ISSN 1342-2812

Research Reports on Mathematical and Computing Sciences

Pseudo Expectation:
A Tool for Analyzing Local Search Algorithms

Osamu Watanabe

Sept. 2004, C–198

Department of
Mathematical and
Computing Sciences
Tokyo Institute of Technology

SERIES C: **Computer Science**

Pseudo Expectation: A Tool for Analyzing Local Search Algorithms

Osamu Watanabe*
Dept. of Math. and Computing Sciences
Tokyo Institute of Technology, W8-25, Tokyo 152-8552
(watanabe@is.titech.ac.jp)

Research Report C-198

Abstract.
In [WST03], the notion of *pseudo expectation* has been proposed for analyzing relatively simple Markov processes, which would be often seen as simple execution models of local search algorithms. In this paper, we first explain how it is used, and then investigate the approximation error bound of pseudo expectations.

1 Introduction and Some Example

Pseudo expectation has been proposed [WST03] for analyzing relatively simple Markov processes, which would be often seen as simple execution models of "local search algorithms". The technical goal of this paper is to give an error bound for this pseudo expectation by mathematical analysis. But before the analysis, we explain, though briefly, our motivation of introducing this pseudo expectation and show some example illustrating how it could be used.

There are many problems that can be formulated as a "constraint satisfaction problem", a problem of searching for a solution that satisfies a given set of constraints. Well known SAT problem, one of the NP-complete problems that have been believed hard to solve in general, is a typical example of such problems. A "local search" is simple yet an important algorithmic approach for solving such constraint satisfaction problems. Figure 1 shows the outline of standard local search algorithms; it is for finding a solution satisfying all constraints $C_1, ..., C_m$ given as an input. (In some situations, for example, if it is impossible to satisfy *all* constraints, we usually consider a weaker goal and ask

*Supported in part by a Grant-in-Aid for Scientific Research on Priority Areas "Statical-Mechanical Approach to Probabilistic Information Processing" 2002-2005.

program `LocalSearch`;
input: A set of constraints $\{C_1, C_2, ..., C_m\}$ on some variables $\boldsymbol{v} = v_1, ..., v_n$;
output: A solution (i.e., an assignment to the variables) satisfying all constraints;
  $\boldsymbol{v} \leftarrow$ some initial assignment (often chosen randomly);
  repeat T steps
    if all constrains are satisfied **then**
      output the current solution $\boldsymbol{v}$ and halt;
    improve solution $\boldsymbol{v}$ by changing the value of one variable v_i;
program-end.

Figure 1: Outline of local search algorithms

for a solution satisfying as many constraints as possible.) Though very simple, various computer experiments have been reported that several constraint satisfaction problems (including the SAT problem) can be solved well on average under many (obviously not all) reasonable circumstances. Unfortunately, however, it seems quite difficult to analyze the behavior (e.g., the average-case performance) of such algorithms, and existing rigorous mathematical analyses are far from verifying observations made through computer experiments.

As a bridge connecting those experimental results and rigorous analyses, we have proposed [WST03] an approach for analyzing the average-case behavior of such local algorithms. (A similar approach has been also proposed by S. Cocco and R. Monasson [CM04].) For a given local search algorithm, the approach takes the following two approximation steps.

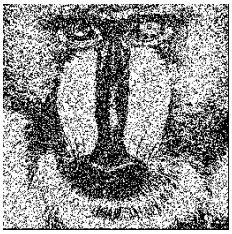
1. Modify the algorithm to a randomized one, and approximate its average execution by a simple Markov process.
2. Approximate this process, more specifically, its average states by simple probabilistic recurrence formula — *pseudo expectation*.

In that paper, the approach has been demonstrated by analyzing some randomized local search algorithm for solving some constraint satisfaction problem — LDPCC Decoding Problem [Gal62, Mac99]. But justifications to these two approximation steps have been left open. The purpose of this paper is to mathematically analyze an error bound of the pseudo expectation, for giving a justification to the second approximation step.

Example: Local Search for Image Restoration

We explain our approch and how pseudo expectation is used by using some concrete problem and its algorithm. For our concrete problem, we consider the image restration problem; that is, recovering the original image from a given image with noise. For example, from a given image like the left one of Figure 2, our task is to obtain the right one.

Here we consider the following simple situation:



restoration
$\Longrightarrow$
$\Longleftarrow$
noise
(by random bit flipping)

Figure 2: An example of our simple image restration

- An image is a 256×256 pixels (or, bits) of black(1)/white(0).
- Noise is just an i.i.d. flip at each bit, with some fixed flipping probability, say, 0.2 (= 20%).

Thus, our task is simply to determine, for each bit of 256×256 bits, whether the original is 0 or 1. Then one natural idea (for determining each bit) is to look around its neighbors, and flip the current bit if most of them have the opposite color. The algorithm of Figure 3, which was suggested by Prof. Kazuyuki Tanaka to the author, is based on this idea. This can be regarded as a simplified version of the algorithm that Tanaka etal have investigated [Tan01]. Clearly, this is a local search algorithm, and we consider this algorithm as our example.

Some explanation on the algorithm may be necessary. In the algorithm (and in the following discussion), we assume that an image data (with noise) is stored in the array `v[·, ·]`, where each `v[i,j]` is called a *variable*. For each variable `v[i,j]`, its *penalty* is the number of horizontal or vertical neighbors with the opposite assignment. Figure 1 shows examples of assignments of each penalty. (Precisely speaking, the notion of "penalty" is for the assignment to each variable; but in this paper, we simply say "the penalty of `v[i,j]`", meaning the penalty of the current assignment to the variable `v[i,j]`.)

At each step, the algorithm chooses one variable, which we call a *flipped variable*, from those with the highest penalty and flip its value. This local improvement process is repeated until some condition is fulfilled. Then the important point that we must determine is when we should stop this repetitions. If we repeated the process too long, then we would destroying the original image by flipping too many times. Tanaka etal proposed to use the "boundary length", the number of pair of adjacent variables with opposite assignments, and to terminate the process when the boundary length becomes the original one. The idea behind this is the conjecture that the image gets almost optimally close to the original at this point. (Here for simplicity, let us use the Hamming distance, the number of different bits, to measure the closeness of given two images.) One of the important contributions of their work is to develop a way to estimate this original boundary length.

Some computer experiments (conducted by the author) suggests that this stopping condition is reasonable (though not perfect for some images). Choose one of the images

program `LS_ImageRestoration`;
input: `v[0..255, 0..255]`, a degraded image data.
output: an image as close as the original one.
  repeat
    if some stopping condition H holds **then** halt;
    choose one `v[i,j]` randomly from variables with the highest penalty;
    flip the value of `v[i,j]`;
program end.

Figure 3: Local search image restration algorithm

that the stopping condition works well, and let L_0 and $L(t;\mathcal{N})$ denote the boundary length of, respectively, the original image and the image obtained after t steps of the local search from an input impage degrated by a randomly generated noise data $\mathcal{N}$. Similary, let $H(t;\mathcal{N})$ denote the Hamming distance from the original after t steps. Then it can be observed, for most noise $\mathcal{N}$, that both $H(t;\mathcal{N})$ and $D(t;\mathcal{N}) = |L(t;\mathcal{N}) - L_0|$ get decrease and both takes the minimum almost at the same step t. (Note that in our experiment we *know* the random noise and the original image, while they should be unknown to the algorithm.) Then we might want to analyze how these functions change on average. Our approach provides a way to give these functions in a relatively simple form.

The first step of our analysis is to regard the execution of the algorithm as a (relatively) simple Markov process. We start with modifying the algorithm to a fully randomized one. Note that the algorithm has a *deterministic* selection; that is, it first selects the set of variables with the highest penalty (and then choose one of them randomly). We introduce weights and change this "hard decision" to a "soft decision", by choosing each variable randomly with the following probability:

$$\Pr\{\text{ a var. of penalty } k \text{ is chosen }\} \;=\; \frac{w(k)X_k}{\sum_{i=0}^{4} w(i)X_i}, \tag{1}$$

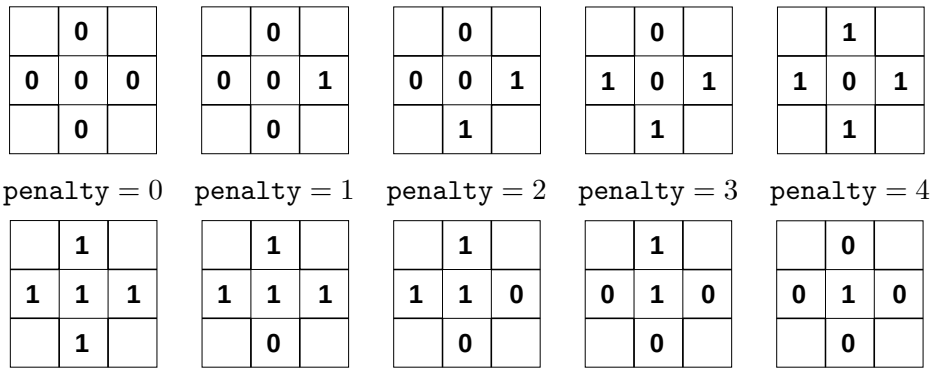


Figure 4: Examples of assignments with each penalty

where, for each i, $0 \leq i \leq 4$, $w(i)$ and X_i are respectively the weight of each variable with penalty i and the number of variables of penalty i. By choosing w such that $w(0) << w(1) << \cdots << w(4)$, we may assume that the modified algorithm executes almost the same way as the original one.

