<table>
<thead>
<tr>
<th>Title</th>
<th>Minimum delay scheduling in scalable hybrid electronic/optical packet switches</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author(s)</td>
<td>Wu, B; Yeung, KL</td>
</tr>
<tr>
<td>Citation</td>
<td>Proceedings of the Global Telecommunications Conference, 2006 (GLOBECOM 2006), San Francisco, CA, USA, 27 November - 1 December 2006</td>
</tr>
<tr>
<td>Issued Date</td>
<td>2006</td>
</tr>
<tr>
<td>URL</td>
<td><a href="http://hdl.handle.net/10722/99457">http://hdl.handle.net/10722/99457</a></td>
</tr>
<tr>
<td>Rights</td>
<td>©2006 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.</td>
</tr>
</tbody>
</table>
Minimum Delay Scheduling in Scalable Hybrid Electronic/Optical Packet Switches

Bin Wu and Kwan L. Yeung
Dept. of Electrical and Electronic Engineering
The University of Hong Kong
Pokfulam, Hong Kong
E-mail: {binwu, kyeung}@eee.hku.hk

Abstract—A hybrid electronic/optical packet switch consists of electronically buffered line-cards interconnected by an optical switch fabric. It provides a scalable switch architecture for next generation high-speed routers. Due to the non-negligible switch reconfiguration overhead, many packet scheduling algorithms are invented to ensure performance guaranteed switching (i.e. 100% throughput with bounded packet delay), at the cost of speedup. In particular, minimum delay performance can be achieved if an algorithm can always find a schedule of no more than \(N\) configurations for any input traffic matrix, where \(N\) is the switch size. Various minimum delay scheduling algorithms (MIN, \(a^2\)-SCALE and QLEF) are proposed. Among them, QLEF requires the lowest speedup bound. In this paper, we show that the existing speedup bound for QLEF is not tight enough. A new bound which is 10% lower than the existing one is derived.

Keywords—Minimum delay scheduling; performance guaranteed switching; reconfiguration overhead; speedup bound.

I. INTRODUCTION

The explosion of Internet traffic has led to ever-increasing demands for larger bandwidth and higher port density in the next generation routers. At present, most Internet backbone routers are based on a single-rack solution using a switched backplane. Typically, a standard telecommunication rack is of size 19 inches in width and 7 feet in height. It can accommodate 14-16 line-cards, with aggregate capacity up to 160 Gb/s. To further increase the capacity, multi-rack solution [1] is adopted, where line-cards in different racks are interconnected to/from the central electronic switch fabric by fibers. This architecture defines the 4th generation router, which can offer an aggregate capacity up to 10 Tb/s [2]. Since electronic switch fabric is used, O/E/O conversions are necessary at the central switch rack. As data is handled in electronic domain, power consumption becomes the key constraint [3]. To solve these issues, the 5th generation router is proposed as shown in Fig. 1, where a hybrid electronic/optical switch architecture is adopted. Compared to the 4th generation, the central electronic switch fabric is replaced by an optical one. This not only removes the O/E/O conversions from the switch rack, but also reduces its power consumption. Following this way, the aggregate capacity can be up to 100 Tb/s [3-4].

However, the optical switch fabric needs a non-negligible amount of time to change its configurations, known as reconfiguration overhead. During this period, no packet can be transmitted across the switch. Reconfiguration overhead is due to three factors [5]. First, the optical fabric needs time to change its interconnection pattern, which can range from 10 ns to several milliseconds depending on the technology adopted [6-9]. Second, time is required to resynchronize the optical transceivers and the switch fabric in each reconfiguration. Finally, because the arriving time of optical signals varies, the clock and its phase have to be aligned, and extra clock margin has to be considered in order to avoid data loss.

To achieve performance guaranteed switching (i.e. 100% throughput with bounded packet delay) [10-13], the switch fabric must run faster to compensate for both the reconfiguration overhead and the scheduling inefficiency. The required speedup \(S\) is defined as the ratio of the internal packet transmission rate to the external line-rate (\(S \geq 1\)).

