Biclustering of the Gene Expression Data by Coevolution Cuckoo Search

Lu Yin[1,2], Yongguo Liu[1*]

[1]*School of Computer Science and Engineering*
University of Electronic Science and Technology of China
ChenDu, SiChuan, 611731, China
E-mails: yinlu_78@163.com, liuyg@uestc.edu.cn

[2]*School of Computer Engineering*
Huaiyin Institute of Technology
HuaiAn, JiangSu, 223003, China

[*]*Corresponding author*

Received: January 17, 2015 **Accepted: June 18, 2015**

Published: June 30, 2015

Abstract: *Biclustering has a potential to discover the local expression patterns analyzing the gene expression data which provide clues about biological processes. However, since it is proven that the biclustering problem is NP-hard, it is necessary to seek more effective algorithm. Cuckoo Search (CS) models the brood parasitism behavior of cuckoo to solve the optimization problem and outperforms the other existing algorithms. In this paper, we introduce a new algorithm for biclustering gene expression data, called cuckoo search biclustering (CSB), which adopts CS to solve the biclustering problem. For further improving the performance of CSB, three modifications to CSB are made with the aim of increasing the convergence rate and the coevolution cuckoo search biclustering (COCSB) is designed. CSB and COCSB are tested on the six gene expression data and their results are compared to those of CC, FLOC, ISA, SEBI, BIC-aiNet, PSOB, SAB and SSB. The comparison shows that CSB and COCSB have achieved great success in biological significance and time performance.*

Keywords: *Biclustering, Gene expression data, Coevolution cuckoo search biclustering.*

Introduction

DNA microarray technology makes simultaneous monitoring of the gene expression level of thousands of genes under different samples which obtain the gene expression data after several preprocessing steps [5]. The gene expression data can be represented as a matrix where each gene corresponds to its row, each sample to its column and each element to the expression level of a gene under a sample. Different methods have been applied for discovering the biological knowledge in gene expression data [1, 8, 15, 19, 21, 33]. Among these methods, biclustering [8] has a potential to discover the local expression patterns of gene expression data, which makes biclustering an important tool in analyzing the gene expression data.

Formally, a gene expression data can be viewed as a $n \times m$ matrix $\mathbf{A}(X,Y)$, in which $X = \{x_1, x_2, \cdots, x_n\}$ denotes the set of the genes, $Y = \{y_1, y_2, \cdots, y_m\}$ denotes the set of the samples, and a_{ij} denotes the expression level of the *i*-th gene under the *j*-th sample. Let that I is the gene subset of X and $I \subseteq X$, $|I| = k$, $k \le n$, J is the sample subset of Y and $J \subseteq Y$, $|J| = l$, $l \le m$. A bicluster is defined as a submatrix $\mathbf{B}(I,J)$ of data $\mathbf{A}(X,Y)$ in which

the subset I shows similar gene expression patterns under the subset J which also shows similar sample expression profiles across the gene subset I, and is denoted as $k \times l$ submatrix $\mathbf{B}(I,J)$. So the biclustering problem can be formulated as follows: given a gene expression data $\mathbf{A}(X,Y)$, construct a bicluster $\mathbf{B}_{\mathbf{opt}}(I,J)$ such that $f(\mathbf{B}_{\mathbf{opt}}) = \max_{\mathbf{B} \in \mathbf{A}} f(\mathbf{B})$ where $f(\mathbf{B})$ is an objective function for evaluating the coherence of a candidate bicluster $\mathbf{B}(I,J)$. It is proved that the biclustering problem is NP-hard [8].

Consequently, many existing biclustering algorithms are based on reducing the searching range or improving the searching efficiency. Since Cheng and Church [8] first proposed CC biclustering algorithm, many algorithms have been proposed and the surveys can be found in [7, 26-27]. Ayadi et al. [3] classified the biclustering algorithms into systematic search algorithms and stochastic search algorithms. Among the representative systematic search algorithms are ISA [21], CC [8], BiMine [3], OPSM [4], Motifs [28], Direct clustering [20], FLOC [36], SAMBA [32], OP-Cluster [24], Bimax [31] and Paid [22], etc. The stochastic searching algorithms have SEBI [13], SSB [29], CMOPSOB [25], BIC-aiNet [11], and SAB [6].

The stochastic search algorithms are known for their strength in avoiding locally optimal solutions especially using some local search strategies, so they can quickly converge toward the global optimum which account for many significant biclusters reported in their work. However, no single existing algorithm is completely successful in solving biclustering problem, so it is useful to seek more effective algorithms for obtaining a better bicluster. Cuckoo Search (CS), a recent metaheuristic algorithm first reported [37], models the brood parasitism behavior of cuckoo to solve the optimization problem and shows that CS is very promising and outperforms the other existing algorithms. In addition, CS only needs one parameter, the fraction of nests to abandon, to be adjusted which makes it easily implemented. In this paper, we introduce a new stochastic search algorithm for biclustering of gene expression data, called cuckoo search biclustering (CSB). Although CSB achieves good performance, it cannot find the global optimal bicluster. For further improving the performance of CSB, some modifications to CSB are made with the aim of increasing the convergence rate but without losing the attractive features of CSB. The first modification is to introduce the coevolution. The second is that the populations of host and cuckoo dynamically change. The third is that unlike CSB the optimizing objective is the egg instead of the nest. As a result, we designed the coevolution cuckoo search biclustering (COCSB) algorithm. To assess the performance of the proposed algorithms, the results of CSB and COCSB on six gene expression data are reported and compared with those of CC, FLOC, ISA, SEBI, BIC-aiNet, PSOB, SAB and SSB. In addition, we perform a biological validation.

The article is organized as follows: Section II describes CSB. In Section III the procedure of COCSB is introduced. In Section IV after briefly introducing the platform and the expression data used by our experiment, the experiment results are reported and discussed in terms of the evaluating criterion and biological significance. Finally, the conclusion and future work is provided in Section V.

