Industrial Engineering & Management
ISSN: 2169-0316

Hirotani et al., Ind Eng Manage 2014, 3:5
DOI: 10.4172/2169-0316.1000146

Review Article Open Access

Policy for Rearranging Workers for a Self-Balancing Production Line with Worker Learning

Daisuke Hirotani*, Katsumi Morikawa and Katsuhiko Takahashi

Faculty of Engineering, Hiroshima University, 1-4-1, Kagamiyama, Higashi-Hiroshima, 7398527, Japan

Abstract

In the traditional production lines such as assembly lines, each worker is usually assigned to a particular fixed position, and the speed of performing the task decreases until the worker masters the assigned work. However, when an imbalance in the speeds of the workers exists, any given worker can delay the overall work on the production line, and the production rate of the particular line will also decrease. To avoid this problem, the "Self-Balancing Production Line" was introduced. In this type of production line, each worker is assigned work dynamically so they can keep the production line balanced while satisfying the specific conditions. A previous paper studying worker learning has been published. When a worker learns, the speed of the worker can be increased. In that paper, the authors analyzed the conditions with and without passing and claimed that if passing is allowed, self-balancing of a production line can be achieved. However, even if the initial sequence is slowest to fastest (this is the best sequence for self-balance in the previous paper) and if passing is allowed, much more time is required to balance some conditions of speed and degree of learning (we call this the learning rate). Therefore, a new policy for rearranging workers that changes the sequence before passing should be considered for rapid balance of the production line. In this paper, the policy for rearranging workers that changes the sequence being learned is proposed, and to verify the policy, numerical experiments are performed under various conditions of speed and learning rate.

Keywords: Production; Self-balancing; Learning

Introduction

In a traditional assembly line, each worker is typically given a fixed assignment, and each worker iterates the assigned work continuously under the traditional assembly line balancing system. For this type of line, assigning workers to balance the work has been studied previously [1]. When a speed imbalance exists in this type of line, the slowest worker will delay the overall work, and the production rate of the production line will decrease. To solve this problem, a "Self-Balancing Production Line" was introduced [2]. This type of line is referred as "Bucket Brigades" and was used in at least two commercial environments: apparel manufacturing and distribution warehousing [3]. In this type of production line, each worker is assigned to work according to the worker's own pace, and when the last worker completes a task, he/she walks back and takes over the next task from his/her predecessor. The predecessor then walks back and takes over the next task from his/her predecessor; this cycle continues until the first worker walks back and starts a new task. Because faster workers are assigned more work in processing a task and vice versa, balance can be maintained. For this line with a constant working speed, the maximum production rate can be achieved if workers are sequenced from slowest to fastest [2]. Furthermore, the balance conditions with two and three workers have been obtained numerically with simulations [3], and the production line performance with n workers has been analyzed mathematically [4]. There is typically assumed to be no walk-back time or hand off time in this type of line. However, Bartholdi and Eisenstein [5] consider a constant walk-back time and hand off time, and Bartholdi et al. [6] and Bratcu and Dolgui [7] consider finite walk-back speeds. Additionally, Armbruster et al. [8] consider the case in which there are two zones according to the speed (i.e., the worker's speed changes according to the zone). Under these conditions, these authors analyzed the cases in which passing is and is not allowed.

Recently, research for a self-balancing production line focused on the characteristics of the line. Lim [9] considered a U-shaped production line. In this line, the entrance and exit for an item were near each other. Therefore, the conditions for self-balance were changed. Lim analyzed the conditions for this type of line. Webster et al. [10] considered the line for order picking with respect to variations in work skills, SKU (Stock Keeping Unit) volume, and walking to the selected work content according to a pick list. This paper analyzes the impact on throughput for the above-mentioned variations.

With a few exceptions, the aforementioned studies focused on constant speed. Each worker processed at a constant speed according to the time. One factor that changes a worker's speed is learning, a topic on which many papers have been written. Relative to this paper, Nembhard [11] considers individual learning and forgetting. Under these conditions, he proposed a heuristic approach for assigning workers, and using ANOVA (ANalysis Of VAriance), he analyzed the effect of this approach. Jaber et al. [12] considered several learning and forgetting models. Using these models, these authors analyzed the effects of learning and forgetting. Shafer et al. [13] analyzed the effect of learning and forgetting in an assembly line by simulation. Villalobos et al. [14] proposed dynamic work sharing methods in assembly lines assuming certain learning curves.

The above-mentioned papers are considered without the aspect of learning. Several previous papers studying learning have been published. Armbruster et al. [15] considered learning with and without allowing one worker to pass the other. These authors claimed that if

***Corresponding author:** Daisuke Hirotani, Faculty of Engineering, Hiroshima University, 1-4-1, Kagamiyama, Higashi-Hiroshima, 7398527, Japan, Tel: +81-82-424-7703; Fax: +81-82-422-7024; E-mail: dhiro@hiroshima-u.ac.jp

Received August 22, 2013; **Accepted** September 25, 2013; **Published** December 15, 2014

Citation: Hirotani D, Morikawa K, Takahashi K (2014) Policy for Rearranging Workers for a Self-Balancing Production Line with Worker Learning. Ind Eng Manage 3: 146. doi: 10.4172/2169-0316.1000146

Copyright: © 2014 Hirotani D, et al. This is an open-access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

passing is allowed, self-balancing can be achieved. Hirotani et al. [16] considered individual learning (i.e., different degrees of learning). These authors claimed that the slowest to fastest sequence is not the best for achieving a higher production rate. However, from previous papers, even if the initial sequence is arbitrary and passing is allowed, more much time is required to balance production for some conditions of speed and degree of learning (we call this the learning rate). Therefore, a new work policy that changes the sequence before passing should be considered to balance production rapidly. In this paper, a work policy that changes the sequence with learning is proposed, and to verify the policy, numerical experiments are performed under various conditions of speed and learning rate.