Assume time is slotted and each time slot can accommodate one fixed-size packet. Several scheduling algorithms are proposed to achieve performance guaranteed switching [10-13]. They all adopt the same four-stage scheduling procedure as shown in Fig. 2. Stage 1 is for traffic accumulation. An \(N \times N\) traffic matrix \(C(T)\) is obtained at the input buffers every \(T\) time slots, where \(N\) is the switch size. Each entry \(c_{ij}\) denotes the number of packets arrived at input \(i\) and destined to output \(j\). As a common assumption [10-13], the entries in each line (i.e. row or column) of \(C(T)\) sum to at most \(T\). In Stage 2, a scheduling algorithm generates a schedule consisting of (at most) \(N_S\) switch configurations in \(H\) time slots. Each

This work is supported by Hong Kong Research Grant Council Earmarked Grant HKU 7032/01E.
configuration is denoted by a permutation matrix \( P_n = \{p_{ij}^{(n)}\} \) \((N \geq n \geq 1)\), where \( p_{ij}^{(n)} = 1 \) means that input port \( i \) is connected to output port \( j \) (in this case, we also say that \( P_n \) covers entry \((i, j)\)). A weight \( \phi_n \) is assigned to each \( P_n \) indicating the number of slots that \( P_n \) should be kept for packet transmission. The set of \( N_S \) configurations generated must cover \( C(T) \), i.e., \( \sum_{i=1}^{N_S} \phi_n p_{ij}^{(n)} \geq \delta \) for any \( i, j \) \( \in \{1, \ldots, N\} \). Then \( \sum_{n=1}^{N_S} \phi_n \) is the number of slots required to transmit all the packets in \( C(T) \). Let each reconfiguration take an overhead of \( \delta \) time slots. Accordingly, sending \( C(T) \) requires \( \delta N_S + \sum_{n=1}^{N_S} \phi_n \) time slots. This is generally larger than the traffic accumulation time \( T \). Without speedup, 100% throughput is not possible. Stage 3 is for actual packet transmission, where the switch fabric is reconfigured according to the \( N_S \) configurations. At a speedup of \( S \), the slot size for a single packet transmission in Stage 3 is shortened by \( S \) times. Then 100% throughput is ensured by having
\[
\delta N_S + \frac{1}{S} \sum_{n=1}^{N_S} \phi_n = T. \tag{1}
\]
The values of \( N_S \) and \( \sum_{n=1}^{N_S} \phi_n \) in (1) are determined by the scheduling algorithm. Note that the total reconfiguration overhead time \( \delta N_S \) cannot be reduced by speedup and thus \( T > \delta N_S \). Finally, Stage 4 takes another \( T \) time slots to send the packets onto the output lines from the output buffers.

Rearranging (1), we have
\[
S = \frac{T - \delta N_S}{\frac{1}{S} \sum_{n=1}^{N_S} \phi_n} = S_{reconfigure} \times S_{schedule}, \tag{2}
\]
where \( S_{reconfigure} \) and \( S_{schedule} \) are defined as
\[
S_{reconfigure} = \frac{T}{T - \delta N_S}, \tag{3}
\]
\[
S_{schedule} = \frac{1}{S} \sum_{n=1}^{N_S} \phi_n. \tag{4}
\]
\( S_{reconfigure} \) is the speedup factor to compensate for the idle time caused by reconfigurations, whereas \( S_{schedule} \) is the speedup factor to compensate for the scheduling inefficiency.

