Solving Biclustering with a GRASP-Like Metaheuristic: Two Case-Studies on Gene Expression Analysis

Angelo Facchiano[1], Paola Festa[2], Anna Marabotti[3], Luciano Milanesi[3], and Francesco Musacchia[2]

[1] Istituto di Scienze dell'Alimentazione - CNR, Italy
angelo.facchiano@isa.cnr.it
[2] University of Napoli "Federico II", Italy
{paola.festa,francesco.musacchia}@unina.it
[3] Istituto di Tecnologie Biomediche - CNR, Italy
{anna.marabotti,luciano.milanesi}@itb.cnr.it

Abstract. The explosion of "omics" data over the past few decades has generated an increasing need of efficiently analyzing high-dimensional gene expression data in several different and heterogenous contexts, such as for example in information retrieval, knowledge discovery, and data mining. For this reason, biclustering, or simultaneous clustering of both genes and conditions has generated considerable interest over the past few decades. Unfortunately, the problem of locating the most significant bicluster has been shown to be NP-complete. We have designed and implemented a GRASP-like heuristic algorithm to efficiently find good solutions in reasonable running times, and to overcome the inner intractability of the problem from a computational point of view.

Experimental results on two datasets of expression data are promising indicating that this algorithm is able to find significant biclusters, especially from a biological point of view.

Keywords: Biclustering, gene expression analysis, GRASP, combinatorial optimization, approximate solutions.

1 Introduction

Traditional clustering tasks take as input a data set and a similarity (or distance) function over the domain, with the aim of finding a partition of the data into groups of mutually similar elements. Biclustering (term coined by Hartigan [1]) is a variant of this task that is needed when the input data comes from two domain sets and some relation over the Cartesian product of these two sets is given. In this case, one could be interested in partitioning each of the sets, such that the subsets from one domain exhibit similar behavior across the subsets of the other domain. Roughly speaking, bi-clustering can be viewed as simultaneous data clustering and feature selection, i.e., detection of significant clusters and the features that are uniquely associated with them, given that not all features are relevant to certain clusters.

E. Biganzoli et al. (Eds.): CIBB 2011, LNBI 7548, pp. 253–267, 2012.

In the pioneering work by Cheng and Church [2] biclustering was first introduced for the purpose of gene expression analysis. In the scientific literature devoted to biclustering [3], several classes of biclusters have been identified, depending on the chosen definition of homogeneity.

The state-of-the-art methods proposed for approaching the problem can be divided into five main classes: 1) exhaustive enumeration algorithms, that exhaustively search in the input matrix the best biclusters with very high computational running times [4, 5]; 2) iterative row and column clustering combination algorithms, that first apply separately clustering algorithms to the rows and columns of the data matrix and then combine the results using some sort of iterative procedure [6, 7]; 3) divide and conquer algorithms, that divide the problem in subproblems and are potentially very fast but usually split good biclusters before they can be identified [1, 8]; 4) greedy iterative search algorithms, that, based on the steepest descent idea, create biclusters by adding and/or removing rows and columns optimizing a local gain criterion [9–11], and 5) distribution parameter identification algorithms, that try to identify the distribution parameters used to generate the data [12–14]. Very recently, Aradhya et al. [15] proposed an approach based on Modular Singular Value Decomposition (Mod-SVD), that first partitions the input data matrix into a set of equally sized submatrices and applies a SVD to each of the submatrices to be then concatenated. It is only in the past few years that the biclustering task has been approached via metaheuristic algorithms, especially in the presence of large scale problem instances. They include a Simulated Annealing [16], a Genetic Algorithm [17], and a Reactive GRASP [18]. The reader can refer to [3] and [19] for recent surveys.

We have implemented Dharan and Nair's proposal and in our experience we observed that, even if robust and elegant, the local search applied at each iteration is time consuming and very rarely improves the constructed solution. In this paper, we propose a novel Reactive GRASP-like that overcomes the drawback of the Reactive GRASP proposed by Dharan and Nair in [18].

The paper is organized as follows. In Section 2 biclustering tasks are formally stated. Section 3 introduces the main issues in designing a GRASP method and Section 4 describes our Reactive GRASP-like proposal. In Section 5 we synthesize the whole analysis process and report and discuss the experimental results obtained on two case studies of gene expression experiments. We also report some consideration on future work in this challenging research area.

2 Problem Formulation

The goal of biclustering techniques is to identify subgroups of genes and subgroups of conditions, by performing simultaneous clustering of both n rows and m columns of a given gene expression matrix $\mathcal{A} \in \mathbb{R}^{n \times m}$, where each element a_{ij} represents the expression level of gene i under condition j. As in [3], in order to coherently represent all possible scenarios of biclustering real–world applications we will consider the general case of a data matrix $\mathcal{A} = (X, Y)$, where $X = \{x_1, \ldots, x_n\}$ is its set of rows, Y is its set of columns $Y = \{y_1, \ldots, y_m\}$,

and the element a_{ij}, $(i \in X,\ j \in Y)$, corresponds to a value representing the relation between row i and column j.