Description of CSB

Cuckoo Search (CS) is inspired by the brood parasitism behavior of cuckoos. In detail, cuckoo does not hatch their eggs, but lay their eggs in the nests of other host birds. If the host does not discover the cuckoo eggs then it hatches the cuckoo eggs and raises the child cuckoo,

otherwise it either destroys the egg or abandons the nest. So for cuckoo the search procedure of the host nest fitting for their eggs is the optimization procedure. After making some assumptions, CS matches the nests with the candidate solution, the quality of the nest with the fitness function of the solution, and gets the better nest by searching nest operation and abandoning nest operation iteratively [37]. Based on CS, we propose the Cuckoo Search Biclustering (CSB) for identifying the bicluster in the gene expression data. The flowchart of CSB is shown in Fig. 1.

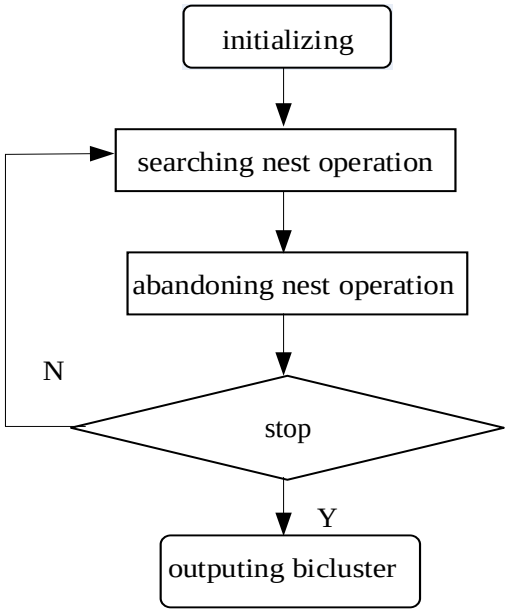


Fig. 1 Flowchart of CSB

Encoding of a bicluster

The represents of a bicluster $\mathbf{B}(I,J)$ are encoded as binary string of length $n+m$ which is represented as a nest of $x_p=\{g_1,\cdots,g_i,\cdots,g_n,s_1,\cdots,s_j,\cdots,s_m\}$, $p=1,\cdots,N$, and N denotes the population size in which n and m are the number of genes and samples of the gene expression matrix $\mathbf{A}(X,Y)$. If the *i*-th gene or *j*-th sample in the $\mathbf{A}(X,Y)$ is chosen by the $\mathbf{B}(I,J)$, $g_i=1$ or $s_j=1$, otherwise, $g_i=0$ or $s_j=0$, where $1\le i\le n$ and $1\le j\le m$.

The fitness function

In this work the index for evaluating the coherence of a candidate bicluster is MSR which has largely been adopted in [4, 6, 8, 11, 13, 25, 29]. Given that the minimal gene and the sample number of a bicluster are 10 as in SAB [6], so the fitness function mainly focuses on the coherence of a bicluster. The fitness function $f(x_p)$ of a nest x_p is as follows:

$$f(x_p)=\sigma / MSR(B)=\sigma / \left(\frac{1}{k\times l}\sum_{i\in I,j\in J}(a_{ij}-a_{iJ}-a_{Ij}+a_{IJ})\right) \tag{1}$$

where a_{iJ}, a_{Ij} and a_{IJ} are the mean of the *i*-th row, the mean of the *j*-th col and the whole mean in $\mathbf{B}(I,J)$ and σ is the threshold of MSR. So the final objective of CSB is to maximize the fitness.

Initialization

The initial population $\mathbf{P}=\{x_1, x_2, \cdots, x_N\}$ is composed of N initial nest $\boldsymbol{x_i}, 1\le i\le N$ which corresponds to the i th bicluster $\mathbf{B}_i(I,J)$. Assume that the number of the gene and the sample of the bicluster $\mathbf{B}_i(I,J), 1\le i\le N$ are $r_{\mathbf{B}_i}$ and $c_{\mathbf{B}_i}$, then the pseudo code of the initializing is as follows:

Step 1: Let $num=1$.

Step 2: while ($num\le N$).

Step 2.1: generating the row vector G of length n in which values are 0, and randomly flipping $r_{\mathbf{B}_i}$ bits from 0 to 1. Similarly, generating the row vector S of length m which values are 0, and randomly flipping $c_{\mathbf{B}_i}$ bit from 0 to 1.

Step 2.2: merging G and S into a row vector represented by the initial nest $x_i(I,J)$ for $1\le i\le N$.

Step 2.3: computing $f(x_i), 1\le i\le n$ according to Eq. (1).

Step 2.4: $num=num+1$.

Searching nest operation

Searching nest means that each nest $x_i(I,J), 1\le i\le N$ in the population $\mathbf{P}$ performs local searching. Let *mSize* is the step length of local searching of a nest, *maxNumber* is the maximal times of local searching of a nest, and δ is the predefined threshold of a nest, the detailed step is as follows:

Step 1: $i=1$, $maxNumber=10$.

Step 2: while ($i\le N$ and $f(x_i)\ge\delta$).

Step 2.1: $k=1$, $mSize=2$.

Step 2.2: Randomly flipping the *mSize* bits from the binary string of x_i to get new $x_i^{'}$, and computing $f(x_i^{'})$.

Step 2.3: If $f(x_i^{'})>f(x_i)$, then updating x_i with $x_i^{'}$ and go to Step 2.5.

Step 2.4: $k=k+1$, if $k>maxNumber$, then $mSize=8$, and randomly flipping *mSize* bits from the binary string of x_i generates $x_i^{'}$ to replace x_i.

Step 2.5: $i=i+1$.

Abandoning nest operation

The aim of abandoning nest is to maintain the diversity of the nest population which helps to escape the local optimal solution. In fact, abandoning nest operating is to update the worst nests with new generated nests at a given proportion $p\in[0,1]$, its detailed step is as follows:

Step 1: Sorting all nests in $\mathbf{P}$ based on the fitness function in ascending order.

Step 2: Computing the number of nests to be updated $replace\_num=N\times p$.

Step 3: For each the first *repalce_num* nests in $\mathbf{P}$, performing the above searching nest operation for obtaining the new nests to replace the above old nests.

Stop condition

The stop condition of CSB is that the fitness function $f(x^*)$ of the best known solution x^* in the population $\mathbf{P}$ is smaller than the predefined threshold.