In Fig. 2, the packet delay is bounded by \( 2T+H \) slots where \( T > \delta N_S \). With a smaller \( N_S \), \( T \) and thus the packet delay bound can be reduced. But \( N_S \) must be no less than \( N \). Otherwise, the \( N_S \) configurations are not sufficient to cover every entry in \( C(T) \) [10-13]. Accordingly, scheduling algorithms that only require \( N_S = N \) configurations are called minimum delay scheduling algorithms. When \( N_S = N \), from (1) \( T \) can be made as close to its lower bound \( \delta N \) as possible by minimizing \( \sum_{n=1}^{N_S} \phi_n \). Generally, we hope to minimize \( S \) for a given packet delay (or equivalently a given \( T \)). In minimum delay scheduling, this translates to minimizing \( S_{schedule} \) according to (1) ~ (4).

Recently, several minimum delay scheduling algorithms (MIN [10], d-SCALE [12] and QLEF [13]) have been proposed to minimize \( S_{schedule} \). Among them, QLEF (Quasi Largest-Entry-First) requires the lowest speedup bound. In this paper, we derive a new speedup bound for QLEF, which is 10% lower than the one in [13]. Because the same scheduling problem is also involved in other communication networks, such as SS/TDMA [14], TWIN [15] and wireless sensor networks [16], our result also contributes to those systems.

The rest of the paper is organized as follows. In Section II, we review the QLEF algorithm [13]. The new speedup bound is derived in Section III. We conclude the paper in Section IV.

---

**QLEF ALGORITHM**

```
Input: An N×N matrix C(T) with maximum line sum not more than T.
Output: N configurations \( P_1, \ldots, P_N \) and weights \( \phi_1, \ldots, \phi_N \).
Step 1: Initialization:
Set \( 0 \rightarrow n \). Initialize \( P_1, \ldots, P_N \) to all-zero matrices and the N×N reference matrix \( R = (r_{ij}) \) to all 1’s.
Step 2: Determine the first “half” configurations \( P_{1-4} \):
(a) Un-shadow \( C(T) \) and \( R \). Set 1 \( \rightarrow w \).
(b) Select the largest entry \( c_{ij} \) in the not-yet-shadowed part of \( C(T) \) and \( R \). If \( w = 1 \), set \( P_{ew} \)'s weight \( \phi_{ew} = w \) and \( w = 0 \). Shadow the corresponding lines in both \( C(T) \) and \( R \) and set \( c_{ij} \) and \( w \) to 0. Set \( 1 \rightarrow w+1 \) where \( w+1 \) is the entry \((i, j)\) of \( P_{ew} \). Repeat this step until \( N \) (2n+1) largest entries are selected.
(c) Construct a bipartite graph \( U_e \) from the remaining not-yet-shadowed part of \( R \) and perform maximum-size matching in \( U_e \) to get \( (2n+1) \) edges. Record the corresponding entries to \( P_{ew} \) by setting \( 1 \rightarrow w+1 \). Set these entries of \( C(T) \) and \( R \) to 0’s. Then set \( n+1 \rightarrow n \).
(d) Repeat Step 2a-2c until \( n = \lceil N/2 \rceil - 1 \).
Step 3: Determine the second “half” configurations \( P_{5-8} \):
(a) Un-shadow \( C(T) \) and \( R \). Find the largest entry \( c_{ij} \) in \( C(T) \) and set \( c_{ij} \) as the weight for all the subsequent configurations.
(b) Find a maximum-size matching in the bipartite graph of \( R \) and set the corresponding entries of \( P_{ew} \) to 1. These entries set to 0 in \( C(T) \) and \( R \), and then set \( n+1 \rightarrow n \). Repeat this step until \( n=N \).
```

Fig. 3. QLEF algorithm.

---

**II. QLEF ALGORITHM**

QLEF algorithm [13] is summarized in Fig. 3. It generates \( N \) non-overlapping configurations (i.e., the entries covered by any two configurations do not overlap) to cover \( C(T) \). It has a time complexity of \( O(N^{3.5}) \), and the correctness proof can be found in [13]. When the values of \( T, N \) and \( \delta \) are given, \( S_{schedule} \) determines the overall speedup \( S \) in minimum delay scheduling. Therefore, QLEF aims at minimizing \( S_{schedule} \).