Let $I \subseteq X$ and $J \subseteq Y$ be subsets of the rows and columns, respectively. Then, $\mathcal{A}_{IJ}$ denotes the submatrix of $\mathcal{A}$ that contains all the elements a_{ij} of $\mathcal{A}$ such that $i \in I$ and $j \in J$ and a *bicluster* $\mathcal{B} = \mathcal{A}_{IJ}$ is a $k \times s$ submatrix of $\mathcal{A}$, where $I = \{x_{i_1}, \ldots, x_{i_k}\} \subseteq X$ and $J = \{y_{j_1}, \ldots, y_{j_s}\} \subseteq Y$, i.e. it is a subset of $k \leq n$ rows defined over a subset of $s \leq m$ columns or, equivalently, a subset of $s \leq m$ columns defined over a subset of $k \leq n$ rows.

A natural representation of data matrices in clustering/biclustering problems is by means of a *complete weighted bipartite graph* $G = (V, E)$[1], where $V = X \cup Y$ (clearly, $X \cap Y = \emptyset$) and $E = \{[x_i, y_j] \mid x_i \in X,\ y_j \in Y\}$. Moreover, a *weight function* $w : E \mapsto \mathbb{R}$ is defined that to each edge $[x_i, y_j] \in E$ assigns a weight $w_{ij} = a_{ij} \in \mathbb{R}$.

This graph theoretical representation helped in understanding the inner intractability of the problem of finding a maximum size bicluster in a data matrix $\mathcal{A}$. In fact, even in its simplest form where $\mathcal{A} \in \{0, 1\}^{n \times m}$ the problem is **NP**-complete, since it reduces to finding the maximum edge biclique in the corresponding bipartite graph G [20]. Generally speaking, given a data matrix $\mathcal{A}$, biclustering aims at identifying a set of biclusters $\{\mathcal{B}_1 = (I_1, J_1), \ldots, \mathcal{B}_k = (I_k, J_k)\}$ such that each bicluster $\mathcal{B}_q$, $q = 1, \ldots, k$, satisfies some specific characteristics of "homogeneity", whose definition varies from approach to approach and plays an important role to evaluate a biclustering algorithm and the quality of the type of biclusters that it is able to find. In this paper, we want to identify "biclusters with coherent values". Therefore, we want to analyze directly the numeric values in the data matrix $\mathcal{A}$ and try to find subsets of rows and subsets of columns with similar behaviors. For this class, biclusters cannot be found simply by considering that the values within the bicluster are given by additive or multiplicative models that consider an adjustment for either the rows or the columns. More sophisticated statistical approaches are needed to evaluate the quality of the resulting bicluster or set of biclusters. According to Cheng and Church [2], we have used as a measure of the coherence of the rows and columns in the bicluster the *mean squared residue score* to be minimized and defined as the sum of the squared residues, where the residue of an element a_{ij} in $\mathcal{B}$ is the difference between its actual value and its expected value predicted from the corresponding row mean, column mean, and bicluster mean.

3 GRASP

GRASP (Greedy Randomized Adaptive Search Procedures) [21, 22] is a multi-start metaheuristic for producing good-quality solutions of hard combinatorial optimization problems and it has been efficiently applied to many problems. Each GRASP iteration is usually made up of a construction phase, where a

[1] A graph $G = (V, E)$ is said a *bipartite graph* if $V = V_1 \cup V_2$, $V_1 \cap V_2 = \emptyset$, and for each $[i, j] \in E$, $i \in V_1$ and $j \in V_2$. Moreover, G is *complete*, if for each $v_1 \in V_1$ and for each $v_2 \in V_2$, $[v_1, v_2] \in E$.

feasible solution is constructed in a *greedy*, *randomized*, and *adaptive* manner, and a local search phase which starts at the constructed solution and applies iterative improvement until a locally optimal solution is found. The reader can refer to [23–25] for a study of a generic GRASP metaheuristic framework and its applications.

Stopping criteria could be maximum number of iterations, maximum number of iterations without improvement of the incumbent solution, maximum running time, or solution quality at least as good as a given target value. A complete solution is iteratively constructed in the construction phase, one element at a time. At each construction iteration, the choice of the next element to be added is determined by ordering all candidate elements (i.e. those that can be added to the solution) in a candidate list C with respect to a greedy function $g : C \rightarrow R$. This function measures the (myopic) benefit of selecting each element. The heuristic is adaptive because the benefits associated with every element are updated at each iteration of the construction phase to reflect the changes brought on by the selection of the previous element. The probabilistic component of a GRASP is characterized by randomly choosing one of the best candidates in the list, but not necessarily the top candidate. The list of best candidates is called the *restricted candidate list* (RCL).

As is the case for many deterministic methods, it is almost always beneficial to apply a local search to attempt to improve each constructed solution. A local search algorithm replaces the current solution by a better solution in the neighborhood of the current solution, until a *locally optimal* solution is found.

In the next section, we describe the details of the Reactive GRASP-like algorithm that we have designed and implemented for the biclustering task.

4 A Reactive GRASP-Like Algorithm for Biclustering

GRASP is a multi-start high-level procedure that coordinates simple heuristics and rules (i.e., construction and local search) to find good (often optimal) approximate solutions of computationally hard combinatorial optimization problems.

In our GRASP-like proposal for biclustering, we adopted the stopping criterion that counts a maximum number of iterations without improvement of the incumbent solution (`MaxNoImpr`) and we implemented the reactive version of the metaheuristic framework. The pseudo-code is reported in Figure 1.

