Abstract - We present a combined use of Genetic Algorithms (GAs) and column generation to approximately solve graph-coloring problems. The proposed method is divided in two phases. The constructive phase builds the initial pool of columns using a Constructive Genetic Algorithm (CGA). Each column forms an independent set. The second phase solves by column generation the set covering formulation. The columns are generated solving weighted independent set problems. Computational experience is given using the CGA alone and also combined with the column generation process.

Key words: Column generation, Constructive Genetic Algorithm, Graph Coloring.

I. INTRODUCTION

Let G = (V,E) be an undirected graph. A k-coloring of G is a partition of V into k subsets Ci, i = 1, ..., k, such that no adjacent vertices belong to the same subset. The Graph-Coloring Problem is to find k-coloring of G with k as small as possible. This optimal value of k corresponds to the chromatic number of G. It is well known that this problem is NP-hard [Garey and Johnson (1978)], and heuristics must be used for large graphs. Each vertex subset is
an independent vertex set, and the coloring problem could be seen as a clustering problem to form independent vertex sets.


The Genetic Algorithms (GA) are very well known, having successful applications in Combinatorial Optimization problems [Boers, E.J.W. et al.]. The GA is based on the controlled evolution of a structured population. The basis of a GA is the recombination and the schema formation and propagation over generations [De Jong (1975), Goldberg (1989), Holland (1975)].

Lorena and Furtado (2000) and Ribeiro Filho (1997) recently proposed the Constructive Genetic Algorithm (CGA) approach. The schemata theory was for a long time the central point in classical GA, but has been less explored in recent years. A typical CGA uses not only complete problem solutions (structures), but also solution parts, known as schemata. The algorithm works with an initial population formed only by schemata. A simple schema is not enough to represent a feasible solution for the coloring problem, as some vertices are not colored. New schemata or structures are generated by recombination. A double fitting process is used to evaluate schemata adaptation and good schemata are preserved. An evolution parameter eliminates schemata that do not satisfy a permanence criterion and the best structure found so far is kept. The process finishes with an empty population or when an iteration limit is reached.
A column generation approach to graph coloring was early studied by Mehrotra and Trick (1995) to provide lower bounds to the chromatic number. The column generation was embedded on a tree search and solves successfully small to moderate size problems.

This paper is organized as follows. Section II presents the modeling of graph coloring to apply CGA, describing the representation of schemata and structures, schemata selection and recombination, and structure mutation. A CGA pseudo-code is also presented. Section III outlines the column generation process and how it combines with CGA. Computational results are presented in the next section.

II. CGA REVIEW

The CGA is proposed to address the problem of evaluating schemata and structures in a common basis. While in the other evolutionary algorithms, evaluations of individuals are based on a single function (the fitness function), in CGA this process relies on two functions, mapping the space of structures and schemata onto $\mathbb{R}_+$.

A. Representation

The representation for structures and schemata uses three symbols. These symbols are: the “do not care” symbol, indicating the vertices which are not assigned to any cluster; a symbol to indicate the vertex is a “seed” to form a cluster; and a third symbol indicating the vertices assigned to some cluster. The number of seed vertices is exactly the number of colors being used, or clusters being formed. The objective is to find clusters composed of independent sets of vertices. The vertex-to-cluster assignment must be made by an appropriate heuristic.
The vertex-to-cluster assignment uses an adaptation of a heuristic known as Recursive Large First (RLF) (Leighton, 1979) that has been compared to others and considered of good computational performance. This can be better understood using an example. Suppose we are looking for a 3-coloring for a graph with ten vertices and the following adjacency matrix:

```
0111000000
1000010000
1001010000
1010101100
0010101100
0010000100
0101000001
0011000100
0000100100
0000001000
```

Let’s consider the following sets:

- $C_i$ is the set of vertices in the $i$-th cluster,
- $U_i$ is the set of all schema vertices adjacent to any vertex in $C_i$,
- $V_{sch}$ is the set of all the schema vertices, and
- $V_i$ is $V_{sch} - U_i$.

