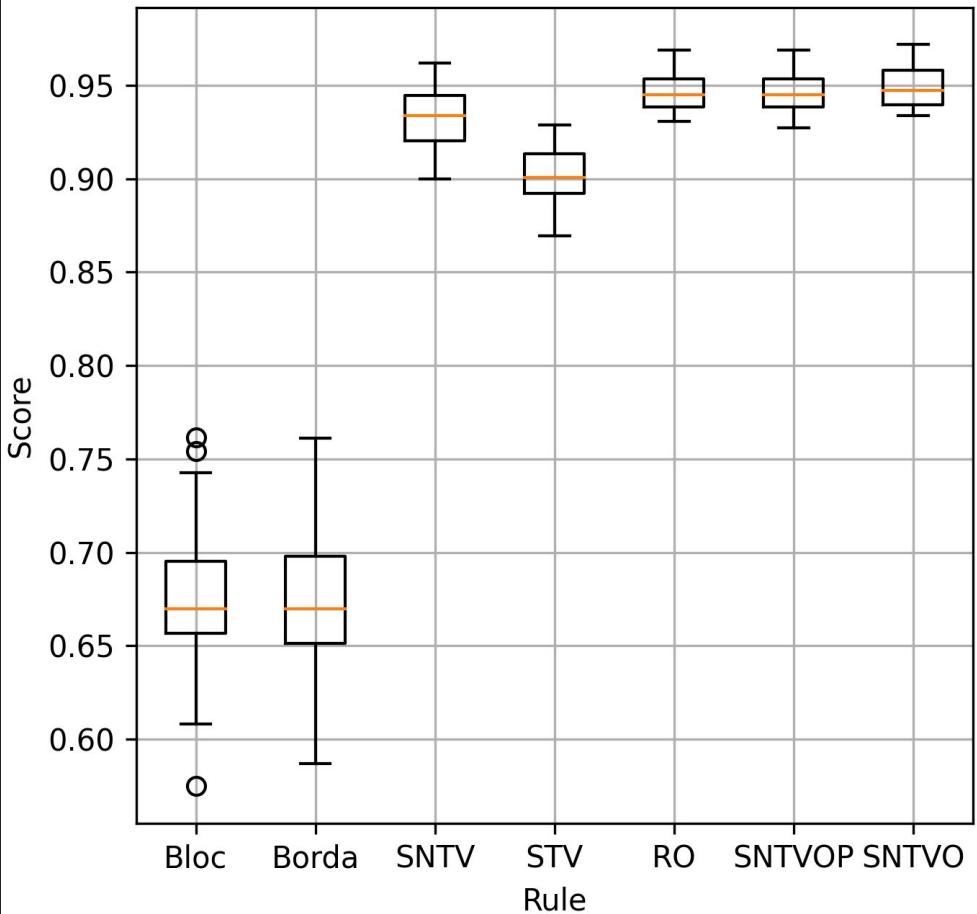
```
0.9494941137430277
```

```
median Score of SNTV-opt rule is 1.0
```