This paper is organized as follows: In section 2, assumptions, and characteristics of this self-balancing production line with learning are explained. In section 3, we propose and explain work policies for learning. In section 4, we compare the proposed policy and analyze the characteristics of the policies under various speeds and learning rates. Finally, concluding remarks are made in section 5.

The Production Line

Assumptions

In this paper, a production line with the following assumptions is considered.

- Each worker processes only one identical item sequentially.

- Workers are sequenced from one to n on the production line (Figure 1), and each worker can pass over the upstream and downstream workers. This assumption is different from previous papers [2-10,15,16].

- Worker i processes with increasing working speed $v_i^{(t)}>0$ at iteration t. The lower and upper limit of working speed for a worker i is defined as v_i^L and v_i^H, respectively. Degree of learning (we call this learning rate in the following sections) is defined as τ_i.

- When the last worker finishes processing an item, worker n walks back to worker n-1 and takes over the next item from worker n-1. Then, worker n-1 walks back to worker n-2 and takes over the next item from worker n-2. Similarly, all workers walk back to their preceding worker and take over the next item from the preceding worker, and worker 1 introduces a new item into the system. The time required to walk back and take over is ignored.

- The position of worker i when he/she starts to process is given by x_i (Figure 1). Then, the position at iteration t is defined as $x_i^{(t)}$. Note that $x_i^{(t)}=0$ for any iteration t because the first worker always starts to process an item.

Self-balancing and convergence

The production line has been proven to maintain balance when workers are sequenced from slowest to fastest, and the working speed is constant for all workers [2]. Subsequently, the position of the workers will converge to a unique fixed point. Under these conditions, the production rate can be calculated as the sum of each worker's speed for each worker. The condition for convergence under which the line can balance for n workers has been found in a previous paper [4]. However, in this paper, the worker's speed can be increased as the worker learns, so the result of the previous paper should be modified as follows:

$$\frac{\sum_{k=1}^{i=1}(-1)^{i+k-1}v_k^{(t)}}{v_n^{(t)}}<1 \tag{1}$$

The difference is only the working speed with iteration t. Additionally we should modify the definition of convergence. In this paper, if the maximum production rate at iteration t can be achieved under $v_i^{(t)}$, we judged that convergence had occurred. If convergence occurs, the above-mentioned fixed point can be defined as follows:

$$x_i^{(t)*}=\frac{\sum_{k=1}^{i=1}v_k^{(t)}}{\sum_{k=1}^{n}v_k^{(t)}} \tag{2}$$

In this paper, if the difference in the fixed points is less than ε, we judge that convergence has occurred. Additionally, under the conditions of convergence, if condition (1) is satisfied for all workers based on the average speeds, we judge that self-balance has occurred.

Learning

In this paper, worker learning is represented as increasing the worker's working speed. Based on the previous papers [15-17], speed is defined as follows:

$$v_i^{(t)}=v_i^L+(v_i^H-v_i^L)\left(1-e^{\frac{t'}{\tau_i}}\right) \tag{3}$$

where t' is number of workings. In this paper, we divide the task into 1000 intervals, a difference from a previous paper [15]. Additionally, the difference from the previous paper [16] is τ_i where τ_i is the same for all workers (i.e., $\tau_i=\tau$) in that paper. Figure 2 shows the learning curve varying under $v_i^L=2$ and $v_i^H=4$.

Worker Rearranging Policy for Learning

In this paper, we propose the following four policies for self-balancing a production line with learning.

Only passing (Policy 1)

This policy is the same as in previous papers [15,16]. When a worker catches up with a downstream worker, the worker can pass the other worker. As mentioned, this policy can self-balance under various speeds and learning rates. A time chart for Policy 1 is shown in Figure 3. In this figure, two lines are crossed: the first worker passes the second worker.

Applying only this policy, passing seldom occurs. Instead, the worker's starting position converges to a specific point in any sequence. This phenomenon is shown in Figure 4. This figure is an orbit of worker starting positions. In this figure, the worker starting position converges three times. Therefore, more time is required to self-balance. As a result, a new rearranging policy should be considered.