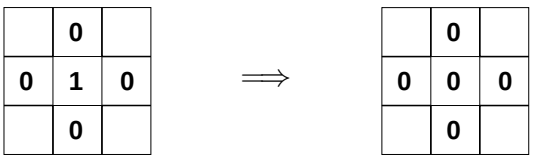
By this modification we can regard the execution of the algorithm as a Markov process; but for this, we need a huge state space, i.e., the set of all possible assignments to variables `v[·,·]`, which is of size 2^N, exponential to our size parameter N, the number of pixels. Our first approximation step is to simulate this algorithm as a Markov process with much smaller number of states, polynomially bounded by N. For example, we may express the state of the algorithm by a tuple $\boldsymbol{X} = (X_{0,+}, X_{0,-}, ..., X_{4,+}, X_{4,-})$, where each $X_{i,+}$ (resp., $X_{i,-}$) is the number of correctly (resp., incorrectly) assigned variables with penalty i. Note that each $X_{i,sg}$ takes an integer in the range $[0, N]$, and one state is determined by 10 parameters; thus, the size of states is bounded by $O(N^{10})$, which is still large but much smaller than 2^N.

Consider the simulation of the algorithm with these states. First we note that the information given by a tuple $\boldsymbol{X}$ is enough to compute the boundary length $L(t, \mathcal{N})$ and $H(t, \mathcal{N})$; in fact, they can computed as follows.

$$L(t,\mathcal{N}) \;=\; \sum_{i=0}^{4} i \cdot (X_{i,+} + X_{i,-})/2, \quad \text{and} \quad H(t,\mathcal{N}) \;=\; \sum_{i=0}^{4} X_{i,-},$$

where each $X_{i,sg}$ is, precisely speaking, a random variable showing the state of the process at tth step from the initial state determined by the given noise data $\mathcal{N}$. Also recall (1), the probability of choosing a flipped variable. Again our tuple $\boldsymbol{X}$ is enough for computing this probability and simulate the random choice of the algorithm. Unfortunately, however, the information is not enough to simulate (precisely) the effect of flipping to its four adjacent variables.

For discussing this point, consider the situation that the following flip is made on some variable `v[i,j]` at step t.



Assume that `v[i,j]` was incorrectly assigned before the flip. Then, by this flip, its assignment becomes correct, and its penalty is changed from 4 to 0; hence, $X_{4,-}$ gets decreased by one, whereas $X_{0,+}$ gets increased by one. Notice that this flip also causes the change of penalty at the four adjacent variables of `v[i,j]`. Suppose, for example, the penalty of `v[i,j-1]`, the left of `v[i,j]`, was 3 before the flip; then the flip changes it to 1, which causes the decrement of $X_{3,sg}$ and the increment of $X_{1,sg}$, where sg is the correct/incorrect status of `v[i,j-1]`. Clearly, the status of adjacent variables is not

At state $\boldsymbol{X} = (X_{0,+}, X_{0,-}, ..., X_{4,+}, X_{4,-})$, the process is executed as follows. (See the text for explanation.)

1. Choose the status (i, sg) of a flipped variable with prob. $w(i)X_{i,sg}/Z$.
2. For each direction $D \in \{\mathrm{N}, \mathrm{E}, \mathrm{S}, \mathrm{W}\}$,
   choose the status (i_D, sg_D) of the variable adjacent to the flipped variable with prob. $X_{i_D,sg_D}/N_{i,sg,D}$.
3. Update $\boldsymbol{X}$ to $\boldsymbol{X} + \boldsymbol{e}_{i,sg} + \boldsymbol{e}_{i_\mathrm{N},sg_\mathrm{N}} + \cdots + \boldsymbol{e}_{i_\mathrm{W},sg_\mathrm{W}}$.

Figure 5: Markov process for simulating `LS_ImageRestoration`

given in our simplified state. Our proposal is to choose them *uniformly at random* from all possible ones. For example, with probability $X_{3,+}/(X_{1,+} + X_{1,-} + \cdots + X_{4,+} + X_{4,-})$, we assume that the variable `v[i,j-1]` has status $(3, +)$, i.e., it has penalty 3 and it is correctly assigned. The same choice is made *independently* for the other three adjacent variables.

In summary, we consider the Markov process stated in Figure 5. Here Z is the total weight, i.e., $Z = \sum_{i=1}^{4} w(i)X_i$, of all candidates of flipped variables; on the other hand, $N_{i,sg,D}$ is the number of variables that can be adjacent to the direction D of the chosen flipped variable. Update vectors $\boldsymbol{e}_{i,sg}$, $\boldsymbol{e}_{i_\mathrm{N},sg_\mathrm{N}}$, ... are determined by the algorithm; for example, $\boldsymbol{e}_{3,+} = (0, 0, 0, +1, 0, 0, -1, 0, 0, 0)$.

This is our first step approximation. We conjecture that the average execution of a given algorithm on random inputs can be simulated well *to certain extent* if sufficiently large number of parameters are used for describing states. For our image restoration algorithm, describing states $\boldsymbol{X} = (X_{0,+}, X_{0,-}, ..., X_{4,+}, X_{4,-})$ by 10 parameters is not fine enough; but our preliminary experiments (see, e.g., Figure 7) show that the simulation becomes accurate if we describe a state by $(X_{i,sg})_{1 \le i \le 512, sg \in \{+,-\}}$, where each $X_{i,sg}$ denotes the number of correctly/incorrectly assigned variables with the ith configuration, like the one illustrated in Figure 6.

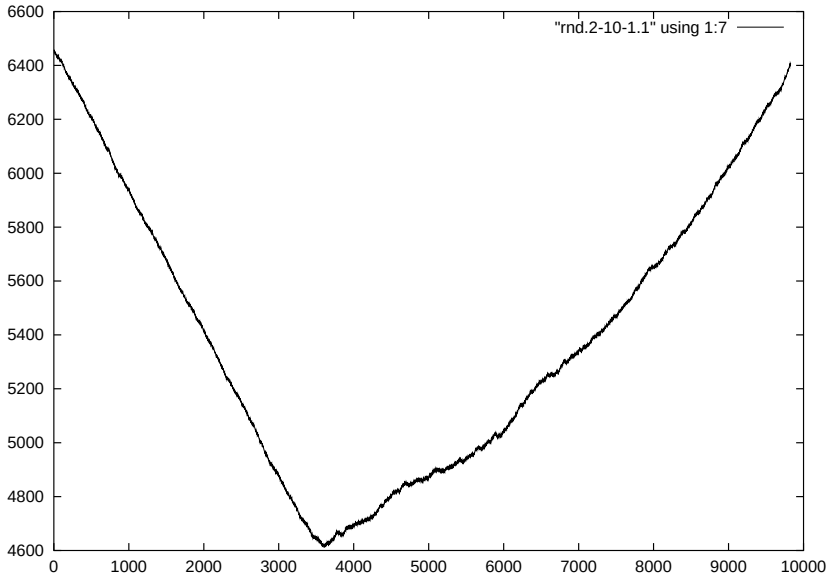
0	0	1
1	0	1
1	0	0

an example

We consider a *configuration* defined by the assignment to the variable considered and the assignments of all 9 neighboring variables. Thus, there are $2^{10} = 512$ configurations, each of which is given some index. The left figure shows an example of such configurations.

Figure 6: Variable configuration

Now assume that our first approximation works and the algorithm can be simulated by a relatively simple Markov process. But it is still hard to define a function, e.g., $H(t) = \mathrm{E}[H(t, \mathcal{N})]$, the average of $H(t, \mathcal{N})$ for a randomly generated noise $\mathcal{N}$. Our second approximation step provides a way to obtain such a formula as a simple recurrence formula.



Two graphs of $H(t, \mathcal{N})$, the total number of incorrectly assigned variables vs. step t, for a given input image created from the image of Figure 2 by randomly generated noise (10%). Solid line: the result of one execution of the algorithm. Dashed line: the result of one simulation (from the same input) by the simplified Markov process using states specified by 512×2 parameters.