And the following schema: (#,1,0,1,#,0,0,#,1,0). Where: 1 = seed vertex, 0 = vertex to be assigned and # = vertex not to be assigned. So, initially we have:

- $C_1 = \{2\}$, $C_2 = \{4\}$, $C_3 = \{9\}$
- $V_1 = \{3,6,10\}$, $V_2 = \{6,10\}$, $V_3 = \{3,6,7,10\}$
- $U_1 = \{7\}$, $U_2 = \{3,7\}$, $U_3 = \{\}$

Now, take the vertex $v$ in $V_i$, $i=1,2,3$ with the largest degree in $U_i$, $i=1,2,3$ and assign $v$ to $C_i$. Then, update the sets $C_i$, $V_i$, and $U_i$. We obtain:
\[ C1 = \{2,10\} \quad C2 = \{4\} \quad C3 = \{9\} \]

\[ V1 = \{3,6\} \quad V2 = \{6\} \quad V3 = \{3,6,7\} \]

\[ U1 = \{7\} \quad U2 = \{3,7\} \quad U3 = \{\} \]

Repeating the process we have:

\[ C1 = \{2,10\} \quad C2 = \{4,6\} \quad C3 = \{9\} \]

\[ V1 = \{3\} \quad V2 = \{\} \quad V3 = \{3,7\} \]

\[ U1 = \{7\} \quad U2 = \{3,7\} \quad U3 = \{\} \]

The process continues until all sets \( V_i \) are empty. At the end, in this example we will have the following clusters:

\[ C1 = \{2,3,10\} \quad C2 = \{4,6\} \quad C3 = \{7,9\} \]

An initial population is created with \( n \) schemata randomly generated, with 20% of labels 0, exactly \( k \) labels 1 and the remaining labels receiving the indetermination \( # \). The population will be increased by recombination but remains composed exclusively of schemata.

\[ B. \text{ CGA modeling} \]

Let \( X \) be the set of all structures and schemata that can be generated by the 0-1-# string representation of section 2.A., and consider two functions \( f \) and \( g \), defined as \( f : X \to \mathbb{R}^+ \) and \( g : X \to \mathbb{R}^+ \) such that \( f(s) \leq g(s) \), for all \( s \in X \). We define the double fitness evaluation of a structure or schema \( s \), due to functions \( f \) and \( g \), as \( fg\text{-fitness} \).
The CGA optimization problem implements the fg-fitness with the following two objectives:

- *(interval minimization)* Search for \(s_i \in X\) of minimal \(\{g(s_i) - f(s_i)\}\), and

- *(g maximization)* Search for \(s_i \in X\) of maximal \(g(s_i)\).

To attain these purposes, a problem to be solved using CGA is modeled as the following Bicriterion Optimization Problems (BOP):

\[
\begin{align*}
\text{Min} & \quad \{g(s_i) - f(s_i)\} \\
\text{Max} & \quad g(s_i) \\
\text{subject to} & \quad g(s_i) \geq f(s_i) \\
& \quad \forall s_i \in X
\end{align*}
\]

Considering the schema representation, the fg-fitness evaluation increases as the number of labels \# decreases, and therefore functions \(f\) and \(g\) must be designed such that structures will have higher fg-fitness evaluation than schemata. Functions \(f\) and \(g\) also must be properly identified to represent optimization objectives of the problem at issue.

C. The fg-fitness

For the coloring problem the fg-fitness functions used are respectively

\[
g(s_i) = \sum_{p=1}^{k} \left|\left|C_{ip} - 1\right|\right| C_{ip} / 2, \quad \text{and}
\]
\[ f(s_i) = g(s_i) - \sum_{p=1}^{k} |E(C_{ip})|. \]

Where \( k \) is a pre-fixed number of colors, \( C_{ip} \) is the set of vertices receiving the color \( p \) on schema \( s_i \) (the notation \( |C_{ip}| \) is the number of vertices in set \( C_{ip} \)), and \( E(C_{ip}) \) is the set of edges with both terminal vertices in \( C_{ip} \). The expression \( \frac{(|C_{ip}|-1)|C_{ip}|}{2} \) gives the number of edges of a complete graph with \( |C_{ip}| \) vertices.

Function \( g(s_i) \) can be interpreted as the total number of edges if \( k \) complete graphs of sizes \( |C_{ip}| \) are considered.