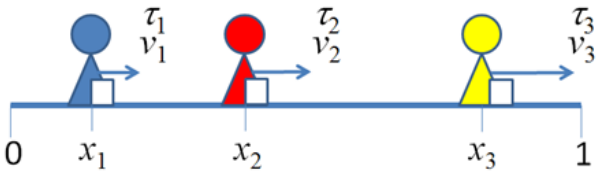

Figure 1: The production line and positions for three workers

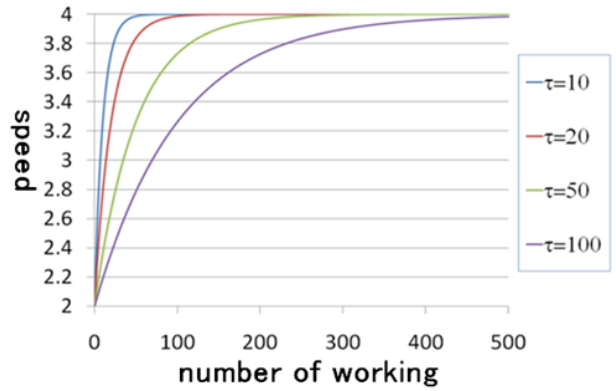

Figure 2: Learning curve with respect to number of worker under various learning rates

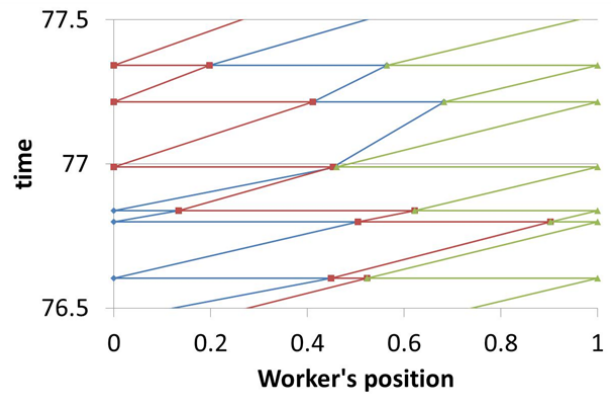

Figure 3: Time chart for Policy 1 under $_c$=3.5, v_2^H =3, v_3^H =2.5, τ_1=50, τ_2=50, τ_3=10 and ABC sequence

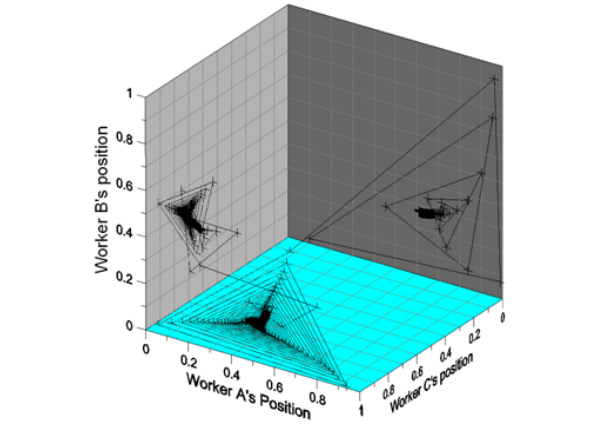

Figure 4: Orbit of worker starting positions for Policy 1 under v_1^H =3.5, v_2^H =3, v_3^H =2.5, τ_1=50,τ_2=50,τ_3=10 and ABC sequence

Rearranging adjacent workers (Policy 2)

Each worker evaluates the average speed for all iterations, and if the average speed is higher than the speed of a downstream worker, the position of each worker changes to the other. This policy arises because if more time is required to pass and yet not self-balance, arranging the sequence is necessary to self-balance sooner. The time chart for Policy 2 is shown in Figure 5. In this figure, the starting position of the first worker is changed to the starting position of the second worker. Therefore, the first worker does not take over an item.

By applying only this policy, rearranging frequently occurs because, after rearranging, speed is decreased compared to the situation before rearranging. This phenomenon is shown in Figure 6. This figure is an orbit of worker starting position. In this figure, rearranging occurs frequently for both first and second workers and second and third workers and is therefore not realistic. As a result, a new rearranging policy should be considered.

Rearranging adjacent workers when workers' positions converge (Policy 3)

Convergence is evaluated by computing the difference between the present and the previous worker's starting positions. If these positions converge, each worker evaluates the average speed, and if the speed is higher than a downstream worker's speed, the position of each worker changes to the other's position. The difference from Policy 2 is the use of convergence. Using convergence, the worker avoids frequent changes in the position. The time chart for Policy 3 is shown in Figure 7. In this figure, the starting position of the first worker is changed to the starting position of the second worker after that worker's position converges to the specific position.

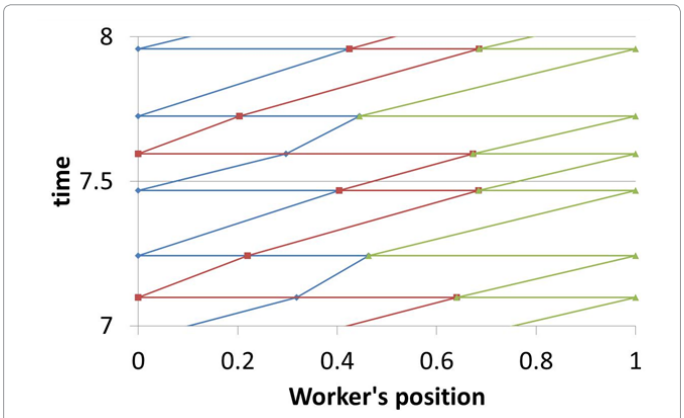

Figure 5: Time chart for Policy 2 under v_1^H =3.5, v_2^H =3, v_3^H =2.5, τ_1=50,τ_2=50,τ_3=10 and ABC sequence