Description of COCSB

In the brood parasitism the cuckoo egg must mimic the host egg for assuring its egg hatched by cuckoo. The hosts also evolve many defenses against the brood parasitism in nature [10]. So the completion of host and cuckoo leads to a co-evolutionary arms race in which each party evolves in response to the other as in Fig. 2.

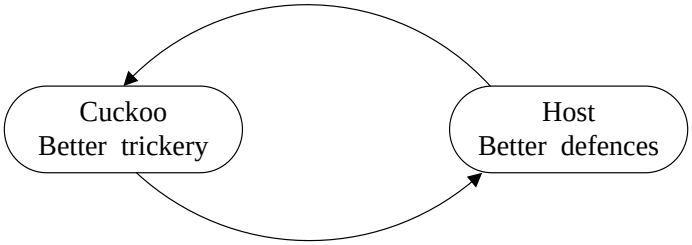


Fig. 2 The cycle of co-evolution between cuckoo and host

For simulating the above phenomena and increasing the convergence rate, three modifications of CSB are made. The first modification is to introduce the coevolution. The brood parasitism behavior simulated by CSB doesn't show the coevolution which usually exists in nature. The second modification is that the populations of host and cuckoo dynamically change. The above competitive relation makes the population size of host and cuckoo dynamically changes and eventually arrives at equilibrium. The third is that the optimizing objective is the egg rather than the nest in CSB. Assuming that the host only possesses one defense against the cuckoo parasitism-reducing intra-clutch variation and increasing inter-clutch variation, for host and cuckoo the procedure of the coevolution in egg is the optimization procedure. Matching the egg with the candidate solution and the quality of the egg with the fitness of the solution, we propose the Coevolution Cuckoo Search Biclustering (COCSB). In COCSB, the evolution of the host egg carries out the exploitation process in the search space while that of the cuckoo egg controls the exploration process. The flowchart of COCSB is Fig. 3.

Encoding of the bicluster and designing the fitness function

In COCSB, a bicluster $\mathbf{B}(I, J)$ is also encoded as binary string of length $n+m$ as CSB. This binary string is represented as an egg $x = g_1, g_2, \cdots, g_n, s_1, s_2, \cdots, s_m$. The fitness function $f(x)$ of an egg x is defined as Eq. (1) as CSB.

Initializing the egg population of cuckoo and host

In COCSB, there are two populations, host and cuckoo egg, which correspond to the bicluster. Assuming that the initial number of host and cuckoo are $N_{cuckooEgg}$ and $N_{hostEgg}$ and that each host lay $eggnum = 5$ eggs considering that each host usually lay 4~6 eggs in nature, so the population of host eggs

$$\mathbf{P}_{eggHost} = \left\{ x_{1,1}, \cdots, x_{i,j}, \cdots, x_{N_{host,4}}, x_{N_{host,5}} \right\},$$

where $x_{i,j}$ $(1 \le i \le N_{hostEgg}, 1 \le j \le 5)$ represent the *j*-th egg of the *i*-th host. Moreover, since in nature each cuckoo lays one egg in each nest of host, so the number of cuckoo eggs is the same as the size of N_{host} and the population of cuckoo eggs

$$\mathbf{P}_{eggCuckoo} = \left\{ y_1, \cdots, y_k, \cdots, y_{N_{host}} \right\},$$

where $y_k, 1 \le k \le N_{host}$ represent the *k*-th cuckoo egg. In COCSB initializing two above egg populations is the same as CSB.

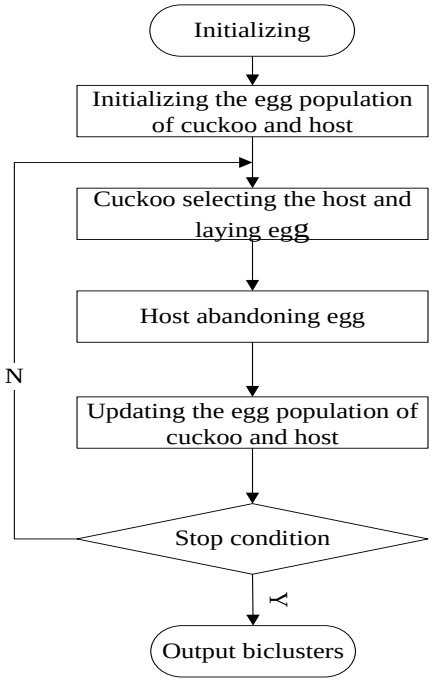


Fig. 3 Flowchart of COCSB

Cuckoo selecting the host and laying egg

Since the cuckoo evolved with the egg mimicry for improving the survival of its offspring, each cuckoo selects the host in which the fitness of eggs is close to the fitness of its own egg. Note that each cuckoo lays only one cuckoo egg in one host nest to avoid the intraspecific competition. The pseudo code of the cuckoo selecting the host is as follows:

Step 1: For each egg $x_{i,j}$ computing the fitness $f\left(x_{i,j}\right)$

Step 2: For each host computing and saving the mean fitness $f\left(x_i\right) = \frac{1}{eggnum}\sum_{j=1}^{5} f\left(x_{i,j}\right)$

Step 3: For each cuckoo egg y_k in $\mathbf{P}_{cuckoo}$

Step 3.1: Computing the fitness $f\left(y_k\right)$ of cuckoo egg y_k.

Step 3.2: Getting the host $x_{minIndex}$ of which the mean fitness is closest to the fitness $f\left(y_k\right)$ of cuckoo egg y_k.

Step 3.4: For the host $x_{minIndex}$ replacing the egg of which the fitness is minimal with the cuckoo egg y_k.

Step 4: Let $f(x_{minIndex}) = inf$.

Host abandoning egg

For simulating the host's ability against the cuckoo' nest parasitism, for each host the difference between the mean fitness of the host eggs before cuckoo laying egg and that of the host and cuckoo eggs after cuckoo laying egg is computed. If the percentage of the above difference is larger than the threshold τ, then the cuckoo egg is abandoned, otherwise no egg is abandoned. Let n_{succ} and p_{succ} denotes the number and the percentage of abandoned egg, its pseudo code is as follows:

Step 1: Computing the population size $N'_{hostEgg}$ and $N'_{cukcooEgg}$ of next cuckoo and host using Step 2 and Step 3.

