Simulated Annealing with a Temperature Dependent Penalty Function

JULIO M. STERN / School of Operations Research and Industrial Engineering, Cornell University, Ithaca, NY 14853, jstern@cs.cornell.edu and the Computer Science Department of the University of São Paulo, Brazil, jstern@ime.usp.br

(Received: June 1990; final revision received: December 1990; accepted: November 1991)

We formulate the problem of permuting a matrix to block angular form as the combinatorial minimization of an objective function. We motivate the use of simulated annealing (SA) as an optimization tool. We then introduce a heuristic temperature dependent penalty function in the simulated annealing cost function, to be used instead of the real objective function being minimized. Finally we show that this temperature dependent penalty function version of simulated annealing consistently outperforms the standard simulated annealing approach, producing, with smaller running times, better solutions. We believe that the use of a temperature dependent penalty function may be useful in developing SA algorithms for other combinatorial problems.

Reducing a matrix to block angular form (BAF) is important for a variety of decomposition methods. In Section 1 we formulate the BAF reduction problem as a combinatorial partitioning problem, where the cost of a given partition or state, cost(p), is the objective function to be minimized.

In Section 2 we present a "standard" simulated annealing (SSA) algorithm to approximately solve this combinatorial problem. The SSA can be seen as a generalization of simulated annealing (SA) algorithms for the graph partitioning problem.\[14\]

In Section 3 we motivate the use of a heuristic cost function,

\[\text{cost}(p, \mu(t)) = \text{cost}(p) - \mu(t) \text{penalty}(p),\]

in the SSA, instead of the real objective function being minimized, cost(p). Parameter \(\mu\) is temperature dependent, and only in the zero temperature limit we have \(\mu = 0\) and \(\text{cost}(p, 0) = \text{cost}(p)\). The main reasons for using \(\text{cost}(p, \mu)\) are:

- it is only marginally more expensive to compute than \(\text{cost}(p)\);
- it helps us "sense" the proximity of low cost states;
- it breaks down massively degenerate states of \(\text{cost}(p)\) that can "trap" the SSA in nonoptimal metastable states.

In Sections 4 and 5 we report numerical experiments. In these experiments the temperature dependent penalty function simulated annealing (TPSA) consistently outperforms the SSA, producing much better solutions with smaller running times. In Section 5 we also relate the temperature dependent penalty function, that can be seen as perturbing the metric of the original problem, to other variants of the SA algorithm. Finally in Section 6 we compare SA to another heuristic to solve the BAF reduction problem.

1. The Block Angular Form

The Column Block Angular Form (CBAF) reduction problem is, by rows and columns permutations, \(P\) and \(Q\), to reduce a given matrix \(A, m \times n\), to CBAF; i.e., we want \(PAQ\) with \(b\) diagonal rectangular blocks, \(B_1, \ldots, B_b\), plus some residual columns \(C\) (Figure 1). This can also be seen as a hypergraph partition problem where we paint all nonzero elements (NZEs) of each vertex or row, \(i \in M = \{1, \ldots, m\}\), with a color \(p(i) \in B = \{1, \ldots, b\}\). The color \(q(j)\) of an edge or column \(j \in N = \{1, \ldots, n\}\), is then the set of all its NZE's colors, and multicolored edges of the hypergraph correspond to residual columns in the CBAF.

A more General Block Angular Form (GBAF), also allows some residual rows, \(B_r\), to remain uncolored, or to receive the color \(0\) (Figure 2). Finally the Row Block Angular Form (RBAF) only allows residual rows (Figure 3). Our interest in reducing a matrix to a BAF relates to techniques for sparse matrix computations,\[7, 21, 25, 26, 30\] and some other general decomposition methods. In these applications we always want:

1. Roughly the same number of rows in each block.
2. Only a few residual rows or columns.

From 1 and 2 it is natural to consider the minimization of the function

\[\text{cost}(p) = \alpha \sum_{k=1}^b \left[m/b - s(k)\right]^2 + \beta c(p) + \gamma r(p)\]

\[c(p) = |\{ j \in N : q(j) > 2 \}|\]

\[r(p) = |\{ i \in M : p(i) = 0 \}|\]

\[s(k) = |\{ i \in M : p(i) = k \}|\].