Figure 7: Simulation by a simplified Markov process

Let M be a Markov process on the set $\mathcal{S}$ of states. Then by definition, we have a *transition function*, a function $\boldsymbol{f}$ on $\mathcal{S}$ such that

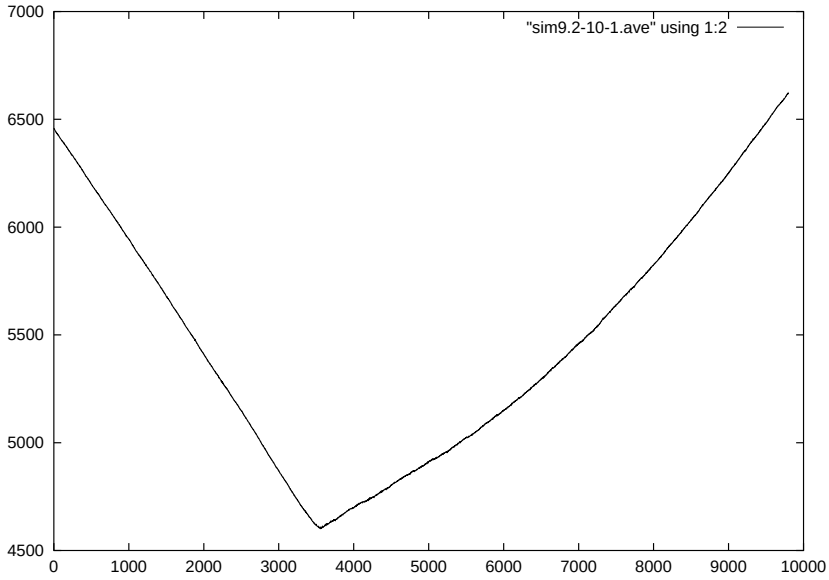
$$\mathrm{E}[\boldsymbol{S}^{(t+1)} | \boldsymbol{S}^{(t)} = \boldsymbol{s}] \;=\; \boldsymbol{f}(\boldsymbol{s})$$

holds for all $\boldsymbol{s} \in \mathcal{S}$, where $\boldsymbol{S}^{(t)}$ is the state of M at the tth step. Now our approach is to approximate $\mathrm{E}[\boldsymbol{S}^{(t)} | \boldsymbol{S}^{(0)} = \boldsymbol{s}]$ as follows.

$$\mathrm{E}[\boldsymbol{S}^{(t)} | \boldsymbol{S}^{(0)} = \boldsymbol{s}] \;\approx\; \boldsymbol{f}^t(\boldsymbol{s}). \tag{2}$$

This righthand side, the value obtained by simply applying $\boldsymbol{f}$ to an initial state $\boldsymbol{s}$ for t times, is called a *pseudo expectation*. We propose to use this pseudo expectation for giving a formula computing an expected state of the algorithm. For example, let $\boldsymbol{g}$ be the transition function for the Markov process simulating our image restoration algorithm. (For simplicity, let us use states expressed by 10 parameters.) Then our approach is to approximate $H(t, \mathcal{N})$ as follows.

$$H(t, \mathcal{N}) \;\approx\; \sum_{i=0}^{4} (\boldsymbol{g}^t(\boldsymbol{x}_{\mathcal{N}}))_{i,-}.$$



For the same data used in Figure 7, expectation and pseudo expectation are compared. Solid line: the average of 10 executions of the Markov process on the same input. Dashed line: the pseudo expectation from the same input.

Figure 8: Expectation and pseudo expectation

where $(\cdot)_{i,-}$ is the component corresponding to the status $(i,-)$, and $\boldsymbol{x}_{\mathcal{N}}$ is the initial state determined by a given noise data $\mathcal{N}$. For obtaining $H(t) = \mathrm{E}[H(t,\mathcal{N})]$, we simply have to average the righthand side over all 10% noise data.

Obviously the pseudo expectation cannot be precise in general, but again, our preliminary experiments (see, e.g., Figure 8) show that it is quite close to the real expectation. The technical goal of this paper is to estimate the difference, the error of the pseudo expectation. More specifically, for a general Markov process M and its transition function defined above, we want to give some upper bound on the following error function, for any index i.

$$err_i(t,\boldsymbol{s}) \stackrel{\text{def}}{=} (\mathrm{E}[\boldsymbol{S}^{(t)}|\boldsymbol{S}^{(0)} = \boldsymbol{s}])_i - (\boldsymbol{f}^t(\boldsymbol{s}))_i. \tag{3}$$

Let us summarize the type of transition functions that we would usually expect for Markov processes simulating local search algorithms. Since we may assume that such Markov processes follow the description stated in Figure 5, we again use it as our typical example. Let $\boldsymbol{g}$ be the transition function for the Markov process stated in Figure 5.

Then it is easy to see that

$$\begin{aligned} \boldsymbol{g}(\boldsymbol{x}) &= \boldsymbol{x} + \sum_{i,sg} p(i,sg)\boldsymbol{e}_{i,sg} + \sum_{D} \sum_{i,sg,i_D,sg_D} p(i,sg)q(i_D,sg_D)\boldsymbol{e}_{i_D,sg_D}, \\ \text{where} \quad p(i,sg) &= \frac{w(i)x_{i,sg}}{Z}, \quad q(i_D,sg_D) = \frac{x_{i_D,sg_D}}{N_{i,sg,D}}, \\ Z &= \sum_{i,sg} w(i)x_{i,sg}, \quad \text{and} \quad N_{i,sg,D} = \sum_{\text{possible } i_D,sg_D} x_{i_D,sg_D}. \end{aligned}$$

Note that both $q(i_D, sg_D)$ and $\boldsymbol{e}_{i_D,sg_D}$ depend on D, i, and sg; hence, more precisely, both $q(i_D, sg_D)$ and $\boldsymbol{e}_{i_D,sg_D}$ should be written as $q(D, i, sg, i_D, sg_D)$ and $\boldsymbol{e}_{D,i,sg,i_D,sg_D}$.

In general, we may assume that a transition function $\boldsymbol{f}$ is of the following form.

$$\begin{aligned} \boldsymbol{f}(\boldsymbol{x}) &= \boldsymbol{x} + \sum_{j} p(j)\boldsymbol{e}_j + \sum_{j}\sum_{k} p(j)q(j,k)\boldsymbol{e}_{j,k}, \qquad (4) \\ \text{where} \quad p(j) &= \frac{w_j x_j}{Z}, \qquad Z = \sum_{j} w_j x_j, \\ q(j,k) &= \frac{x_k}{N_{j,k}}, \quad N_{j,k} = \sum_{h \in A_j} x_h, \quad \text{and} \end{aligned}$$

A_j is the set of variable types that could be affected by the change on a type j variable.

Finally, we remark that the pseudo expectation becomes the real expectation if all divisors (i.e., Z and N_j's) defining probabilities were regarded as constants. For example, for the image restoration algorithm, if we used the same weights for all penalties (including 0 penalty), we would have $Z = N$ at any step; then ignoring the changes on $N_{i,sg,D}$, we can easily see that our pseudo expectation is close to the real one. But as we have seen, weights are important for simulating hard decisions with soft ones.

2 Error Bound: One-Dimensional Case

We analyze the following simple one-dimensional Markov process and its pseudo expectation.

Markov Process:

Let N and W be given integers. For any state $A^{(t)}$, $0 \le A^{(t)} \le N$, at step $t \ge 0$, the next state $A^{(t+1)}$ is determined as follows.

$$A^{(t+1)} = \begin{cases} A^{(t)} - 1 & \text{with prob. } p(A^{(t)}), \text{ and} \\ A^{(t)} + 1 & \text{with prob. } q(A^{(t)}), \end{cases}$$

where $q(x) = 1 - p(x)$, and $p(x)$ is defined as follows.

$$p(x) = \frac{Wx}{Wx + (N - x)} = \frac{Wx}{N + (W - 1)x}.$$

If $W < 1$, then we may consider $N - A^{(t)}$ as a state, which uses a weight $1/W > 1$. Thus, without losing generality, we may assume $W \geq 1$.

It would be more intuitive if we express a state by a pair $(X_1^{(t)}, X_2^{(t)})$ of random variables and interpret $X_i^{(t)}$ as a number of balls that player i has after the tth step. At each step t, with probability $p(X_1^{(t)})$ player 1 passes one ball (which is selected randomly) to player 2, and the other way around with probability $1 - p(X_1^{(t)})$. Note that the weight W is given to player 1's balls, while each ball of player 2 has weight 1. Thus, this is the simplest case of the Markov process defined in the previous section. Since the total number of balls, $X_1^{(t)} + X_2^{(t)}$, does not change, we can express this process by a single variable $A^{(t)}$.

Define $f(x)$ as follows:

$$f(x) \;=\; p(x)(x-1) + q(x)(x+1) \;=\; x + 1 - \frac{2Wx}{N + (W-1)x}.$$

Then we have $f(A^{(t)}) = \mathrm{E}[A^{(t+1)}|A^{(t)}]$. Thus, for any a, $0 \leq a \leq N$, $f^t(a)$ is the pseudo expectation at the tth step when started from $A^{(0)} = a$, and the error at the tth step is defined by

$$\mathit{err}(t, a) \;=\; \mathrm{E}[A^{(t)}|A^{(0)} = a] - f^t(a).$$

(We will see below (Lemma 2) that $\mathrm{E}[A^{(t)}|A^{(0)} = a] \geq f^t(a)$ for this simple process. In the following, we will often write simply, e.g., $\mathrm{E}[A^{(t)}|a]$ for $\mathrm{E}[A^{(t)}|A^{(0)} = a]$.)

Note that $p(0) = 0$ and $p(N) = 1$ (hence, $f(0) = 1$ and $f(N) = N - 1$). Hence, $A^{(t)}$ takes values from 0 to N (and so is $f(x)$ if $x \in [0, N]$). Thus, we have an obvious bound $\mathit{err}(t, a) \leq N$. What we would like to show is something significantly smaller than N; for example, a constant independent from N (nor W). We will prove the following upper bound.