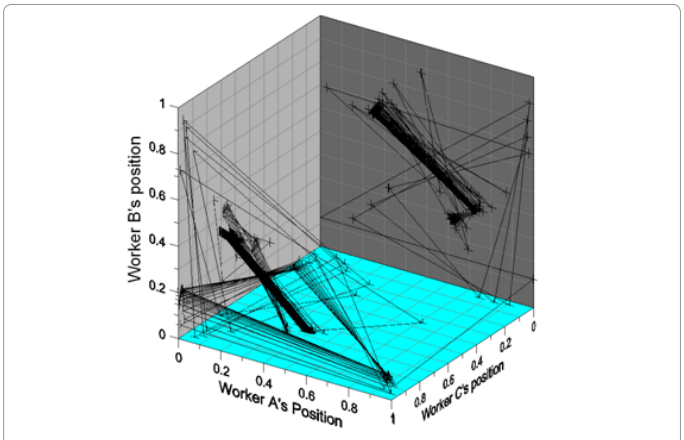

Figure 6: Orbit of worker starting positions for Policy 2 under v_1^H =3.5, v_2^H =3, v_3^H =2.5, τ_1=50, τ_2=50, τ_3=10 and ABC sequence

Applying only this policy, rearrangement occurs only for adjacent workers. An orbit of starting positions for the workers is shown in Figure 8. In this figure, rearrangement seldom occurs. However, even if rearrangement occurs, the production line may have more time to self-balance because, after rearrangement, no one can be sure that the sequence after rearrangement satisfies the self-balancing condition. Therefore, as a result, a new rearrangement policy should be considered.

Rearranging all workers when the worker's positions converge (Policy 4)

Convergence is evaluated by computing the difference between the present and previous workers' starting positions. If the positions converge, each worker evaluates the average speed, and if the speed does not satisfy the self-balancing condition shown in formula (1), the starting position of each worker changes to slowest to fastest at the next iteration. The difference from Policy 3 is using the self-balancing condition. If the change to slowest to fastest is rapid, the line can self-balance earlier. A time chart is shown in Figure 9. In this figure, after the positions converge to a specific position, the position of the first worker changes to the position of the second worker, the position of the second worker changes to the position of the third worker, and the position of third worker changes to the position of the first worker. The changed sequence is now slowest to fastest. An orbit of worker starting positions is shown in Figure 10. In this figure, rearrangement also seldom occurs.

Comparing the Work Policies

We compare four policies and evaluate these policies to determine the best policy. We assume three workers (i.e., n=3). Therefore, the self-balancing conditions achieved by using formula (1) are $v_1<v_3$ and $v_1+v_3>v_2$, meaning that the speed of the first worker is slower than the speed of the last worker, and the sum of the speed of the first and the last workers is faster than the speed of the second worker. Additionally, we set $\varepsilon=10^{-4}$ for evaluation of the convergence.

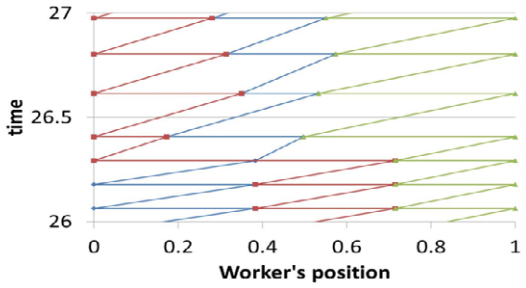

Figure 7: Time chart for Policy 3 under v_1^H =3.5, v_2^H =3, v_3^H =2.5, τ_1=50,τ_2=50,τ_3=10 and ABC sequence

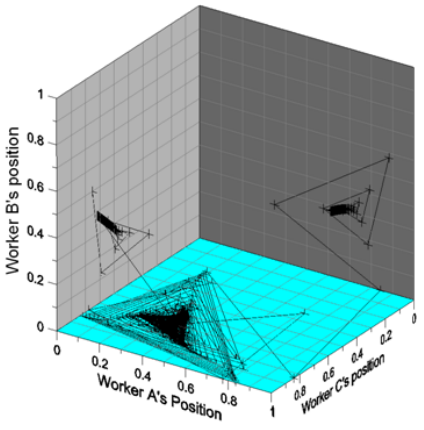

Figure 8: Orbit of worker starting positions for Policy 3 under v_1^H =3.5, v_2^H =3, v_3^H =2.5, τ_1=50,τ_2=50,τ_3=10 and ABC sequence

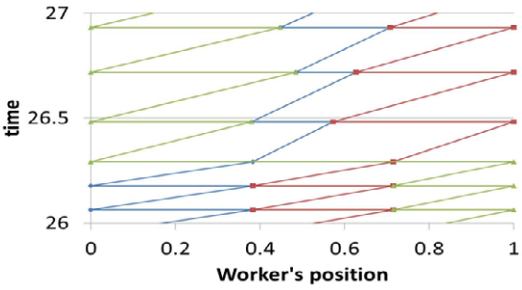

Figure 9: Time chart for Policy 4 for three workers under v_1^H =3.5, v_2^H =3, v_3^H =2.5, τ_1=50,τ_2=50,τ_3=10 and ABC sequence