Step 2: If $N'_{hostEgg} < N_{hostEgg}$ then performing accelerative evolving $\mathbf{P}_{eggHost}$, otherwise performing constantly evolving $\mathbf{P}_{eggHost}$.

Step 3: If $N'_{cukcooEgg} < N_{cuckooEgg}$ then performing accelerative evolving $\mathbf{P}_{eggCuckoo}$, otherwise performing constantly evolving $\mathbf{P}_{eggCuckoo}$.

Stop condition

As CSB, the stop condition of the COCSSB is that the fitness function $f(x^*)$ of the best solution x^* in the $\mathbf{P}_{eggHost}$ or $\mathbf{P}_{eggCuckoo}$ is smaller than the predefined threshold.

Experiments

CSB and COCSB algorithms are implemented by Matlab 2012b and are run on a PC which uses Intel Core i3-2120 with 3.29 GHz, 8.0 Gb RAM and 64 bit Windows OS. In addition, CC, ISA, FLOC, SEBI, BIC-aiNet, PSOB, SAB and SSB are implemented in the above platform. All chosen algorithms get 100 biclusters for each data.

Identifying of CSB and COCSB parameter

For identifying the proportion of nests to abandon p in CSB, the index of the 100 biclusters obtained on the yeast cell cycle data is reported in Table 1when specifying p with different value and the size of a population with 100. As shown Table 1, in the different p MSR of the biclusters is less than the thresholds 300, but the time is best when $p = 0.25$. So p is designed as 0.25 in the following experiments of other data.

In COCSB, $N_{hostEgg}$, $N_{cuckooEgg}$ and $eggnum$ are is easily obtained since its value has little impact, so $N_{hostEgg} = 100$, $N_{cuckooEgg} = 20$ and $eggnum = 5$. Considering that in nature to a certain degree the host should accept the cuckoo parasitism, $\tau = 10\%$ [10]. As for p_{death}, because any population almost maintains a more stable state after a long term evolution, this is, $N'_{hostEgg} \approx N_{hostEgg}$ and $N'_{cuckooEgg} \approx N_{cuckooEgg}$. The above condition is met when $p_{death} = 60.00\%$ and the trend of the size of next population $N'_{hostEgg}$ and $N'_{cukcooEgg}$ is shown in Fig. 4.

Table 1. The mean MSR and time of CSB under the different *p*

p	**0.10**	**0.15**	**0.20**	**0.25**	**0.30**	**0.35**	**0.40**
MSR	299.87	299.86	299.84	299.84	299.83	299.82	299.86
Time	6.67	6.35	6.41	6.32	6.47	6.55	6.41

Gene expression data and its threshold

The aim of the experiment is to assess the potential of CSB and COCSB as the biclustering algorithm, so the data used in this work are as extensive as possible in the scale of the gene expression data. So the following six data are chosen and its more detailed information are shown in Table 2.

In six data the first three are widely used in biclustering problem and are preprocessed in [8, 31]. However, the other three data are downloaded in GEO [14] which accession number is GSE2403, GSE2034 and GSE952 respectively. Its raw data are preprocessed by R package 'affy' [18] and scaled by 100. For the first three data the MSR thresholds are available in [8, 31]. However, for the other three data the MSR thresholds are not off-the-peg. Considering that ISA need not MSR as evaluating the bicluster, so the minimal value of MSR obtained by ISA on BCLL, PBC and Rat Strain are used to the threshold as shown in Table 2.

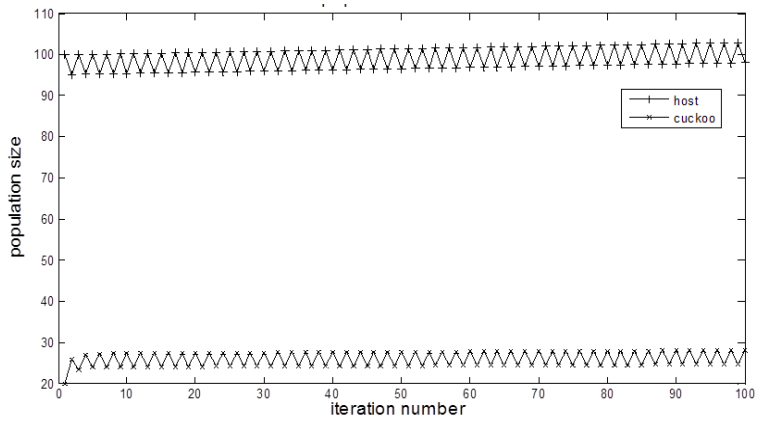


Fig. 4 The trend of the size of the population of cuckoo and host

Table 2. The data used in the experimentation

Data	**Name**	**#Gene**	**#Sample**	**Threshold**	**Ref**
Yeast Cycle	Yeast cell cycle	2884	17	300	[9]
DLBCL	diffuse large B-cell lymphoma	4026	96	1200	[1]
Gasch	Yeast stress conditions	2993	173	500	[17]
BCLL	B-cell chronic lymphocytic	12625	21	50	[16]
PBC	Primary breast cancer	22284	286	50	[35]
RatStrain	rat multiple tissue in strain	8799	122	5	[34]

Analysis of experiment

We aim to test whether CSB and COCSB can yield the discovery of biclusters characterized by small MSR and high volume. Moreover, we are interested in comparing the results achieved by CSB and COCSB against the results obtained by the other algorithms especially the stochastic search algorithms, so the compared algorithms will not only have CC, FLOC and ISA but also have SEBI, PSOB, BIC-aiNet, SAB and SSB. In this subsection the

following content mainly presents and analyzes the experiment results of eight compared biclustering algorithms and new proposed CSB and COCSB on the six gene expression data. The experiment result lists the gene number, sample number, MSR and time of a bicluster in which the gene number and the sample number together reflect the volume of a bicluster, MSR reflects the coherence and the time measured in seconds reflects the time performance of an algorithm. On each data the 100 biclusters are obtained for each algorithm, the mean and the standard variance of the gene number, the sample number, MSR, and time are presented in Tables 3-6.