Our novel Reactive GRASP-like overcomes the drawback of the Reactive GRASP proposed by Dharan and Nair in [18]. We have implemented Dharan and Nair's proposal and in our experience we observed that, even if robust and elegant, the local search applied at each iteration is time consuming and very rarely improves the constructed solution, since it decides if to add or not a new element in the current solution on the only merit function basis. To overcome this drawback, we have designed a completely different local search strategy that uses two local improvement procedures that successively replace a bicluster in the current solution by a better bicluster in its neighborhood made of all biclusters that differ either because they have one more element (row or column)

```
algorithm GRASP-like-bicluster(A,MaxNoImpr,MaxDist,δ)
1   Δ := {α_1, ..., α_ℓ};            /* α_i ∈ [0,1], i = 1, ..., ℓ */
2   for i = 1 to ℓ do
3       p_{α_i} := 1/ℓ;
4   endfor
5   B = {B_1, ..., B_k} :=filtered-Kmeans(A);   /* H(B_q) ≤ δ, q = 1, ..., k */
6   for q = 1 to k do
7       B̂_q :=grasp(B_q,Δ,A,MaxNoImpr,MaxDist);
8   endfor
9   return (B̂ = {B̂_1, ..., B̂_k});
end
```

Fig. 1. Pseudo-code of our GRASP-like algorithm for biclustering

```
procedure grasp(B_q,Δ,A,MaxNoImpr,MaxDist)
1   count := 0;
2   repeat
3       (c,B̄_q):=build-columns(B_q,Δ,A);
3       (bool,B'_q):=local-improvement-columns(c,Δ,B̄_q,A,MaxDist);
4       if (bool) then count := 0;
5       else count := count + 1;
6       endif
7   until (count =MaxNoImpr)
8   count := 0;
9   repeat
10      (c,B̄_q):=build-rows(B'_q,Δ,A);
11      (bool,B'_q):=local-improvement-rows(c,Δ,B̄_q,A,MaxDist);
12      if (bool) then count := 0;
13      else count := count + 1;
14      endif
15  until (count =MaxNoImpr)
16  return (B');
end
```

Fig. 2. Pseudo-code of `grasp` procedure invoked in our GRASP-like algorithm

and/or one less element. The specific element to be removed and/or added is chosen on the basis either of the diversity or of the improvement in terms of objective function value given by the mean squared residue.

In more detail, our algorithm starts from a partial solution made of a set $\mathcal{B} = \{\mathcal{B}_1, \ldots, \mathcal{B}_k\}$ of k biclusters found by applying a k-means procedure and retaining only biclusters with small mean squared residue, i.e. those biclusters $\mathcal{B}_q = (I_q, J_q)$ such that $H(\mathcal{B}_q) \leq \delta$, where δ is a given input parameter.

As shown in Figure 2, the method proceeds in the attempt of finding a larger and locally better solution iteratively considering first the columns in Y (lines 1–7) and then the rows in X (lines 8–15), until `MaxNoImpr` iterations are performed

```
procedure build-columns(B_q,Δ,A)
1  C := ∅; g_min := large; g_max := 0;          /* B_q = (I_q, J_q), A = (X, Y) */
2  for each y ∈ Y \ J_q do
3      C := C ∪ {y};
4      g(y) := H(I_q, J_q ∪ {y});              /* mean squared residue */
5      if (g_min > g(y)) then g_min := g(y);
6      if (g_max < g(y)) then g_max := g(y);
7  endfor
8  α :=select(Δ);
9  μ := g_min + α(g_max − g_min);
10 RCL:= {c ∈ C | g(c) ≤ μ};
11 c :=select(RCL); J_q := J_q ∪ {c};
12 return (c,B_q = (I_q, J_q));
end
```

Fig. 3. Pseudo-code of `build-columns` procedure invoked in our GRASP-like algorithm

```
procedure build-rows(B_q,Δ,A)
1  C := ∅; g_min := large; g_max := 0;          /* B_q = (I_q, J_q), A = (X, Y) */
2  for each x ∈ X \ I_q do
3      C := C ∪ {x};
4      g(x) := H(I_q ∪ {x}, J_q);              /* mean squared residue */
5      if (g_min > g(x)) then g_min := g(x);
6      if (g_max < g(x)) then g_max := g(x);
7  endfor
8  α :=select(Δ);
9  μ := g_min + α(g_max − g_min);
10 RCL:= {c ∈ C | g(c) ≤ μ};
11 c :=select(RCL); I_q := I_q ∪ {c};
12 return (c,B_q = (I_q, J_q));
end
```

Fig. 4. Pseudo-code of `build-rows` procedure invoked in our GRASP-like algorithm

without improving the current better bicluster. The best incumbent bicluster is returned in line 16. Both procedures `build-columns` (Figure 3) and `build-rows` (Figure 4) take as input a partial bicluster $\mathcal{B}_q$ and try to enlarge it. The choice of the next element to be added to the partial solution is determined by ordering all candidate elements in a candidate list C with respect to their incremental costs given by evaluating a greedy function $g : C \rightarrow R$ that is the mean squared residue. The RCL then is the list of best candidates. The heuristic is adaptive because the incremental costs associated with every element are updated at each iteration to reflect the changes brought on by the selection of previous elements. The probabilistic component of our GRASP-like algorithm is characterized by randomly choosing one element from the RCL, but not necessarily the top candidate. In more detail, let $g(i)$ be the incremental cost associated with

```
procedure local-improvement-rows(c,Δ,B̄_q,A,MaxDist)
1  score := H(Ī_q, J̄_q);            /* B̄_q = (Ī_q, J̄_q), A = (X, Y) */
2  D := Ī_q; new :=select(X); dist :=distance(new,c);
3  if (new ∈ D) then
4      if (dist >MaxDist) then D := D \ {new};
5  else
6      if (dist ≤MaxDist) then D := D ∪ {new};
7  endif
8  new := argmin_{d∈D} H(D \ {d}, J̄_q);        /* mean squared residue */
9  D := D \ {d}; new :=select(D);
10 dist :=distance(new,c);
11 if (dist >MaxDist and H(D \ {new}, J̄_q) < H(D, J̄_q)) then D := D \ {new};
12 if (score > H(D, J_q)) then
13     reevaluate-probabilities(Δ);
14     Ī_q := D; bool := true;
15 else
16     bool := false;
17 endif
18 return (bool,B̄_q = (Ī_q, J̄_q));
end
```