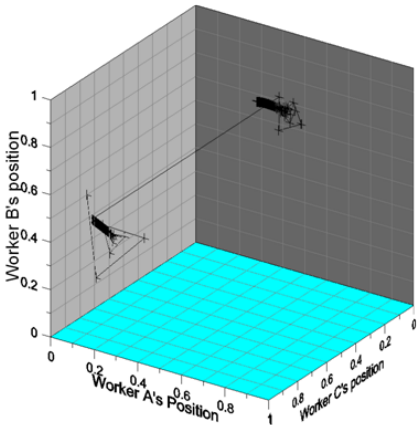

Figure 10: Orbit of worker starting positions for Policy 4 under v_1^H =3.5, v_2^H =3, v_3^H =2.5, τ_1=50,τ_2=50,τ_3=10 and ABC sequence

Comparison of results

First we evaluate the number of iterations for self-balance by varying learning rate, initial sequence and final sequence. Additionally, we evaluate the number of rearrangements (not including number of passes). For both measures, a low value is better. Because for the number of iterations, a low value means rapid convergence for self-balancing, and for the number of rearrangements, a low value means seldom rearranging so that the worker does not have to change work range. The result for the fastest to slowest final sequence is shown in Table 1. As shown above, the fastest to slowest sequence is the worst case. Therefore, the sequence should be changed. We choose this sequence for that reason. In this table, ABC is a worker sequence, i.e., the first worker is A, who has an initial speed of 1 (i.e., v_A^L =1), the second worker is B, who has an initial speed of 1.5 (i.e., v_B^L =1.5), and the last worker is C, who has an initial speed of 2 (i.e., v_C^L =2). In this case, the speed of the first worker (A) increases from 1 to 3.5, the speed of the second worker increases from 1.5 to 3, and the speed of the last worker increases from 2 to 2.5. Additionally, the term "-" indicates that this sequence does not self-balance under $\varepsilon=10^{-4}$. However, in

analyzing this sequence, we find that there is a small difference, but the sequence seems to self-balance.

In Table 1, Policy 1 is the worst for cases of ABC, ACB, and BAC. In this case, each worker has to pass at least once. For Policy 1, the worker starting positions converge once. However, because this sequence does not satisfy the self-balancing condition later, the position frequently changes. If this behavior is repeated, the line can ultimately self-balance, though it takes more time to do so. For the other policies, by arranging forcibly, the line can self-balance earlier.

Changing frequently occurs in Policy 2. Changing frequently is not a good situation because the worker is confused when working with frequent changing. However, the number of iterations is the best for some cases. In Policy 3, the number of iterations is not better than the number of iterations of Policy 2. On the contrary, the number of rearrangements is better than the number of rearrangements for Policy 2 because it takes more time to converge to the specific position. Additionally, to utilize the convergence state, frequent rearrangements seldom occur. In Policy 4, to overcome the disadvantages of Policies 2 and 3, better results can be obtained because the number of rearrangements is low and the number of iterations is the best for some cases. However, for some cases, Policy 2 is the best with regard to the number of iterations. We consider why Policy 2 is the best about the number of iterations for some cases. Figure 11 shows that in one of the cases, Policy 2 is better than Policy 4. In Policy 4, rearrangement occurs several times, so it takes more time to self-balance. In this case, frequent rearrangement is better; thus, Policy 2 is better.

On the contrary, we analyze the condition where the final sequence is slowest to fastest. This sequence is the best case in a previous paper [2]. Therefore, the sequence does not have to change in some cases. This result is shown in Table 2. In Table 2, similar results can be obtained. Policy 1 is the worst for cases of BCA, CAB, and CBA. In this case, each worker has to pass at least once. For the reason as in Table 1, Policy 1 is not better. Additionally, in Policy 2, the number of rearrangements is high instead of the number of iterations being the best for some cases. In Policy 3, the number of iterations is high for cases of BCA, CAB and CBA, and the number of rearrangements is better than Policy 2. In Policy 4, the number of rearrangements is the best for all cases, and the number of iterations is the best for most cases. However, comparing Policies 3 and 4, the number of iterations is almost the same for cases ABC, ACB and BAC. We consider why this result is almost the same for the above-mentioned cases. As mentioned above, the initial sequence satisfies the self-balancing condition, as well as the final sequence. Therefore, there is no need to rearrange, and the result is almost the same for most cases of ABC, ACB and BAC.

Compared to the other three policies, Policy 4 is the best. At first, we compare Policy 4 with Policy 2. To avoid the use convergence, frequent rearrangement occurs. Therefore, using the convergence condition is necessary. Next, we compare Policy 3 and 4 by adding self-balancing conditions, and the number of rearrangements decreases.

As a result, Policy 4 is the best even if the number of convergence is not the best.

Sensitivity analysis for the convergence condition of workers' positions

In previous subsections, $\varepsilon=10^{-4}$. However, as seen in Figures 10 and 11, more iteration is required for self-balance. Therefore, we perform sensitivity analysis for ε. ε is set as $\varepsilon=10^{-2}$, 10^{-3}. The results are shown in Table 3. In Table 3, comparing to $\varepsilon=10^{-4}$, cases for BCA, CAB and CBA are almost the same because there is no need to rearrange. For cases ABC, ACB, and BAC, $\varepsilon=10^{-2}$ is better for most cases. We consider why 10^{-2} is better. Figure 12 shows orbits for worker starting position under $\varepsilon=10^{-2}$ and $\varepsilon=10^{-4}$. In this figure, for $\varepsilon=10^{-4}$, it takes more time to rearrange. Therefore, we claim that evaluation for convergence must not be set rigidly. The result is worse for some parameters because, to determine the convergence earlier, the other sequence that is not desired is changed. This circumstance is shown in Figure 13. In this figure, to change earlier, more rearrangement occurs. Therefore, roughly setting for ε is not recommended. Next, as in the previous section, the case for that initial sequence of slowest to fastest is analyzed. The results are shown in Table 4. In Table 4, comparing to $\varepsilon=10^{-4}$, cases for ABC, ACB and BAC are almost the same because there is no need to rearrange. For cases BCA, CAB, and CBA, $\varepsilon=10^{-2}$ is better for most cases for almost the same reason as previously. Therefore, we claim that evaluation for convergence must not be set rigidly. However, the result is worse for some parameters because, to determine the convergence earlier, another sequence that is not desired is changed. Therefore, roughly setting for ε is not recommended.