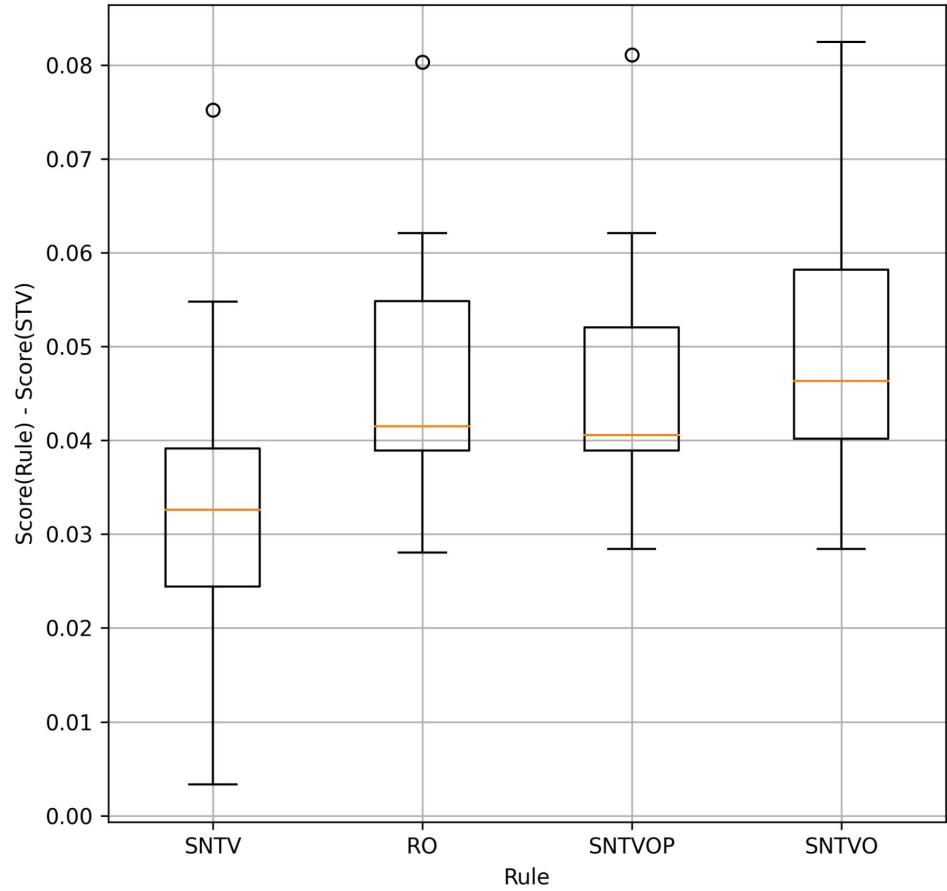








Для такого распределения
оптимизационные алгоритмы
также оказываются в среднем
на 4.5% лучше STV



Последние тесты говорят о том, что оптимизационные алгоритмы в среднем стабильно выдают Score, больше, чем у неоптимизационных алгоритмов.

Кроме того, разные подходы к начальному комитету в алгоритмах оптимизации статистически не влияют на получаемое значение скора.

Однако для последнего теста, где $k/|C| = 0.5$, верно, что SNTV-opt немного лучше, чем SNTV-opt с очищением поискового пространства, что впрочем логично, ведь с увеличением $k/|C|$ экстремальные значения PS и BS уменьшаются(увеличиваются), а значит вероятность, что удаляются кандидаты, увеличивающие Score. Это можно исправить, введя зависимость пороговых параметров очищения от отношения $k/|C|$, но тогда и ускорение вычислений будет падать с ростом этого отношения