The first term in \(\text{cost}(p)\) measures the deviation of each block from the ideal size \(m/b\); \(c(p)\) is the number of residual columns, and \(r(p)\) is the number of residual rows.

In most applications one can view the diagonal blocks as

Subject classifications: Programming; integer; heuristics; mathematics: matrices; networks/graphs. heuristics.
2. Standard Simulated Annealing

The NP-hardness of the CBAF reduction problem, and the many degrees of freedom in its formulation, suggests Simulated Annealing (SA) as an optimization tool. We follow the approach of Johnson et al.[14] From the matrix A, it suffices to have, for any row i, a list of columns in which there are NZE’s, i.e.

\[
\text{aijs}(i) = \{ j \in N: A[i, j] \neq 0 \}
\]

We begin the standard simulated annealing (SSA) setting an initial random state or coloring \( p : M \rightarrow B \). We then compute the number of NZEs of each color per column, i.e., the \( n \times b \) matrix of weights:

\[
W[j, k] = \{ i \in M : A[i, j] \neq 0 \land p(i) = k \}
\]

and from \( W \) it is trivial to compute \( s(k) \) and \( \text{cost}(p) \).

From state \( p \) we propose a possible move. In our case a move is a random change in the color of a row

\[
p(i) \rightarrow p'(i) \in B
\]

The set of states \( p' \) reachable from \( p \) by a single move is the neighborhood, \( \text{nb}d(p) \) of \( p \). In our problem, any state \( p \) has a neighborhood of the same size, \( \text{NBDSIZE} \). We can choose one of the \( m \) rows, and then choose one of the \( b \) colors for this row, so:

\[
\text{NBDSIZE} = |\text{nb}d(p)| = m \ast b
\]

Next compute the cost of this proposed move \( \Delta_0 \).

\[
\Delta_0 = \text{cost}(p') - \text{cost}(p)
\]

The move is then accepted with probability \( \text{prob}(\Delta_0) \)

\[
\text{prob}(\Delta_0) = \begin{cases} 1 & \text{if } \Delta_0 \leq 0 \\ \text{exp}(-\Delta_0/\text{temp}) & \text{if } \Delta_0 > 0 \end{cases}
\]

To compute \( \Delta_0 \) it suffices to scan \( W[j, k] \) for \( j \in \text{aijs}(i) \).

If the proposed move was accepted we update \( p, W, \) and \( \text{cost}(p) \). The temperature parameter \( \text{temp} \) is initially set at \( \text{temp} = \text{STARTTEMP} \).

New moves are proposed until we either:

- make \( \text{LENGTH} = \text{SIZEFACTOR} \ast \text{NBDSIZE} \) proposals, or
- accept \( \text{CUTOFF} \ast \text{LENGTH} \) moves.

Then we go to a cooling step, where we:

- compute the acceptance rate for this temperature:

\[
\text{accrate} = |\text{accepted moves}|/|\text{proposals}|
\]

- cool the temperature setting \( \text{temp} \leftarrow \text{temp} \ast \text{TFAC} \)

The SSA is said to be frozen at a given temperature if \( \text{accrate} < \text{MINACCEPT} \). The algorithm terminates after \( \text{FROZENMAX} \) consecutive frozen temperatures.

The parameters \( \text{STARTTEMP}, \text{TFAC}, \text{SIZEFACTOR}, \text{CUTOFF}, \text{MINACCEPT}, \) and \( \text{FROZENMAX} \) are set by the user, as discussed in [14] and Section 4 of this.
paper. Our reliance on [14] to define a “standard” SA is convenient for our study, but by no means the only possibility. Alternative approaches, methodologies and views of the field can be found in [1, 2, 5, 6, 20 and 29].

