Simultaneously Identifying Multiple Pathways with Alternative Scoring Criteria

Bowen Deng

Dept. of Prob. and Stat.

We'll discuss in the following aspects:

- Simultaneously Detection of Multiple Pathways with Alternative Scoring Criteria
- Danger in Simulation Study
- Difficulty in Stability Criteria
- Further Work

Simultaneously Detecting Multiple Pathways

Objective:

$$\max O(M_1, \cdots, M_t) = \sum_{i=1,\cdots,t} W(M_i)$$
$$s.t. |M_i| \in [k_{\min}, k_{\max}]$$
$$M_i \cap M_j = \emptyset$$

Greedy approach: Iteratively find maximizer $\hat{M}_i, i = 1, \cdots, t$ of $W(M)$.

Integer Linear Programming

$$\max O(M_1, \cdots, M_t) = \sum_{\rho=1}^{t} \sum_{i=1}^{m} (2C_i(M_\rho) - \sum_{j=1}^{n} I_{M_\rho}(j) A_{ij})$$

$$s.t. \sum_{j=1}^{n} I_{M_\rho}(j) A_{ij} \geqslant C_i(M_\rho)$$

$$\sum_{\rho=1}^{t} I_{M_\rho}(j) \leqslant 1$$

$$k_{\min} \leqslant \sum_{j=1}^{n} I_{M_\rho}(j) \leqslant k_{\max}$$

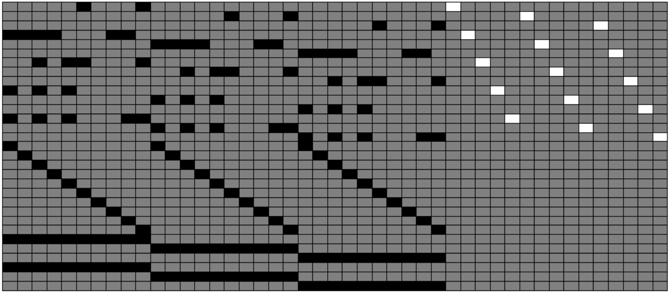


Figure: The coefficients of the ILP

MDendrix and IterDendrix

To find t pathways, one method is to solve the ILP directly. (MDendrix)
Another iterative approach is to solve the ILP with $t = 1$, delete the identified genes, and run iteratively. (iterDendrix)
By theory, the time cost of MDendrix and iterDendrix is comparable with small t.

$$\begin{aligned} TC(M) &\geqslant TC(\text{iter once}) \\ &= \frac{\sum_{\rho=1}^{t} TC(\text{iter once})}{t} \\ &= \frac{TC(\text{iterDendrix})}{t} \\ &\geqslant \frac{TC(\text{mDendrix})}{t} \end{aligned}$$

The authors used CPLEX v12.3 for implementation. We use lpSolve package in R. Set simulation data with $m = 200$, $n = 1000$, $l = 10$, $t = 3$. $k_{\text{min}} = 8$, $k_{\text{max}} = 12$.

MDendrix:

Time Cost	184.32	
Result	Score	Score of Standard
1 ∼ 10 761 774	30	32
11 ∼ 20 752 973	35	40
21 ∼ 30 34 109	32	36

IterDendrix:

Time Cost	7.06	
Result	Score	Score of Standard
1 ∼ 10 214 774	30	32
21 ∼ 30 34 109	32	36
11 ∼ 20 652 752	35	40

Candidate Criteria

For mutation matrix A, p takes value in m patients, and g takes value in n genes. M is a set of genes.
We borrow the criteria from RME, an alternative approach for driver pathway identification.
Coverage Score:

$$C(M) = \frac{\#(\exists g \in M \text{ mutates in } p)}{m}$$

Exclusivity Score:

$$E(M) = \frac{\#(\text{exactly one } g \in M \text{ mutates in } p)}{\#(\exists g \in M \text{ mutates in } p)}$$

Denote $I_M(p)$ the occurrence indicator of mutations of patient p in M.

$$\begin{aligned} S(M) &= C(M) + E(M) \\ &= \frac{\#\{I_M(p) > 0\}}{m} + \frac{\#\{I_M(p) = 1\}}{\#\{I_M(p) > 0\}} \\ W(M) &= 2\#\{I_M(p) > 0\} - \sum_p I_M(p) \end{aligned} \tag{1}$$

The objective is to find $\hat{M}$ as the maximizer of $S(M)$. We could also restrict $|M| = k$.

