

# Supplementary Material for TFBS Identification Based on Genetic Algorithm with Combined Representations and Adaptive Post-processing

Tak-Ming Chan<sup>1\*</sup>, Kwong-Sak Leung<sup>1</sup>, Kin-Hong Lee<sup>1</sup>

<sup>1</sup>Department of Computer Science & Engineering, The Chinese University of Hong Kong, Shatin, N. T., Hong Kong

## 1 METHODS

### 1.1 Information Content in Post-processing

The information content adopted in GALF

$$IC = \sum_{j=1}^w IC(j) = \sum_{j=1}^w \sum_{f_b(j) > 0} f_b(j) \log \frac{f_b(j)}{p_b} \quad (1)$$

eliminates the difficulty to compute zero logarithms and works well for the assumption of  $k_i = 1$  per sequence. However, when it comes to the post-processing of adding and removing to handle  $0 \leq k_i \leq l - w + 1$  based on GALF's best individual, this fitness has to be relaxed for additional weaker instances and modified to include pseudo-counts for removing false positives appropriately.

The concern is that after the procedure GALF, the best individual  $I_{Best} = \{m_1, \dots, m_N\}$  consists of the most conserved instance from each sequence, i.e., each instance is the best in terms of fitness among all the instances in the same sequence. The instances in  $I_{Best}$  may be highly conserved or, if possible, even completely conserved. If adding is only based on improving  $IC_{I_{Best}}$  from Equation 1, it is unlikely to take in any weaker instances. This can be illustrated by a simplified case of calculating the positional  $IC(j)$  of column  $j$  with uniform background frequencies ( $p_b = 0.25$  for all  $b$ ). We consider a nucleotide  $b_\alpha$ , whose frequency  $f_{b_\alpha}$  will be changed by adding or removing. The ratio of any other nucleotide  $b_q$  with non-zero frequency is defined as  $r_{b_q}$ ,  $q \neq \alpha$ , where  $\sum_{q \neq \alpha} r_{b_q} = 1$ , and the number of these nucleotides is denoted as  $|\{q\}|$  and  $|\{q\}| > 0$  ( $|\{q\}| = 0$  is a trivial case where  $f_{b_\alpha} = 1$  as the only frequency). We have  $\sum_b f_b = 1$  and  $f_{b_q} = r_{b_q} * (1 - f_{b_\alpha})$  (for clarity of concise notations,  $j$  is ignored). Then  $IC(j)$  can be expressed as

$IC(j)$  ( $j$  will be ignored for clarity)

$$\begin{aligned} &= \sum_{b=b_\alpha, b_q} f_b \log \frac{f_b}{p_b} \\ &= \sum_{b=b_\alpha, b_q} f_b \log f_b + Const \\ &= f_{b_\alpha} \log f_{b_\alpha} + (1 - f_{b_\alpha}) (\log(1 - f_{b_\alpha}) + \sum_q r_{b_q} \log r_{b_q}) \\ &\quad + Const \end{aligned}$$

where  $Const$  is a constant. By differentiation with respect to  $f_{b_\alpha}$ , we can show that  $IC(j)$  is monotonically increasing as

$$f_{b_\alpha} \geq \frac{1}{1 + 2^{-\sum_q r_{b_q} \log r_{b_q}}} \quad (2)$$

and vice versa. Note that  $-\sum_q r_{b_q} \log r_{b_q}$  is exactly the entropy of the non-zero frequencies. So  $2^{-\sum_q r_{b_q} \log r_{b_q}}$  is constrained within the range  $(1, |\{q\}|]$ , and

$$\frac{1}{1 + |\{q\}|} \leq \frac{1}{1 + 2^{-\sum_q r_{b_q} \log r_{b_q}}} < \frac{1}{2} \quad (3)$$