3. A Temperature Dependent Cost Function
The SSA behaves poorly in the CBAF problem, mainly because it is very difficult to sense the proximity of low cost states, i.e.,

1. Most of the neighbors of a low cost state, \( p \), can have much higher costs.
2. The problem is massively degenerate, i.e., often a “connected by single moves” set of equal cost states, \( S = \{ p_1, \ldots, p_n \} \), has such a large degeneracy \( d \) that, even rejecting all proposals that would take us out of \( S \), would still give us a significant acceptance rate.

Difficulty 2 implies, in particular, the failure of the SSA termination criterion: A degenerate locally minimum connected component of \( S_c = \{ p, \text{cost}(p) = c \} \), could trap the SSA into forever sustaining an acceptance rate above the threshold MINACCEPT.

The best way we found to overcome difficulties 1 and 2 is to use a temperature-dependent cost function:

\[
\text{cost}(p, \mu(t)) = \text{cost}(p) - \mu(t) \text{penalty}(p)
\]

\[
\text{penalty}(p) = \sum_{j, |q(j)| > 1} (b - |q(j)|)
\]

The additional term in \( \text{cost}(p, \mu) \) can be seen as an heuristic penalty function that rewards multicolored columns for using fewer colors. This penalty function, and some possible variants, are inspired by the tally function \( \theta(j) \) used in the \( P3 \) heuristic\(^{[23]} \) for sparse LU factorization. The temperature dependent parameter \( \mu(t) \) gives the relative weight of the penalty function in \( \text{cost}(p, \mu) \).

Function \( \text{cost}(p, \mu) \) also has the following properties:

1. \( \text{cost}(p, 0) = \text{cost}(p) \)
2. \( \text{cost}(p, \mu) \) is linear in \( \mu \).

Properties 1 and 2 suggest that we can cool the parameter \( \mu \) along with the temperature, much in the same way we decrease a parameter of the barrier functions in some constrained optimization algorithms.\(^{[19]} \) We use \( \text{cost}(p, \mu) \) in the temperature-dependent penalty function simulated annealing (TPSA)\(^{[20]} \) as follows:

- Initially set \( \mu = \text{STARTMU} \);
  set \( p \) an initial partition, and initialize \( w, s, c, \) and \( \text{cost} = \text{cost}(p) \)
- For each proposal compute
  \[
  \Delta_0 = \text{cost}(p') - \text{cost}(p)
  \]
  \[
  \Delta_\mu = \text{cost}(p', \mu) - \text{cost}(p, \mu)
  \]
- Accept the move with probability \( \text{prob}(\Delta_\mu) \)
  and then update \( w, s, c, \) and \( \text{cost} = \text{cost} + \Delta_0 \)
- Keep track of the best, i.e. smallest \( \text{cost}(p) \), solution found
- At each cooling step set \( \mu = \mu + \text{MUFACCTOR} \).

To compute \( \Delta_\mu \) and \( \Delta_0 \) we scan the same columns of \( W \) that would be scanned to compute \( \Delta_\mu \) alone. It only takes a few more arithmetic operations to compute \( \Delta_\mu \) along with \( \Delta_0 \), which has little impact on the running time of the SA. The \( \text{cost}(p, \mu) \) is never computed (only the differentials \( \Delta_\mu \)).

The main goals for the temperature dependent penalty function, namely to accelerate the SA convergence to the global optimum and to avoid premature convergence to locally optimal solutions, have motivated many others interesting strategies. Closest to our approach, directly involving the cost function, are \([23] \) and \([28]\). Alternative approaches deal with the cooling schedules, \([13] \) and \([31]\), or with the topology of the neighborhood, \([11] \) and \([17]\). We believe that the TPSA method, which can be seen as a perturbation on the metric of the problem, could be useful in developing SA algorithms for other problems where the user is aware of some heuristic merit or penalty function. Also, there is no a priori impediment to combining metric perturbations with better topologies (i.e., neighborhood structures) or improved cooling schedules.\(^{[27]} \)

4. Numerical Experiments
We tested the SSA and the TPSA for CBAF reduction on three different matrices. These matrices are coefficient matrices from linear programs in the NETLIB collection of test problems. A portrait of the sparsity structure of these matrices can be found in [18].