The key idea of QLEF is to cover large entries in \( C(T) \) first. A reference matrix \( R \) is initialized to all 1’s, where a “1” indicates that the corresponding entry of \( C(T) \) is not yet covered. Assume that configuration \( P_{n+1} \) is being constructed. To avoid configuration overlaps, QLEF only selects \( N \) (2n+1) largest entries from \( C(T) \) in its Step 2b, which are called selected-entries. The corresponding lines of the selected-entries are shadowed in both \( C(T) \) and \( R \) (see Fig. 4a). Then, QLEF applies maximum-size matching (MSM) [17] to the remaining not-yet-shadowed part of \( R \) to get \( (2n+1) \) entries in Step 2c, which are called MSM-entries. The \( N \) (2n+1) selected-entries combine with the \( (2n+1) \) MSM-entries to form \( P_{n+1} \). Accordingly, those entries covered by \( P_{n+1} \) are set to 0 in both \( C(T) \) and \( R \). Then both \( C(T) \) and \( R \) are un-shadowed and QLEF repeats the above process to construct the next configuration.

To ensure \( N \) (2n+1) \( > 0 \), only the first \( \lceil N/2 \rceil - 1 \) configurations are constructed according to the above mechanism (in Step 2). Then in Step 3, the remaining \( N \) (2n+1) \( \geq 1 \) configurations are determined by maximum-size matching [17].

---

**III. SPEEDUP BOUND**

In QLEF, the \( N \) configurations are sequentially constructed from \( P_1 \) to \( P_N \). We now focus on the first half of them \( P_1, \ldots, P_{N/2+1} \). Assume that an entry \( c_{ij} \in C(T) \) is covered
by \( P_{n+1} \). If \( c_{ij} \) is shadowed (see Fig. 4a) in the construction of \( P_k \)
\((n \geq k \geq 1)\), then \( P_k \) is called an \( s \)-configuration of \( c_{ij} \). Otherwise,
\( P_k \) is called a \( g \)-configuration of \( c_{ij} \).

Fig. 4b shows the conceptual QLEF scheduling procedure. We use a “scheduling trace”
to represent the trend of \( c_{ij} \) values covered. It is usually a wave rather than
a monotonically decreasing curve, although QLEF always selects the largest
entry in the not-yet-shadowed part. Due to the shadowing operation, a large entry may be shadowed by an \( s \)-configuration
and thus can only be covered later by another configuration.

For a particular configuration \( P_{n+1} \), its weight \( \phi_{n+1} \) also appears
as an entry in \( C(T) \) and is the first selected-entry in \( P_{n+1} \). Therefore, in the following discussion, we treat \( \phi_{n+1} \) as
an entry in \( C(T) \) rather than a weight. Among the \( n \) configurations constructed before
\( P_{n+1} \) (i.e. \( P_1 \sim P_n \)), we assume that \( \Delta \) of them are \( g \)-configurations of \( \phi_{n+1} \) and the
other \( n-\Delta \) configurations are its \( s \)-configurations, as in Fig. 4b.

A. General Idea

Define the entries larger than or equal to entry \( \phi_{n+1} \) in the
original \( C(T) \) as larger-entries (or LEs). To bound the value of
\( \phi_{n+1} \), we count the minimum total number of LEs (denoted by
\( M \)) that are covered by \( P_1 \sim P_n \). These \( M \) LEs reside in \( N \) lines
(rows or columns) of \( C(T) \), and the line that contains the
maximum number of LEs must contain at least the average
number of \( \lceil M/N \rceil \) LEs. As a result, the smallest LE in this line
must be smaller than or equal to the \( \lceil M/N \rceil \)-th largest entry of
this line. Yet it is not smaller than \( \phi_{n+1} \). Because the maximum line sum of \( C(T) \) is \( T \), according to Lemma 1 in [13], we have

\[
\phi_{n+1} \leq \frac{T}{\lceil M/N \rceil} \quad (5)
\]