Function \( f(s_i) \) decreases this number by the number of edges actually linking vertices on the sets \( C_{ip} \). When \( f(s_i) = g(s_i) \) the \( k \) sets \( C_{ip} \) are independent sets.

**D. The evolution process**

The set \( X \) defined in section 2.A. is not completely known and the problem BOP can not be directly considered. Instead, an evolution process is conducted to attain the objectives (interval minimization and \( g \) maximization) of the BOP. At the beginning of the process, the following two expected values are given to these objectives. A non-negative real number \( g_{\text{max}} > \text{Max}_{s_i \in X} g(s_i) \), that is an upper bound to \( g(s_i) \), for each \( s_i \in X \), and the interval length \( d g_{\text{max}} \), obtained from \( g_{\text{max}} \) using a real number \( 0 < d \leq 1 \).

The upper bound for coloring problems is \( g_{\text{max}} = \text{mult}.k\left[\frac{\left\lfloor \frac{n}{k} \right\rfloor - 1}{2}\right] \left\lfloor \frac{n}{k} \right\rfloor \). To obtain this bound, first is considered to divide the number of vertices \( n \) in \( k \) sets with approximately the same number of elements (the
expression \( \lceil n/k \rceil \) gives the large integer smaller than \( n/k \), then the same procedure used for \( g(s_i) \) is applied, where the positive integer \( \text{mult} \) is considered to certify that \( g_{\text{max}} > \max_{s_i \in S} g(s_i) \).

The evolution process is then conducted considering an adaptive rejection threshold, which contemplates both objectives in BOP. Given a parameter \( \alpha \geq 0 \), the expression

\[
g(s_i) - f(s_i) \geq d g_{\text{max}} - \alpha. d[g_{\text{max}} - g(s_i)]
\]  

(2.1)

presents a condition for rejection from the current population of a schema or structure \( s_i \).

The right hand side of (2.1) is the threshold, composed of the expected value to the interval minimization \( d g_{\text{max}} \), and the measure \( [g_{\text{max}} - g(s_i)] \), that shows the difference of \( g(s_i) \) and \( g_{\text{max}} \) evaluations. For \( \alpha = 0 \), (2.1) is equivalent to comparing the interval length obtained by \( s_i \) and the expected length \( d g_{\text{max}} \). Schemata are discarded if expression (2.1) is satisfied. When \( \alpha > 0 \), schemata with large number of labels \#'s have higher possibility of being discarded than the ones with small number of labels \#'s, as they present, in general, smaller differences \( [g_{\text{max}} - g(s_i)] \).

Parameter \( \alpha \) is related to time in the evolution process. Considering that the good schemata need to be preserved for recombination, the evolution parameter \( \alpha \) starts from 0, and then increases slowly, in small time intervals, from generation to generation. The population at the evolution time \( \alpha \), denoted by \( P_{\alpha} \), is dynamic in size according to the value of the adaptive parameter \( \alpha \), and can be emptied during the process.
The parameter $\alpha$ is now isolated in expression (2.1), thus yielding the following expression and corresponding rank to $s_i$: 

$$\alpha \geq \frac{d g_{\text{max}} - \left[g(s_i) - f(s_i)\right]}{d \left[g_{\text{max}} - g(s_i)\right]} = \delta(s_i).$$

At the time they are created, the schemata receive their corresponding rank value $\delta(s_i)$. The rank of each schema is compared with the current evolution parameter $\alpha$. At the moment a schema is created, it is then possible to have some figure of its survivability. The higher the value of $\delta(s_i)$, and better is the schema to the BOP, and they also have more surviving and recombination time.

### E. Selection and recombination

The population was kept in a non-decreasing order according to the following key

$$\Delta(s_i) = (1 + d_i)(n - n_\#),$$

where

$$d_i = \frac{g(s_i) - f(s_i)}{g(s_i)},$$

$n_\#$ is the number of labels #’s in $s_i$. Schemata with small $n_\#$ and/or presenting small $d_i$ are better and appear in first order positions.