Properties of New Criteria

- The score is fixed for patients who has mutations in the gene set.
- Fixing coverage, $E(M)$ promotes more ones.
- Fixing $\#\{I_M(p) = 1\}$, coverage tends to obtain either extreme: whole coverage or all ones.

Simultaneous Detection with New Criteria

We combine this new criteria with mDendrix to detect a mutually exclusive set of genes $M = \{M_1, \cdots, M_t\}$ which maximizes:

$$S(M) = \sum_{\rho=1}^{t} S(M_\rho)$$

$k_{\text{min}} \leqslant |M_\rho| \leqslant k_{\text{max}}$.

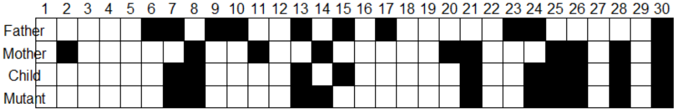
Computation

The objective is to maximize $S(M)$. GA (Genetic Algorithm) is one of the top choices:

- The problem is no longer an ILP (Integer Linear Programming) task.
- MCMC might trap in a local maxima.
- The return value of GA is a set of solutions, suboptimal solutions are obtained as bonus.
- GA's time cost is tractable.
- GA is flexible for further integrated approach and variable scoring settings.

However, we should generalize the GA because it only works for binary case.

Multiple Value Genetic Algorithm



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
Father	□	□	□	□	□	■	■	□	■	■	□	□	■	□	■	□	■	□	□	□	□	□	■	■	□	□	□	□	□	■
Mother	□	■	□	□	□	□	□	■	□	□	■	□	□	■	□	□	□	□	□	■	■	□	□	□	■	■	□	■	□	■
Child	□	□	□	□	□	□	■	■	□	□	□	□	■	□	■	□	□	□	□	□	■	□	□	■	■	■	□	■	□	■
Mutant	□	□	□	□	□	□	■	■	□	□	□	□	■	■	□	□	□	□	□	□	■	□	□	■	■	■	□	■	□	■

Figure: Genetic Algorithm with $0-1$ values

We generalize GA to MVGA (Multiple Value Genetic Algorithm), it is analogous to the GA.

Review of GA

- Select a population
- Crossproduct
- Mutation
- Form a new population

However, the crossproduct should deal with multiple value vectors rather than $0 - 1$ vectors.

Toy Example of GA

We would like to maxmize $f(x) = e^{-x^2}, x \in [-5, 5]$.

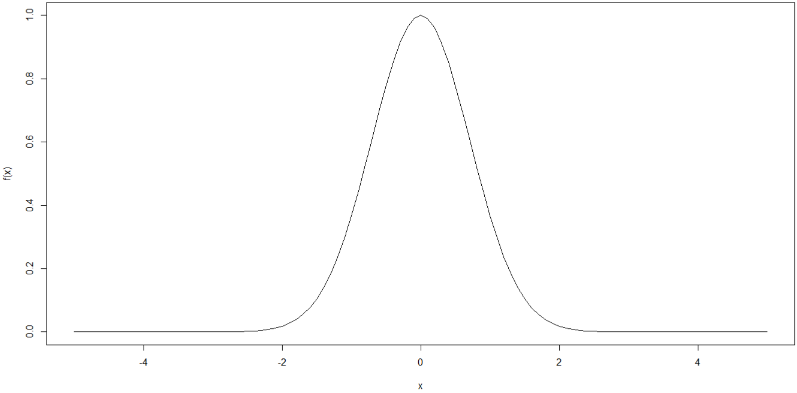


Figure: Objective Function

Select Population

We first select a parent generation of size 10 from $[-5, 5]$.
2.42 2.33 2.04 2.00 -0.71 1.16 -1.16 1.75 3.66 4.20
We calculate $f(x)$ for these parents:
2.75e-03 4.22e-03 1.51e-02 1.79e-02 5.99e-01 2.58e-01 2.59e-01 4.66e-02 1.47e-06 2.01e-08
We select two of the parents to produce a child.
The probability of x being selected is proportional to $f(x)$, therefore, an elite is more likely to be chosen than a normal person candidate.
The crossproduct is flexible with specific problems, here we could use $cp(x, y) = \frac{x+y}{2}$.
For example, the parents are -0.715157 1.163802, the child is 0.22.

