Peltokorpi, Jaakko; Tokola, Henri; Niemi, Esko

Worker coordination policies in parallel station systems: performance models for a set of jobs and for continuous arrival of jobs

Published in:
International Journal of Production Research

DOI:
10.1080/00207543.2014.918290

Published: 01/01/2014

Document Version
Peer reviewed version

Please cite the original version:
Peltokorpi, J., Tokola, H., & Niemi, E. (2014). Worker coordination policies in parallel station systems: performance models for a set of jobs and for continuous arrival of jobs. International Journal of Production Research, 53(6). https://doi.org/10.1080/00207543.2014.918290

This is the authors accepted manuscript of an article published as the version of record in International Journal of Production Research on 21st May 2014, available online: http://www.tandfonline.com/10.1080/00207543.2014.918290.

Worker coordination policies in parallel station systems: performance models for a set of jobs and for continuous arrival of jobs

Jaakko Peltokorpi*

jaakko.peltokorpi@aalto.fi

Department of Engineering Design and Production, Aalto University School of Engineering, Espoo, Finland

Henri Tokola

henri.tokola@aalto.fi

Department of Engineering Design and Production, Aalto University School of Engineering, Espoo, Finland

Esko Niemi

esko.niemi@aalto.fi

Department of Engineering Design and Production, Aalto University School of Engineering, Espoo, Finland,

*Corresponding author: Jaakko Peltokorpi, jaakko.peltokorpi@aalto.fi, +358 50 566 2382, Address: Puumiehenkuja 3, 02150 Espoo, Finland

Worker coordination policies in parallel station systems: performance models for a set of jobs and for continuous arrival of jobs

Varying workloads and uncertain processing times in parallel assembly cause idle times for skilled, high-cost workers. This idleness can be avoided and the utilisation of the workers improved by allowing workers to move between the stations to help each other. Worker movement between assembly stations needs efficient and feasible coordination, and therefore, this paper compares four different worker coordination policies: no helping, floater, pairs and complete helping. The dynamics of the policies are modelled by studying the parallel assembly as a continuous-time Markov process. The system is studied with two different job release cases for non-identical jobs (customised products). In the first case, a given number of jobs have to be completed by the entire system. In the second case, new jobs arrive with a Poisson-distributed rate. The models assume that when one worker helps another, their collaborative inefficiency reduces the productivity. The models are used in numerical experiments to compare the performances of worker coordination policies as average job cycle times. The main conclusions from the results suggest the use of the complete helping policy in minor collaborative inefficiency conditions, especially with a given set of jobs. The pairs policy is a reasonable alternative in major inefficiency conditions with the continuous arrival of jobs.

Keywords: parallel assembly; worker coordination policy; moving workers; flexibility; collaborative efficiency

1. Introduction

As companies seek to meet increasing customer requirements through customisation assembly production faces variation, mostly caused by varying work content in each customised product. It is often reasonable to implement such production with a parallel stations design in which the stations are independent of each other. However, varying workloads and uncertain processing times cause idleness for specialised, high-cost assembly workers, which reduces the cost-efficiency of the system. As a solution, agile companies have responded to this problem by enabling flexible, cross-trained workers to move from station to station to help with the workload. An easily controlled and efficient worker coordination policy certainly increases the performance of assembly operations.

Worker coordination policies have been widely studied with production systems with different designs. There are some studies of systems with parallel design (e.g. Graves, 2008; Slomp and Molleman, 2002; Bokhorst, Slomp and Gaalman, 2004 and Bokhorst and Gaalman, 2009) and others with line design (e.g. Inman, Jordan and Blumenfeld, 2004; Hopp, Tekin and Van Oyen, 2004 and Sennott, Van Oyen and Iravani, 2006). Typically, worker coordination policies are studied from the perspective of cross-training. The present study, however, focuses on the actual dynamics and performances of the principles for workers helping each other.