Fig. 5. Pseudo-code of `local-improvement-rows` procedure invoked in our GRASP-like algorithm

the incorporation of element i in the solution under construction and let g_{min} and g_{max} be the smallest and the largest incremental costs, respectively, i.e.

$$g_{min} = \min_{c \in C} g(c), \qquad g_{max} = \max_{c \in C} g(c). \tag{1}$$

The restricted candidate list RCL is made up of elements $c \in C$ with the best (i.e., the smallest) incremental costs $g(c)$. There are two main mechanisms to build this list: a *cardinality-based* (CB) and a *value-based* (VB) mechanism. In the CB case, the RCL is made up of the z elements with the best incremental costs, where z is a parameter. In the VB case, the mechanism that we adopted, the RCL is associated with a parameter $\alpha \in [0, 1]$ and a threshold value $\mu = g_{min} + \alpha(g_{max} - g_{min})$. In fact, all candidate elements c whose incremental cost $g(c)$ is no greater than the threshold value are inserted into the RCL. Note that, the case $\alpha = 0$ corresponds to a pure greedy algorithm, while $\alpha = 1$ is equivalent to a random construction.

As already underlined, we have implemented the *reactive* version of the GRASP metaheuristic framework. Reactive GRASP is the first enhancement that incorporates a learning mechanism in the memoryless construction phase of the basic GRASP. In Reactive GRASP, the value of the RCL parameter α is selected in each iteration from a discrete set of possible values with a probability that depends on the solution values found along the previous iterations. One way to accomplish this is to use the rule proposed in [26]. Let $\Delta = \{\alpha_1, \alpha_2, \ldots, \alpha_\ell\}$ (Figure 1 line 1)

```
procedure local-improvement-columns(c,Δ,B̄_q,A,MaxDist)
1  score := H(Ī_q, J̄_q);            /* B̄_q = (Ī_q, J̄_q), A = (X, Y) */
2  D := J̄_q; new :=select(Y); dist :=distance(new,c);
3  if (new ∈ D) then
4      if (dist >MaxDist) then D := D \ {new};
5  else
6      if (dist ≤MaxDist) then D := D ∪ {new};
7  endif
8  new := argmin_{d∈D} H(Ī_q, D \ {d});          /* mean squared residue */
9  D := D \ {d}; new :=select(D);
10 dist :=distance(new,c);
11 if (dist >MaxDist and H(Ī_q, D \ {new}) < H(Ī_q, D)) then D := D \ {new};
12 if (score > H(I_q, D)) then
13     reevaluate-probabilities(Δ);
14     J̄_q := D; bool := true;
15 else
16     bool := false;
17 endif
18 return (bool,B̄_q = (Ī_q, J̄_q));
end
```

Fig. 6. Pseudo-code of `local-improvement-columns` procedure invoked in our GRASP-like algorithm

be the set of possible values for α. At the first GRASP iteration, all ℓ values have the same probability to be selected (lines 2–4), i.e.

$$p_{\alpha_i} = \frac{1}{\ell}, \qquad i = 1, \ldots, \ell. \tag{2}$$

At any subsequent iteration, let $\hat{z}$ be the incumbent solution objective function value and let A_i be the average objective function value of all solutions found using $\alpha = \alpha_i, i = 1, \ldots, \ell$. The selection probabilities are periodically reevaluated (Figure 5 line 13 and Figure 6 line 13) as follows:

$$p_i = \frac{q_i}{\sum_{j=1}^{\ell} q_j}, \tag{3}$$

where $q_i = \hat{z}/A_i$, $i = 1, \ldots, \ell$. If values of $\alpha = \alpha_i$ ($i \in \{1, \ldots, \ell\}$) lead to the best solutions on average, then the value of q_i is increased and larger values of q_i correspond to more suitable values for α. The probabilities associated with these more appropriate values will then increase when they are reevaluated.

Due to greater diversification and less reliance on parameter tuning, Reactive GRASP has lead to improvements over the basic GRASP in terms of robustness and solution quality. In fact, it has been successfully applied in power system transmission network planning [27] and in a capacitated location problem [28].