<table>
<thead>
<tr>
<th>NETLIB LP</th>
<th>m</th>
<th>n</th>
<th>NZEs</th>
<th>Matrix Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>scatp1</td>
<td>300</td>
<td>660</td>
<td>1872</td>
<td>9 steps stair case</td>
</tr>
<tr>
<td>scfxm1</td>
<td>300</td>
<td>600</td>
<td>2732</td>
<td>4 diagonal blocks of diff. sizes</td>
</tr>
<tr>
<td>grow15</td>
<td>300</td>
<td>645</td>
<td>5620</td>
<td>narrow band with dense clusters</td>
</tr>
</tbody>
</table>

For each matrix we used all possible combinations of the parameters:

\( \text{STARTMU} \in (0.5, 1.0) \) and \( \text{MUFACCTOR} \in (0.90, 0.95, 0.98) \) plus \( \text{STARTMU} = 0.0 \) that corresponds to the SSA, in which case \( \text{MUFACCTOR} \) is irrelevant. The cost function parameters were chosen according to our specific applications.\(^{[23, 26]} \) always \( b = 4, \alpha = 0.01 \) and \( \beta = 1. \) Finally we always had \( \text{SIZEFACTOR} = 16, \text{CUTOFF} = 0.125, \text{TEMPFACTOR} = 0.95, \) and \( \text{STARTTEMP} \) was set to give us an initial acceptance rate of \( \approx 40\%. \) We set these last four parameters following the guidelines in \([14]\), but the relative performances of the various annealings do not depend heavily on them.

To avoid the problems with termination criteria mentioned in Section 2, we gave each SA run a fixed “budget” of \( 1.0E6 (1.0 \times 10^6) \) proposals. In each run, after completion of \( 0.1E6, 0.2E6, 0.3E6, 0.4E6, 0.5E6 \) and \( 1.0E6 \) proposals, we recorded \( \text{bestcost} = \) the best \( \text{cost}(p) \) already found, and \( \text{totmove} = \) the total (cumulative) number of accepted
moves. With bestcost we can monitor the progress of the annealing; totmove and totprop (the total number of proposals) are measures of the running time of the SA. Variable totprop is more important for a pure sequential implementation, while totmove may dominate in parallel machines.

In each SA run, the initial state, row selection, color change and acceptance processes were controlled by 4 different streams of a C-coded 100-streams 32-bits linear congruential random number generator. For each matrix and parameter setting we performed an experiment consisting of 82 runs, each controlled by a different 4-stream set.

5. Experimental Results

Our results are summarized by the box-plots of bestcost and totmove for each experiment (Figures 4–9). The three horizontal lines of a box, inside, lower and upper ends, correspond to the median, lower and upper quartiles; from the ends of the box “whiskers” may extend for up to 1.5 times the inter-quartile distance, and beyond that, detached points indicate extreme outliers. Each figure contains seven sequences of five plots. Each sequence corresponds to budgets of 0.1E6, 0.2E6, 0.3E6, 0.5E6 and 1.0E6 proposals. The seven sequences correspond to the parameters STARTMU and MUFACmTOR set to:

\[
\begin{align*}
[0.0, 0.5, 0.90] & \quad [0.5, 0.95] & \quad [0.5, 0.98] \\
[1.0, 0.90] & \quad [1.0, 0.95] & \quad [1.0, 0.98]
\end{align*}
\]

As noted in the last section, the leftmost sequence of plots correspond to the SSA, since STARTMU = 0.0.