Theorem 1. Assume that $N \geq W$. Then there are some constants c_1 and c_2 independent from N and W such that the following holds for any t and a.

$$\mathit{err}(t, a) \;=\; \mathrm{E}[A_t|a] - f^t(a) \;\leq\; c_1 W + c_2.$$

Remark on the Error in the Limit

Since our process is simple, we can give a precise formula for its stationary distribution[1] as follows. (This analysis has been done by Johannes Schneider in his Master's thesis, and the proof is omitted here.)

[1]This Markov process is periodic with period 2. Thus, no stationary distribution exists. In the following, we consider states at even and odd steps separately. By the "average stationary state", we mean the arithmetic mean of the expectations for even and odd steps.

Proposition 2. Consider the process starting from $A^{(0)} = a$. Then we have

$$\begin{aligned}\lim_{h\to\infty} \Pr[\, A^{(2h)} = m \,] &= \begin{cases} 2P_\infty, & \text{if both } a+m \text{ is even, and} \\ 0, & \text{otherwise.} \end{cases} \\ \lim_{h\to\infty} \Pr[\, A^{(2h+1)} = m \,] &= \begin{cases} 2P_\infty, & \text{if both } a+m \text{ is odd, and} \\ 0, & \text{otherwise.} \end{cases}\end{aligned}$$

Where

$$P_\infty \;=\; \frac{1}{2} \cdot \frac{(W^m \cdot N + W^{m-1} \cdot m \cdot (1-W))}{N \cdot (1+W)^{N-1}} \cdot \binom{N}{m}.$$

The average stationary distribution is then calculated as follows.

Corollary 3.

$$a_{\mathrm{lim}} \stackrel{\mathrm{def}}{=} \frac{\lim \mathrm{E}[A^{(2h)}|a] + \mathrm{E}[A^{(2h+1)}|a]}{2} \;=\; \frac{N + (W-1)/2}{W+1}.$$

On the other hand, the fixed point a_{fix} of f is calculated as follows; note that $\lim_{t\to\infty} f^t(a) = a_{\mathrm{fix}}$ for any a.

$$a_{\mathrm{fix}} \;=\; \frac{N}{W+1}. \tag{5}$$

Hence, in the limit, the error of the pseudo expectation is quite small.

$$\lim_{t\to\infty} \mathit{err}(t,a) \;=\; a_{\mathrm{lim}} - a_{\mathrm{fix}} \;=\; \frac{1}{2} - \frac{1}{W+1}.$$

But unfortunately, $\mathit{err}(t,a)$ is not monotone in general, and this is not a general upper bound.

Proof of Theorem 1

As remarked in the previous section, the pseudo expectation is accurate when $W = 1$, because the function f becomes "linear". Our approach is to estimate the nonlinearity of f^t. More specifically, we introduce the following function, which we call *nonlinearity* of f^t at x.

$$d\mathit{L}(t,x) \;=\; p(x)f^t(x-1) + q(x)f^t(x+1) - f^t\Big(p(x)(x-1) + q(x)(x+1)\Big),$$

To make this definition valid at the boundaries, we let $f^t(-1) = f^t(N+1) = 0$. In fact, it is easy to see that $d\mathit{L}(t,0) = d\mathit{L}(t,N) = 0$.

The following relation, though easy to show, by induction, plays a key role in our analysis.

Lemma 1. For any $t \geq 1$ and any a, $0 \leq a \leq N$,

$$\mathit{err}(t,a) \;=\; p(a)\mathit{err}(t-1,a-1) + q(a)\mathit{err}(t-1,a+1) + d\mathit{L}(t-1,a).$$

Here and in the following, we let $\mathit{err}(t,-1) = \mathit{err}(t,N+1) = 0$.

Proof. Consider the state $A^{(1)}$ after the first transition from $A^{(0)} = a$. Since either $A^{(1)} = a - 1$ with probability $p(a)$, or $A^{(1)} = a + 1$ with probability $q(a)$, we have

$$\begin{aligned} \mathrm{E}[A^{(t)}|a] &= p(a)\mathrm{E}[A^{(t)}|A^{(1)} = a - 1] + q(a)\mathrm{E}[A^{(t)}|A^{(1)} = a + 1] \\ &= p(a)\mathrm{E}[A^{(t-1)}|a - 1] + q(a)\mathrm{E}[A^{(t-1)}|a + 1]. \end{aligned}$$

On the other hand, we have

$$f^t(a) \;=\; f^{t-1}(f(a)) \;=\; f^{t-1}\Big(p(a)(a-1) + q(a)(a+1)\Big).$$

Then the recurrence formula of the lemma is immediate from these two equations. □

From this lemma, we have the following bound:

$$err(t, a) \;\leq\; \sum_{1 \leq h \leq t-1} \max_{x \in [N]} dL(h, x). \tag{6}$$

Recall that $dL(h, 0) = dL(h, N) = 0$; hence, for the maximum, we only have to consider the range $\{1, ..., N-1\}$. Thus, in the following, by $\max_x dL(h, x)$ we mean $\max_{x \in \{1,...,N-1\}} dL(h, x)$. In more general, $dL(h, x)$ is estimated not only on integral x; in such a case, we assume that x is taken from $(0, N)$.

Note the following basic properties of our f^t. (Since these properties are proved easily by simple induction, their proofs are omitted here.)

Fact 1. For any t, the following holds:
(1) both $f^t(x)$ and $(f^t)'(x)$ are monotone and increasing,
(2) $0 \leq f^t(x) \leq x$, $0 \leq (f^t)'(x) \leq 1$, and
(3) $f^{t+1}(x + 1) - f^{t+1}(x - 1) \leq f^t(x + 1) - f^t(x - 1) \leq 2$.

From the monotonicity of $(f^t)'(x)$, the function $f^t(x)$ must be convex, which implies the following properties.

Lemma 2. The nonlinearity $dL(h, x)$ is nonnegative for any $h \geq 0$ and x. Thus, $\mathrm{E}[A_t|a] \geq f^t(a)$ for any t and a.

Let us replace $dL(h, x)$ with a function that is easier to compute. As one can easily expect, the nonlinearity $dL(h, x)$ can be bounded by the second derivative. But here by using the monotonicity of f^h and $(f^t)'$, we can also prove the following bound. (Since a more general bound will be proved for Lemma 5, the proof is omitted here.)

Lemma 3. For any $h \geq 0$ and x, we have

$$dL(h, x) \;\leq\; \frac{1}{2}((f^h)'(x + 1) - (f^h)'(x - 1)).$$

Thus, define $df(h,x) = (f^h)'(x+1) - (f^h)'(x-1)$, and in the following, we will bound $\max_x df(h,x)$. Again unless explicitly stated, we will assume that $x \in \{1, ..., N-1\}$, or $x \in (0, N)$ in general. Also h will be a nonnegative integer.

Note first that $df(0,x) = (f^0)'(x+1) - (f^0)'(x-1) = 0$. On the other hand, for any $h \geq 1$, we have the following bound.

$$\begin{aligned}
df(h,x) &= (f^h)'(x+1) - (f^h)'(x-1) \\
&= f'((f^{h-1}(x+1))(f^{h-1})'(x+1) - f'((f^{h-1}(x-1))(f^{h-1})'(x-1) \\
&= \left(1 - \frac{2WN}{(N+(W-1)f^{h-1}(x+1))^2}\right)(f^{h-1})'(x+1) \\
&\qquad - \left(1 - \frac{2WN}{(N+(W-1)f^{h-1}(x-1))^2}\right)(f^{h-1})'(x-1) \\
&= df(h-1,x)\left(1 - \frac{2WN}{(N+(W-1)f^{h-1}(x+1))^2}\right) \\
&\qquad + (f^{h-1})'(x-1)\left(\frac{2WN}{(N+(W-1)f^{h-1}(x-1))^2} - \frac{2WN}{(N+(W-1)f^{h-1}(x+1))^2}\right) \\
&\leq df(h-1,x)\left(1 - \frac{2WN}{(N+(W-1)f^{h-1}(x+1))^2}\right) + \frac{8W^2N \cdot (f^{h-1})'(x-1)}{(N+(W-1)f^{h-1}(x-1))^3}.
\end{aligned}$$

For the last bound, we used the fact that $f^{h-1}(x+1) - f^{h-1}(x-1) \leq 2$. Here note that

$$(f^{h-1})'(x-1) \;\leq\; (f^{h-1})'(x+1) \;=\; \prod_{i=0}^{h-1}\left(1 - \frac{2WN}{(N+(W-1)f^i(x+1))^2}\right).$$

Hence by letting

$$\alpha(i) \;=\; 1 - \frac{2WN}{(N+(W-1)f^i(x+1))^2}, \quad \text{and} \quad \beta(i) \;=\; \frac{8W^2N}{(N+(W-1)f^i(x-1))^3},$$

we can write the bound as

$$df(h,x) \;\leq\; \alpha(h-1)df(h-1,x) + \beta(h-1)\prod_{i=0}^{h-2}\alpha(i),$$

which is in a closed form as follows.

$$df(h,x) \;\leq\; \sum_{i=0}^{h-1}\left(\prod_{j=0}^{h-1}\alpha(j)\right)\beta(i)\alpha(i)^{-1}. \tag{7}$$

Let us define $u(h,x)$ by the righthand side expression, and in the following, we discuss a bound for this $u(h,x)$.