Table 3. Average gene number obtained on each data

Data / Algorithm	Yeast Cycle	DLBCL	Gasch Yeast	BCLL	PBC	RatStrain
ISA	62.53±103.94	370.23±327.88	256.56±190.62	363.34±616.85	1763.56±1470.5	116.43±86.31
CC	340.83±47.65	39.87±9.17	63.02±76.92	315.67±93.69	35.80±14.44	204.80±65.80
FLOC	595.12±10.95	1330.69±135.99	798.15±108.10	359.24±175.20	1020.90±133.09	1694.54±46.13
BIC-aiNet	1266.10±296.61	2189.62±512.22	761.34±925.77	418.26±459.46	1620.27±430.57	1174.17±552.40
SEBI	1183.21±196.66	407.05±369.77	24.89±11.49	483.83±434.93	494.42±792.26	1250.73±501.63
PSOB	1360.23±77.59	379.67±231.78	22.48±14.09	465.33±410.63	308.60±631.75	1194.55±356.75
SAB	1200.46±154.38	829.26±541.93	214.46±62.45	505.03±281.11	1097.80±735.04	1082.30±357.07
SSB	695.36±178.92	196.79±232.27	16.40±6.62	454.25±104.31	65.46±20.42	1202.55±214.34
CSB	1435.34±185.86	688.15±251.80	257.55±134.61	373.21±190.44	254.07±83.00	1489.05±304.97
COCSB	684.20±79.37	397.27±297.43	243.82±20.45	3402.06±339.41	249.88±93.47	1383.34±366.15

Table 4. Average sample number obtained on each data

Data / Algorithm	Yeast Cycle	DLBCL	Gasch Yeast	BCLL	PBC	RatStrain
ISA	14.66±4.35	12.77±4.66	45.94±23.71	19.29±6.48	202.45±61.19	41.50±19.15
CC	16.92±0.307	90.57±11.10	47.56±18.43	24.00±0.00	44.25±12.53	24.25±7.15
FLOC	17.00±0.00	25.77±2.70	34.03±5.18	10.00±0.00	10.00±0.00	33.91±3.35
BIC-aiNet	10.18±0.41	12.80±5.35	79.09±57.37	10.02±0.14	11.68±16.40	14.72±8.38
SEBI	11.07±1.70	60.68±17.71	116.13±11.75	10.06±0.28	161.85±92.63	85.14±20.61
PSOB	10.01±0.10	58.02±14.56	120.37±12.99	10.09±0.40	176.67±80.15	85.03±13.30
SAB	10.33±0.91	37.47±17.05	72.20±7.04	10.00±0.00	38.74±37.38	70.69±17.90
SSB	15.24±1.53	72.34±16.74	120.77±11.54	12.29±3.86	179.50±29.38	97.60±12.56
CSB	11.56±1.32	36.36±9.20	59.31±13.34	14.95±2.46	74.19±33.35	88.40±15.87
COCSB	15.62+0.72	69.71+14.66	103.03±11.14	16.80±1.54	85.36±23.47	102.83±14.02

From Table 3 we can see that different algorithms show different performance in gene number for different data. It can be seen that for Yeast Cycle data PSOB is the best, for DLBCL BIC-aiNet is the best, for Gasch Yeast FLOC is the best, for BCLL COCSB is the best, for PBC ISA is the best and for RatStrain CSB is the best. It is worth noting that gene number of COCSB is better than most of the other algorithms.

Table 5. Average MSR obtained on each data

Data / Algorithm	Yeast Cycle	DLBCL	Gasch Yeast	BCLL	PBC	RatStrain
ISA	317.20±289.3	57094.01±176	0.93±0.42	419.15±450.6	913.97±119.3	1.43±1.02
CC	223.76±13.17	1010.20±96.6	0.03±0.00	97.00±1.56	46.13±3.48	2.41±1.01
FLOC	299.71±0.11	1199.61±0.25	0.05±0.00	68.82±30.03	30.20±4.04	0.82±0.12
BIC-aiNet	298.18±1.87	1199.61±0.96	0.05±0.00	99.99±4.47	50.01±0.31	5.00±0.03
SEBI	299.85±0.11	1198.48±1.35	0.05±0.00	99.99±4.42	49.91±0.07	4.99±0.01
PSOB	299.84±0.12	1198.31±1.50	0.05±0.00	100.00±4.86	49.90±0.08	4.99±0.01
SAB	299.86±0.10	1199.08±0.94	0.05±0.00	100.00±3.58	49.93±0.06	4.99±0.01
SSB	299.75±0.11	1197.35±2.54	0.05±0.00	100.00±4.13	49.90±0.13	4.99±0.01
CSB	299.84±0.11	1199.96±0.04	0.05±0.00	100.00±0.28	49.99±0.01	5.00±0.00
COCSB	299.75±0.10	1199.52+1.42	0.05±0.00	99.99±0.38	49.85±0.02	4.98±0.01

Table 6. Average time obtained on each data

Data / Algorithm	Yeast Cycle	DLBCL	Gasch Yeast	BCLL	PBC	RatStrain
ISA	0.12	0.01	0.04	4.5	0.28	0.42
CC	0.02	0.06	0.05	0.04	0.13	0.03
FLOC	4.05	14.98	14.14	49.95	77.33	95.54
BIC-aiNet	10.14	25.06	17.86	81.51	367.85	68.99
SEBI	6.66	15.79	14.72	43.95	357.23	81.81
PSOB	5.78	7.46	7.98	20.81	220.48	35.50
SAB	4.44	9.36	8.44	122.98	307.45	111.61
SSB	3.14	7.88	10.30	28.00	84.35	22.91
CSB	4.77	8.58	7.22	31.25	65.17	32.86
COCSB	3.95	6.49	12.76	20.50	58.36	34.19

As Table 4 shows, in sample number the systematic searching algorithms shows excellent performance. It can be seen that CC is the best for DLBCL and BCLL, ISA is the best for PBC and FLOC for Yeast Cycle. Even so, COCSB proposed in this work is the best for RatStrain and SSB for Gasch yeast.