The objective of this paper is to compare the performances of four different worker coordination policies in a parallel station system. The policies involve different amounts of worker collaboration in terms of helping others, starting from the *no-helping* policy and ending with the *complete helping* policy in which everyone helps everyone. The dynamics of the policies are modelled by studying the system as a continuous-time Markov process. The system is studied with two different job release cases: that of a *given set of jobs* and a *continuous arrival of jobs*. A given set of jobs can be thought of as a pre-determined set of jobs to be processed in its entirety in a given period (e.g. one week). This case is based on the common practice in which rough-cut planning assigns a weekly load for production. With the *continuous arrival of jobs* there is a Poisson-distributed arrival rate of jobs into the system and the jobs are processed according to the First-Come, First-Served (FCFS) principle. As the above two different release cases are both possible in assembly operations, it is relevant to study how they affect the performances of worker coordination policies. In both cases, jobs are first released to a single queue from which they are centrally assigned to a station when there is a need for jobs. The system that is studied has as many stations as workers. All the workers are assumed to be identical, but the customisation of products affects the exponentially distributed processing times of the jobs.

As a varying workload affects the idle times at a station the motivation to move workers between the stations to help each other is always to avoid idleness and improve worker utilisation. Therefore, it also makes sense to study the effects of collaborative inefficiency (i.e. the loss of efficiency caused by helping) on the performances of the policies. In general, a practical option for moving workers between stations would be to transfer a worker to another department. The profitability of that option can be better assessed on a case-by-case basis on the basis of the results of the present study. The models constructed in this study are used in numerical experiments to compare the performances of the policies in an average job cycle time.

The rest of the paper is organised as follows. Next, Section 2 presents a literature review of worker coordination policies and of studies dealing with efficiency in worker collaboration. Then Section 3 describes the assembly system under study and presents the Markov process models for the different worker coordination policies under study. Section 4 presents the experimental results and compares the performances of the policies. Finally, Section 5 draws conclusions and suggests some aspects for future research.

2. Literature review

This section contains the literature review of worker coordination policies. The review has been divided into two parts. First, different worker coordination policies and their performances are reviewed. Second, studies dealing with efficiency in worker collaboration are reviewed.

2.1 *Worker coordination policies*

Hopp and Oyen (2004) stated that a *worker coordination policy* 'allocates workers to tasks (or tasks to workers) over time'. Several earlier studies examined worker coordination policies to show how they influence system performance (see the paper of Saadat et al., 2013, in which challenges and trends concerning workforce allocation are reviewed). Since in production systems workers are typically specialised in one task of one station, workforce flexibility and coordination between the stations are based on cross-training that enables workers to perform multiple task types.

The literature shows that the existing worker coordination policies differ from each other in their degrees of workforce flexibility. A well-known strategy that enables one to increase flexibility in small increments is chaining, a notion that was originally introduced by Jordan and Graves (1995) in their study of flexibility in parallel manufacturing processes. The study compared different configurations in the context of a set of plants building a set of products. The principle of chaining was applied by Inman, Jordan and Blumenfeld (2004) in the context of cross-training workers between the sections of an automotive assembly line. The study compared how different cross-training policies compensate for absenteeism while minimising investments in cross-training. It was shown that cross-training the existing utility worker of each section for a task in the next section (the sections can be viewed as being parallel) by means of a closed, complete chaining produced the best result. In addition, e.g. cross-training the utility workers of two adjacent sections as pairs 'as a collection of small chains' was found to perform well. In any case, the study concluded that 'each step towards chaining is valuable'.

Graves (2008) examined cross-training for one additional product in an assembly system of six parallel stations with a varying weekly demand for each different product of each station. The purpose of cross-training was to avoid the over- and under-utilisation of specialised workers. In relation to the system with no cross-training, the closed chain policy reduced the amount of overtime almost twice in comparison to cross-training in pairs.

Hopp, Tekin and Van Oyen (2004) compared cherry-picking (CP), in which all workers can directly help the bottleneck station, and two-skill chaining (2SC), which enables each worker to help the next station on the assembly line. The results from this simulation study provided that it can be ineffective to cross-train the capacity directly for a certain station, as in CP. In contrast, strategies based on 2SC that shift the capacity indirectly between the stations are usually more effective, especially in circumstances of high variability in processing speed and low *WIP* (Work-In-Process).

Bokhorst and Gaalman (2009) studied a parallel system with fewer workers than machines. In their system, machines can process jobs with either a high or low mean processing time and were divided into two groups accordingly. The study found that the system benefits from fully cross-training only with at most a moderate difference in mean processing times between the groups. Otherwise, cross-training between groups is not advised.