Crossproduct

With probability 0.1, the child might mutate.
For example, 0.22 might mutate to 0.21 or 0.23 if the score could be higher.
In this case, 0.22 will mutate to 0.21.
After ten productions, we get ten parents and ten children.
Parents:
2.42 2.33 2.04 2.00 -0.71 1.16 -1.16 1.75 3.66 4.20
Children:
-0.9384129 -0.9384129 0.2243223 -0.9384129 0.2243223 0.8115732 0.2243223 -0.9384129 -0.9384129 -0.9384129

Iteration

We pool the parents and the children and get 20 candidates.
We remain the 10 top scored ones.
0.2243223 -0.7151570 0.8115732 -0.9384129 -1.1616687 1.1638017 1.7504537 2.0054783 2.0472774 2.3383034 2.4277411 3.6646631 4.2097401
We use them to produce the next generation iteratively until convergence.
-7.244348e-09 5.279007e-09 2.856562e-10 -9.826706e-10 -5.023779e-09 -3.485072e-10 -7.144341e-09 4.050658e-09 2.782331e-09 6.547334e-09

Crossproduct

Denote father: $a_1, \cdots, a_n$, $b_1, \cdots, b_n$, $a_i, b_i \in \{0, 1, \cdots, t\}$.
$a_i = \rho$ means i-th gene is in the ρ-th set, if $\rho = 0$, i-th gene is not selected.
The motivation is to find a feasible solution corresponding to child $c_1, \cdots, c_n$ under constraint:

$$\sum_{i=1}^{n} [(1 - x_i) I(a_i = \rho) + x_i I(b_i = \rho)] \in [k_{\min}, k_{\max}]$$

where $x_i = 0$ represents $c_i = a_i$, and $x_i = 1$ represents $c_i = b_i$.
We also wish the child to deviates from the parents so that $\sum_i x_i \approx \frac{n}{2}$.
Therefore, we would like to maximize $\sum_i x_i$, with constraint $\sum_i x_i \leqslant \frac{n}{2}$.

Selection of $k_{\min}$ and $k_{\max}$

The hunt for an appropriate k not only meets difficult in theory but also deviates results in simulation study.

Theoretically, the peak value often takes at $l = k \pm 1$ rather than exact k.

Practically, the overlapping criteria is not unimodal and unstable with small t.

On the other hand, large t is computationally expensive.

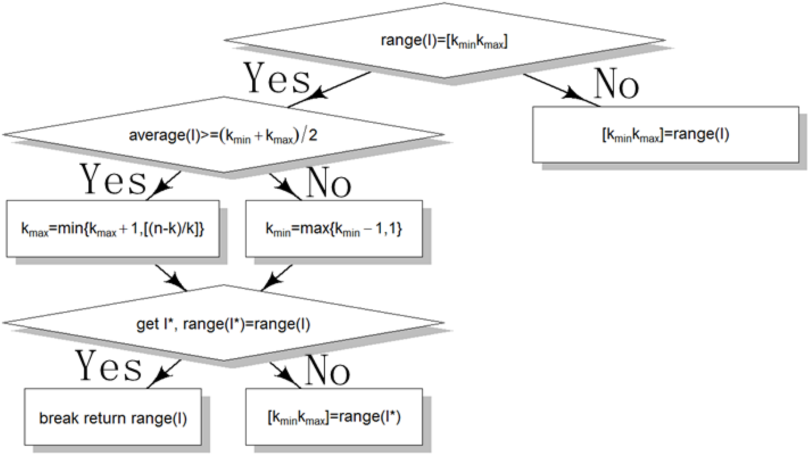
Moreover, biologically speaking, we should not restrict ourselves to one certain k.

We could try estimating the bound of k: $k_{\min}$ and $k_{\max}$.

It would make much more sense.

The idea is that for appropriate $k_{\min}$ and $k_{\max}$, the length of $M_1, \cdots, M_t$ should be dispersed in $[k_{\min}, k_{\max}]$ uniformly.