In the motif discovery problems, certain degree of conservation is preserved, so we assume the dominant nucleotides in the motifs should be generated with a probability more than 50%. Now suppose  $b_\alpha$  is the dominant nucleotide, and thus the frequency

$$f_{b_\alpha} \geq 0.5 > \frac{1}{1 + 2^{-\sum_q r_{b_q} \log r_{b_q}}} \quad (4)$$

is in the monotonically increasing region. By adding a new instance with  $b_\alpha$  at column  $j$  (equivalent to increasing  $f_{b_\alpha}$ ), the positional  $IC(j)$  will increase. On the other hand, suppose  $b_\alpha$  is a non-dominant nucleotide, which are assumed to be generated randomly, and thus (note that  $|\{q\}| > 0$  and the frequency of the dominant nucleotide is  $\geq 0.5$ )

$$f_{b_\alpha} \leq \frac{1 - 0.5}{|\{q\}|} \leq \frac{1}{1 + |\{q\}|} \leq \frac{1}{1 + 2^{-\sum_q r_{b_q} \log r_{b_q}}} \quad (5)$$

i.e.,  $f_{b_\alpha}$  is in the monotonically decreasing region. Hence, adding an instance with  $b_\alpha$  at column  $j$  will decrease the positional  $IC(j)$ .

\*to whom correspondence should be addressed

As a result, only instances with the largest number of dominant nucleotides are likely to be added based on increasing the  $IC_{I_{Best}}$  of the best individual output by GALF. In other words, weaker instances can hardly be added. So the threshold of fitness  $IC_{I_{Best}}$  has to be slightly relaxed for adding weaker instances. Pseudo-counts also have some effect on relaxing the conservation. Moreover, since they are used in the removing stage, they are also included in the adding stage for the consistency in both stages.

To remove false positives correctly, pseudo-counts are in fact necessary. Consider again the calculation of positional  $IC(j)$  in Equation 1. If a dominant nucleotide  $b_\alpha$  has total conservation with  $f_{b_\alpha} = 1$ ,  $IC(j)$  will not decrease (always 1) when removing any occurrence at column  $j$  with  $b_\alpha$ . Similar to the case for adding, if  $b_\alpha$  is a non-dominant nucleotide, removing  $b_\alpha$  (equivalent to decreasing  $f_{b_\alpha}$ ) will increase  $IC(j)$  since  $f_{b_\alpha}$  is in the monotonically decreasing region based on Equation 5. As a result, the removing procedure will continue to be wrong until the remaining few instances containing only the dominant nucleotides. Thus the  $IC$  has to include pseudo-counts and the threshold has to be adjusted accordingly as well for the removing in post-processing.

## 2 RESULTS

### 2.1 Design of the Datasets for Parameter Setting

To test the parameter settings, two types of synthetic datasets are generated randomly with equal probability for each nucleotide: type A contains 25 200bp sequences while type B contains 20 500bp sequences. We adopt the concepts of short (8bp) and long (16bp) motif widths, high (91% probability for the dominant nucleotides) and low (70%) conservation from (Wei and Jensen, 2006). The different synthetic datasets include (denoted as: dataset{Motif width, Conservation, type}): data1{short, high, A}, data2{short, high, B}, data3{long, low, A}, data4{long, low, B}. The purpose of embedding low conservation patterns is to set “traps” to test whether GALF is able to find the most conserved patterns with scores not lower than the embedding ones. On the other hand, if high conservation patterns are embedded, they serve as the global optima. So the target is to find as high an  $IC$  in Equation 1 as possible in all tests. For short motifs, the low conservation “traps” are not necessary because there are already many short subsequences with higher conservation. For long motifs with high conservation, GALF always succeeds to find global optima with any setting, so we neglect them in the analysis.

Note that we generate the synthetic datasets for parameter setting differently from the datasets for simulation experiment in order to avoid over-training that favors our approach.

### 2.2 The Full Table of Comparisons with Published Results

Table 1 shows the results when GALF-P is compared with the reported performance from (Wei and Jensen, 2006) of GAME, MEME, Bioprosector, BioOptimizers based on MEME and Bioprosector (denoted by BioOpt. M. and BioOpt. B. respectively in the table) on the 8 real data sets. The data in parentheses are precisions of GALF before postprocessing for reference. Since GALF assumes  $k_i = 1$ , the precisions are exactly the recalls and  $F$ -scores when the datasets (labelled with “\*”s) satisfy the assumption. In each set, the best  $F$ -scores are bolded. We can see GALF achieves superior average precision (0.87) to other methods (0.64 – 0.78),