Проверим теорию для

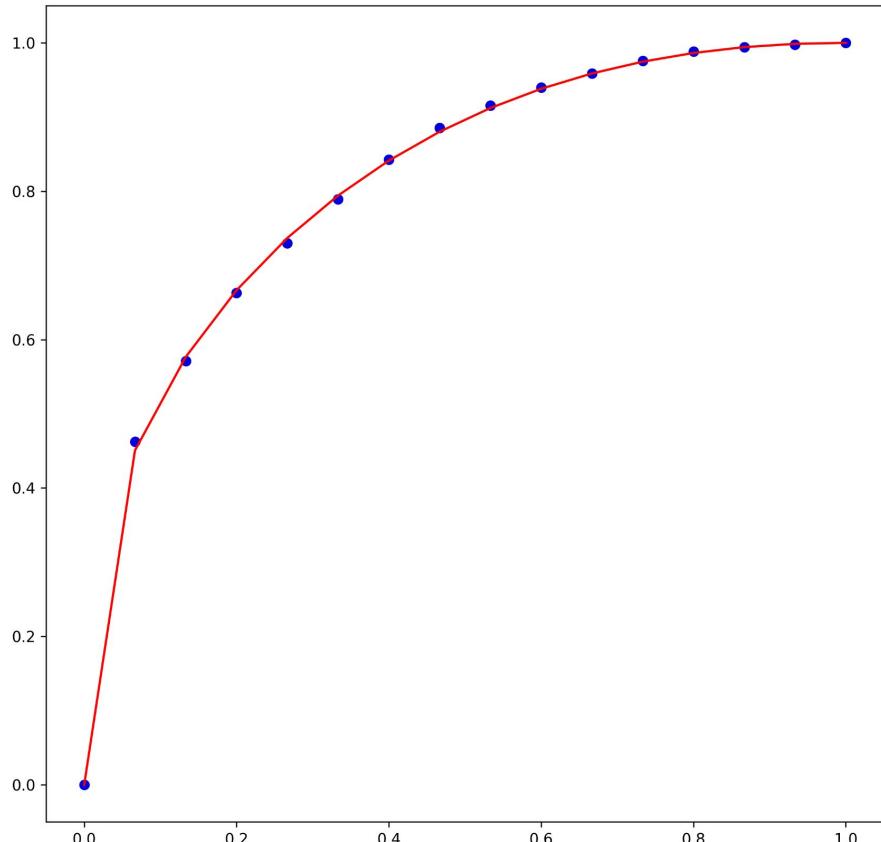
$|V| = 300$

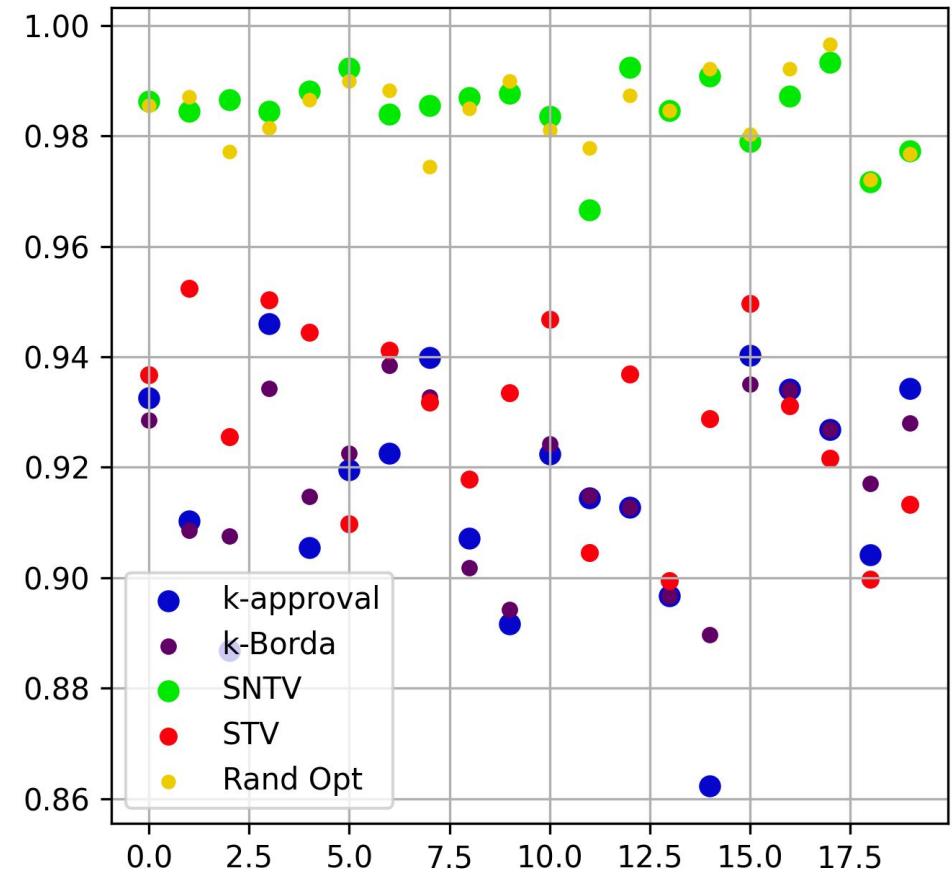
$|C| = 80$

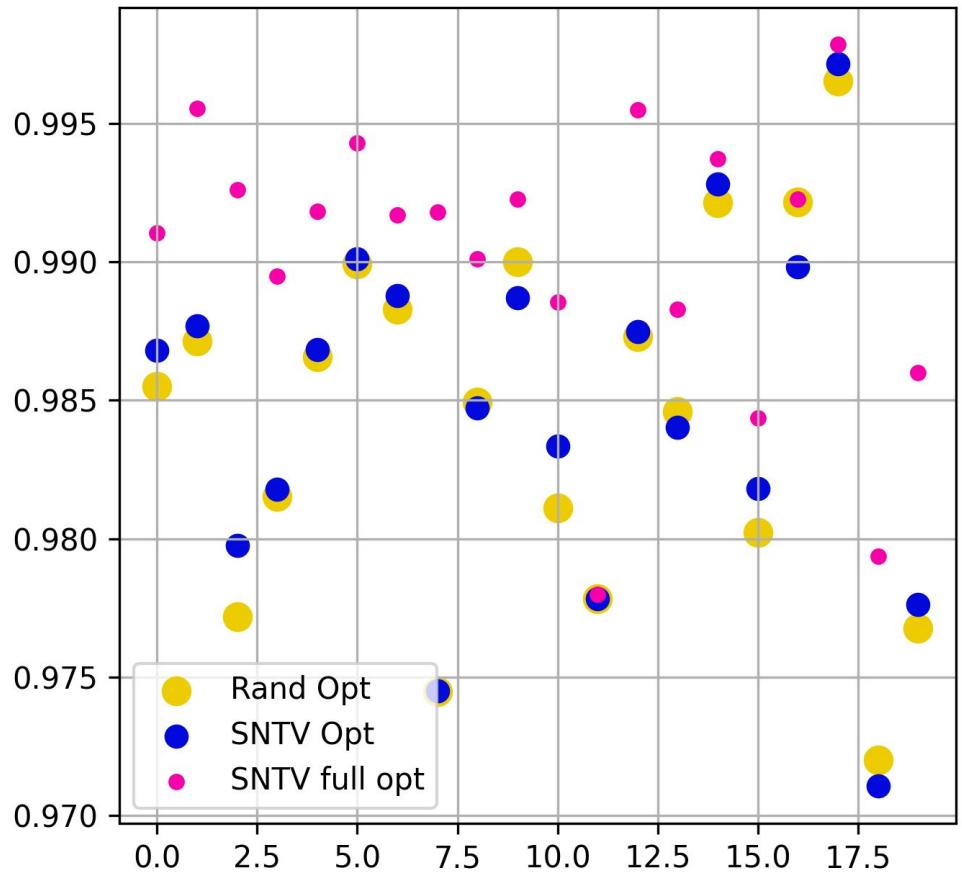
$K = 60$ ($k/|C| = 0.75$)

