Polynomial-time Local Improvement Algorithm for Consecutive Block Minimization

S. Haddadi[1], S. Chenche[1], M. Cheraitia[1], and F. Guessoum[1]

LabSTIC
8 Mai 1945 University
BP 401, 24000 Guelma, Algeria
http://www.univ-guelma.dz/recherche/labs/labstic/index.html
salim.haddadi@yahoo.com
sara.chenche@hotmail.fr
meryem.cheraitia@hotmail.fr
fatima_guessoum@yahoo.fr

1 Introduction

Given a binary matrix, a block of consecutive ones (bco for short) is any maximal sequence of consecutive ones occurring in the same row. Consecutive block minimization (CBM) seeks for a permutation of the columns of the binary matrix so as to minimize the number of bco's. CBM is well known to be NP-hard. In view of this negative result, it would be necessary for dealing with the large instances arising in practice, to adopt a heuristic approach. This is what we do by proposing a polynomial-time local improvement algorithm. An empirical analysis of the algorithm is performed on a large set of randomly generated instances, as well as on five real-world instances.

2 The local improvement heuristic

Regarding CBM, we are not confronted with any "feasibility" problem. We are just seeking a permutation of the columns of the binary matrix that minimizes the number of bco's, and any arbitrary permutation can serve as a starting point for improvement. Let us prove some preliminary lemmas.

Lemma 1. *Let B be a binary $m \times 3$-matrix. The number of occurrences of the row sequences* 101 *or* 010 *is*

$$\sum_{k=1}^{k=m} \left(B_2^k - B_1^k \times B_2^k - B_2^k \times B_3^k + B_1^k \times B_3^k\right) \tag{1}$$

Lemma 2. *If we insert column j between the columns i and $i+1$ in a binary $m \times n$-matrix B we create δ new bco's where*

$$\delta = \sum_{k=1}^{k=m} \left(B_j^k - B_i^k \times B_j^k - B_j^k \times B_{i+1}^k + B_i^k \times B_{i+1}^k\right) \tag{2}$$

Lemma 3. *If we remove column j from between the columns $j-1$ and $j+1$ in B we remove δ bco's where*

$$\delta = \sum_{k=1}^{k=m} \left(B_j^k - B_{j-1}^k \times B_j^k - B_j^k \times B_{j+1}^k + B_{j-1}^k \times B_{j+1}^k\right) \tag{3}$$

We consider two different ways for improving π (by decreasing the number of bco's): either by interchanging two distinct columns or by shifting a single column.

2.1 Improvement by interchange

Suppose that we are presently inspecting the matrix A_π associated to some permutation π such that the number of bco's is σ. We consider the neighborhood $\mathcal{N}(\pi)$ to be the set of all the permutations that result from π by interchanging two columns. We explore $\mathcal{N}(\pi)$ searching for a permutation giving a smaller number of bco's. If no such permutation exists, the procedure ends. When an improvement δ is achieved via an interchange of columns $\pi(i)$ and $\pi(j), i \neq j$ (call this new permutation π'), we update $\sigma \longleftarrow \sigma - \delta, \pi'(i) \longleftarrow \pi(j), \pi'(j) \longleftarrow \pi(i)$, and $\pi'(k) \longleftarrow \pi(k), k \neq i, k \neq j$. The entire process is repeated with π'.

Suppose that we interchange columns $\pi(i)$ and $\pi(i+1)$. Consider first the special case where $\pi(i)$ and $\pi(j)$ are adjacent (i.e. $j = i+1$).

Lemma 4. *There is an improvement by interchanging columns $\pi(i)$ and $\pi(i+1)$ if and only if $\delta > 0$ where δ is computed below. Furthermore, the resulting number of bco's by interchanging the two columns is $\sigma - \delta$.*

$$\delta^+ = \sum_{k=1}^{k=m} \left(A^k_{\pi(i-1)} \times A^k_{\pi(i+1)} + A^k_{\pi(i)} \times A^k_{\pi(i+2)}\right) \tag{4}$$

$$\delta^- = \sum_{k=1}^{k=m} \left(A^k_{\pi(i-1)} \times A^k_{\pi(i)} + A^k_{\pi(i+1)} \times A^k_{\pi(i+2)}\right) \tag{5}$$

$$\delta = \delta^+ - \delta^- \tag{6}$$

Consider now the general case where columns $\pi(i)$ and $\pi(j)$ are non adjacent (i.e. there exists at least one distinct column separating them).