For MSR, as shown Table 5 which MSR of almost all algorithms are smaller than the threshold while ISA which does not use MSR are far higher than the threshold. Further observation shows that MSR obtained by CC and FLOC is smallest for most data. This is because that CC and FLOC mainly focuses on the decreasing of MSR and ignores the volume.

Table 6 shows that the time of ISA and CC in systematic search algorithm is the smallest in all algorithms for each data. This is because the procedure of ISA and CC is specific. Even so, the time of FLOC is comparable with that of stochastic search algorithm. In stochastic search algorithm SSB are the best for Yeast Cycle and RatStrain and CSB and COCSB are the best for other data. In detailed, CSB for Gasch Yeast and COCSB for DLBCL, BCLL and PBC of which the gene number is very large shows that COCSB shows the potential of identifying the bicluster in large scale data.

Biological validation

The above analysis shows that CSB and COCSB proposed in this work can obtain the bicluster which is better than that of other algorithms. For further evaluating the significance of CSB and COCSB, the biological validation of the results obtained by all considered algorithms on two well-studied data: Yeast Cycle and Gasch Yeast. GO [2] can be used to investigate if a group of genes belonging to a bicluster presents significant enrichment about a specific GO term. Following the methodology [31] the performance of all algorithms is evaluated biologically with the proportion of the biclusters significantly enriched by GO and the weight enrichment score.

A bicluster is said to be significantly enrichment if p-values of one or more terms produced by the bicluster annotated by GO is smaller than the predefined significance level. So the performance of a biclustering algorithm can be evaluated by the proportion of the biclusters significantly enriched by GO, and the higher it is the better the performance of the algorithm in biological significance. Weight Enrichment score (*WEscore*) is used to accurately evaluate the quality of a bicluster and the performance of a biclustering algorithm. *WEscore* of a bicluster is described below:

$$WEscore = \sum_{i=1}^{n} x_i s_i / r_{IJ}, \tag{2}$$

where n is the number of GO terms annotated by GO, x_i and s_i are the gene number and -log10 transformed p-value of i-th GO terms, and r_{IJ} is the gene number of this bicluster. It can be seen from Eq. (2) that the higher the biological significance of a bicluster is, the larger is *WEscore*.

We implement the module of GO annotation in Matlab. This module of GO annotation first performs GO enrichment analysis in biological process, then uses the hyper geometric tests for statistical analysis to compute p-value, and finally uses Benjamin-Hochberg False Discovery Rate (FDR) procedure to perform the multiple testing corrections. In addition, since the probability of random chosen is relative large, the GO term only involving one gene is discarded. The eight significant levels selected in this work are 0.001%, 0.005%, 0.01%, 0.05%, 0.1%, 0.5%, 1% and 5%.

Table 7 and Table 8 represent the proportion of biclusters significantly enriched by GO in different significant levels and *WEscore* for Yeast Cycle and Gasch Yeast. Standard variance of *WEscore* are also reported in Tables 7-8.

As Table 7 shown, for Yeast Cycle data the proportion of biclusters significantly of the stochastic search algorithm are higher than that of systematic search algorithm while its *WEscore* are smaller than that of that of systematic search algorithm. Under $p < 0.001\%$ the proportion of biclusters significantly of ISA is 12.00%, of CC is 88.00%, of FLOC is 100.00%, of BIC-aiNet is 94.00%, of SEBI is 98.00%, of PSOB is 100.00%, of SAB is 99.00%, of SSB is 78.00% and of CSB and COCSB proposed in our work are 96.00% and 98.00%. However *WEscore* of ISA, CC and FLOC are 1.54, 163 and 1.45 which are higher than that of stochastic search algorithm. This is due to the volume of biclusters since it is easier to find functional enrichment from larger groups of genes than from small groups [29] which is validated by the gene number of ISA, CC and FLOC being far smaller than that of stochastic search algorithm as shown in Table 3. However, for all stochastic search algorithms

CSB and COCSB achieve the best biological performance according to *WEscore* of which CSB and COCSB are 1.38 and 1.49 and greater than that of other stochastic search algorithms.

Table 7. The proportion of biclusters significantly and *WEscore* on Yeast Cycle data

Algorithm	$p < 0.001\%$	$p < 0.005\%$	$p < 0.01\%$	$p < 0.05\%$	$p < 0.1\%$	$p < 0.5\%$	$p < 1\%$	$p < 5\%$	*WEscore*
ISA	12.00%	16.00%	17.00%	82.00%	84.00%	87.00%	90.00%	92.00%	1.54±0.67
CC	88.00%	96.00%	97.00%	100.00%	100.00%	100.00%	100.00%	100.00%	1.63±0.18
FLOC	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	1.45±0.09
BIC-aiNet	94.00%	94.00%	96.00%	97.00%	98.00%	99.00%	100.00%	100.00%	1.23±0.14
SEBI	98.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	1.36±0.18
PSOB	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	1.23±0.09
SAB	99.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	1.36±0.14
SSB	78.00%	97.00%	98.00%	100.00%	100.00%	100.00%	100.00%	100.00%	1.37±0.12
CSB	96.00%	99.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	1.38±0.23
COCSB	96.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	1.49±0.15

Table 8. The proportion of biclusters significantly and *WEscore* on Gasch Yeast data

Algorithm	$p < 0.001\%$	$p < 0.005\%$	$p < 0.01\%$	$p < 0.05\%$	$p < 0.1\%$	$p < 0.5\%$	$p < 1\%$	$p < 5\%$	*WEscore*
ISA	83.00%	83.00%	86.00%	93.00%	97.00%	99.00%	100.00%	100.00%	8.92±8.06
CC	14.00%	28.00%	33.00%	50.00%	61.00%	79.00%	89.00%	97.00%	1.60±0.63
FLOC	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	2.58±0.18
BIC-aiNet	51.00%	62.00%	67.00%	75.00%	78.00%	85.00%	88.00%	93.00%	1.53±0.69
SEBI	3.00%	11.00%	17.00%	34.00%	50.00%	85.00%	93.00%	99.00%	1.51±0.39
PSOB	3.00%	12.00%	21.00%	40.00%	55.00%	81.00%	88.00%	98.00%	1.63±0.52
SAB	80.00%	90.00%	95.00%	98.00%	100.00%	100.00%	100.00%	100.00%	1.95±0.42
SSB	2.00%	4.00%	8.00%	31.00%	43.00%	68.00%	83.00%	97.00%	1.62±0.55
CSB	67.00%	83.00%	87.00%	97.00%	99.00%	100.00%	100.00%	100.00%	1.97±0.39
COCSB	69.00%	89.00%	91.00%	100.00%	100.00%	100.00%	100.00%	100.00%	2.85±0.39

For Gasch Yeast data, as the above analysis of the biological performance of CSB and COCSB is higher than that of other stochastic search algorithm in Table 8, and it is smaller than that of ISA while it is slightly greater than that of CC and FLOC. This fully shows that in all stochastic search algorithms the biological performance of CSB and COCSB has been greatly enhanced.