Slomp and Molleman (2002) studied cross-training policies for a team with fewer workers than tasks. The policies were related to the question 'who should be cross-trained

next and for which task?' The study shows that, when developing a team, more effort should be invested into the multi-functionality of workers than cross-training workers for critical tasks. Yang (2007) compared a set of cross-training policies which comprise different numbers of cross-trained workers, of additional skills per cross-trained worker and of additional machines of each department in a job shop. The study shows that for a fixed total number of skills it is more beneficial to distribute the skills over more workers than to cross-train a few workers in more skills. The study also reported that cross-training is highly recommended in environments with e.g. relatively good proficiency of each worker in a cross-trained department, a high labour utilisation rate and high variability in processing times. Easton (2011) reports that, when cross-trained workers are less proficient than specialists, a higher proportion of cross-trained workers in total workforce reduces capacity shortages but requires a larger total number of workers to respond to the demand. Thus, in this case, the optimum cross-training level is a trade-off between workforce size and capacity shortages.

Nembhard and Bentefouet (2012) examined assigning of workers, each of them for one of the parallel tasks and for the same length of time period at a time. The study assumed that workers are never starved for work, tasks have stochastic processing times and workers have learning and forgetting properties. The results of the study suggest assigning workers in such a way that each worker is specialised in one task. However, in the case of production requirements on each task, workers may need to perform additional task(s). In this context, pure specialisation was still found to be more significant than the possibility of cross-training. These results were later extended by Nembhard and Bentefouet (2014) who showed that, when implementing cross-training, selecting workers from a general pool for a set of tasks based on their highest level of previous experience at any task provides better results than selection based on their expected projected output across all tasks.

Sennott, Van Oyen and Iravani (2006) studied fully cross-trained floating workers (floaters), and compared a system with only specialists to a flexible system with one floater and specialists. On the basis of the results of the tests, with a two-station line a system with four specialists did not perform much better than a flexible system with three workers (depending on the server utilisation rate that was generated). Thus, the study pointed out that floaters were valuable in providing line capacity balancing.

Andradottir, Ayhan and Down (2001) studied worker coordination in a finite queuing system in order to obtain the near-optimal long-run average throughput. They suggest that workers should not idle but should help others when the processing times for each task are independent of the worker or, alternatively, when each worker processes at about the same speed on all tasks. In contrast, when the processing times depend both on the worker and on the task, the study suggests assigning a worker to each task in such a way that the product of the processing rates of the workers at their assigned tasks is maximised. This requires workers to be instructed to avoid idleness by working on tasks that will enable them to get back to work at their primary task as soon as possible, and that the worker processing rates do not vary a lot for the tasks that are not their primary

ones. These findings, as well as the results of other studies, explain the influence of worker coordination policies on system performance.

In relation to most of the studies reviewed above, instead of studying workforce flexibility through cross-training, the present paper studies actual worker coordination policies in terms of workers helping each other and, specifically, in a parallel system. Finally, another innovative feature of the present paper is that we study the dynamics of the policies in two different job release cases: that of a *given set of jobs* and a *continuous arrival of jobs*.

2.2 *Efficiency in worker collaboration*

If a single worker's task is processed by multiple workers at the same time, the collaborative processing might be faster or slower than just the sum of the processing times of single workers. According to Hopp and Oyen (2004), a basic measure of *collaborative efficiency*, a term they use, is 'the relative percentage increase in average task speed (or labor productivity) that results from assigning multiple workers to the same task'. In this paper, the term *collaborative inefficiency* is used, since collaborative inefficiency has only the negative meaning that two workers (as the maximum number of collaborative workers in this paper) can process the same task only less than twice as fast as one. Collaborative inefficiency can occur e.g. as a result of worker interaction or inadequate work space for two workers (see the paper of Hytönen, Niemi and Pérez, 2010, in which inefficiency caused by e.g. the overmanning of an assembly station is considered).

The effects of collaborative inefficiency in different assembly cases and circumstances should be identified in order to build up effective collaboration practices. However, it is not easy as realistic values for productivity loss need empirical and laborious case-by-case studies. The loss of labour productivity as a result of overmanning is reported in an empirical study of mechanical and sheet metal contractors by Hanna et al. (2007). In their report, with 100% overmanning the productivity loss varies from 4 to 20%, depending on the manpower (from 5 up to 50). Gunduz (2004) summarises the variation in the loss of productivity as being between 25% and 47% with 100% overmanning. On the basis of the results of the two reports cited above, productivity loss values may vary a lot on a case-by-case basis. Moreover, the studies above deal with the construction industry, where products and task times are different and the numbers of workers who collaborate are much larger than in this study, which deals with assembly stations.