From the box-plots we see that the TPSA outperforms the SSA for all of our test matrices, for all TPSA parameter setting, and all budgets we tried. Also, the TPSA usually continued to make progress with bigger budgets, while the SSA “saturated” much earlier. Moreover, for a given totprop budget, the totmove component of the running time is usually smaller in the TPSA than in the SSA!

For the SSA in matrix grow15, note the convergence of bestcost to a very precise level \(\min(\text{cost}(p))\). Most of the connected components of the macro-state \(S_c = \{p, \text{cost}(p) = c\}\) are not only massively degenerate, but also local minima, which give us an intuitive explanation of the metastable character of \(S_c\). The penalty term in \(\text{cost}(p, \mu)\) decreases degeneracy and local minima overall, thereby destroying the metastability of \(S_c\).

The ratio MUFACmTOR/TEMPFACTOR is very important for the TPSA performance. In our experiments we set MUFACmTOR to 0.90, 0.95, and 0.98, respectively a slow, neutral, or fast cooling of \(\mu\) relative to TEMPFACmTOR = 0.95. The slow cooling of \(\mu\) very effectively avoids degeneracy, at the expense of optimizing a biased \(\text{cost}(p, \mu)\) with a relatively big penalty factor \(\mu\). In Table I we show some statistics of the TPSA performance. Each line of Table I corresponds to a series of 82 independent annealings, for a given test matrix and setting of parameters STARTMU and MUFACmTOR. The statistics in Table I are: \(aoc\), the approxi-
SARTMU=(0.0,0.5,1.0) X MUFACCTOR=(0.90,0.95,0.98) X budget=\{1,2,3,5,10\}E5

Figure 5. \textit{totmove} for matrix sctap1.

SARTMU=(0.0,0.5,1.0) X MUFACCTOR=(0.90,0.95,0.98) X budget=\{1,2,3,5,10\}E5

Figure 6. \textit{bestcost} for matrix scfxm1.
Figure 7. \( \text{totmove} \) for matrix scfxm1.

Figure 8. \( \text{bestcost} \) for matrix grow15.
Annealing with Temperature Dependent Penalty

Figure 9. \textit{totmove} for matrix grow15.

Figure 10. Contour heuristic costs.
Table I. TPSA Dependence on STARTMU and MUFACCTOR

<table>
<thead>
<tr>
<th>SMU</th>
<th>MUF</th>
<th>aoc</th>
<th>naoc</th>
<th>mbc</th>
<th>sdhc</th>
<th>mtm</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>*</td>
<td>103.3</td>
<td>1</td>
<td>128.1</td>
<td>12.3</td>
<td>54972</td>
</tr>
<tr>
<td>0.5</td>
<td>0.90</td>
<td>81.8</td>
<td>2</td>
<td>96.6</td>
<td>7.7</td>
<td>74578</td>
</tr>
<tr>
<td>0.5</td>
<td>0.95</td>
<td>81.9</td>
<td>2</td>
<td>98.7</td>
<td>6.7</td>
<td>64098</td>
</tr>
<tr>
<td>0.5</td>
<td>0.98</td>
<td>80.6</td>
<td>1</td>
<td>99.4</td>
<td>8.2</td>
<td>45460</td>
</tr>
<tr>
<td>1.0</td>
<td>0.90</td>
<td>80.5</td>
<td>4</td>
<td>91.2</td>
<td>7.1</td>
<td>79760</td>
</tr>
<tr>
<td>1.0</td>
<td>0.95</td>
<td>80.5</td>
<td>4</td>
<td>91.2</td>
<td>7.2</td>
<td>61654</td>
</tr>
<tr>
<td>1.0</td>
<td>0.98</td>
<td>80.5</td>
<td>4</td>
<td>96.1</td>
<td>8.5</td>
<td>48483</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Matrix scfxm1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
</tr>
<tr>
<td>0.5</td>
</tr>
<tr>
<td>0.5</td>
</tr>
<tr>
<td>0.5</td>
</tr>
<tr>
<td>1.0</td>
</tr>
<tr>
<td>1.0</td>
</tr>
<tr>
<td>1.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Matrix grow15</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
</tr>
<tr>
<td>0.5</td>
</tr>
<tr>
<td>0.5</td>
</tr>
<tr>
<td>0.5</td>
</tr>
<tr>
<td>1.0</td>
</tr>
<tr>
<td>1.0</td>
</tr>
<tr>
<td>1.0</td>
</tr>
</tbody>
</table>