Conclusion

In this paper, we consider the self-balancing production line with the additional factor of learning. When considering learning, more time is required for the production line to self-balance. To overcome this

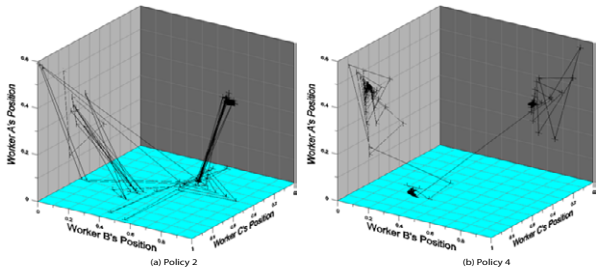

Figure 11: Orbit of worker starting positions under v_1^H =3.5, v_2^H =3, v_3^H =2.5, τ_1=50,τ_2=50,τ_3=10 and ABC sequence

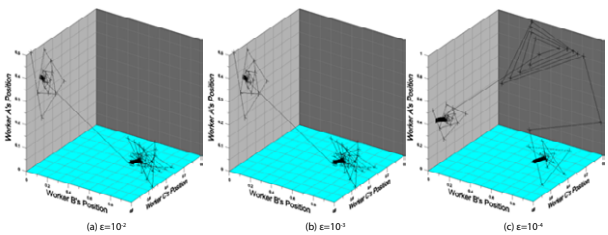

Figure 12: Orbit of worker starting positions under BAC sequence and τ_1=50,τ_2=50,τ_3=10.

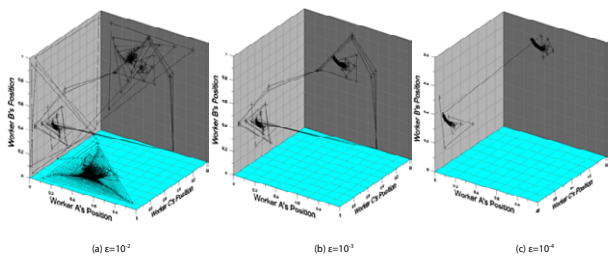

Figure 13: Orbit of worker starting positions under ABC sequence andτ_1=100,τ_2=50,τ_3=10

(a) Policy 1

τ_1	τ_2	τ_3	Number of iterations						Number of iterations					
			ABC	ACB	BAC	BCA	CAB	CBA	ABC	ACB	BAC	BCA	CAB	CBA
50	50	50	1702	717	1928	141	158	99	0	0	0	0	0	0
10	50	50	1118	138	1385	168	201	147	0	0	0	0	0	0
100	50	50	2478	1496	2611	217	1202	89	0	0	0	0	0	0
50	10	50	1437	782	1289	192	149	180	0	0	0	0	0	0
50	100	50	2505	564	3065	115	303	126	0	0	0	0	0	0
50	50	10	1732	758	1980	327	1802	290	0	0	0	0	0	0
50	50	100	1576	536	1786	233	175	237	0	0	0	0	0	0

(b) Policy 2

τ_1	τ_2	τ_3	Number of iterations						Number of rearrangements					
			ABC	ACB	BAC	BCA	CAB	CBA	ABC	ACB	BAC	BCA	CAB	CBA
50	50	50	455	411	261	-	266	109	320	195	163	80	113	3
10	50	50	189	168	329	171	290	192	51	23	183	10	143	3
100	50	50	866	921	235	503	226	88	676	639	121	244	107	3
50	10	50	458	443	350	331	130	174	341	291	176	142	26	95
50	100	50	304	361	594	378	586	182	171	192	415	49	367	3
50	50	10	485	456	323	239	349	223	345	255	211	122	191	111
50	50	100	383	364	247	199	259	228	234	170	133	78	59	3

(c) Policy 3

τ_1	τ_2	τ_3	Number of iterations						Number of rearrangements					
			ABC	ACB	BAC	BCA	CAB	CBA	ABC	ACB	BAC	BCA	CAB	CBA
50	50	50	923	842	597	-	332	179	6	7	5	1	1	1
10	50	50	366	278	412	277	377	270	1	2	2	1	1	1
100	50	50	1172	1582	693	464	377	435	6	15	2	1	2	2
50	10	50	813	940	464	287	149	-	7	12	2	1	0	1
50	100	50	1080	682	718	258	547	503	5	4	3	1	1	3
50	50	10	1189	1075	545	691	575	290	2	8	3	4	2	0
50	50	100	820	678	457	284	378	329	6	12	3	1	1	1