Because of a lack of empirical studies on collaborative inefficiency in assembly work, it is attractive to make assumptions or simplifications on the effects of inefficiency. As an example, Sengupta and Jacobs (2004) compared assembly line design without collaboration to the parallel, cell-based design of two single tasks with collaboration. The paper introduced an inefficiency factor (from 1 down to 0.4) covering the loss of efficiency as a result of worker transfers, tool constraints, space to work in or just working in teams. The factor values indicate the efficiency from no impact of collaboration (main experiments) to a negative impact in which the productivity of each worker in

collaboration decreases, and finally, that collaboration even increases the completion time for the current task. The revised remaining time for task completion was calculated as [remaining time at task/(inefficiency factor * number of workers at task after transfer)]. In the conditions of a low setup time and a low level of variation in processing times, line design was found to perform better than parallel design when the inefficiency factor in parallel design equalled 0.7 or lower. With a high setup time and a high level of variation in processing times, the corresponding value was 0.5.

The present paper tests collaborative efficiency factors equivalent to the factors of Sengupta and Jacobs (2004). We define minor collaborative inefficiency in pair-working as being 10% (with a collaborative efficiency factor of 0.9) and major inefficiency as being 30% (with a collaborative efficiency factor of 0.7), respectively.

3. Worker coordination policies for parallel stations

This section studies the different worker coordination policies in the parallel station assembly system. We define a worker coordination policy as simply describing how the workers move between the assembly stations or how jobs that have arrived are assigned to the workers under varying work circumstances. The following four policies are studied:

(1) no helping
(2) floater
(3) pairs
(4) complete helping

In order to do this, first, the parallel station system, its properties and the notation that is used are presented. Second, the above policies are described in detail. Third, Markov models for each policy are formed for two different job release cases, first, in the case of a *given set of jobs* and, second, in the case of a *continuous arrival of jobs*.

3.1 *Parallel station system*

The parallel station system under study has the following properties:

i. the number of parallel work stations N is even and it is the same as the number of workers;
ii. the products are customised (non-identical products), workers have equal mean processing rates (identical workers) and the processing time of each job is exponentially distributed;
iii. the jobs are centrally assigned to a station from a single queue in the event that there is a need for jobs. If a new job arrives while all the stations are busy, the job is added to the queue;
iv. when there is a need for jobs, but the queue is empty, the worker can go to help others according to the worker coordination policy that is in use;
v. the maximum number of workers collaborating on the same job is two;

vi. when two workers collaborate, their collaborative inefficiency slows the processing.

The system is examined with two job release types. The first type, a *given set of jobs*, studies the case in which the system has to assemble exactly W jobs in a given period. This set of jobs can be e.g. a weekly load for production. The second type, *continuous arrival of jobs*, means that there is a Poisson-distributed arrival rate of jobs with a mean of λ and the jobs are processed according to the First-Come, First-Served (FCFS) principle. In both types, varying work content in each product causes exponentially distributed processing times of jobs, and thus each worker has a Poisson-distributed processing rate with a mean of μ.

When there is a need for jobs at a station and if no jobs exist in the queue, the worker is allowed to help other stations, according to the policy that is used. Collaborative efficiency, calculated by the factor α, takes place in the situations in which one worker helps another worker. If there is no inefficiency in collaboration, the factor α is one and when inefficiency occurs, α is below one. The processing rate of a pair is the processing rate of two single workers multiplied by the collaborative efficiency factor as follows: $\alpha 2\mu$. When the system is studied as a Markov process, at each moment the state of the system can be determined by the number of single workers (S) and pairs (P) who are processing, as well as by the number of jobs in the queue (Q) waiting for processing.