We next focus on the local search strategy that we have designed. We have implemented two local search algorithms, whose pseudo-codes are reported in Figures 5 and 6, respectively. Both the procedures successively replace a bicluster $\bar{\mathcal{B}}_q = (\bar{I}_q, \bar{J}_q)$ in the current solution by a better bicluster in the neighborhood of $\bar{\mathcal{B}}_q$ made of all biclusters that differ from $\bar{\mathcal{B}}_q$ either because they have one more element (row or column) and/or one less element. The element *new* to be removed and/or added is chosen on the basis either of the diversity (Figures 5 and 6 lines 3–7) or of the improvement in terms of objective function value given by the mean squared residue (Figures 5 and 6 lines 8–11). If a better mean squared residue neighbor bicluster is found (Figures 5 and 6 line 12), then the selection probabilities of the α's in Δ are accordingly reevaluated (Figures 5 and 6 line 13).

Example. Suppose that we have a matrix $\mathcal{A}$ of 10 genes (rows) and 5 conditions (columns). Fixed as input the number of sets of genes and conditions (in the following example, 3 and 2, respectively), k-means algorithm will provide as output the required sets. Those sets may contain common elements but there will never be identical.

Then, biclusters seeds ($\mathcal{B}$) are created: in our example, 3×2 combinations are made to find a match between each set of genes and each set of conditions $(\mathcal{B}_1, \ldots, \mathcal{B}_6)$. Among these combinations, only those whose mean squared residue (hScore) is less than or equal to a given threshold δ are saved. Suppose that only 3 out of the 6 candidates are selected ($\mathcal{B}_1$,$\mathcal{B}_2$,$\mathcal{B}_3$).

Now, the 3 biclusters are given as input to an iterative refinement procedure that tries to add and/or remove items, considering first the columns and then the rows.

Let us suppose that $\mathcal{B}_1$ has 6 rows and 3 columns (1,3,5). The procedure evaluates the improvement that could be obtained from the insertion of one of the remaining columns (2 and 4), in terms of hScore. If hScore is below a given threshold μ, then the corresponding column is inserted within a list of elements (RCL). Suppose that both columns 2 and 4 are included. One element (suppose column 4) is selected at random from the RCL and added to $\mathcal{B}_1$.

Once modified the current solution, the local search tries to improve it, by performing the following three steps.

1. Extract a new random element from the columns not included in the current solution (in our example, column 2). If the distance of column 2 from the column previously extracted from RCL (column 4) is less than or equal to a threshold given in input (`MaxDist`), column 2 is added to $\mathcal{B}_1$. Let us suppose this is the case.
2. From $\mathcal{B}_1$ the column that makes the hScore value of the bicluster worst is then eliminated. Suppose that this column is 3. Therefore, the set of columns in the new bicluster $\mathcal{B}_1$ is 1,2,4,5.
3. A further column is selected at random from all the columns of $\mathcal{B}_1$. It will be removed only if an improvement in terms of hScore is obtained. Supposing that this happens for column 5, the final bicluster consists of columns 1,2,4.

This procedure stops after a certain number of iterations without improvement (`MaxNoImpr`) and performs the above described operations on the set of rows of the bicluster under the same stopping condition.

The whole iterative procedure is applied on each selected bicluster ($\mathcal{B}_1$, $\mathcal{B}_2$, and $\mathcal{B}_3$).

5 Experimental Results and Biological Significance

Our Reactive GRASP-like algorithm has been implemented in C language, compiled with the Apple Xcode 3.1, and run on a MacBookPro 2GHz Intel Core Duo running MAC OSX 10.6. We have performed several iterations adopting the stopping criterion that counts a maximum number of iterations without improvement of the incumbent solution and inspected the results obtained.

A series of experiments has been conducted on the Yeast (Saccharomyces cerevisiae) cell cycle expression dataset [29] and on the dataset coming from the Lymphoma/Leukemia Molecular Profiling Project [30] to evaluate the quality of the proposed algorithm.

The first dataset includes 2884 genes and 17 conditions, with the expression level reported as an integer value in the range 0 to 600. Missing values in Yeast dataset are represented by -1. The second dataset is formed by 4026 genes and 96 conditions, with the expression level reported as an integer value in the range -300 to 300.

Table 1. Statistics on results of biclustering on the Yeast cell cycle expression dataset and on the Lymphoma/Leukemia molecular profiling project. The table lists the mean values of number of genes, number of conditions, volume, squared residue H, and running time over 10 trials using 10 different random number generator seeds. The last row reports the mean squared residual Hr obtained on a set of 33 (for yeast dataset) and 11 (for Lymphoma dataset) biclusters with the same cardinality of the bicluster obtained by our Reactive GRASP-like algorithm but with randomly selected membership.

Statistics	Yeast Dataset	Lymphoma Dataset
mean number of genes	97,33	59,63
mean number of conditions	10,52	8,18
mean volume	1000,06	478,93
mean H value	195,73	0,03
mean running time (in secs)	4044,43	5012,03
mean H_r value	1821,76	0,56

Table 1 shows results for a set of 33 biclusters generated for Yeast, and 11 biclusters generated for Lymphoma, in terms of number of genes, number of conditions, mean volume, mean squared residue H and mean running time over 10 trials using 10 different random number generator seeds. The differences in the values of H scores and volumes for the two datasets depend on the numerical

values of the data included in each bicluster. In the last row, Table 1 reports the mean squared residual H_r obtained on a set of 33 and 11 biclusters with the same cardinality of the biclusters obtained by our Reactive GRASP-like algorithm but with randomly selected membership. Comparing this mean squared residual with the mean squared residual obtained with our approach, it is evident that our proposal is outperforming a simple random approach, since the H_r value is in both cases about one order of magnitude larger than the H.