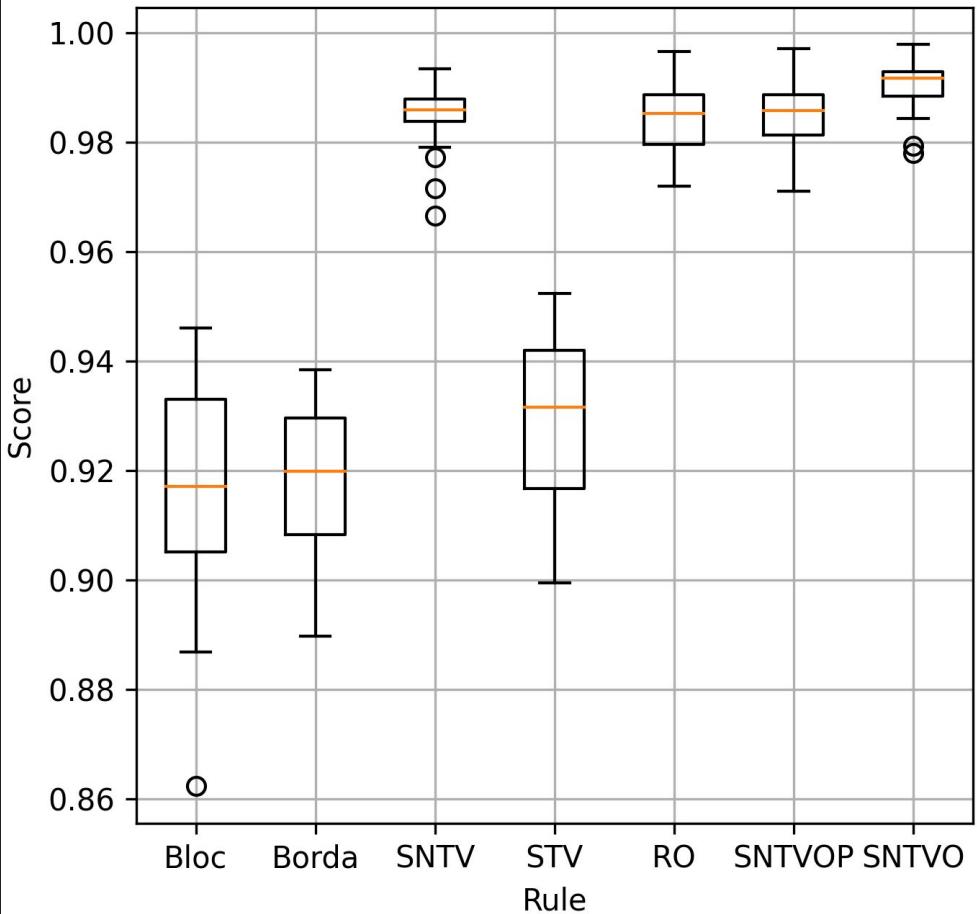
normal/uniform, 7/14

Помним, что для $k/|C| = 0.75$
максимально достижимый Score
уже достаточно близок к
единице