Therefore, we follow the procedure to search for the best $k_{\min}, k_{\max}$.



range(l)=[$k_{min}k_{max}$]

Yes

No

average(l)>=($k_{min} + k_{max}$)/2

[$k_{min}k_{max}$]=range(l)

Yes

No

k_{max}=min{k_{max} + 1,[(n-k)/k]}

k_{min}=max{k_{min} − 1,1}

get l*, range(l*)=range(l)

Yes

No

break return range(l)

[$k_{min}k_{max}$]=range(l*)

Toy Example

Suppose we would like to find

$$
\begin{aligned}
(x_1, x_2, \cdots, x_{10}) &= \arg\min O(\vec{x}) \\
&= \sum_{i=1}^{5} (x_i - 4)^2 + \sum_{i=6}^{10} (x_i - 6)^2 \\
s.t. k_{\min} \leqslant & x_i \leqslant k_{\max}
\end{aligned}
$$

The ideal $k_{\text{min}} = 4$, $k_{\text{max}} = 6$.
Take initial $k_{\text{min}} = 3$, $k_{\text{max}} = 5$, the iteration runs as the following:

The ideal $k_{\text{min}} = 4$, $k_{\text{max}} = 6$.
Take initial $k_{\text{min}} = 3$, $k_{\text{max}} = 5$, the iteration runs as the following:

- $k_{\text{min}} = 3, k_{\text{max}} = 5$
- 4,4,4,4,4,5,5,5,5,5

The ideal $k_{\text{min}} = 4$, $k_{\text{max}} = 6$.
Take initial $k_{\text{min}} = 3$, $k_{\text{max}} = 5$, the iteration runs as the following:

- $k_{\text{min}} = 3, k_{\text{max}} = 5$
- 4,4,4,4,4,5,5,5,5,5
- $k_{\text{min}} = 4, k_{\text{max}} = 5$
- 4,4,4,4,4,5,5,5,5,5

The ideal $k_{\text{min}} = 4$, $k_{\text{max}} = 6$.
Take initial $k_{\text{min}} = 3$, $k_{\text{max}} = 5$, the iteration runs as the following:

- $k_{\text{min}} = 3, k_{\text{max}} = 5$
- 4,4,4,4,4,5,5,5,5,5
- $k_{\text{min}} = 4, k_{\text{max}} = 5$
- 4,4,4,4,4,5,5,5,5,5
- $k_{\text{min}} = 3, k_{\text{max}} = 6$
- 4,4,4,4,4,6,6,6,6,6

The ideal $k_{\min} = 4$, $k_{\max} = 6$.
Take initial $k_{\min} = 3$, $k_{\max} = 5$, the iteration runs as the following:

- $k_{\min} = 3, k_{\max} = 5$
- 4,4,4,4,4,5,5,5,5,5
- $k_{\min} = 4, k_{\max} = 5$
- 4,4,4,4,4,5,5,5,5,5
- $k_{\min} = 3, k_{\max} = 6$
- 4,4,4,4,4,6,6,6,6,6
- $k_{\min} = 4, k_{\max} = 6$
- 4,4,4,4,4,6,6,6,6,6

The ideal $k_{\min} = 4$, $k_{\max} = 6$.
Take initial $k_{\min} = 3$, $k_{\max} = 5$, the iteration runs as the following:

- $k_{\min} = 3, k_{\max} = 5$
- 4,4,4,4,4,5,5,5,5,5
- $k_{\min} = 4, k_{\max} = 5$
- 4,4,4,4,4,5,5,5,5,5
- $k_{\min} = 3, k_{\max} = 6$
- 4,4,4,4,4,6,6,6,6,6
- $k_{\min} = 4, k_{\max} = 6$
- 4,4,4,4,4,6,6,6,6,6
- $k_{\min} = 3, k_{\max} = 7$
- 4,4,4,4,4,6,6,6,6,6

The ideal $k_{\text{min}} = 4$, $k_{\text{max}} = 6$.
Take initial $k_{\text{min}} = 3$, $k_{\text{max}} = 5$, the iteration runs as the following:

- $k_{\text{min}} = 3, k_{\text{max}} = 5$
- 4,4,4,4,4,5,5,5,5,5
- $k_{\text{min}} = 4, k_{\text{max}} = 5$
- 4,4,4,4,4,5,5,5,5,5
- $k_{\text{min}} = 3, k_{\text{max}} = 6$
- 4,4,4,4,4,6,6,6,6,6
- $k_{\text{min}} = 4, k_{\text{max}} = 6$
- 4,4,4,4,4,6,6,6,6,6
- $k_{\text{min}} = 3, k_{\text{max}} = 7$
- 4,4,4,4,4,6,6,6,6,6

Therefore, the best $k_{\text{min}} = 4$, $k_{\text{max}} = 6$.