Looking at the bicluster plots in Figure 7 and 8, one can notice that the genes in sample biclusters present a similar behavior under a set of conditions. This proves that our method is able to identify coherent biclusters from gene expression data.

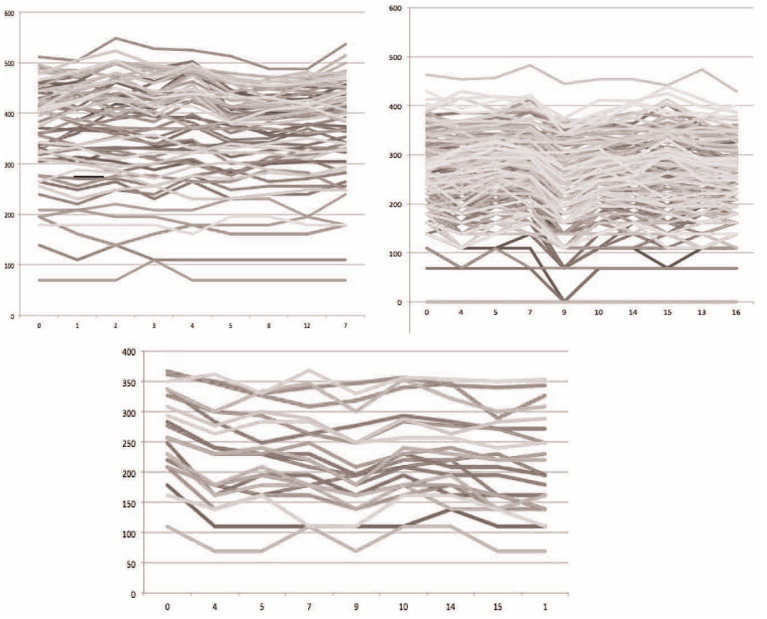

Fig. 7. Graphical representation of the expression levels for sample biclusters obtained in our analysis on Yeast dataset ([29]). On the rows we have the gene behaviour and on columns the conditions.

In order to verify the biological significance of biclusters obtained, we used GO annotation database and tools online. In GO annotation, terms describing biological processes, cellular components, and molecular functions are assigned to genes, so that a list of genes can be analyzed, looking for terms associated. The statistical significance to which the genes matches with the different GO terms or categories can be indicated by p-value. We used the Yeast Genome Gene Ontology Term Finder [31] to evaluate the biological significance of the 33 biclusters obtained in our analysis of the Yeast dataset, and the PANTHER (Protein ANalysis THrough Evolutionary Relationships) Classification System [32] for the analysis on Lymphoma results.

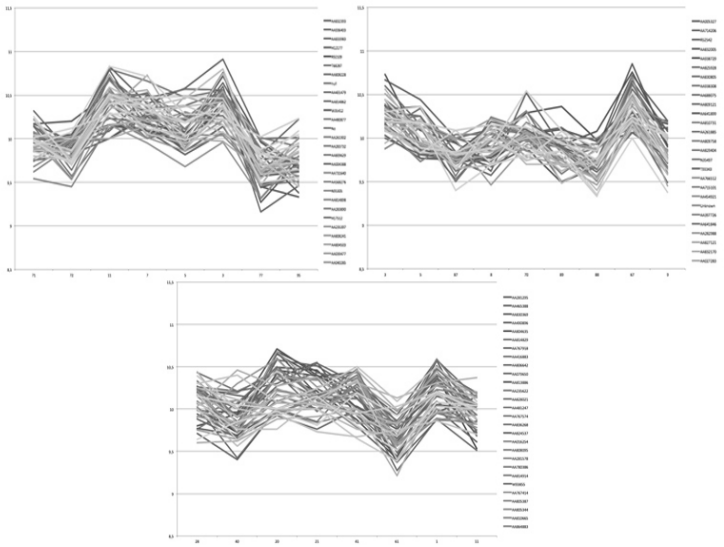

Fig. 8. Graphical representation of the expression levels for sample biclusters obtained in our analysis on Lymphoma dataset ([30]). On the rows we have the gene behaviour and on columns the conditions.

Table 2. Statistics on results of biclustering on the Yeast cell cycle expression dataset and on the Lymphoma/Leukemia molecular profiling project. The table shows a summary of the results in terms of p-values.

Statistics	Yeast Dataset	Lymphoma Dataset
mean p-value for biological process	1,83E-03	1,15E-03
mean p-value for molecular function	9,28E-04	5,88E-03
mean p-value for cellular component	1,60E-03	1,38E-01
minimum p-value for biological process	3,89E-15	5,25E-05
minimum p-value for molecular function	5,08E-17	3,27E-08
minimum p-value for cellular component	6,62E-22	1,05E-03

We reported in Table 2 a summary of the results, in terms of the mean p-value and the best p-value obtained for each of the three main categories, i.e. biological process, cellular component, molecular function. The analysis has been performed by submitting each gene list to the tool, and when a significant result was obtained, the p-value was selected. When two or more GO terms were significantly associated to the gene list, only the lowest p-value was selected. Figure 9 shows an example of the graphical result of the analysis for one of the biclusters analyzed. The analysis has found at least one GO term significantly associated to the gene list for 29 out of 33 biclusters in Yeast database, and for 11 out of 11 biclusters in Lymphoma dataset. This means that biclusters are made of