The method used for selection takes one schema from the n first positions in the population (base) and the second schema from the whole population (guide). Before recombination, the first schema is complemented to generate a structure representing a feasible solution, i.e., all #’s are replaced by 0’s. This structure suffers mutation and is compared to the best solution found so far (which is kept throughout the process). The recombination merges information from both selected schemata, but preserves the number of labels 1 (number of colors) in the new generated schema or structure. The recombination is best described as (to be executed in this order)
Recombination

\[
\begin{align*}
\text{if } s_{\text{base}}(j) = s_{\text{guide}}(j) \text{ then } s_{\text{new}}(j) &\leftarrow s_{\text{base}}(j) \\
\text{if } s_{\text{guide}}(j) = \# \text{ then } s_{\text{new}}(j) &\leftarrow s_{\text{base}}(j) \\
\text{if } s_{\text{base}}(j) = \# \text{ or } 0 \text{ and } s_{\text{guide}}(j) = 1 \text{ then } &s_{\text{new}}(j) \leftarrow 1 \text{ and } s_{\text{new}}(i) \leftarrow 0 \text{ for some } s_{\text{new}}(i) = 1 \\
\text{if } s_{\text{base}}(j) = 1 \text{ and } s_{\text{guide}}(j) = 0 \text{ then } &s_{\text{new}}(j) \leftarrow 0 \text{ and } s_{\text{new}}(i) \leftarrow 1 \text{ for some } s_{\text{new}}(i) = 0
\end{align*}
\]

At each generation, exactly n new individuals are created by recombination. If a new individual does not represent a feasible solution (is a schema), then it is inserted into the population; otherwise it suffers mutation and is compared to the best solution found so far. The following pseudo-code describes the mutation process:

**Mutation Process**

1: For each cluster
   Move the seed to the vertex with the largest degree in the cluster
   Re-assign the vertices using the RLF approach
   Count conflicts and save the best in this loop
2: If the best found in the loop above is better than the original solution
   Replace the original by this best and return to pass 1
   Else
   Stop.

\[F. \text{ The algorithm}\]

The Constructive Genetic Algorithm can be summed up by the pseudo-code:
Given $g_{\text{max}}$ and $d$;

$\alpha := 0$;

$\varepsilon := 0.05$; \hspace{1cm} \{ time interval \}

Initialize $P_\alpha$; \hspace{1cm} \{ initial population \}

Evaluate $P_\alpha$; \hspace{1cm} \{ fg-fitness \}

For all $s_i \in P_\alpha$ compute $\delta(s_i)$ \hspace{1cm} \{ rank computation \}

end_for

While (not stop condition) do

For all $s_i \in P_\alpha$ satisfying $\alpha < \delta(s_i)$ do \hspace{1cm} \{ evolution test \}

$\alpha := \alpha + \varepsilon$;

Select $P_\alpha$ from $P_{\alpha - \varepsilon}$; \hspace{1cm} \{ reproduction operator \}

Recombine $P_\alpha$; \hspace{1cm} \{ recombination operators \}

Evaluate $P_\alpha$; \hspace{1cm} \{ fg-fitness \}

end_for

For all new $s_i \in P_\alpha$ compute $\delta(s_i)$ \hspace{1cm} \{ rank computation \}

end_for

end_while

---

III. THE COLUMN GENERATION

The use of column generation for coloring problems was first proposed by Mehrotra and Trick (1995). In a classical fashion of column generation, the algorithm iterates between a column generation sub-problem and a restricted master problem. Solving the master problem yields a certain dual solution, which is used in the sub-problem to determine whether there is any column that might be an incoming column. The master problem (MP) is

\[
\begin{align*}
\text{Min} & \quad \sum_{j \in \mathcal{J}} x_j \\
\text{Subject to} & \quad \sum_{j \in \mathcal{J}} a_{ij} x_j = 1, \quad i = 1, \ldots, |\mathcal{V}|, \\
& \quad x_j \in \{0,1\}.
\end{align*}
\]


Where \( S = \{ S_1, S_2, \ldots, S_m \} \), \( S_i \subseteq V \), is the set of all maximal independent sets of \( G \) and \( A = [ a_{ij} ]_{n \times m} \), is a matrix with \( a_{ij} = 1 \) if \( i \in S_j \), and \( a_{ij} = 0 \) otherwise.