**Table 1.** Comparisons on the 8 datasets. Datasets satisfying one instance per sequence are labelled with “\*”s.

Data	Approaches	Precision	Recall	$F$ -score
CREB	GALF-P	(13/17) 16/23	16/19	<b>0.76</b>
	GAME	15/22	15/19	0.73
	BioOpt. M.	10/15	10/19	0.59
	BioOpt. B.	12/17	12/19	0.67
	MEME	10/15	10/19	0.59
	BioProspector	13/20	13/19	0.67
CRP	GALF-P	(17/18) 17/17	17/23	<b>0.85</b>
	GAME	16/17	16/23	0.80
	BioOpt. M.	12/13	12/23	0.67
	BioOpt. B.	12/13	12/23	0.67
	MEME	12/13	12/23	0.67
	BioProspector	16/18	16/23	0.78
ERE *	GALF-P	(19/25) 19/23	19/25	<b>0.79</b>
	GAME	19/26	19/25	0.75
	BioOpt. M.	17/22	17/25	0.72
	BioOpt. B.	18/23	18/25	0.75
	MEME	15/17	15/25	0.71
	BioProspector	14/16	14/25	0.68
E2F	GALF-P	(19/25) 23/30	23/27	0.81
	GAME	23/24	23/27	<b>0.90</b>
	BioOpt. M.	20/27	20/27	0.74
	BioOpt.B.	19/27	19/27	0.70
	MEME	19/23	19/27	0.76
	BioPros.	11/21	11/27	0.46
MEF2 *	GALF-P	(17/17) 17/18	17/17	<b>0.97</b>
	GAME	15/17	15/17	0.88
	BioOpt. M.	14/15	14/17	0.88
	BioOpt.B.	11/19	11/17	0.61
	MEME	14/15	14/17	0.88
	BioPros.	12/17	12/17	0.71
MYOD	GALF-P	(15/17) 21/37	21/21	<b>0.72</b>
	GAME	10/21	10/21	0.48
	BioOpt. M.	0/10	0/21	0.00
	BioOpt.B.	0/11	0/21	0.00
	MEME	0/8	0/21	0.00
	BioPros.	0/18	0/21	0.00
SRF	GALF-P	(18/20) 33/43	33/36	<b>0.84</b>
	GAME	33/47	33/36	0.80
	BioOpt. M.	32/51	32/36	0.74
	BioOpt.B.	32/50	32/36	0.74
	MEME	28/48	28/36	0.67
	BioPros.	25/35	25/36	0.70
TBP *	GALF-P	(86/95) 83/92	83/95	<b>0.89</b>
	GAME	78/91	78/95	0.84
	BioOpt. M.	35/79	35/95	0.40
	BioOpt.B.	65/78	65/95	0.75
	MEME	26/50	26/95	0.36
	BioPros.	58/69	58/95	0.71
Average	GALF-P	<b>(0.87) 0.81</b>	<b>0.87</b>	<b>0.83</b>
	GAME	0.78	0.77	0.77
	BioOpt. M.	0.64	0.57	0.59
	BioOpt. B.	0.65	0.60	0.61
	MEME	0.67	0.53	0.58
	BioPros.	0.66	0.51	0.56

demonstrating its capacity to identify the optimal TFBSs correctly with the assumption of a single instance per sequence.

### 2.3 Details of Complexity Comparison

Suppose there are  $N$  sequences, each with the same length  $l$ . Motif width is  $w$ . Population size is  $P$  which is the same for GALF and GAME. We consider the cost in each generation first. The complexity consists of the following parts: fitness evaluation ( $Eval$ )

of the population, genetic operations (*GenOP*) including single-point mutation and crossover (for GALF, shift operator is considered though it is rarely triggered), and selection (*Select*) including parent and survivor selections (i.e. replacement). For GALF, additional overhead of the local filtering (*LFOP*) has to be considered. Here we will discuss each part in detail.

*Eval*: For both GALF and GAME, the complexity is both  $O(P * N * w)$  because  $O(N * w)$  operations are needed to extract and compute the PWM. The calculation of fitness after PWM is constructed is proportionate to  $w$  only, so it can be ignored. Specifically in GAME, except the first generation, in each generation  $O(P)$  new individuals are created for selection and to be evaluated, so the complexity is  $O(P * N * w)$ . For GALF, because the mutation rate is as high as 0.9, we just assume for each generation, all  $P$  individuals need to be evaluated, resulting in  $O(P * N * w)$  in complexity.