(d) Policy 4

τ_1	τ_2	τ_3	Number of iterations						Number of rearrangements					
			ABC	ACB	BAC	BCA	CAB	CBA	ABC	ACB	BAC	BCA	CAB	CBA
50	50	50	302	308	299	141	158	99	1	1	1	0	0	0
10	50	50	283	138	377	168	201	147	1	0	1	0	0	0
100	50	50	503	528	681	217	296	89	1	1	2	0	1	0
50	10	50	382	364	318	192	149	180	1	1	1	0	0	0
50	100	50	429	276	530	115	303	126	1	1	1	0	0	0
50	50	10	345	347	355	327	336	290	1	1	1	0	1	0
50	50	100	266	295	263	233	175	237	1	1	1	0	0	0

Table 1: Comparison with Policies under v_1^H =3.5, v_2^H =3, v_3^H =2.5,

(a) Policy 1

τ_1	τ_2	τ_3	Number of iterations						Number of rearrangements					
			ABC	ACB	BAC	BCA	CAB	CBA	ABC	ACB	BAC	BCA	CAB	CBA
50	50	50	106	110	76	1965	150	932	0	0	0	0	0	0
10	50	50	290	-	261	1495	1699	761	0	0	0	0	0	0
100	50	50	236	158	79	1998	121	1083	0	0	0	0	0	0
50	10	50	145	158	116	2331	673	2059	0	0	0	0	0	0
50	100	50	168	93	192	1128	3181	433	0	0	0	0	0	0
50	50	10	187	180	186	199	205	800	0	0	0	0	0	0
50	50	100	82	124	90	3864	149	2514	0	0	0	0	0	0

(b) Policy 2

τ_1	τ_2	τ_3	Number of iterations						Number of rearrangements					
			ABC	ACB	BAC	BCA	CAB	CBA	ABC	ACB	BAC	BCA	CAB	CBA
50	50	50	106	-	275	417	275	444	0	76	116	197	164	313
10	50	50	226	252	348	458	321	423	110	126	188	264	297	329
100	50	50	236	202	260	370	258	393	00	76	60	167	131	229
50	10	50	236	328	120	443	289	454	92	140	15	292	168	334
50	100	50	168	400	589	366	590	247	0	46	366	195	416	170
50	50	10	187	172	290	64	330	199	0	8	142	21	182	64
50	50	100	82	506	230	921	238	861	0	242	108	633	124	673

(c) Policy 3

τ_1	τ_2	τ_3	Number of iterations						Number of rearrangements					
			ABC	ACB	BAC	BCA	CAB	CBA	ABC	ACB	BAC	BCA	CAB	CBA
50	50	50	106	110	76	1016	341	756	0	0	0	4	3	4
10	50	50	204	-	261	927	465	776	2	0	0	10	2	4
100	50	50	236	158	79	1123	282	553	0	0	0	5	1	3
50	10	50	145	158	116	933	471	882	0	0	0	10	4	4
50	100	50	168	93	192	691	597	835	0	0	0	7	2	4
50	50	10	187	180	186	308	367	374	0	0	0	1	1	2
50	50	100	106	396	208	1726	282	1172	0	2	2	9	1	8

(d) Policy 4

τ_1	τ_2	τ_3	Number of iterations						Number of rearrangements					
			ABC	ACB	BAC	BCA	CAB	CBA	ABC	ACB	BAC	BCA	CAB	CBA
50	50	50	106	110	76	364	150	446	0	0	0	1	0	2
10	50	50	290	-	261	472	927	546	0	0	0	2	2	2
100	50	50	236	158	79	375	121	402	0	0	0	1	0	2
50	10	50	145	158	116	371	224	489	0	0	0	1	1	1
50	100	50	168	93	192	300	448	600	0	0	0	1	3	4
50	50	10	187	180	186	199	205	361	0	0	0	0	0	1
50	50	100	82	124	90	575	149	671	0	0	0	1	0	2

Table 2: Comparison with Policies under v_1^H =2.5, v_2^H =3, v_3^H =3.5,

(a) $\varepsilon=10^{-2}$

τ_1	τ_2	τ_3	Number of iterations						Number of rearrangements					
			ABC	ACB	BAC	BCA	CAB	CBA	ABC	ACB	BAC	BCA	CAB	CBA
50	50	50	438	170	156	141	158	99	2	1	1	0	0	0
10	50	50	190	138	263	168	201	147	2	0	1	0	0	0
100	50	50	986	325	492	217	182	89	4	1	1	0	1	0
50	10	50	281	214	198	192	149	180	2	1	2	0	0	0
50	100	50	225	168	315	115	303	126	2	1	1	0	0	0
50	50	10	447	243	100	327	-	290	2	2	1	0	1	0
50	50	100	220	199	139	233	175	237	2	1	1	0	0	0

(b) $\varepsilon=10^{-3}$

τ_1	τ_2	τ_3	Number of iterations						Number of rearrangements					
			ABC	ACB	BAC	BCA	CAB	CBA	ABC	ACB	BAC	BCA	CAB	CBA
50	50	50	460	183	150	141	158	99	1	1	1	0	0	0
10	50	50	214	138	263	168	201	147	2	0	1	0	0	0
100	50	50	367	325	492	217	182	89	3	1	1	0	1	0
50	10	50	381	232	206	192	149	180	2	1	2	0	0	0
50	100	50	320	170	316	115	303	126	2	1	1	0	0	0
50	50	10	458	237	130	327	-	290	2	2	1	0	1	0
50	50	100	170	219	158	233	175	237	1	1	1	0	0	0