On the other hand, because \( \phi_{n+1} \) is shadowed by \( n-\Delta \) \( s \)-
configurations, from Lemma 2 in [13], we have

\[
\phi_{n+1} \leq \frac{T}{\Delta + 1} \quad (6)
\]

Combining (5) and (6), we can bound \( \phi_{n+1} \) as follows:

\[
\phi_{n+1} \leq \max_{0 \leq \Delta \leq N-1} \left\{ \min \left\{ \frac{T}{\lceil M/N \rceil}, \frac{T}{\Delta + 1} \right\} \right\} \quad (7)
\]

Formula (7) indicates that no matter what is the value of \( \Delta \)
\((0 \leq \Delta \leq n)\), the bound always holds because we have taken the
worst case into consideration (i.e. the max function for \( \Delta \)).

For the remaining \( N - \lceil N/2 \rceil + 1 \) configurations constructed in
Step 3 in Fig. 3, QLEF uses a small constant as the weights.
According to QLEF, this constant is not larger than any weight
of the first \( \lceil N/2 \rceil - 1 \) configurations (since the weights are
monotonically decreasing as shown by the dashed line in Fig.
4b). Therefore, it can be bounded by the weight of the last one
among the first \( \lceil N/2 \rceil - 1 \) configurations. That is

\[
\phi_{n+1} \leq \min_{0 \leq \Delta \leq N-1} \left\{ \frac{T}{\Delta + 1} \right\} \quad (8)
\]

After the \( N \) weights \( \phi_{n+1} \) are bounded by (7) and (8),
we can calculate \( S_{\text{schedule}} \) bound in (4). Note that the key is to count
the minimum total number of LEs (i.e. \( M \) in (7)).

B. Speedup Bound Formulation

For entry \( \phi_{n+1} \), we consider its \( \Delta \) \( g \)-configurations and the
other \( n-\Delta \) \( s \)-configurations (see Fig. 4b). In QLEF, all the
selected-entries in the \( g \)-configurations are LEs. On the other
hand, each \( s \)-configuration must cover one LE in the same line as
\( \phi_{n+1} \). However, in addition to this LE, the \( s \)-configuration
may also cover other LEs in different lines. Assume that a set
of consecutive \( s \)-configurations \( \{P_x \} \) shadow \( \phi_{n+1} \), and \( P_x \) is
the first \( g \)-configuration after \( \{P_y \} \). From Lemma 1 in the
Appendix, the number of LEs covered by each \( P_x \in \{P_y \} \) is at
least half of the number of the selected-entries in \( P_y \).

In Fig. 4b, the \( \Delta \) \( g \)-configurations and the \( n-\Delta \) \( s \)-
configurations may queue in any order. From Lemma 2 in the
Appendix, in order to minimize \( M \), the \( n-\Delta \) \( s \)-configurations
should consecutively locate at either the very beginning or
the very end of the configuration sequence \( P_1 \sim P_x \).

Case 1: The \( n-\Delta \) \( s \)-configurations consecutively locate at
the very end of the \( n \) configurations. In this case, all the
selected-entries covered by the \( \Delta \) \( g \)-configurations are LEs, but
the number of LEs covered by the \( n-\Delta \) \( s \)-configurations is
trivial and is ignored when counting \( M \). So, \( M=(N-1)+(N-3)+\cdots+(N-2\Delta+1)=(N-\Delta)\Delta \). Note that none of the \( g \)-
configurations shadows \( \phi_{n+1} \). So, these LEs reside in \( N-1 \) lines
of \( C(T) \) instead of \( N \) lines. Replacing \( N \) in (7) by \( N-1 \), we have

\[
\phi_{n+1} \leq \min_{0 \leq \Delta \leq N-1} \left\{ \frac{N-\Delta}{N-1} \right\} \quad (9)
\]

Mathematically, this is equivalent to (10) below.
\[
\phi_{n+1} \leq \max \left( \frac{T}{n-\Delta+1} \right) \text{ with } \Delta = \frac{3(N-1)-3(N-1)^2-8(N-1)(\Delta+2)}{4}.
\]