*GenOP*: In both methods, single-point mutation and crossover operators take  $O(1)$  and  $O(N)$  respectively. The complexity is  $O(P * N)$  for GAME ( $O(P * (0.001 * O(1) + 1.0 * O(N)))$ ). Though different mutation and crossover rates are employed, similar complexity can be obtained as  $O(P * (0.9 * O(1) + 0.5 * O(N)))$  in GALF. Shift operator in GALF may introduce extra computation. However, it is only triggered on the best individual when it stagnates for 10 generations, so the complexity never exceeds  $O(0.1 * w * N)$ . So the overall complexity in GALF is  $O(P * N + w * N)$ .

*Select*: In GAME, random pairing is applied twice, one for offspring generation  $O(P)$  and one for binary tournament selection  $O(2 * P)$  (because after reproduction the population size becomes  $2 * P$ ). As a result the overall complexity for *Select* in GAME is still  $O(P)$ . For GALF, random pairing is applied once for reproduction,  $O(P + N)$  is required to pick up and store the best individual, and  $O(P * w)$  is required in replacement, where  $O(w)$  is for calculating the Hamming distance between the pairs of parent and offspring. So the overall complexity for *Select* in GALF  $O(P * w + N)$ .

*LFOP*: The overhead for GALF can be divided into 3 parts for each individual. The first part is the calculation of similarity scores, which requires  $O(N * w)$ . The second part is for sorting the instances, which takes  $O(N * \log N)$  on average with a quick sort (for a bubble sort  $O(N * N)$ ). The third part is for scanning the sequences. The average number of sequences scanned drops significantly according to previous work (Chan *et al.*, 2007). We assume the average ratio of sequences scanned is  $1/k$ . So in total the complexity is  $O(P * 0.1 * (N * w + N * \log N + N/k * l))$ , where 0.1 indicates local filtering is triggered once every 10 generations.

To summarize, the complexity for GAME is

$$\begin{aligned} C_{GAME} &= G_1 * (C(Eval) + C(GenOP) + C(Select)) \\ &= O(G_1 * (O(P * N * w) + O(P * N) + O(P))) \\ &= O(G_1 * P * N * w) \end{aligned}$$

and the complexity for GALF is

$$\begin{aligned} C_{GALF} &= G_2 * (C(Eval) + C(GenOP) + C(Select) + C(LFOP)) \\ &= O(G_2 * (O(P * N * w) \\ &\quad + O(P * N + w * N) + O(P * w + N) \\ &\quad + O(P * 0.1 * (N * w + N * \log N + N/k * l)))) \\ &= O(G_2 * P * N * (w + 0.1 * (\log N + l/k))) \end{aligned}$$

where 0.1 indicates local filtering is triggered once every 10 generations,  $1/k$  is the averaged percentage of sequences scanned in local filtering, and  $G_1$  and  $G_2$  denote the different maximum generations required in GAME and GALF respectively.

## 2.4 Computation Time of GALF-P and GAME

In the previous experiments, GALF-P and GAME are both executed on the same Pentium D 3.00 GHz machine with 1GB memory, running Windows XP Professional platform. The average real computation time at each run (in one run the GA in GAME and GALF are run 20 times) for each dataset is shown in Table 2.

**Table 2.** Average computation time on the 8 datasets between GAME and GALF-P.

	GAME	GALF-P	Speedup
CREB	133.00	42.75	3.11
CRP	380.05	98.20	3.87
ERE	334.20	83.20	4.02
E2F	288.65	86.95	3.32
MEF2	112.05	34.40	3.26
MYOD	91.05	26.25	3.47
SRF	224.05	49.10	4.56
TBP	765.80	74.40	10.29
<b>Average</b>	291.11	61.91	4.49

## REFERENCES

- Chan, T.-M., Leung, K.-S., and Lee, K.-H. (2007). TFBS identification by position- and consensus-led genetic algorithm with local filtering. In *GECCO '07: Proceedings of the 2007 conference on Genetic and evolutionary computation*, pages 377–384.
- Wei, Z. and Jensen, S. T. (2006). GAME: detecting cis-regulatory elements using a genetic algorithm. *Bioinformatics*, **22**(13), 1577–1584.