...in rows of B and in rows outside B). The cost of a block is defined as the cardinality of its cut, i.e.

$$\text{cost}(B) = |\text{cut}(B)|.$$  

The CH forms the row blocks in A (vertex clusters in the hypergraph) one at a time. To form the first block, B, we take at random the first vertex. At step $t$, $t = 2, 3 \ldots$ we then add to the block a vertex $v_t$ that minimizes $\text{cost}(B_t) = \text{cost}(B_{t-1} + v_t)$. For each $t$ we store $v_t$ and $\text{cost}(B_t)$. Our final step is to select at which $\tilde{t}$ to terminate the block $B = B_t$. To avoid having too small or too large a block, we impose

$$(1 - \omega)m/b \leq \tilde{t} \leq (1 + \omega)m/b$$

where the parameter $\omega \in [0, 1]$ is set by the user, and then take $\tilde{t}$ so to minimize $\text{cost}(B_t)$. After we finish block B we eliminate from A the rows of B, and the columns having any NZEs in rows of B. If $b > 1$ we use the CH to form a new block in the reduced A, with parameters $m \leftarrow m - \tilde{t}$ and $b \leftarrow b - 1$.

As in the TPSA, we tried to improve the performance of the CH by using a heuristic penalty function. When selecting the vertices for the sequence $v_1, v_2, \ldots, v_{|1 + \omega|m/b}$ we use the perturbed cost function:

$$\text{cost}(B_t, \mu) = \text{cost}(B_t) - \mu \cdot \text{penalty}(B_t)$$

$$\text{penalty}(B_t) = ||e \in \text{cut}(B_t),$$

$e$ adjacent to exactly 1 vertex not in B].

The interpretation of $\text{penalty}(B_t)$ is similar to the penalty term in the TPSA, and even closer to the tally function of Hellerman and Rarick[12].

We tested the CH on our three test matrices. We tested parameter $\mu \in \{0.0, 0.5, 1.0\}$. The case $\mu = 0.0$ corresponds to the unperturbed block cost. Experimentation indicated that taking $\omega = 0.2$ was a good choice. We used the CH to form 3 blocks, considering the remaining vertices as the fourth block, and then evaluated the final configuration with the cost function $\text{cost}(p)$, as described in Section 1, and using the parameters given in Section 4. In Figure 10 we have box-plots of these final costs. For each test matrix and parameter setting, we ran the CH 600 times, which demands a computing time roughly equivalent to 10 runs of the SA. Although the CH outperforms the SSA for one of the test matrices (grow15), the TPSA clearly outperforms the CH in all cases. Also the "response" of the CH performance to the value of parameter $\mu$ is not as uniform as in the TPSA.

Acknowledgments

This work was supported by grant 87-0540-9 from FAPESP, Fundação de Amparo à Pesquisa do Estado de São Paulo, and the Computer Science Department of the University of São Paulo, Brasil. I am grateful to Stephen A. Vavasis and Russell R. Barton, from Cornell University, for many useful suggestions.

References


15. M.E. Johnson (Editor), 1988. Simulated Annealing & Optimization, American Science Press, Syracuse, NY (This book is also the volume 8 of the American Journal of Mathematical and Management Sciences.)


28. C.A. Tovey, 1988. Simulated Simulated Annealing, in Johnson.[15]