To summarise, the following notation is used:

N	number of parallel stations in the system (even)
W	initial number of jobs in the case of the *given set of jobs*
λ	mean arrival rate of jobs into the system (Poisson-distributed) in the case of the *continuous arrival of jobs*
μ	mean processing rate for one worker (Poisson-distributed)
α	factor of collaborative efficiency, indicating the performance of a work pair in relation to the performance of two single workers.
n	number of unfinished jobs in the system
S	number of single workers processing
P	number of pairs of workers processing
Q	number of jobs in a queue waiting for processing

For the case of the *continuous arrival of jobs*, the following three equations hold. The average number of products within the system (*WIP* = Work-In-Process) is the following (Curry and Feldman, 2011)

$$WIP = \sum_{n=0}^{\infty} np_n \qquad (1)$$

where p_n is the steady-state probability, i.e. the percentage of time a system has n jobs.

According to Little's Law (Little, 1961), the average job cycle time, CT, is

$$CT = \frac{WIP}{\lambda} \qquad (2)$$

The average system utilisation rate *u* is

$$u = \frac{\lambda}{N\mu} \qquad (3)$$

3.2 *Worker coordination policies under study*

Worker coordination policies are used to keep workers utilised in situations where the workload varies. Section 2.1 discussed the principles of some previously studied policies. In this study, the policies differ from each other purposely by having different restrictions on worker collaboration. In this manner the influence of practical worker movement and collaboration on system performance can be examined. The four policies that are studied were discussed in a previous simulation study (Peltokorpi, Tokola and Niemi, 2012). The principles of worker movements are explained below.

3.2.1 *No helping*

In the *no-helping* policy, there is no movement or collaboration of workers. This policy acts as a reference for the performances of actual worker coordination policies.

3.2.2 *Floater*

In the *floater* policy, the system has one flexible floater. After completing the job at his home station, the floater moves to help a randomly selected station that has work left. If the work at these stations is subsequently completed, the floater moves to another station that has work left. Since there is only one floater in the system other workers who complete their jobs will be idle. Therefore worker flexibility in this policy is strongly limited.

3.2.3 *Pairs*

In the *pairs* policy, workers form a fixed pair with the neighbouring station. A worker who has no work at his home station will help his pair partner. This policy restricts the number of stations helped by a worker to one.

3.2.4 *Complete helping*

In the *complete helping* policy, a worker will move to help a randomly selected station that has work left, but is not yet being helped. The amount of collaboration, measured as the number of different stations helped, is not limited; it is complete. Thus with this policy

everyone helps everyone.

Figure 1 illustrates the movement of workers with different worker coordination policies in the case of six parallel stations (N = 6). The assignment of jobs that have arrived is explained in the next section.