Independence Gene Model (IGM)

Let A be an $m \times n$ mutation matrix so that $\hat{M}$ is a submatrix of A and $|\hat{M}| = k$.

IGM conditions:

1. Each gene $g \notin \hat{M}$ is mutated independently in each patient with probability $p_g \in [p_L, p_U]$. (Low rate passenger mutation)
2. $W(\hat{M}) = rm$. (Average row score r)
3. $\forall l$, any subset $M \subset \hat{M}$ of cardinality $|M| = l$ satisfies: $W(M) \leqslant \frac{l+d}{k} W(\hat{M})$, for a constant $0 \leqslant d < 1$. (No dominant submatrix)

IGM is a standard assumption for somatic single nucleotide mutations.

The genes in $\hat{M}$ are modelled to be the driver pathway.

Generation of Simulation Data

In our study, we often use the following simulation data based on IGM requirements.

Mutation matrix A is constructed as following:

- The top $l \cdot t$ columns represent t groups of driver genes, each group contains l genes.
- For each group, we set the row sums follows a distribution concentrating at 1. (e.g. $p(x) = (0.9 - 0.85|x - 1|)^{+}$)
- For other genes, we consider them as passenger genes which mutates with probability $p_m = O(1/m)$.

The Reason for Failure

Take WMSP problem as an example, if we want to select $k \in [k_{\min}, k_{\max}]$ columns such that the submatrix M obtains maximum $W(M)$.
We first select the top k columns M_0. Suppose the patients mutated in M_0 are $\Gamma(M) \subset \{1, \cdots, m\}$. Consider an outsider gene g, an insider gene g_0, such that $W(M \setminus \{g_0\}) \triangleq W(M') \geqslant \frac{k-1}{k} W(M)$.

$$
\begin{aligned}
W(M' \cup \{g\}) &= W(M') + \#\{\text{patients mutated in } g\text{, but not in } \Gamma(M')\} \\
&\qquad\qquad - \#\{\text{patients mutated in both } g \text{ and } \Gamma(M')\} \\
&\triangleq W(M') + |\Gamma(g) \setminus \Gamma(M')| - |\Gamma(g) \cup \Gamma(M')| \\
&\geqslant \frac{k-1}{k} W(M) + |\Gamma(g) \setminus \Gamma(M')| - |\Gamma(g) \cup \Gamma(M')|
\end{aligned}
$$

Denote $x_i = I\{i \in \Gamma(g)\}$. Assume $\Gamma(M') = \{1, \cdots, m_0\}$, $m_0 < m$, $m - m_0 \ll m$.

$$\begin{aligned}
\Pr(\omega_g) &= \Pr(W(M' \cup \{g\}) > W(M)) \\
&\geqslant \Pr(|\Gamma(g) \setminus \Gamma(M')| - |\Gamma(g) \cup \Gamma(M')| > \frac{W(M)}{k}) \\
&= \Pr(x_{m_0+1} + \cdots + x_m > x_1 + \cdots + x_{m_0} + r) \\
&= \Pr(Y > Z + r)(\text{where } Y \sim B(m - m_0, p), Z \sim B(m_0, p))
\end{aligned}$$

where r is the density of M: $W(M) = rm$.
Numerically, if $p = 0.01$, $m = 100$, $m_0 = 95$, $n = 1000$, $r = 0.9$, the probability $\Pr(\omega_g) > 0.02$.

$$\Pr(\cup_g \omega_g) \approx 1$$

Conclusion

With high probability, the submatrix found that maximize $W(M)$ contains passenger genes.

The accuracy of the methods could not be calculated only by its symmetric difference (or other similar criterion) with expected result $(1, \cdots, 10$ in our case).

We could actually use Brute Force strategy to get the optimal solution for smaller datasets, and use these results as a ruler.