As shown in Tables 7 and 8, the proportion of biclusters significantly and *WEscore* of COCSB are better than that of CSB which clearly shows that the improved COCSB has achieved great success compared to CSB. For Yeast Cycle *WEscore* of CSB is 1.38 while that of COCSB is 1.49, for Gasch yeast *WEscore* of CSB is 1.97 while that of COCSB is 2.85.

Conclusions

In this paper, we proposed two new stochastic search algorithms CSB and COCSB for biclustering gene expression data. Compared with other stochastic search algorithm, CSB and COCSB have many advantages. First, they are easy to implement and its number of the parameter is small compared to existing stochastic biclustering. Secondly, they have achieved great success in biological significance. Thirdly, they are able to efficiently explore the search space which makes identifying the biclusters in gene expression data faster than most of the stochastic algorithm. The performance of CSB and COCSB is compared in gene number, sample number, MSR and time as well as the biological validation based on GO. The compared result shows that CSB and COCSB are highly competitive in comparison with the biclustering algorithm chosen in this paper.

However, as mentioned in [12, 23] the threshold of MSR for each data decides the performances of the biclustering algorithm, and identifying the right threshold for each data is not emphasized in this work since the main focus of our work is putting forward and ameliorating biclustering algorithm, so the further work will focus on identifying the right threshold and using the other quality index such as VE, ACV and ASR for obtaining a better bicluster in biological significance.

Acknowledgments

This research was supported in part by the National Natural Science Foundation of China (NSFC) under grants 60903074, the National High Technology Research and Development Program of China (863 Program) under grant 2008AA01Z119, the Natural Science Foundation of the Jiangsu Province under grant BK20130417.

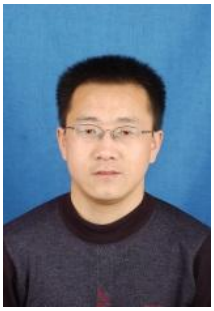
References

1. Alizadeh A. A., M. B. Eisen, R. E. Davis, C. Ma, I. S. Lossos, A. Rosenwald, J. C. Boldrick, H. Sabet, T. Tran, X. Yu, J. I. Powell, L. Yang, G. E. Marti, T. Moore, J. Hudson, L. Lu, D. B. Lewis, R. Tibshirani, G. Sherlock, W. C. Chan, T. C. Greiner, D. D. Weisen-Burger, J. O. Armitage, R. Warnke, R. Levy, W. Wilson, M. R. Grever, J. C. Byrd, D. Botstein, P. O. Brown, L. M. Staudt (2000). Distinct Types of Diffuse Large B-cell Lymphoma Identified by Gene Expression Profiling, Nature, 403, 503-511.
2. Ashburner M., C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, M. A. Harris, D. P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J. C. Matese, J. E. Richardson, M. Ringwald, G. M. Rubin, G. Sherlock (2000). Gene Ontology: Tool for the Unification of Biology. The Gene Ontology Consortium, Nature Genetics, 25(1), 25-29.
3. Ayadi W., M. Elloumi, J.-K. Hao (2009). A Biclustering Algorithm Based on a Bicluster Enumeration Tree: Application to DNA Microarray Data, BioData Mining, 2, 9-24.
4. Ben-Dor A., B. Chor, R. Karp, Z. Yakhini (2002). Discovering Local Structure in Gene Expression Data: The Order-preserving Submatrix Problem, Proceedings of the 6th Annual International Conference on Research in Computational Molecular Biology (RECOMB'02), Washington D.C., USA, 49-57.
5. Brown P., D. Botstein (1999). Exploring the New World of the Genome with DNA Microarrays, Nature Gentics, 21(S1), 33-37.
6. Bryan K., P. Cunningham, N. Bolshakova (2006). Application of Simulated Annealing to the Biclustering of Gene Expression Data, IEEE Transactions on Information Technology in Biomedicine, 10(3), 519-525.
7. Busygin S., O. Prokopyev, P. M. Pardalos (2008). Biclustering in Data Mining, Computers and Operations Research, 35(9), 2964-62987.