Table 3: Sensitivity analysis for policy 4 under v_1^H =3.5, v_2^H =3, v_3^H =2.5,

(a) $\varepsilon=10^{-2}$

τ_1	τ_2	τ_3	Number of iterations						Number of rearrangements					
			ABC	ACB	BAC	BCA	CAB	CBA	ABC	ACB	BAC	BCA	CAB	CBA
50	50	50	106	110	76	251	150	225	0	0	0	1	0	3
10	50	50	290	-	261	272	388	303	0	0	0	5	7	3
100	50	50	236	158	79	271	121	200	0	0	0	1	0	3
50	10	50	145	158	116	291	203	335	0	0	0	1	1	1
50	100	50	168	93	192	202	433	500	0	0	0	1	5	10
50	50	10	187	180	186	199	205	371	0	0	0	0	0	1
50	50	100	82	124	90	462	149	448	0	0	0	2	0	7

(b) $\varepsilon=10^{-3}$

τ_1	τ_2	τ_3	Number of iterations						Number of rearrangements					
			ABC	ACB	BAC	BCA	CAB	CBA	ABC	ACB	BAC	BCA	CAB	CBA
50	50	50	106	110	76	251	150	247	0	0	0	1	0	2
10	50	50	290	-	261	278	225	310	0	0	0	4	3	3
100	50	50	236	158	79	271	121	221	0	0	0	1	0	2
50	10	50	145	158	116	291	219	335	0	0	0	1	1	1
50	100	50	168	93	192	197	323	500	0	0	0	1	3	7
50	50	10	187	180	186	199	205	371	0	0	0	0	0	1
50	50	100	82	124	90	464	149	489	0	0	0	2	0	5

Table 4: Sensitivity analysis for Policy 4 under v_1^H =2.5, v_2^H =3, v_3^H =3.5,

problem, we propose four policies: only passing (Policy 1), rearranging adjacent workers (Policy 2), rearranging adjacent workers when workers' positions converge (Policy 3) and rearranging all workers when the workers positions' converge (Policy 4). Next, we compare the four policies according to the number of iterations of self-balancing and the number of rearrangements. As a result, Policy 4 is the best using convergence and the self-balancing conditions. Additionally, we performed sensitivity analysis of convergence condition ε. As a result, roughly setting for ε is not recommended.

We assume only learning (i.e., worker speed is always increased). Considering the case of decreasing worker speed by forgetting, worker behavior is changed, and it is hard to self-balance compared to only increasing speed. Therefore, this topic will be studied in future work.

References

1. Scholl A (1995) Balancing and Sequencing of Assembly Lines. Physica-Verlag.
2. Bartholdi JJ, Eisenstein DD (1996) A Production Line that Balances Itself. Oper Res 44: 21-34.
3. Bartholdi JJ, Bunimovich LA (1999) Dynamics of two- and three-worker "Bucket Brigades" production lines. Oper Res 47: 488-491.
4. Hirotani D, Morikawa K, Takahashi K (2006) Analysis and Design of Self-Balancing Production Line. Computers and Industrial Engineering 50: 488-502.
5. Bartholdi JJ, Eisenstein DD (2005) Using bucket brigades to migrate from craft manufacturing to assembly lines. Manufacturing & Service Operations Management 7: 121-129.
6. Bartholdi JJ, Eisenstein DD, Lim YF (2009) Deterministic chaos in a model of discrete manufacturing. Nav Res Log 56: 294-299.
7. Bratcu AI, Dolgui A (2009) Some new results on the analysis and simulation of bucket brigades (self-balancing production lines). Int J Prod Res 47: 369-387.
8. Armbruster D, Gel ES (2006) Bucket brigades revisited: are they always effective? Eur J Oper Res 172: 213-229.
9. Lim YF (2011) Cellular bucket brigades. Oper Res 59: 1539-1545.
10. Webster S, Ruben RA, Yang K-K (2012) Impact of storage assignment decisions on a bucket brigades order picking line. Prod Oper Manag 21: 276-290.
11. Nemberd DA, (2001) Heuristic approach for assigning workers to tasks based on individual learning rates. Int J Prod Res 39: 1955-1968.
12. Jaber MY, Kher HV, Davis DJ (2003) Countering forgetting through training and deployment. Int J Prod Econ 85: 33-46.
13. Shafer SM, Nembhard DA, Uzumeri MV (2001) The effects of worker learning, forgetting, and heterogeneity on assembly line productivity. Management Science 47: 1639-1653.
14. Villalobos JR, Gutierrez MA, Mar LR, Sanchez O, Ahumada O (2011) The use of dynamic work sharing production methods to reduce the impact of labour turnover in serial assembly lines. International Journal of Manufacturing Technology and Management 23: 34-53.
15. Armbruster D, Gel ES, Murakami J (2007) Bucket Brigades with worker learning. Eur J Oper Res 176: 264-274.
16. Hirotani D, Morikawa K, Takahashi K (2009) Analysis of self-balancing production line with individual learning. Proceeding of 20th International Conference on Production Research.
17. Wright TP (1936) Factors affecting the cost of airplanes. Journal of Aeronautical Sciences 3: 122-128.