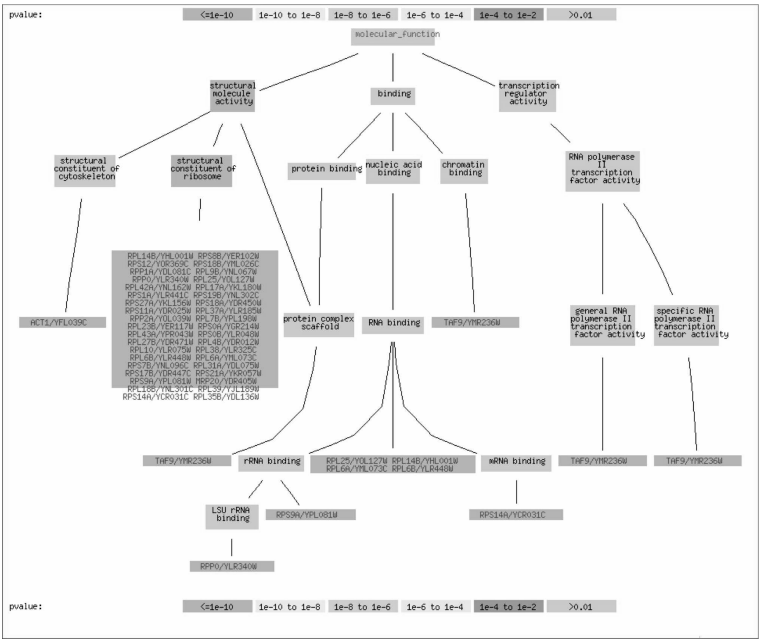

Fig. 9. The graphical output of the Yeast Genome Gene Ontology Term Finder tool for a sample bicluster obtained in our analysis

genes not only associated in terms of similar expression levels in the experimental data, but also with biological relationships, with a statistical confirm of the significance of this relationship. In the example of Figure 9, the GO term "structural costituent of ribosome" and its parent "structural molecule activity", have p-value ¡=1e-10, and are those most significantly associated to the list of genes included in the specific bicluster analyzed. This example shows that our analysis identified in this case a bicluster enriched by genes whose function, at level of protein expressed, is focalized on structural functions of ribosome. Therefore, the GO analysis confirms from a biological point of view the coherence of the bicluster analysis, being most of the gene clusters characterized by a common function, or cellular localization, or by the involvement in a biological process.

In conclusion, our Reactive GRASP-like algorithm is able to overcome several drawbacks of previous approaches for biclustering of biological data. We plan to perform further validation with other datasets from literature, as well as to design further variants of the algorithm to incorporate an intensification procedure by means of path-relinking [33, 34] and/or designing variable neighborhood structures [35, 36].

Acknowledgments. We acknowledge the support by MIUR FIRB ITALBIONET (RBPR05ZK2Z and RBIN064YAT_003) for A.M. and L.M. contribution to the work, and by "Programma Italia-USA Farmacogenomica Oncologica" to A.F.. This work has been made in the frame of the Flagship project InterOmics and CNR-Bioinformatics project.

References

1. Hartigan, J.: Direct clustering of a data matrix. J. Am. Stat. Assoc. 67, 123–127 (1972)
2. Cheng, Y., Church, G.M.: Biclustering of expression data. In: Altman, R., Bailey, T., Bourne, P., Gribskov, M., Lengauer, T., Shindyalov, I. (eds.) Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology (ISMB 2000), pp. 93–103 (2000)
3. Madeira, S., Oliveira, A.: Biclustering algorithms for biological data analysis: A survey. IEEE/ACM Trans. Comput. Biol. Bioinform. 1, 24–45 (2004)
4. Tanay, A., Sharan, R., Shamir, R.: Discovering statistically significant biclusters in gene expression data. Bioinformatics 18(suppl. 1), S136–S144 (2002)
5. Wang, H., Wang, W., Yang, J., Yu, P.: Clustering by pattern similarity in large data sets. In: Proc. 2002 ACM SIGMOD Int'l Conf. Management of Data, pp. 394–405 (2002)
6. Getz, G., Levine, E., Domany, E.: Coupled two-way clustering analysis of gene microarray data. Proc. Natl. Acad. Sci. USA 97 22, 12079–12084 (2000)
7. Tang, C., Zhang, L., Zhang, I., Ramanathan, M.: Interrelated two-way clustering: An unsupervised approach for gene expression data analysis. In: Proc. Second IEEE Int'l Symp. Bioinformatics and Bioeng., pp. 41–48 (2001)
8. Duffy, D., Quiroz, A.: A permutation based algorithm for block clustering. J. Classif. 8, 65–91 (1991)
9. Cho, H., Dhillon, I., Guan, Y., Sra, S.: Minimum Sum-Squared Residue Co-clustering of Gene Expression Data. In: Berry, M., Dayal, U. (eds.) Proceedings of the 4th SIAM Int'l Conf. Data Mining (2004)
10. Yang, J., Wang, W., Wang, H., Yu, P.: δ-clusters: Capturing subspace correlation in a large data set. In: Proc. 18th IEEE Int'l Conf. Data Eng., pp. 517–528 (2002)
11. Yang, J., Wang, W., Wang, H., Yu, P.: Enhanced biclustering on expression data. In: Proc. Third IEEE Conf. Bioinformatics and Bioeng., pp. 321–327 (2003)
12. Klugar, Y., Basri, R., Chang, J., Gerstein, M.: Spectral biclustering of microarray data: Coclustering genes and conditions. Genome Res. 13, 703–716 (2003)
13. Segal, E., Taskar, B., Gasch, A., Friedman, N., Koller, D.: Rich probabilistic models for gene expression. Bioinformatics 17(suppl. 1), S243–S252 (2001)
14. Sheng, Q., Moreau, Y., Moor, B.D.: Biclustering microarray data by gibbs sampling. Bioinformatics 19(suppl. 2), ii196–ii205 (2003)
15. Manjunath Aradhya, V.N., Masulli, F., Rovetta, S.: A Novel Approach for Biclustering Gene Expression Data Using Modular Singular Value Decomposition. In: Masulli, F., Peterson, L.E., Tagliaferri, R. (eds.) CIBB 2009. LNCS, vol. 6160, pp. 254–265. Springer, Heidelberg (2010)
16. Bryan, K., Cunningham, P., Bolshakova, N.: Application of simulated annealing to the biclustering of gene expression data. IEEE Trans. Inf. Technol. Biomed. 10(3), 519–525 (2006)