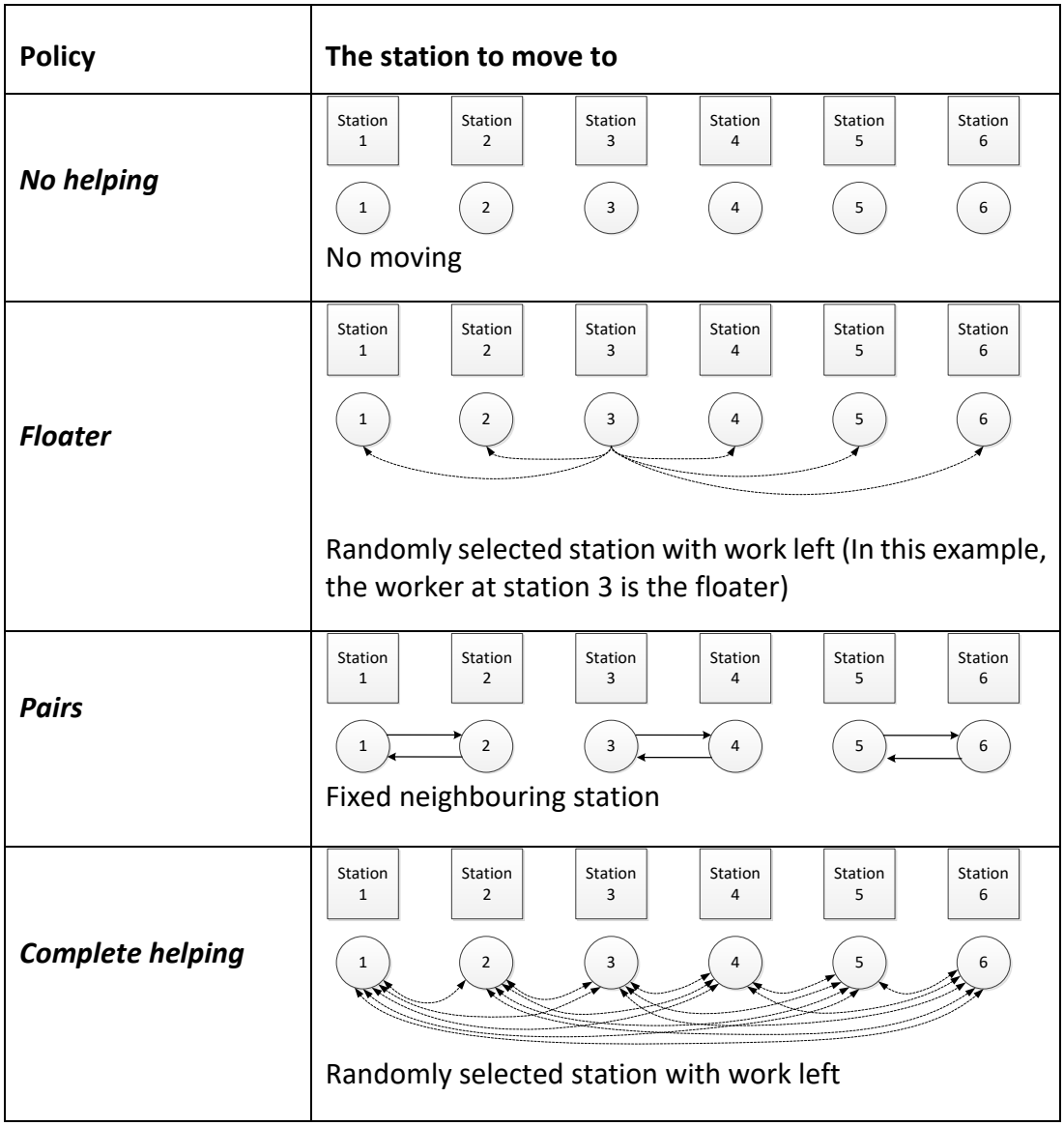


Figure 1. Movement of workers with different worker coordination policies (N = 6).

3.3 *Markov models*

The policies are studied as Markov models. For each policy, two performance models are created: one for the case with a *given set of jobs* and another for the case with a *continuous arrival of jobs*.

3.3.1 *The no-helping policy*

The *no-helping* policy does not allow any movement or collaboration of workers. The

system with this policy can be modelled similarly to a basic M/M/s multi-server queuing model as presented in the review of Hillier and Lieberman (1995). There are *N* parallel stations (*s* = *N*). In such a model, there is an infinite number of states, numbered from 0 to infinity. Now, the notation *n* indicates the number of a single state. *n* also corresponds to the number of unfinished jobs in the system. In *no helping*, there is no pair working, and therefore the number of active pair workers (*P*) is always zero. Instead, the number of single workers (*S*) can vary from zero to that of the number of stations, *N*, and equals the number of unfinished jobs, *n*, in this range. If there are more than *N* jobs (*n* > *N*), then the remaining jobs will be added to the queue, which increases the number of items in the queue (i.e. the length of the queue is *Q* = *n*-*N*). A job is then completed at the rate of $N\mu$ and single workers always take another job from the queue (this is also assumed with the models of other worker coordination policies later). The processing rate in the state *n*, denoted by μ_n, is the number of active single workers multiplied by the processing rate, μ, in the following way:

$$\mu_n = \begin{cases} n\mu & if\ 0 < n < N \\ N\mu & if\ n \geq N \end{cases} \qquad (4)$$

This is also the transition rate from the state *n* to the state *n*-1 when the completion of any job occurs. If there is a job that is arriving, another transition can occur from the state *n* to the state *n*+1 at the arrival rate of jobs, λ. When *n* < *N*, a job that arrives is assigned to an idle worker. Otherwise, the length of the queue (*Q*) is increased by 1. Figure 2 presents the transition rates in a Markov state diagram for six parallel stations (*N* = 6) with the *no-helping* policy.