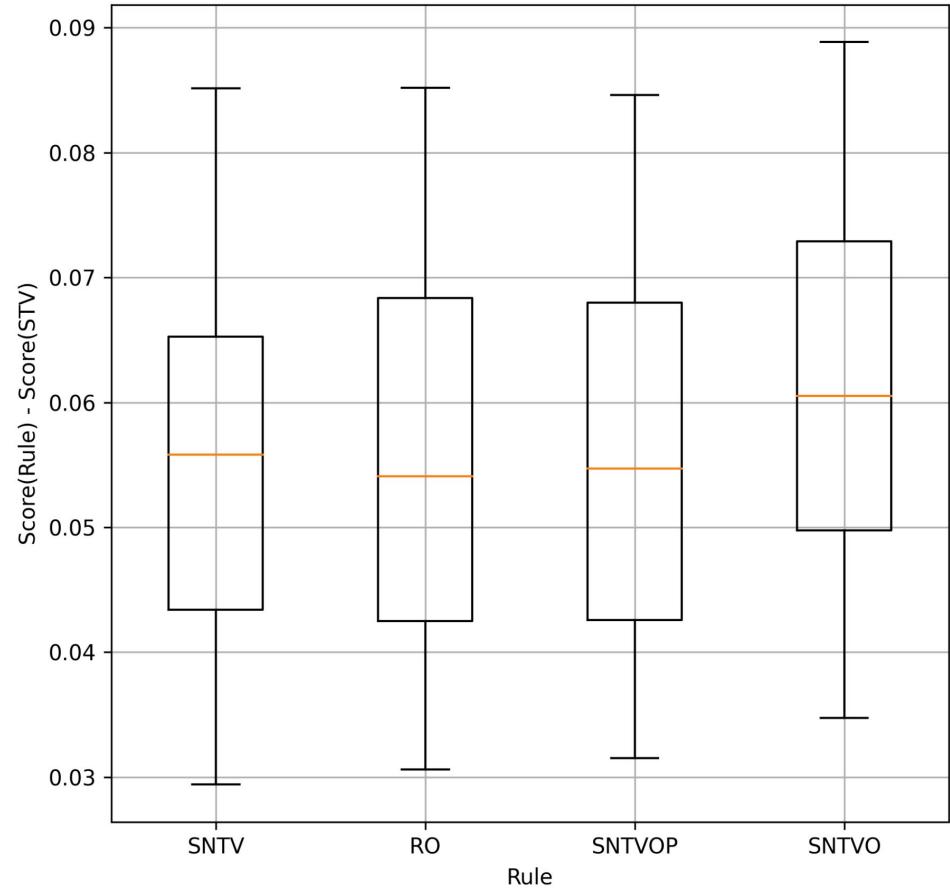




Видим, что при большом
отношении $k/|C|$:

Во-первых, SNTV перестаёт
быть в среднем хуже
оптимизационных алгоритмов

Во-вторых, алгоритм с
отсутствием очищения
действительно оказывается
немного лучше других
оптимизационных алгоритмов