Consider any x and h, and let them be fixed for a while. Now we introduce an important parameter $R = N/W$, which is larger than equal to 1 when $W \leq N$ (as assumed in our theorem). Recall that $a_{\text{fix}} = N/(W+1)$ ($\approx R$) is the fixed point of f;

hence, $f^i(x-1)$ gets decreased if $x-1$ is much larger than R, say, $x-1 > 2R$. A technical key of our analysis is to bound terms in $u(h,x)$ by stages depending on $f^i(x-1)$ w.r.t. R. More specifically, we consider the following K stages, where $h_0 = 0$, $K = W-2$, and $h_{K+1} = h$.

$$\begin{array}{lrcccl}
i = h_0 \sim h_1 - 1 & (W-1)R & < & f^i(x-1) & \leq & WR\ (= N) \\
i = h_1 \sim h_2 - 1 & (W-2)R & < & f^i(x-1) & \leq & (W-1)R \\
i = h_2 \sim h_3 - 1 & (W-3)R & < & f^i(x-1) & \leq & (W-2)R \\
\quad\vdots & & & \vdots & & \\
i = h_{K-1} \sim h_K - 1 & 2R & < & f^i(x-1) & \leq & 3R \\
i = h_K \sim h - 1 & 0 & \leq & f^i(x-1) & \leq & 2R.
\end{array}$$

Since $f^i(x-1)$ gets decreased if $x-1 > 2R$, the sequence $h_0 \leq h_1 \leq \cdots \leq h_K$ can be defined. Note that if $x-1 \leq cR$ for some integer $c < W$, then we would have $h_1 = h_2 = \cdots = h_{W-c} = 0$. Similarly, if $cR < f^{h-1}(x-1) \leq (c+1)R$ for some integer $c \geq 2$, then we would have $h_{W-c} = h_{W-c+1} = \cdots = h_K = h$. Thus, these definitions are valid for all $x-1$ and h. For each k, $0 \leq k \leq K$, let $c_k = W-k$ and $s_k = h_{k+1} - h_k$; $s_k \geq 0$ is the length of each stage.

Consider any $i \in \{h_k, ..., h_{k+1}-1\}$ such that $(c_k-1)R < f^i(x-1) \leq c_k R$ holds with some $c_k \geq 3$. Then $(c_k-1)R < f^i(x+1) \leq c_k R + 2$; hence, we have

$$\begin{aligned}
\alpha(i) &= 1 - \frac{2WN}{(N+(W-1)f^i(x+1))^2} = 1 - \frac{2}{R(c_k+1+2/R)^2} \leq 1 - \frac{2}{R(c_k+3)^2}, \text{ and} \\
\beta(i) &= \frac{8W^2N}{(N+(W-1)f^i(x-1))^3} = \frac{8}{(c_k-1)^3R^2}.
\end{aligned}$$

By another simple calculation, we can also show the same bound (with $c_K = 2$) for any i such that $0 \leq f^i(x-1) \leq 2R$.

We denote the above bound for $\alpha(i)$ by $\gamma(c)$. That is, define $\gamma(c)$ by

$$\gamma(c) = 1 - \frac{2}{R(c+3)^2}.$$

Then by using the above bounds for α and β, we can bound $u(h,x)$ (the bound of (7)) as follows. (Recall $c_K = 2$.)

$$\begin{aligned}
u(h,x) &= \sum_{k=0}^{K} \sum_{i=h_k}^{h_{k+1}-1} \left(\prod_{j=0}^{h-1} \alpha(j) \right) \beta(i)\alpha(i)^{-1} \leq \sum_{k=0}^{K} \frac{8s_k}{(c_k-1)^3R^2} \left(\prod_{k=0}^{K} \gamma(c_k)^{s_k} \right) \gamma(c_k)^{-1} \\
&\leq \sum_{k=0}^{K-1} \frac{8s_k\gamma(2)^{s_K}}{(c_k-1)^3R^2} + \frac{8s_K\gamma(2)^{s_K-1}}{R^2}.
\end{aligned}$$

Here we used the fact that $\gamma(c) \leq 1$ for any $c \geq 2$, which follows from $R \geq 1$; this is where the assumption that $W \leq N$ is used.

We need some knowledge on the convergence speed of f^t. The following simple bound is enough for our analysis.

Fact 2. For any $x \in (0, N)$ and for any t, we have

$$f^t(x) \geq 2R \;\Rightarrow\; f^{t+1}(x) < f^t(x) - 1/3.$$

Now consider the last bound for $u(h, x)$. Note that what is dependent on x is only on the choice of $s_0, ..., s_K$. On the other hand, we know from the above fact that each s_k is at most $3R$ (except for s_K). Thus, we have, for any x,

$$\begin{aligned} u(h,x) &\leq \sum_{k=0}^{K-1} \frac{8s_k\gamma(2)^{s_K}}{(c_k-1)^3R^2} + \frac{8s_K\gamma(2)^{s_K-1}}{R^2} \;\leq\; \frac{\gamma(2)^s}{R}\sum_{k=0}^{K-1}\frac{24}{(c_k-1)^3} + \frac{8s_K\gamma(2)^{s_K-1}}{R^2} \\ &\leq \frac{6\gamma(2)^s}{R} + \frac{8s_K\gamma(2)^{s_K-1}}{R^2}, \end{aligned}$$

where $s = \max(h - 3RW, 0)$, which choice follows from $s_K \geq h - 3R(W-2) \geq h - 3RW$. (Note that $s_K = 0$ if $h \leq 3R(W-2)$.) The last bound is from the fact that $2^{-3}+3^{-3}+\cdots < 1/4$. Noting that $t\gamma(2)^t$ is increasing for $t < 11.5R$ and decreasing for $t \geq 11.5$, we further have

$$u(h,x) \;\leq\; \frac{6\gamma(2)^s}{R} + \begin{cases} \dfrac{92R\gamma(2)^{11.5R}}{R^2}, & \text{if } h < (3R(W-2) + 11.5R, \text{ and} \\ \dfrac{8t\gamma(2)^t}{R^2}, & \text{otherwise (here let } t = h - 3R(W-2)). \end{cases}$$

Therefore, we finally have the following desired bound.

$$\begin{aligned} \sum_{h\geq 0}\max_x u(h,x) &\leq \sum_{h=0}^{3RW-1}\frac{6}{R} + \sum_{s\geq 0}\frac{6\gamma(2)^s}{R} + \frac{(3R(W-2)+1)(92R)\gamma(2)^{11.5R}}{R^2} + \sum_{t\geq 0}\frac{8t\gamma(2)^t}{R^2} \\ &\leq 18W + 75 + (3\cdot 92\cdot e^{-11.5/12.5})W + 1250 \;\leq\; 128W + 1325. \end{aligned}$$

Bound for Large Weight W

Theorem 1 gives a reasonable bound when W is regarded as a constant. On the other hand, it would not be satisfiable for large W; in fact, our above analysis is not applicable for the case $W > N$. On the other hand, for the case that W is proportional to N, *some* bound is provable by a much simpler argument.

Consider, for example, the case that $W = N$. Note first that $a_{\text{lim}} \approx a_{\text{fix}} \approx N/W = 1$; that is, from any state, the process converges to 1. In fact, we have

$$p(x) \;=\; \frac{Wx}{(N + (W-1)x)} \;\approx\; \frac{Nx}{N(1+x)} \;=\; \frac{x}{1+x}.$$

Hence $p(x)$ is close to 1 for large x; this means that for such large x, x gets decreased by 1 almost deterministically. Similarly, since we have

$$f^t(x) \;=\; f^{t-1}(x) - 1 + \frac{2Wf^{t-1}(x)}{N + (W-1)f^{t-1}(x)} \;\approx\; f^{t-1}(x) - 1 + \frac{2}{1 + f^{t-1}(x)},$$

$f^t(x)$ gets decreased almost by 1 from $f^{t-1}(x)$ so long as $f^{t-1}(x)$ is large. Thus, in such a situation that we may assume that $A^{(t)}$ is large, both the real expectation $\mathrm{E}[A^{(t)}|a]$ and the pseudo expectation $f^t(a)$ can be approximated as $a-t$, and hence, only small difference would be expected between them. This observation leads us to the following bound.

Theorem 4. Let $\alpha = (W-1)/N > 0$. Then there exists a constant c_3 independent from N and W such that the following holds for any t and a.

$$err(t,a) \;\le\; \frac{c_3}{\sqrt{\alpha}} \cdot \sqrt{N}.$$

Proof. We may assume that $\alpha > 1/N$; otherwise, we can assume that W is small, and Theorem 1 is applicable. Note that both a_{fix} and a_{lim}, which can be approximated as $1/\alpha$, are much less than $\sqrt{N/\alpha}$ $(= \sqrt{\alpha N} \cdot (1/\alpha))$. Hence, for any initial state $a \le \sqrt{N/\alpha}$, and for any t, we certainly have both $\mathrm{E}[A^{(t)}|a] \le \sqrt{N/\alpha}$ and $f^t(a) \le \sqrt{N/\alpha}$; that is, the bound trivially holds with $c_3 = 1$. Thus, we only need to consider the case that the process starts from some $a > \sqrt{N/\alpha}$.