Lemma 5. *There is an improvement by interchanging non adjacent columns $\pi(i)$ and $\pi(j)$ if and only if $\delta > 0$ where δ is computed below. Furthermore, the resulting number of bco's is $\sigma - \delta$.*

$$\delta_1^+ = \sum_{k=1}^{k=m} \left(A^k_{\pi(i-1)} \times A^k_{\pi(j)} + A^k_{\pi(j)} \times A^k_{\pi(i+1)}\right) \tag{7}$$

$$\delta_2^+ = \sum_{k=1}^{k=m} \left(A^k_{\pi(j-1)} \times A^k_{\pi(i)} + A^k_{\pi(i)} \times A^k_{\pi(j+1)}\right) \tag{8}$$

$$\delta_1^- = \sum_{k=1}^{k=m} \left(A^k_{\pi(i-1)} \times A^k_{\pi(i)} + A^k_{\pi(i)} \times A^k_{\pi(i+1)}\right) \tag{9}$$

$$\delta_2^- = \sum_{k=1}^{k=m} \left(A^k_{\pi(j-1)} \times A^k_{\pi(j)} + A^k_{\pi(j)} \times A^k_{\pi(j+1)}\right) \tag{10}$$

$$\delta = \delta_1^+ + \delta_2^+ - \delta_1^- - \delta_2^- \tag{11}$$

Lemma 6. *The complexity of the interchange procedure is $O\left(mn^2 (f-m)\right)$ where f is the number of nonzero entries in A.*

Unfortunately, we cannot take advantage of the sparsity of the binary matrix as we may presume. The 0's are as important as the 1's for the computations in (4-6) and (7-10).

2.2 Improvement by shifting

We consider the neighborhood $\mathcal{N}'(\pi)$ to be the set of all the permutations that result from π by shifting a single column, which means that the column in question leaves its position and is inserted elsewhere between two other columns. The set $\mathcal{N}'(\pi)$ is scanned searching for a permutation giving a smaller number of bco's. If no such permutation exists, the procedure ends. Suppose that an improvement δ is achieved by shifting column $\pi(i)$, and let π' be the resulting permutation. In the case of shifting, the necessary updates are a bit trickier. They depend on whether the value of

$\pi(i)$ is less or greater than its future position since we have to move several columns. The detailed updates are postponed until the statement of the pseudo-code. When all the necessary updates are performed, we repeat the procedure by considering π'. The proofs of the remaining claims are similar to the proofs of the previous section.

If $\pi(i) > \pi(j)$, we shift $\pi(i)$ between columns $\pi(j-1)$ and $\pi(j)$ (i.e. $\pi(i)$ is removed from between columns $\pi(i-1)$ and $\pi(i+1)$ and is inserted between columns $\pi(j-1)$ and $\pi(j)$).

Lemma 7. *There is an improvement by shifting column $\pi(i)$ between columns $\pi(j-1)$ and $\pi(j)$ if and only if $\delta > 0$ where δ is computed in (12-14)*

$$\delta^{+} = \sum_{k=1}^{k=m} \left(-A^{k}_{\pi(i-1)} \times A^{k}_{\pi(i)} - A^{k}_{\pi(i)} \times A^{k}_{\pi(i+1)} + A^{k}_{\pi(i-1)} \times A^{k}_{\pi(i+1)}\right) \tag{12}$$

$$\delta^{-} = \sum_{k=1}^{k=m} \left(-A^{k}_{\pi(j-1)} \times A^{k}_{\pi(i)} - A^{k}_{\pi(i)} \times A^{k}_{\pi(j)} + A^{k}_{\pi(j-1)} \times A^{k}_{\pi(j)}\right) \tag{13}$$

$$\delta = \delta^{+} - \delta^{-} \tag{14}$$

If $\pi(i) < \pi(j)$, we shift $\pi(i)$ between columns $\pi(j)$ and $\pi(j+1)$.

Lemma 8. *There is an improvement by shifting column $\pi(i)$ between columns $\pi(j)$ and $\pi(j+1)$ if and only if $\delta > 0$ where δ is computed in (15-17)*

$$\delta^{+} = \sum_{k=1}^{k=m} \left(-A^{k}_{\pi(i-1)} \times A^{k}_{\pi(i)} - A^{k}_{\pi(i)} \times A^{k}_{\pi(i+1)} + A^{k}_{\pi(i-1)} \times A^{k}_{\pi(i+1)}\right) \tag{15}$$

$$\delta^{-} = \sum_{k=1}^{k=m} \left(-A^{k}_{\pi(j)} \times A^{k}_{\pi(i)} - A^{k}_{\pi(i)} \times A^{k}_{\pi(j+1)} + A^{k}_{\pi(j)} \times A^{k}_{\pi(j+1)}\right) \tag{16}$$

$$\delta = \delta^{+} - \delta^{-} \tag{17}$$

Invoking similar arguments as previously, we prove that the complexity of the shifting procedure is $O\left(mn^{2}(f-m)\right)$.

3 Computational experience

We experimented the local-improvement heuristic on a large number of real-world, as well as randomly generated, instances. Computational experience shows that the proposed algorithm constitutes a viable heuristic method for CBM, especially in the lack of existing methods. One direction for future research emerges. Using the two local improvement procedures, the design of a meta-heuristic can help escaping the local optimum and continuing the improvement process.