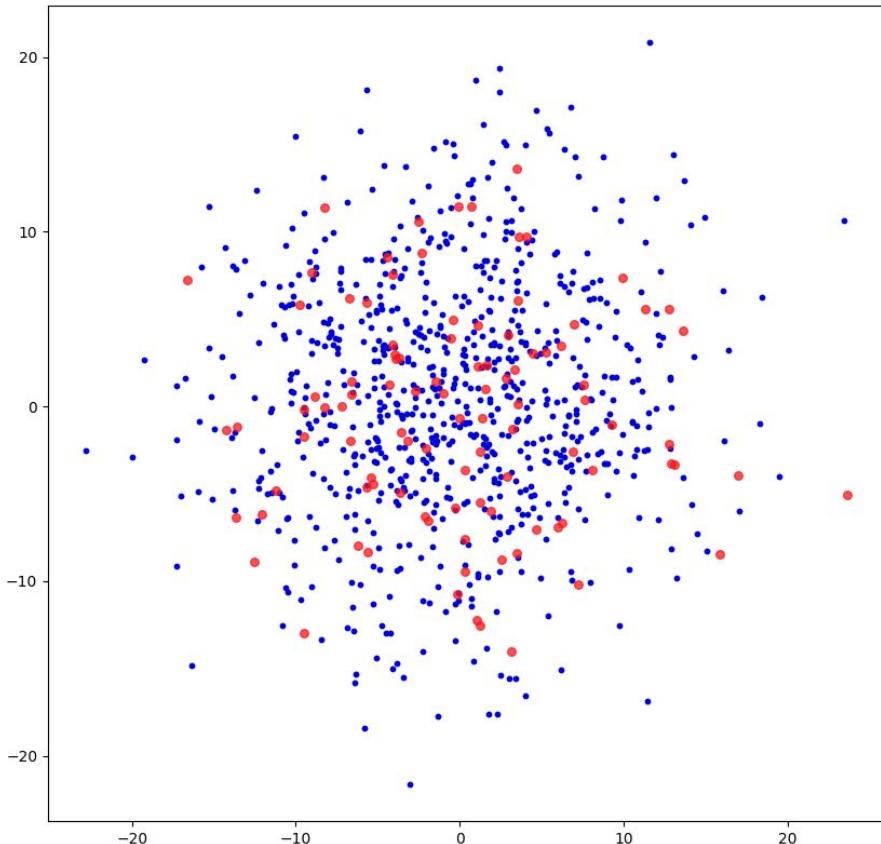
Переменное $|V|$

$|C| = 100$

$k = 20$

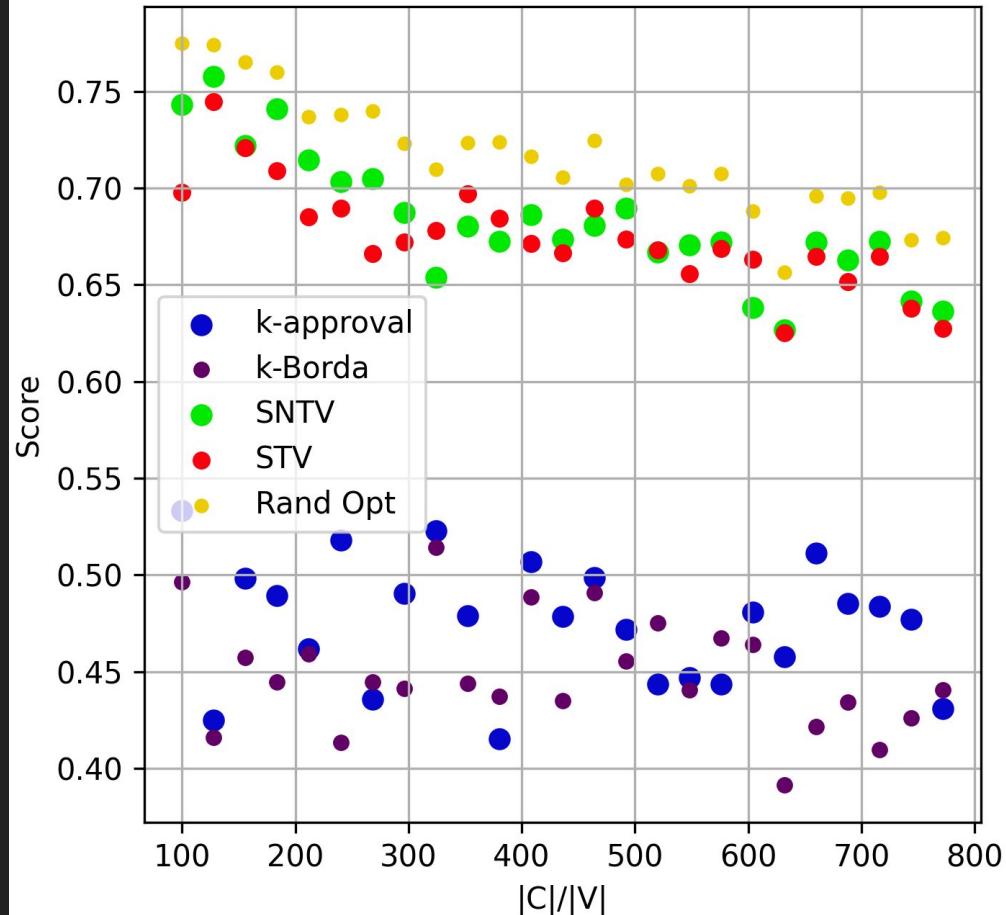
$|V| = 100, 128, 156, \dots, 772$

normal/normal, 7/7

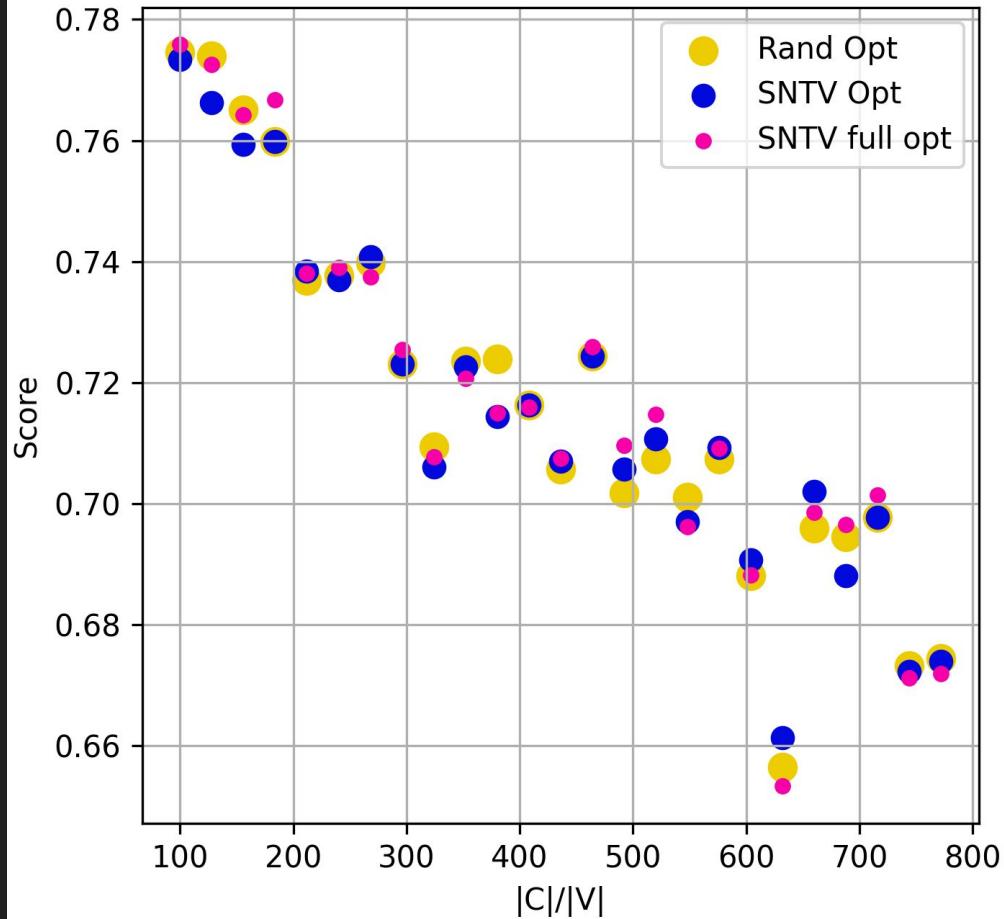


С ростом количества избирателей при неизменных остальных параметрах логично, что