Consider the following simple Markov process $\{X_t\}_{t\ge 0}$:

for any $x > \sqrt{N/\alpha}$,

$$\begin{aligned}
\Pr[\, X_t = X_{t-1} - 1 \,|\, X_{t-1} = x \,\} &= p_0 \stackrel{\mathrm{def}}{=} \frac{\sqrt{\alpha N}}{1+\sqrt{\alpha N}},\\
\Pr[\, X_t = X_{t-1} + 1 \,|\, X_{t-1} = x \,\} &= 1 - p_0 = \frac{1}{1+\sqrt{\alpha N}};\quad \text{and}
\end{aligned}$$

for any $x \le \sqrt{N/\alpha}$,

$$\Pr[\, X_t = X_{t-1} \,|\, X_{t-1} = x \,\} = 1.$$

Note that for any $x > \sqrt{N/\alpha}$, we have

$$\Big(\Pr[\, A^{(t)} = A^{(t-1)} - 1 \,|\, A^{(t-1)} = x \,] = \Big) \;=\; p(x) \;=\; \frac{Wx}{N+(W-1)x} \;\ge\; \frac{\alpha x}{1+\alpha x} \;>\; \frac{\sqrt{\alpha N}}{1+\sqrt{\alpha N}}.$$

Hence it holds that

$$\mathrm{E}[X_t|a] \;\ge\; \mathrm{E}[A^{(t)}|a] \;\ge\; f^t(a).$$

On the other hand, since

$$\mathrm{E}[X_t|X_{t-1}=x] \;=\; x - \frac{\sqrt{\alpha N}}{1+\sqrt{\alpha N}} + \frac{1}{1+\sqrt{\alpha N}} \;=\; x - 1 + \frac{2}{1+\sqrt{\alpha N}} \;\le\; x - 1 + \frac{2}{\sqrt{\alpha N}}$$

and $f^t(x) \ge f^{t-1}(x) - 1$, we have

$$a - t + \frac{2t}{\sqrt{\alpha N}} \;\ge\; \mathrm{E}[X_t|a] \;\ge\; \mathrm{E}[A^{(t)}|a] \;\ge\; f^t(a) \;\ge\; a - t.$$

This implies the following bound for any $t \leq N$.

$$err(t,a) \;=\; \mathrm{E}[A^{(t)}|a] - f^t(a) \;\leq\; \frac{2N}{\sqrt{\alpha N}} \;\leq\; 2\sqrt{\frac{N}{\alpha}}.$$

On the other hand, if $t > N$, the above error bound is clearly satisfied because both $\mathrm{E}[X_t|a]$ and $f^t(x)$ are smaller than $\sqrt{N/\alpha}$. □

3 Error Bound: General Case

Although simple, the analysis of the one-dimensional Markov process suggested us a way to obtain a general error bound. Unfortunately, the detail analysis we did for the one-dimensional case is impossible for the general case; nevertheless, following the same approach, we still can get some upper bound under certain conditions.

We begin with clarifying the conditions we will assume in our analysis. For this, let us recall our general Markov process discussed in Section 1, and define symbols and constants that will be used below. We consider a Markov process whose state is expressed by a D-dimensional vector $\boldsymbol{x} = (x_1, ..., x_D)$, where each x_i takes an integer[2] value in $\{0, ..., N\}$, and we assume that $\sum_i x_i = N$. Let $\mathcal{X}$ denote the set of all such vectors; this is the domain of states of our Markov process. Intuitively, x_i is the number of variables of type i; then w_i is the weight of a variable of type i. The transition of our Markov process is given as (4). Here we specify it a bit more in detail. Let $\mathcal{E}$ denote the set of all update vectors, i.e., $\boldsymbol{e}_j$'s and $\boldsymbol{e}_{j,k}$'s, where the range of j is $\{1, ..., D\}$, and that of k is $\{1, ..., K\}$ for some constant K. Define also the following parameters and constants.

$$w_{\max} \;=\; \max_j w_j, \quad W \;=\; \sum_j w_j, \quad \text{and} \quad e_{\max} \;=\; \max_{\boldsymbol{e}\in\mathcal{E},j} |(\boldsymbol{e})_j|.$$

Throughout this section, let t be any nonnegative integer, and let i, j, and k be any indices, $1 \leq i, j \leq D$, and $1 \leq k \leq K$. Also let $\boldsymbol{x}$ be any element of $\mathcal{X}$. By f_i^t we denote a function computing the ith coordinate of $\boldsymbol{f}^t$.

For a given $\boldsymbol{x} \in \mathcal{X}$, let $\mathcal{N}(\boldsymbol{x})$ denote the convex hull of $\boldsymbol{x} + \mathcal{E} \stackrel{\text{def}}{=} \{\boldsymbol{x} + \boldsymbol{e} | \boldsymbol{e} \in \mathcal{E}\}$, which we consider as a neighbor of $\boldsymbol{x}$. In fact, for Markov processes simulating local search algorithms, we may assume that updates are small; hence, $\mathcal{N}(\boldsymbol{x})$ can be regarded as a small neighbor in the domain $\mathcal{X}$.

Now we state our conditions.

(1) f_i^t is convex on $\mathcal{N}(\boldsymbol{x})$.

(2) The absolute value of the partial derivative $\frac{\partial f_j^t}{\partial x_i}$ is bounded by some constant c_{pd}. Hence, for any $\boldsymbol{y} \in \mathcal{N}(\boldsymbol{x})$, $|f_i^t(\boldsymbol{y}) - f_i^t(\boldsymbol{x})|$ is bounded by some constant c_{f}.

(3) During the execution, total weight Z is larger than $c_{\mathrm{Z}}N$ for some constant $c_{\mathrm{Z}} > 0$.

[2]Later in the analysis, we will consider, for x_i, any number in $[0, N]$.

(4) Each $q(j,k)$ can be regarded as a constant.

Except for the last one, these conditions are reasonable to expect. The last one is for simplifying our calculation; without it, we need more complicated but tedious calculation, but it is not essential. In fact, even if $q(j,k)$ is not a constant, it changes very mildly because $q(j,k) = x_k/(\sum_{h\in A_j} x_h)$ is defined without any weight, and the change of each x_i is small. Under this condition (4), the transition at the ith coordinate can be stated as follows with some constants a_j, $1 \le j \le D$.

$$f_i(\boldsymbol{x}) \;=\; x_i + \frac{\sum_j a_j w_j x_j}{Z}. \tag{8}$$

We will use this for our analysis.

Now we are ready to state our bound.

Theorem 5. Assume that the conditions (1) $\sim$ (4) hold. Then there are some constants d_1 and d_2 such that the following holds for any i, $1 \le i \le D$, $t \le N/(d_1 W)$, and $\boldsymbol{x} \in \mathcal{X}$.

$$err_i(t,\boldsymbol{x}) \;\le\; d_2 w_{\max}.$$

Remark. We can choose constants d_1 and d_2 as follows. (These choices are from rough estimation, and by more careful analysis, much smaller constants could be chosen.)

$$\begin{aligned} d_1 &= c_{\mathrm{Z}}/(4e_{\max}K), \quad \text{and} \\ d_2 &= ec_{\mathrm{f}}c_{\mathrm{pd}}e_{\max}^2 D\log_2(DK)/c_{\mathrm{Z}}. \end{aligned}$$

The proof follows almost the same outline as the one-dimensional case. In the following, consider any i and let it be fixed. As before, we first bound the error by the linearity of f_i^t, which is now defined as follows.

$$dL_i(t,\boldsymbol{x}) \;=\; \left| \sum_j \sum_k p(j)q(j,k) f_i^t(\boldsymbol{x}+\boldsymbol{e}_j+\boldsymbol{e}_{j,k}) - f_i^t\left(\sum_j \sum_k p(j)q(j,k)(\boldsymbol{x}+\boldsymbol{e}_j+\boldsymbol{e}_{j,k}) \right) \right|.$$

Then the following bound is provable by an argument almost the same as before.

Lemma 4. For any $t \ge 1$ and any $\boldsymbol{x}$,

$$err_i(t,\boldsymbol{x}) \;=\; dL_i(t-1,\boldsymbol{x}) + \sum_j \sum_k p(j)q(j,k)\, err_i(t-1,\boldsymbol{x}+\boldsymbol{e}_j+\boldsymbol{e}_{j,k}).$$

Then we have the following bound.

$$err_i(t,\boldsymbol{x}) \;\le\; \sum_{1\le u\le t-1} \max_{\boldsymbol{y}} dL_i(u,\boldsymbol{y}). \tag{9}$$

Hence, our goal is to bound $dL_i(t,\boldsymbol{x})$, and for this, we prove the following lemma corresponding to Lemma 3.