**Case 2:** The \( n-\Delta \) s-configurations consecutively locate at the beginning of the configuration sequence \( P_1 \sim P_n \). In this case, each s-configuration covers at least \( (N-1)/2 - (n-\Delta) \) LEs according to Lemma 1 in the Appendix. Taking the LEs covered by the \( \Delta \) g-configurations into account, we get \( M = Nn - n^2 - (N+1)(n-\Delta)/2 \) by simple calculation. From (7) we have

\[
\phi_{n+1} \leq \max \min \left( \frac{T}{2Nn-2n^2 - (N+1)(n-\Delta)} \right) \text{ and } \Delta = \frac{2n^2+2N}{2N+1}.
\]

Again, this is equivalent to

\[
\phi_{n+1} \leq \frac{T}{n-\Delta+1} \text{ with } \Delta = \frac{2n^2+2N}{2N+1}.
\]

Combining (10), (12) and (8), we get the bound for \( \phi_{n+1} \) as shown in formula (13). We can replace \( \phi_n \) in (4) by the bound in (13) to calculate the \( S_{\text{schedule}} \) bound.

**C. Results**

Fig. 5 shows the new speedup bound we derived for QLEF. As an example, the new \( S_{\text{schedule}} \) bound gives a gain of 11.29% over the existing bound in [13] for \( N = 200 \). Fig. 5b shows the \( S_{\text{schedule}} \) bound evolution for the minimum delay scheduling problem. Particularly, the bound \( S_{\text{schedule}} = 4(4+\log_2 N) \) is given in [10] for MIN, which is refined in [12] to produce the sawtooth-curve. A tighter bound is then provided by \( \alpha' \)-SCALE [12]. This is then followed by QLEF in [13]. In this paper we further push the QLEF speedup bound to be 10% lower.

**IV. Conclusion**

Hybrid electronic/optical packet switch provides a scalable switch architecture for huge-capacity backbone Internet routers. Because of the reconfiguration overhead of the optical switch fabric, packet delay is minimized by using \( N \) configurations (where \( N \) is the switch size) to schedule the traffic. However, this requires a very large speedup to achieve performance guaranteed switching. QLEF (Quasi Largest-Entry-First) is the most efficient minimum delay scheduling algorithm that gives the lowest speedup bound for a given packet delay. In this paper, we derived a new speedup bound for QLEF, which is 10% lower than the existing bound.

**APPENDIX**

**Lemma 1:** Assume that \( \{P_x\} \) is a set of consecutive s-configurations of \( \phi_{n+1} \), and \( P_1 \) is the first g-configuration of \( \phi_{n+1} \) after \( \{P_x\} \). Then, any \( P \in \{P_x\} \) must cover at least \( h \) LEs, where \( h \) is half of the number of the selected-entries in \( P_x \).

**Proof:** Since \( P_x \) is a g-configuration of \( \phi_{n+1} \), any selected-entry \( a \) covered by \( P_x \) is an LE. Because \( P_x \) is constructed before \( P_y \) and \( a \) is not covered in \( P_y \), either 1) all the selected-entries covered by \( P_x \) are not smaller than \( a \), or 2) \( a \) is shadowed in \( P_x \) construction.

In case 1), all the selected-entries in \( P_x \) are LEs. Since \( P_x \) is constructed earlier than \( P_y \), it contains more selected-entries than \( P_y \). So, the number of LEs covered by \( P_x \) is larger than \( h \). In case 2), any \( a \) covered by \( P_y \) must have been shadowed in \( P_x \) construction. Since a selected-entry in \( P_x \) can shadow at most two smaller/equal selected-entries in \( P_y \) (in row and column, respectively), \( P_x \) must cover at least \( h \) LEs. Obviously, this is true for the first g-configuration \( P_y \) after \( \{P_x\} \).