It is a set-partitioning problem with a large number of (generally) unknown columns that are generated when necessary, solving the following weighted maximum independent set problem (WMIP)

\[
\text{Max } \sum_{i \in V} \lambda_i z_i \\
\text{Subject to } z_i + z_j \geq 1, \forall (i, j) \in E \\
z_i \in \{0,1\}, \forall i \in V.
\]

Where \( \lambda_i \) are dual variables for each constraint in MP.

A linear programming relaxation of MP (called \( MP_{LP} \)) will be solved by column generation. An initial pool of columns must be given to form the initial \( MP_{LP} \), and columns are elected to enter the \( MP_{LP} \) if they return bounds larger than 1 when solving WMIP. The CGA described in section II was used here to form the initial pool of columns to \( MP_{LP} \).  

Initially, it is set a number of colors, and if the CGA find this specified coloring, this number is reduced, until no more improvement is found. A number of independent sets found during the last CGA application is stored to compose the first pool of columns. The optimal solution of \( MP_{LP} \) may give a lower bound to the coloring problem, and the dual variables are saved to be used on problem WMIP.
In the sequence, the same CGA is used to approximately solve problem WMIP, setting the last used number of colors minus one, and storing the independent sets found. The new independent sets with bounds greater than 1 on problem WMIP are appended to the previous pool of columns and problem $MP_{LP}$ is resolved. The process continues until no more columns are found to be added to the $MP_{LP}$.

A lower bound to the coloring problem is obtained at each iteration applying the Farley´s bound (Farley, (1990)), given by $v(MP_{LP})/v(WMIP)$, where $v(.)$ is the optimal value of the corresponding problem. These values change at each process iteration.

IV. COMPUTATIONAL TESTS

Computational tests were made with several instances taken from different groups: book graphs (Anna, David, Huck and Jean - each vertex represents a character and two vertices are connected if the corresponding characters encounter each other in the book); game graphs (Games120 – each vertex represents a team and two vertices are connected if they played each other during the season); miles graphs (Miles250, Miles500 and Miles750 – vertices representing cities are linked if the cities are close enough); register graphs (Musol_1, Musol_2, Zeroin_1 and Zeroin_2 - based on register allocation for variables in program code); Mycielski graphs (Myciel5, Myciel6 and Myciel7 - graphs based on the Mycielski transformation); queen graphs (Queen55, 66, 77, 88 and 99 - a graph with $N^2$ vertices, each corresponding to a square in NxN chess board, and two vertices connected if the corresponding squares are in the same row, column or diagonal).

The table 1 bellow gives initial computational results. The table contains the number of instances, average numbers of vertices and edges, conflicts remaining after the CGA application for each instance group and computer times for
a C code running on a Pentium II - 266 MHz machine. All the experiments were made with three runs for each instance, and all of them for the optimal number of colors.

<table>
<thead>
<tr>
<th>Group</th>
<th>Instances</th>
<th>Vertices</th>
<th>Edges</th>
<th>CGA</th>
<th>Times (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Books</td>
<td>4</td>
<td>94.7</td>
<td>363.5</td>
<td>0</td>
<td>1.9</td>
</tr>
<tr>
<td>Games</td>
<td>1</td>
<td>120</td>
<td>638</td>
<td>0</td>
<td>3.0</td>
</tr>
<tr>
<td>Miles</td>
<td>3</td>
<td>128</td>
<td>1223.3</td>
<td>0</td>
<td>17.4</td>
</tr>
<tr>
<td>Register</td>
<td>4</td>
<td>204.8</td>
<td>3848.6</td>
<td>0</td>
<td>508.9</td>
</tr>
<tr>
<td>Mycielski</td>
<td>3</td>
<td>111</td>
<td>1117</td>
<td>0</td>
<td>3.3</td>
</tr>
<tr>
<td>Queen</td>
<td>5</td>
<td>51</td>
<td>753.2</td>
<td>0.5</td>
<td>1021.5</td>
</tr>
</tbody>
</table>

Table 1: CGA computational results

The quality of the results can be easily seen, especially for the queen graphs, considered hard. Problem Queen99 was the only one for which the chromatic number was not reached by the CGA.