Lemma 5. Let f be any function from $\mathcal{X}$ to $\mathbf{R}$. For any $\boldsymbol{x}$ in $\mathcal{X}$, and for any $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$ such that both $\boldsymbol{x}+\boldsymbol{e}_1$ and $\boldsymbol{x}+\boldsymbol{e}_2$ belong to $\mathcal{X}$, suppose that the following function F is convex on $[0,1]$.

$$F(\alpha) \;=\; f(\boldsymbol{x}+\boldsymbol{e}_1+\alpha(\boldsymbol{e}_2-\boldsymbol{e}_1)).$$

Then for any p_1 and p_2 such that $p_1, p_2 \geq 0$ and $p_1+p_2=1$, we have

$$dL \;\stackrel{\text{def}}{=}\; |(p_1 f(\boldsymbol{x}+\boldsymbol{e}_1)+p_2 f(\boldsymbol{x}+\boldsymbol{e}_2))-f(\boldsymbol{x}+\boldsymbol{e})| \;\leq\; \frac{1}{4}(F'(1)-F'(0)),$$

Proof. We prove for the case that $(p_1 f(\boldsymbol{x}+\boldsymbol{e}_1)+p_2 f(\boldsymbol{x}+\boldsymbol{e}_2))-f(\boldsymbol{x}+\boldsymbol{e})>0$. Because F is convex, we have

$$\begin{aligned} f(\boldsymbol{x}+\boldsymbol{e}) &= F(p_2) \;\geq\; F(0)+p_2F'(0), \quad \text{and} \\ f(\boldsymbol{x}+\boldsymbol{e}) &= F(1-p_1) \;\geq\; F(1)-p_1F'(1). \end{aligned}$$

Hence

$$\begin{aligned} dL &= (p_1 f(\boldsymbol{x}+\boldsymbol{e}_1)+p_2 f(\boldsymbol{x}+\boldsymbol{e}_2))-f(\boldsymbol{x}+\boldsymbol{e}) \\ &\leq (p_1 f(\boldsymbol{x}+\boldsymbol{e}_1)+p_2 f(\boldsymbol{x}+\boldsymbol{e}_2))-(p_1(F(0)+p_2F'(0))+p_2(F(1)-p_1F'(1))) \\ &\leq p_1p_2(F'(1)-F'(0)) \;\leq\; \tfrac{1}{4}(F'(1)-F'(0)). \end{aligned}$$

□

Before using this lemma, we define the following value for each $\boldsymbol{x}$.

$$dfmax_i(t,\boldsymbol{x}) \;=\; \max_h \max_{\boldsymbol{u},\boldsymbol{v}\in\mathcal{N}(\boldsymbol{x})} \left| \frac{\partial}{\partial x_h} f_i^t(\boldsymbol{u}) - \frac{\partial}{\partial x_h} f_i^t(\boldsymbol{v}) \right|. \tag{10}$$

Now we are ready to state the following corollary.

Corollary 6. For any $t \geq 1$ and $\boldsymbol{x}$, we have

$$dL_i(t,\boldsymbol{x}) \;\leq\; \frac{De_{\max}\log_2(DK)}{2} dfmax_i(\boldsymbol{x}).$$

Proof. We prove by a small example. Suppose that $dL_i(t,\boldsymbol{x})$ is defined by

$$dL_i(t,\boldsymbol{x}) \;=\; \sum_{i=k}^{4} p_k f_i^t(\boldsymbol{x}+\boldsymbol{e}_k) - f_i^t\left(\sum_{i=k}^{4} p_k(\boldsymbol{x}+\boldsymbol{e}_k)\right) \;>\; 0.$$

Let $q_1=p_1+p_2$ and $q_2=p_3+p_4$, and introduce the following vectors.

$$\boldsymbol{e}_{12} \;=\; \frac{p_1}{q_1}\boldsymbol{e}_1+\frac{p_2}{q_1}\boldsymbol{e}_2, \quad \text{and} \quad \boldsymbol{e}_{34} \;=\; \frac{p_3}{q_2}\boldsymbol{e}_3+\frac{p_3}{q_2}\boldsymbol{e}_4.$$

Note that

$$f_i^t(q_1(\boldsymbol{x}+\boldsymbol{e}_{12})+q_2(\boldsymbol{x}+\boldsymbol{e}_{34})) \;=\; f_i^t\left(\sum_{i=k}^{4} p_k(\boldsymbol{x}+\boldsymbol{e}_k)\right). \tag{11}$$

First by using Lemma 5, we derive

$$q_1 f_i^t(\boldsymbol{x}+\boldsymbol{e}_{12}) + q_2 f_i^t(\boldsymbol{x}+\boldsymbol{e}_{34}) - f_i^t(q_1(\boldsymbol{x}+\boldsymbol{e}_{12}) + q_2(\boldsymbol{x}+\boldsymbol{e}_{34})) \;\leq\; \frac{1}{4}(F'(1) - F'(0)),$$

where $F(\alpha) = f_i^t(\boldsymbol{x} + \boldsymbol{e}_{12} + \alpha(\boldsymbol{e}_{34} - \boldsymbol{e}_{12}))$. Note that

$$\begin{aligned} F'(0) &= \sum_j \frac{\partial}{\partial x_j} f_i^t(\boldsymbol{x}+\boldsymbol{e}_{12}) \cdot (\boldsymbol{e}_{34} - \boldsymbol{e}_{12})_j, \quad \text{and} \\ F'(1) &= \sum_j \frac{\partial}{\partial x_j} f_i^t(\boldsymbol{x}+\boldsymbol{e}_{34}) \cdot (\boldsymbol{e}_{34} - \boldsymbol{e}_{12})_j. \end{aligned}$$

Thus,

$$\begin{aligned} F'(1) - F'(0) &= \sum_j \left(\frac{\partial}{\partial x_j} f_i^t(\boldsymbol{x}+\boldsymbol{e}_{12}) - \frac{\partial}{\partial x_j} f_i^t(\boldsymbol{x}+\boldsymbol{e}_{34}) \right) \cdot (\boldsymbol{e}_{34} - \boldsymbol{e}_{12})_j \\ &\leq D \cdot dfmax_i(t, \boldsymbol{x}) \cdot (2e_{\max}) \;=\; (2De_{\max}) \cdot dfmax_i(t, \boldsymbol{x}). \end{aligned}$$

That is,

$$q_1 f_i^t(\boldsymbol{x}+\boldsymbol{e}_{12}) + q_2 f_i^t(\boldsymbol{x}+\boldsymbol{e}_{34}) - f_i^t(q_1(\boldsymbol{x}+\boldsymbol{e}_{12}) + q_2(\boldsymbol{x}+\boldsymbol{e}_{34})) \;\leq\; \frac{De_{\max}}{2} \cdot dfmax_i(t, \boldsymbol{x}). \quad (12)$$

Then from (11) and (12), we have

$$\begin{aligned} &\sum_{i=k}^{4} p_k f_i^t(\boldsymbol{x}+\boldsymbol{e}_k) - f_i^t\left(\sum_{i=k}^{4} p_k(\boldsymbol{x}+\boldsymbol{e}_k) \right) \\ &\leq \sum_{i=k}^{4} p_k f_i^t(\boldsymbol{x}+\boldsymbol{e}_k) - (q_1 f_i^t(\boldsymbol{x}+\boldsymbol{e}_{12}) + q_2 f_i^t(\boldsymbol{x}+\boldsymbol{e}_{34})) + \frac{De_{\max}}{2} \cdot dfmax_i(t, \boldsymbol{x}). \end{aligned}$$

On the other hand, note that

$$\boldsymbol{x}+\boldsymbol{e}_{12} \;=\; \frac{p_1}{q_1}(\boldsymbol{x}+\boldsymbol{e}_1) + \frac{p_2}{q_1}(\boldsymbol{x}+\boldsymbol{e}_2), \quad \text{and} \quad \boldsymbol{x}+\boldsymbol{e}_{34} \;=\; \frac{p_3}{q_2}(\boldsymbol{x}+\boldsymbol{e}_3) + \frac{p_4}{q_2}(\boldsymbol{x}+\boldsymbol{e}_4).$$

Then we have the following by the same argument.

$$\begin{aligned} q_1 \left(\frac{p_1}{q_1} f_i^t(\boldsymbol{x}+\boldsymbol{e}_1) + \frac{p_2}{q_1} f_i^t(\boldsymbol{x}+\boldsymbol{e}_2) - f_i^t(\boldsymbol{x}+\boldsymbol{e}_{12}) \right) &\leq q_1 \cdot \frac{De_{\max}}{2} \cdot dfmax_i(t, \boldsymbol{x}), \quad \text{and} \\ q_2 \left(\frac{p_3}{q_2} f_i^t(\boldsymbol{x}+\boldsymbol{e}_3) + \frac{p_4}{q_2} f_i^t(\boldsymbol{x}+\boldsymbol{e}_4) - f_i^t(\boldsymbol{x}+\boldsymbol{e}_{34}) \right) &\leq q_2 \cdot \frac{De_{\max}}{2} \cdot dfmax_i(t, \boldsymbol{x}). \end{aligned}$$

Thus,

$$dL_i(t, \boldsymbol{x}) \;\leq\; (1 + q_1 + q_2) \cdot \frac{De_{\max}}{2} \cdot dfmax_i(t, \boldsymbol{x}) \;=\; \frac{2De_{\max}}{2} \cdot dfmax_i(t, \boldsymbol{x}).$$

In general, noting that there are at most DK terms in general, it is easy to derive the bound of the lemma. □

Finally our task is to bound $dfmax_i(t, \boldsymbol{x})$. Let Δ_t denote this bound (for t). For the analysis, we use the equation (8). From this equation, it is not so hard to see that the bound is easier if the maximum of (10) is achieved by some $h \neq i$. Thus, we consider below the hardest case, i.e., the case $h = i$. Furthermore, assume that $\boldsymbol{u}$ and $\boldsymbol{v} \in \mathcal{N}(\boldsymbol{x})$ define $dfmax_i(t, \boldsymbol{x})$. That is, we bound $dfmax_i(t, \boldsymbol{x})$ for the case

$$dfmax_i(t, \boldsymbol{x}) \;=\; \frac{\partial}{\partial x_i} f_i^t(\boldsymbol{u}) - \frac{\partial}{\partial x_i} f_i^t(\boldsymbol{v}).$$

Also we assume that $\Delta_t = dfmax_i(t, \boldsymbol{x})$, and for any $s < t$, it holds inductively that Δ_s is a bound for $dfmax_j(s, \boldsymbol{x})$ for all j.