**Lemma 2:** To minimize \( M \), all the s-configurations of \( \phi_{n+1} \) should be consecutively located at either the very beginning or the very end of the configuration sequence \( P_1 \sim P_n \).

**Proof:** In Fig. 6, let \( y \)-axis denote the number of selected-entries covered in each configuration, and \( x \)-axis denote the scheduling sequence. Without loss of generality, assume there are three sets of consecutive s-configurations of \( \phi_{n+1} \), denoted by \( A_1 \), \( A_2 \) and \( A_3 \) (others are g-configurations). Particularly, \( A_1 \) and \( A_2 \) contain \( x_1 \) and \( x_2 \) s-configurations respectively, and \( A_3 \) locates at the very end of the configuration sequence \( P_1 \sim P_n \).
Although each s-configuration in $A_1$ covers $y_1$ selected-entries, and the first g-configuration after $A_1$ covers $(y_1^2-2x_1)/2$ selected-entries. Similarly, the first s-configuration in $A_2$ covers $y_2$ selected-entries, and the first g-configuration after $A_2$ covers $(y_2^2-2x_2)/2$ LEs. We do not count any LEs in $A_3$ because there is no g-configuration after it. Although each s-configuration in $A_3$ covers one LE in the same line as $\phi_{n+1}$, it is trivial and is ignored when counting $M$.

We first consider $A_1$ and $A_2$. Since all selected-entries covered by g-configurations are LEs, minimizing $M$ is equivalent to maximizing the number of selected-entries in the shaded areas in $A_1$ and $A_2$. That is

$$
\max \left( \frac{y_1^2-2x_1}{2} + \frac{x_1(y_1-1)}{2} \times 2 \right) + \left( \frac{y_2^2-2x_2}{2} + \frac{x_2(y_2-1)}{2} \times 2 \right)
$$

or

$$
\max \left( \frac{(y_1+2)x_1 + (y_2+2)x_2}{2} \right).
$$

Let $\lambda$ be the number of g-configurations between $A_1$ and $A_2$. We have $y_2^2=y_1^2-2(x_1+\lambda)$. So the above formula is equivalent to

$$
\max \left( \frac{(y_1+2)(x_1+x_2) - 2x_1(y_1+\lambda)}{2} \right).
$$

To maximize the above formula for any given $x_1$ and $x_2$, it is necessary that $y_1$ takes the maximum possible value and $\lambda=0$. This entails that $A_1$ and $A_2$ should be located consecutively at the very beginning of the configuration sequence $P_1 \sim P_n$. It is easy to see that this point can be generalized to the case where more sets of consecutive s-configurations are involved.

However, we still need to consider $A_3$ in Fig. 6. In fact, the $n$ configurations before $P_{n+1}$ may also be located as shown in Fig. 7, where the g-configurations are in the middle and the $n-\Delta$ s-configurations are at the both ends (the $\beta$ s-configurations locate consecutively at the beginning of $P_1 \sim P_n$ as argued above). In Fig. 7, minimizing $M$ is equivalent to maximizing the number of selected-entries in the not-shadowed areas, i.e.

$$
\max \left( \frac{N-1-2\beta}{2} + \beta + \frac{\beta-2\lambda}{2} \times 2 \right) + \left( \frac{(n-\Delta-\beta)(N-2n+1)}{2} + \frac{(n-\Delta-\beta)(n-\Delta-\beta-1)}{2} \times 2 \right)
$$

or

$$
\max \left( \beta^2 + \frac{4n-N+1}{2} \beta + (n-\Delta)(N-n-\Delta) \right).
$$

Because the above formula is a quadratic function of $\beta$, it can be maximized only if $\beta=0$ or $\beta=n-\Delta$. From Fig. 7, obviously all the $n-\Delta$ s-configurations should be consecutively located at either the very beginning ($\beta=0$) or the very end ($\beta=n-\Delta$) of the configuration sequence $P_1 \sim P_n$. The specific location is determined by the values of $N$, $n$ and $\Delta$.

REFERENCES