Laguna and Martí (1999) also reported computational results for the same instances (excluding the Queen class) applying a GRASP algorithm, and compare the results with other known heuristics. Table 2 shows the comparison made and includes an extra column referring to the CGA application to the same instances. The instances book, games and miles are grouped and named as SGB, and a larger number of instances is investigated. Columns show the number of instances, average numbers of vertices and edges, the average chromatic numbers, and the averaged minimum number of colors resulted after the application of CGA and the following heuristics: GRASP (Laguna e Martí, 1999); TABUCOL (Hertz e de Werra, 1987) and Simulated Annealing (SA) (Johnson et al., 1991). Times are
not reported as the computational tests are performed on distinct machines. The CGA reproduced all the GRASP best results and can be conjectured as an efficient heuristic.

<table>
<thead>
<tr>
<th>Group</th>
<th>Instances</th>
<th>Vertices</th>
<th>Edges</th>
<th>Colors</th>
<th>TABUCOL</th>
<th>SA</th>
<th>GRASP</th>
<th>CGA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mycielski</td>
<td>5</td>
<td>73.4</td>
<td>688.4</td>
<td>6.0</td>
<td>6.0</td>
<td>6.0</td>
<td>6.0</td>
<td>6.0</td>
</tr>
<tr>
<td>Register</td>
<td>14</td>
<td>362.1</td>
<td>7608.1</td>
<td>37.4</td>
<td>57.1</td>
<td>40.1</td>
<td>37.4</td>
<td>37.4</td>
</tr>
<tr>
<td>SGB</td>
<td>10</td>
<td>113.9</td>
<td>2835.2</td>
<td>22.6</td>
<td>24.1</td>
<td>26.0</td>
<td>22.7</td>
<td>22.7</td>
</tr>
</tbody>
</table>

Table2: CGA comparison with other heuristics

Then we have proceeded with tests for column generation using the Queen99 instance. Table 3 shows the results.

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Number of colors</th>
<th>Best CGA number of conflicts</th>
<th>$MP_{LP}$ bound</th>
<th>Farley’s bound</th>
<th>Time(sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>10</td>
<td>2</td>
<td>9.226</td>
<td>8.359</td>
<td>295</td>
</tr>
<tr>
<td>1</td>
<td>9</td>
<td>10</td>
<td>9.059</td>
<td>8.155</td>
<td>283</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>25</td>
<td>9.007</td>
<td>8.542</td>
<td>246</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>47</td>
<td>9.000</td>
<td>-</td>
<td>177</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>75</td>
<td>-</td>
<td>-</td>
<td>121</td>
</tr>
</tbody>
</table>

Table3: Column generation process for Queen99
The times in table 3 correspond to the CGA application. The CPLEX 6.5 [ILOG (1999)] solves the $MP_{LP}$ problem very fast, and their times are not reported. The best solution found by the CGA was 11 colors. Considering the best $MP_{LP}$ and Farley’s bounds, it can be conjectured that the best number of colors to Queen99 is 9, 10 or 11 (actually the best number is 10). As the number of colors decreases, the number of conflicts in CGA solutions increases, but the new columns improve the $MP_{LP}$ bounds. The Farley’s bound have an oscillating behavior due to the fact that problem WMIP was not exactly solved. We have used a SUN-ULTRA30 and the CGA parameters was $iteration\_limit=20$, $\epsilon = 0.01$, $d = 0.15$ and $mult = 2.0$.

V. CONCLUSION

The graph-coloring problem is an important and very studied problem. We presented in this paper a new evolutionary approach to this problem. The Constructive Genetic Algorithm treats the problem as a clustering problem, introducing some new aspects, such as the direct evaluation of schemata and the use of a known graph coloring heuristic as a part of the individuals representation.

The CGA was also used as an auxiliary tool to column generation. The initial pool of columns and the extra columns generated uses the CGA in an original fashion. The process gives upper and lower bounds to the chromatic number, that can be useful on tree search algorithms.

The computational tests on instances available in literature have show good results to the CGA application when compared with other approaches and have opened a view of integrating evolutionary algorithms to well accepted mathematical programming approaches, like the column generation process.

Acknowledgments: The second author acknowledges Conselho Nacional de Desenvolvimento Científico e Tecnológico - CNPq (proc. 300837/89-5) and Fundação para o Amparo a Pesquisa no Estado de S. Paulo - FAPESP (proc. 99/06954-7) for partial financial support.
REFERENCES