Let us derive a formula for $\frac{\partial}{\partial x_i} f_i^t(\boldsymbol{u})$. Here again we follow the same outline as before, and use the inductive definition of the partial derivative. That is, we use the following.

$$\frac{\partial}{\partial x_i} f_i^t(\boldsymbol{u}) \;=\; \sum_h \frac{\partial}{\partial x_h} f_i(\boldsymbol{f}^{t-1}(\boldsymbol{u})) \cdot \frac{\partial}{\partial x_i} f_h^{t-1}(\boldsymbol{u}).$$

By computing $\frac{\partial f_i}{\partial x_i} = 1 + \cdots$ from (8), we further have

$$\frac{\partial}{\partial x_i} f_i^t(\boldsymbol{u}) \;=\; \frac{\partial}{\partial x_i} f_h^{t-1}(\boldsymbol{u}) + \sum_h w_h \frac{\sum_j (a_h - a_j) w_j \widehat{u}_j}{Z(\widehat{\boldsymbol{u}})^2} \cdot \frac{\partial}{\partial x_i} f_h^{t-1}(\boldsymbol{u}),$$

where $\widehat{\boldsymbol{u}} = \boldsymbol{f}^{t-1}(\boldsymbol{u})$, $\widehat{u}_j = (\widehat{\boldsymbol{u}})_j$, and $Z(\widehat{\boldsymbol{u}}) = \sum_j w_j \widehat{u}_j$, i.e., the total weight computed from $\boldsymbol{f}^{t-1}(\boldsymbol{u})$. We have a similar formula for $\frac{\partial}{\partial x_i} f_i^t(\boldsymbol{v})$; for this one, $\widehat{\boldsymbol{v}} = \boldsymbol{f}^{t-1}(\boldsymbol{v})$, $\widehat{v}_j = (\widehat{\boldsymbol{v}})_j$, and $Z(\widehat{\boldsymbol{v}}) = \sum_j w_j \widehat{v}_j$ are used.

Now we have

$$\begin{aligned}
dfmax_i(t, \boldsymbol{x}) \;&=\; \left(\frac{\partial}{\partial x_i} f_h^{t-1}(\boldsymbol{u}) - \frac{\partial}{\partial x_i} f_h^{t-1}(\boldsymbol{v}) \right) \\
&\quad +\; \sum_h w_h \left(\frac{\sum_j (a_h - a_j) w_j \widehat{u}_j}{Z(\widehat{\boldsymbol{u}})^2} \cdot \frac{\partial}{\partial x_i} f_h^{t-1}(\boldsymbol{u}) - \frac{\sum_j (a_h - a_j) w_j \widehat{v}_j}{Z(\widehat{\boldsymbol{v}})^2} \cdot \frac{\partial}{\partial x_i} f_h^{t-1}(\boldsymbol{v}) \right) \\
&\leq\; \Delta_{t-1} + \sum_h w_h \frac{\sum_j (a_h - a_j) w_j \widehat{u}_j}{Z(\widehat{\boldsymbol{u}})^2} \cdot \Delta_{t-1} \\
&\quad +\; \sum_h w_h \left(\frac{\sum_j (a_h - a_j) w_j \widehat{u}_j}{Z(\widehat{\boldsymbol{u}})^2} - \frac{\sum_j (a_h - a_j) w_j \widehat{v}_j}{Z(\widehat{\boldsymbol{v}})^2} \right) \cdot \frac{\partial}{\partial x_i} f_h^{t-1}(\boldsymbol{v}) \\
&\leq\; \Delta_{t-1} \left(1 + \frac{aW}{Z(\widehat{\boldsymbol{u}})} \right) + \sum_h w_h \cdot \frac{a c_{\mathrm{f}} W}{Z(\widehat{\boldsymbol{v}})^2} \cdot c_{\mathrm{pd}} \;\leq\; \Delta_{t-1} \left(1 + \frac{aW}{c_{\mathrm{Z}} N} \right) + \frac{a c_{\mathrm{f}} c_{\mathrm{pd}} W^2}{(c_{\mathrm{Z}} N)^2}.
\end{aligned}$$

Here constants are those defined in conditions (1) $\sim$ (4). On the other hand, $a = \max |a_h - a_j|$, which is at most $4 e_{\max} K$.

By solving this recurrence, we obtain

$$\Delta_t \;\leq\; \Delta_1 \left(1 + \frac{aW}{c_{\mathrm{Z}} N} \right)^t + \frac{a c_{\mathrm{f}} c_{\mathrm{pd}} W}{c_{\mathrm{Z}}^2 N} \left(1 + \frac{aW}{c_{\mathrm{Z}} N} \right)^t.$$

Since Δ_1 is bounded by $2ae_{\max}w_{\max}W/(c_{\mathrm{Z}}N)^2$, we have (a bit roughly)

$$\Delta_t \;\leq\; \frac{a(c_{\mathrm{f}}c_{\mathrm{pd}}+2e_{\max})w_{\max}W}{c_{\mathrm{Z}}^2N}\left(1+\frac{aW}{c_{\mathrm{Z}}N}\right)^t \;\leq\; \frac{e(2c_{\mathrm{f}}c_{\mathrm{pd}}e_{\max})aW\cdot w_{\max}}{c_{\mathrm{Z}}^2N},$$

for any $t \leq (c_{\mathrm{Z}}N/aW)$.

Finally, with the bound of Corollary 6 and equation (9), we have

$$err_i(t,\boldsymbol{x}) \;\leq\; \frac{(ec_{\mathrm{f}}c_{\mathrm{pd}}e_{\max}^2D\log_2(DK))w_{\max}}{c_{\mathrm{Z}}},$$

while $t \leq c_{\mathrm{Z}}N/aW$.

4 Concluding Remarks

We have shown that some approximation error bounds for the pseudo expectation. When W is small, i.e., it is regarded as a constant, the bound of Theorem 1 (and also of Theorem 5 for the general case) would be reasonable. On the other hand, for the one-dimensional case, some $o(N)$ bound is also provable (i.e., Theorem 4) for the case that W is proportional to N; altogether the error can be bounded by $o(N)$. We, however, conjecture that the error could be bounded by a much smaller function, maybe, a constant independent from N and W.

Notice that the argument of Theorem 1 could be called a *mechanical approach*, because no statistical property of the process is used. On the otherhand, we would expect some strong concentration [TN03], which may be helpful to get a tighter bound. For the one-dimensional case, since the model is simple, one may expect to solve the formula for the expected state completely by solving a corresponding differential equation. Unfortunately, though, a straightforward application of such method only provides us $o(N)$ bound, and more careful consideration may be necessary to obtain a better bound.

Acknowledgments

A part of the work has been done jointly with Y. Niikura and J. Schneider as their Master thesis projects. I would like to thank them for their collaboration. Also I would like to thank H. Takahashi, K. Tanaka, and R. Monasson, for their interest to this work and various useful suggestions. In particular, I thank to K. Tanaka for teaching me the image restoration algorithm, and to R. Monasson for pointing out the idea leading me to the proof of Theorem 4.

References

[Gal62] R.G. Gallager, Low density parity check codes, *IRE Trans. Inform. Theory*, **IT-8**(21), 21–28, 1962.

[Mac99] D. MacKay, Good error-correcting codes based on very sparse matrices, *IEEE Trans. Inform. Theory*, **IT-45**(2), 399–431, 1999.

[CM04] S. Cocco and R. Monasson, Heuristic average-case analysis of the backtrack resolution of random 3-satisfiability instances, *Theoret. Comput. Sci.* 320(2-3), 345–372, 2004.

[TN03] H. Takahashi and Y. Niikura, An extension of Azuma-Hoeffding inequalities and its application to an analysis of randomized local search algorithms, in *Proc. Workshop IBIS2003*, 177–181, 2003.

[Tan01] K. Tanaka, Maximum marginal likelihood estimation and constrained optimization in image restoration, *J. Japanese Society of Aritificial Intelligence*, 16(2), 246–258, 2001.

[WST03] O. Watanabe, T. Sawai, and H. Takahashi, Analysis of a randomized local search algorithm for LDPCC decoding problem, in *Proc. SAGA'03*, Lecture Notes in Comp. Sci. 2827,50–60, 2003.