удовлетворить всех становится всё сложнее, а потому закономерно, что в среднем снижается и скор.

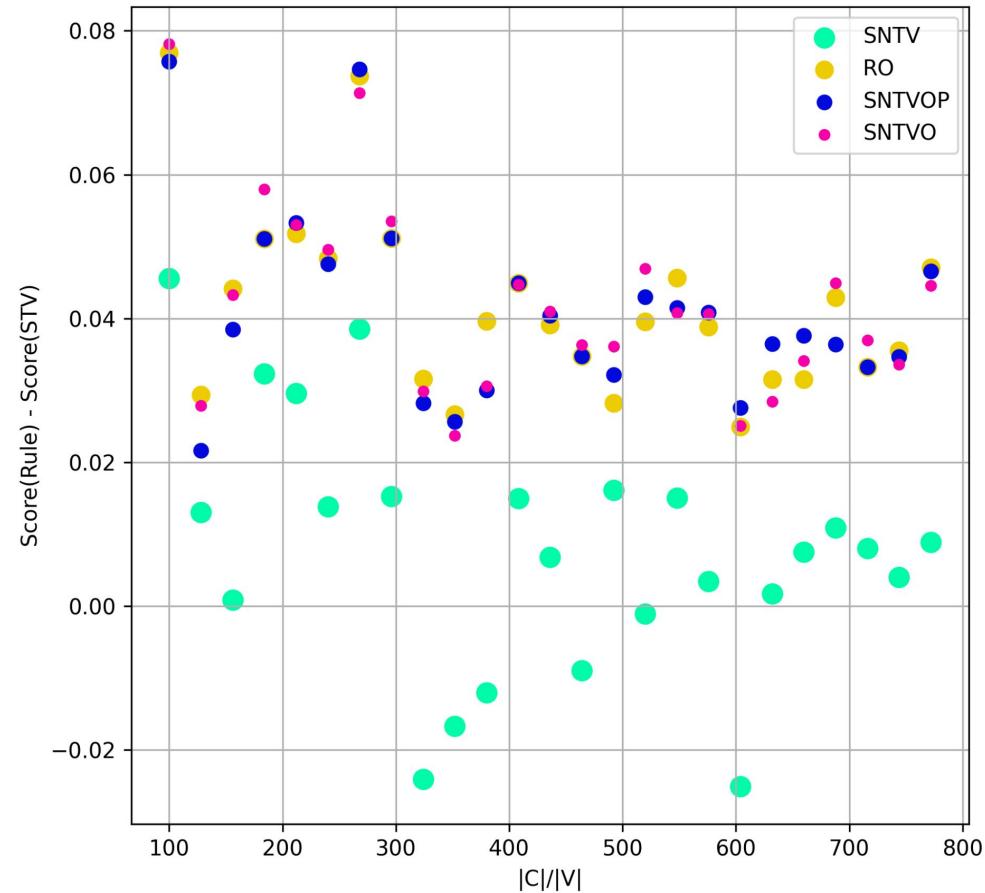
Ранее уже было показано (плохим графиком, который естественно надо сделать хорошо и со средними значениями) что и максимально возможный скор убывает