Motivation

We would like to select the l important genes from all genes $A = \{1, 2, \cdots, n\}$, when the true pathway gene set is $\hat{M} = \{1, 2, \cdots, k\}$. If $l \neq k$, the result would be unstable. Therefore, we would like to build a criteria to determine the l without prior knowledge on k.

Model

The idea is to abstract the result from the IGM (Independence Gene Model). We set a probability model for the result gene set.
Denote $\vec{x} = (x_1, \cdots, x_n)$ the indicator vector for $M \subset A$, $|M| = l$.
Assume the distribution $\triangleq M(n, k, l)$:

$$f(\vec{x}) = p^{x_{k+1} + \cdots + x_n} / A(n, k, l)$$

where $p = O(1/n)$.

$$A(n, k, l) = \sum_u C_k^u C_{n-k}^{l-u} p^{l-u}$$

Problem

Run pairs of results $I_i, J_i \overset{i.i.d.}{\sim} M(n,k,l), i = 1, 2, \cdots, t$.
We would like to estimate EOS (Expectation of Overlapping Size) defined as:

$$C(l) = \frac{\sum_{i=1}^{t} |I_i \cap J_i|}{t}$$

By LLN (Law of Large Numbers), $C(l) \approx \mathbb{E}C(l) = \mathbb{E}|I_1 \cap J_1|$.
After calculations, the problem reduces to

$$\begin{aligned} C(l) &\approx \mathbb{E}|I_1 \cap J_1| \\ &= k(\frac{A(n-1,k-1,l-1)}{A(n,k,l)})^2 + (n-k)(p\frac{A(n-1,k,l-1)}{A(n,k,l)})^2 \end{aligned}$$

Simulation Study

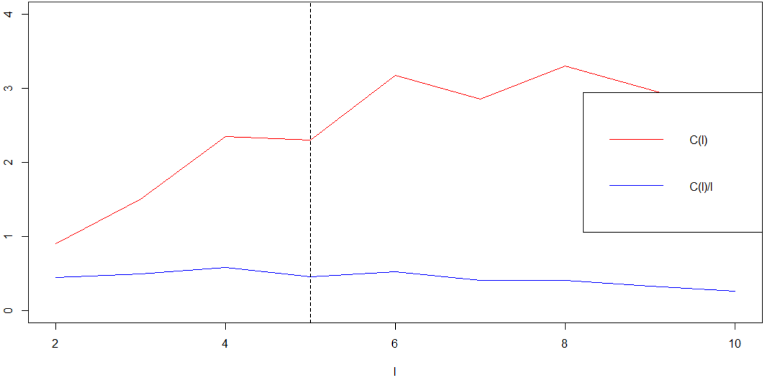


Figure: The relation of l and average of $C(l)$ for 10 tests, simulated data of 50 patients, 200 genes, 5 driver mutation genes

With the settings in the simulation data, the mutation rate in the model p should be 0.1.

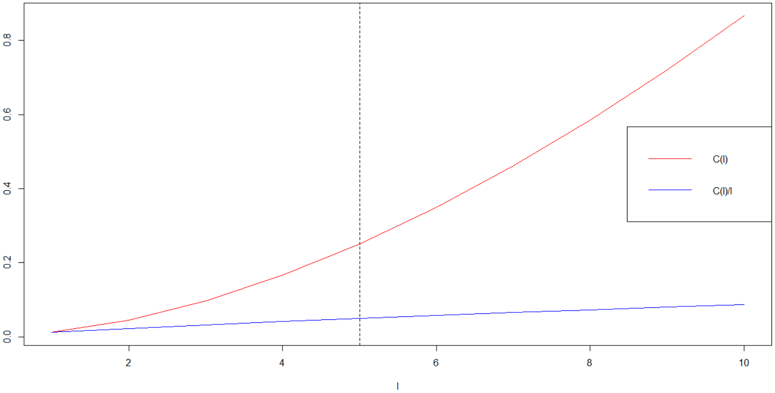


Figure: $\mathbb{E}C(l)$ with respect to l, where $p = 0.1$, $n = 200$, $k = 5$

Further Study Directions

- Find better crossproduct method for MVGA
- Use both simulation data and biological data to test the new method
- Selection of t in simultaneous identification
- Mathematically prove the reliability of parameter selection of k_{min} and k_{max}
- Illustration of the reason why candidate scoring scheme is preferable