8. Cheng Y., G. M. Church (2000). Biclustering of Expression Data, Proceeding of the 8th International Conference on Intelligent System for Molecular Biology (Bourne P., M. Gribskov, Eds.), California, USA, 93-103.
9. Cho R. J., M. J. Campbell, E. A. Winzeler, L. Steinmetz, A. Conway, L. Wodicka, T. G. Wolfsberg, A. E. Gabrielian, D. Landsman, D. J. Lockhart, R. W. A. Davis (1998). Genome-wide Transcriptional Analysis of the Mitotic Cell Cycle, Mol Cell, 2(1), 65-73.
10. Davies N. B. (1999). Cuckoos and Cowbirds Versus Hosts: Co-evolutionary Lag and Equilibrium, Journal of African Ornithology, 70(1), 71-79.
11. de Franca F. O., G. Bezerra, F. J. Von Zuben (2006). New Perspective for the Biclustering Problem, Proceeding of 2006 IEEE Congress on Evolutionary Computation, Vancouver, BC, Canada, 753-760.
12. Divina F., B. Pontes, R. Giraldez, J. S. Aguilar-Ruiz (2012). An Effective Measure for Assessing the Quality of Biclusters, Computers in Biology and Medicine, 42(2), 245-256.
13. Divina F., J. S. Aguilar-Ruiz (2006). Biclustering of Expression Data with Evolutionary Computation, IEEE Transaction on Knowledge and Data Engineering, 18(5), 590-602.
14. Edgar R., M. Domrachev, A. E. Lash (2002). Gene Expression Omnibus: NCBI Gene Expression and Hybridization Array Data Repository, Nucleic Acids Research, 30(1), 207-210.
15. Eisen M. B., P. T. Spellman, P. O. Brown, D. Botstein (1998). Cluster Analysis and Display of Genome-wide Expression Patterns, PNAS, 95(25), 14863-14868.
16. Fält S., M. Merup, G. Gahrton, B. Lambert, A. Wennborg (2005). Identification of Progression Markers in B-CLL by Gene Expression Profiling, Exp Hematol, 33(8), 883-893.
17. Gasch A. P., P. T. Spellman, C. M. Kao, O. Carmel-Harel, M. B. Eisen, G. Storz, D. Botstein, P. O. Brown (2000). Genomic Expression Programs in the Response of Yeast Cells to Environmental Changes, Molecular and Cellular Biology, 11(12), 4241-4257.
18. Gautier L., L. Cope, B. M. Bolstad, R. A. Irizarry (2004). affy--analysis of Affymetrix GeneChip Data at the Probe Level, Bioinformatics, 20(3), 307-315.
19. Getz G., E. Levine, E. Domany (2000). Coupled Two-way Clustering Analysis of Gene Microarray Data, PNAS, 97(22), 12079-12084.
20. Hartigan J. (1972). Direct Clustering of a Data Matrix, Journal of the American Statistical Association, 337(67), 123-129.
21. Ihmels J., G. Friedlanders, S. Bergmann, O. Sarig, Y. Ziv, N. Barkai (2002). Revealing Modular Organization in the Yeast Transcriptional Network, Nature Genetics, 31, 370-377.
22. Lazzeroni L., A. Owen (2002). Plaid Models for Gene Expression data, Statistica Sinica, 12, 61-86.
23. Li X., U. Lu, M. Wang (2013). A Hybrid Gene Selection Method for Multi-category Tumor Classification Using Microarray Data, International Journal Bioautomation, 17(4), 249-258.
24. Liu J., W. Wang (2003). OP-cluster: Clustering by Tendency in High Dimensional Space, Proceedings of the 3rd IEEE International Conference on Data Mining, Bangalore, India, 87-194.
25. Liu J., Z. Liu (2009). Biclustering of Microarray with MOPSO Based on Crowding Distance, BMC Informatics, 10(4), s9.
26. Maderia S. C., A. L. Oliverial (2004). Biclustering Algorithms for Biological Data Analysis: A Survey, IEEE/ACM Trans Computation Biological Bioinformatics, 1(1), 24-45.
27. Mukhopadhyay A., U. Maulik, S. Bandyopadhyay (2010). On Biclustering of Gene Expression Data, Current Bioinformatics, 5, 204-216.

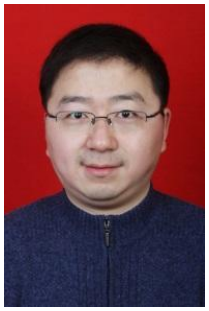
28. Murali T. M., S. Kasif (2003). Extracting Conserved Gene Expression Motifs from Gene Expression Data, Pacific Symposium Biocomputing, 8, 77-88.
29. Nepormuceno J. A., A. Troncoso, J. S. Aguilar-Ruiz (2011). Biclustering of Gene Expression Data by Correlation-based Scatter Search, BioData Mining, 4(3), 1-17.
30. Nowak M. A. (2006). Evolutionary Dynamics: Exploring the Equations of Life, Belknap Press, 65-66.
31. Prelić A., S. Bleuler, P. Zimmermann, A. Wille, P. Bühlmann, W. Gruissem, L. Hennig, L. Thiele, E. Zitzler (2006). A Systematic Comparison and Evaluation of Biclustering Methos for Gene Expression Data, Bioinformatics, 22(9), 1122-1129.
32. Tanay A., R. Sharan, R. Shamir (2002). Discovering Statistically Significant Biclusters in Gene Expression Data, Bioinformatics, 18, 136-144.
33. Tavazoie S., J. D. Hughes, M. J. Campbell, R. J. Cho, G. M. Church (1999). Systematic Determination of Genetic Network Architecture, Nature Genetic, 22, 281-285.
34. Walker J. R., A. I. Su, D. W. Self, J. B. Hogenesch, H. Lapp, R. Maier, D. Hoyer, G. Bilbe (2004). Applications of a Rat Multiple Tissue Gene Expression Data Set, Genome Research, 14(4), 742-749.
35. Wang Y., J. G. Klijn, Y. Zhang, A. M. Sieuwerts, M. P. Look, F. Yang, D. Talantov, M. Timmermans, M. E. Meijer-van Gelder, J. Yu, T. Jatkoe, E. M. Berns, D. Atkins, J. A. Foekens (2005). Gene-expression Profiles to Predict Distant Metastasis of Lymph-node-negative Primary Breast Cancer, Lancet, 365(9460), 671-679.
36. Yang J., W. Wang, H. Wang, P. Yu (2002). δ-Clusters: Capturing Subspace Correlation in a Large Data Set, Proceedings of the 18th IEEE International Conference on Data Engineering, California, USA, 517-528.
37. Yang X., S. Deb (2009). Cuckoo Search via Lévy Flights, Proceeding of World Congress on Nature & Biologically Inspired Computing, India, 210-214.

Lu Yin, Ph.D. Student
E-mail: yinlu_78@163.com



Lu Yin received his M.E. degree in Computer Software and Theory from TaiYuan University of Technology in 2005. Currently he is a Ph.D. student in computer software and theory at the University of Electronic Science and Technology of China. His research interests are in the field of bioinformatics, data mining and machine learning.

Prof. Yongguo Liu, Ph.D.
E-mails: liuyg@uestc.edu.cn,
liuyg_cn@163.com



Prof. Yongguo Liu received his Ph.D. in Engineering in Computer Science from Chongqing University in 2003 and has been out of post-doctoral stations at the Shanghai Jiaotong University in 2005. His research interests are in the field of data mining and TCM Information Science.