17. Mitra, S., Banka, H.: Multi-objective evolutionary biclustering of gene expression data. Pattern Recogn. 39, 2464–2477 (2006)
18. Dharan, S., Nair, A.: Biclustering of gene expression data using reactive greedy randomized adaptive search procedure. BMC Bioinformatics 10(suppl. 1), S27 (2009)
19. Tanay, A., Sharan, R., Shamir, R.: Biclustering Algorithms: A Survey. In: Aluru, S. (ed.) Handbook of Computational Molecular Biology. Computer and Information Science Series. S. Chapman & Hall/CRC (2005)
20. Peeters, R.: The maximum edge biclique problem is NP-Complete. Discrete Appl. Math. 131(3), 651–654 (2003)
21. Feo, T., Resende, M.: A probabilistic heuristic for a computationally difficult set covering problem. Oper. Res. Lett. 8, 67–71 (1989)
22. Feo, T., Resende, M.: Greedy randomized adaptive search procedures. J. Global Optim. 6, 109–133 (1995)
23. Festa, P., Resende, M.: GRASP: An annotated bibliography. In: Ribeiro, C., Hansen, P. (eds.) Essays and Surveys on Metaheuristics, pp. 325–367. Kluwer Academic Publishers (2002)
24. Festa, P., Resende, M.: An annotated bibliography of GRASP – Part I: Algorithms. International Transactions in Operational Research 16(1), 1–24 (2009)
25. Festa, P., Resende, M.: An annotated bibliography of GRASP – Part II: Applications. International Transactions in Operational Research 16(2), 131–172 (2009)
26. Prais, M., Ribeiro, C.: Reactive GRASP: An application to a matrix decomposition problem in TDMA traffic assignment. INFORMS J. Comput. 12, 164–176 (2000)
27. Binato, S., Oliveira, G.: A Reactive GRASP for transmission network expansion planning. In: Ribeiro, C., Hansen, P. (eds.) Essays and Surveys on Metaheuristics, pp. 81–100. Kluwer Academic Publishers (2002)
28. Delmaire, H., Díaz, J., Fernández, E., Ortega, M.: Reactive GRASP and tabu search based heuristics for the single source capacitated plant location problem. INFOR 37, 194–225 (1999)
29. Tavazoie, S., Hughes, J., Campbell, M.J., Cho, R.J., Church, G.M.: Systematic determination of genetic network architecture. Nat. Genet. 22, 281–285 (1999)
30. Alizadeh, A., Eisen, M., Davis, R., Ma, C., Lossos, I., Rosenwald, A., Boldrick, J., Sabet, H., Tran, T., Yu, X., Powell, J., Yang, L., Marti, G., Moore, T., Hudson, J., Lu, L., Lewis, D., Tibshirani, R., Sherlock, G., Chan, W., Greiner, T., Weisenburger, D., Armitage, J., Warnke, R., Levy, R., Wilson, W., Grever, M., Byrd, J., Botstein, D., Brown, P., Staudt, L.: Distinct types of diffuse large b-cell lymphoma identified by gene expression profiling. Nature 403, 503–511 (2000)
31. `http://www.yeastgenome.org/cgi-bin/GO/goTermFinder`
32. Mi, H., Dong, Q., Muruganujan, A., Gaudet, P., Lewis, S., Thomas, P.: PANTHER version 7: improved phylogenetic trees, orthologs and collaboration with the gene ontology consortium. Nucleic Acids Res. 38, D204–D210 (2010)
33. Frinhani, R.M.D., Silva, R.M.A., Mateus, G.R., Festa, P., Resende, M.G.C.: GRASP with Path-Relinking for Data Clustering: A Case Study for Biological Data. In: Pardalos, P.M., Rebennack, S. (eds.) SEA 2011. LNCS, vol. 6630, pp. 410–420. Springer, Heidelberg (2011)
34. Laguna, M., Martí, R.: GRASP and path relinking for 2-layer straight line crossing minimization. INFORMS J. Comput. 11, 44–52 (1999)
35. Festa, P., Pardalos, P., Resende, M., Ribeiro, C.: Randomized heuristics for the MAX-CUT problem. Optim. Methods Softw. 7, 1033–1058 (2002)
36. Mladenović, N., Hansen, P.: Variable neighborhood search. Comput. Oper. Res. 24, 1097–1100 (1997)