При этом скоры всех правил
убывают примерно одинаково

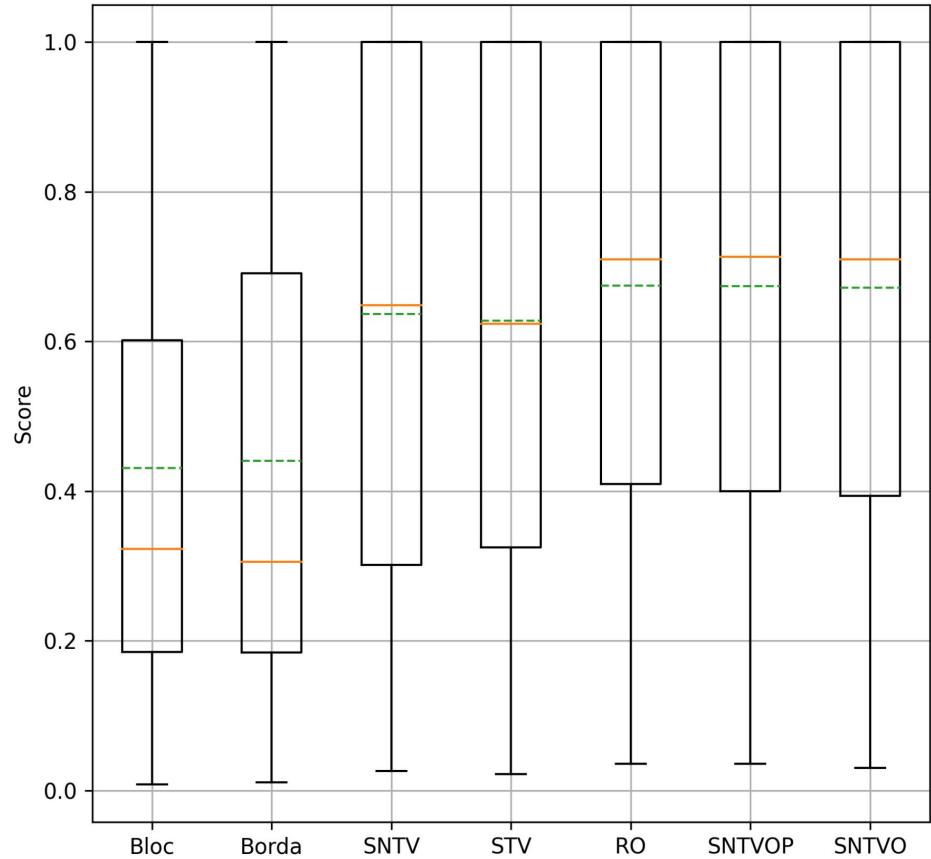


Видно, что с ростом $|V|$
практически не меняется
разность с Score(STV), значит
скоры всех правил меняются
примерно одинаково



Распределение избирательских скоров

Теперь рассмотрим не среднее значение по всем избирателям, а на распределение скора (правда пока только для одного теста, но всё же)



Какие никакие выводы

- Значение Score действительно почти не зависит от соотношения среднеквадратичных отклонений распределений избирателей и кандидатов, как и от функций распределений (при условии, что соотношение остаётся неизменным)
- Оптимизационные алгоритмы по качеству превосходят остальные алгоритмы, хоть те и выигрывают по времени работы
- Для большого $k/|C|$ целесообразнее использовать SNTV, который при больших значениях этого параметра не уступает оптимизационным алгоритмам, но превосходит их по времени работы
- Максимально возможный Score странно зависит от $k/|C|$, как-то зависит от $|C|/|V|$, и является случайной величиной
- Оптимизационные алгоритмы кроме того обеспечивают меньший разброс в значениях Score(v_i), а значит являются более социал-демократичными

Что можно ещё сделать?

- Определить зависимость a и b в среднем
- Реализовать другой метод дискретной оптимизации и сравнить
- Реализовать поиск оптимального s-score на основе методов численной оптимизации
- Проверить эффективность алгоритмов с точки зрения времени вычисления
- Увеличить \dim пространства, проверить зависимость максимального скора от размерности

Качественное изменение задачи

Можно изменить условие: пусть у избирателя нет конкретной точки в пространстве, ему соответствующей, но есть некоторая область, кандидаты внутри которой считаются приемлемыми, а вне которой - неприемлемыми. Тогда возникают вопросы:

- Как выбрать победителей, чтобы как можно большее число избирателей осталось удовлетворено
- Какой наименьший размер у комитета, удовлетворяющего всех избирателей
- Это по сути аналог задачи *Covering graph*, что также является задачей дискретной оптимизации, которую можно решать методом MIP