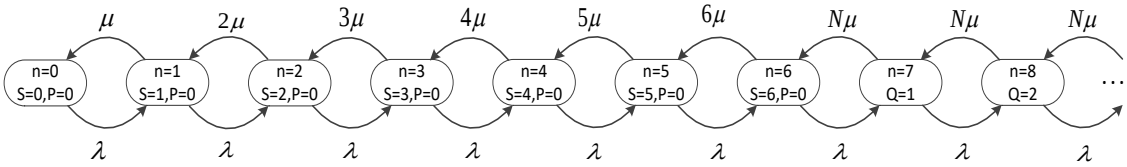


Figure 2. Markov state diagram for six parallel stations with the *no-helping* policy.

Using the above model, first, the case with a given set of *W* jobs is studied. In this case, there is no arrival of new jobs, i.e. λ = 0. The total time spent in each state *n* is calculated by multiplying the processing time (i.e. the inverse of the processing rate from Equation (4)) by the number of unfinished jobs, which is *n* as well. Then the average cycle time, CT, is calculated by summing these times in the given job set *W* and dividing it by *W* as follows:

$$CT = \frac{\sum_{n=1}^{W} (1/\mu_n)\, n}{W}$$

$$= \begin{cases} 1/\mu & if\ 0 < W < N \\ \dfrac{(N-1)\,(1/\mu) + (1/(2N\mu))(W-N+1)(N+W)}{W} & if\ W \geq N \end{cases} \qquad (5)$$

In Equation (5), when $W \geq N$ the first term sums the processing times up to N-1 and the second term sums the processing times as an arithmetical series from N up to W.

Second, the case of the *continuous arrival of jobs* is studied. The steady-state probability for each state n, denoted by p_n, has to be solved. The balance equations for each state n are solved by the cut partitioning method between the nodes n and n+1 as presented e.g. in the example of Curry and Feldman (2011). By applying their procedure, first, the following balance equations are needed:

$$\begin{aligned} \lambda p_0 &= \mu p_1 \\ \lambda p_1 &= 2\mu p_2 \\ &\vdots \\ \lambda p_{N-1} &= N\mu p_N \\ \lambda p_N &= N\mu p_{N+1} \\ \lambda p_{N+1} &= N\mu p_{N+2} \\ &\vdots \end{aligned}$$

$$\sum_{n=0}^{\infty} p_n = 1. \qquad (6)$$

Second, the steady-state probabilities p_n can be solved from Equation (6) and presented recursively in terms of p_0 as follows (presented e.g. by Hillier and Lieberman, 1995):

$$p_n = \begin{cases} \dfrac{\rho^n}{n!} p_0 & if\ 0 < n < N \\ \dfrac{\rho^n}{N!\, N^{n-N}} p_0 & if\ n \geq N \end{cases} \qquad (7)$$

where $\rho = \lambda/\mu$. Third, using Equations (6) and (7), the actual value for p_0 is calculated and it is the following (Hillier and Lieberman, 1995):

$$p_0 = 1/\left(\sum_{n=0}^{N-1}\frac{\rho^n}{n!} + \frac{\rho^N/N!}{1-(\rho/N)}\right) \quad (8)$$

where the last term in the summation is based on the sum of geometric series. The value p_0 from Equation (8) is used in Equation (7) to calculate the values for p_n. This can again be combined with Equations (1) and (2) in order to calculate the cycle time.

3.3.2 *The floater policy*

In the *floater* policy there is a single floater who helps others. Now the difference to the *no-helping* policy is that the effect of the floater on the processing rate is taken into account. The principle of the floater is modelled as follows. When the floater does not have work at his station he can help a randomly selected station with work left. After finishing at that station, the floater can move again to help another station with work left. When the floater helps someone, the processing speed at that station increases from the single worker processing rate μ to the pair working processing rate $\alpha 2\mu$, where α is the factor of collaborative efficiency.

With the *floater* policy a single state cannot always be defined only by the number of unfinished jobs, n, but as a set of (n, S, P), where $S+P = \min(n, N)$. However, the number of *pairs*, P, can only be either one (the floater does not have a job at his station and is helping) or zero (the floater has a job at his station). The average processing rate $\mu_{(n,S,P)}$ in a state is the sum of the processing rates of single workers ($S\mu$) and of a pair of the floater and another worker ($\alpha P 2\mu$) as follows:

$$\mu_{(n,S,P)} = \begin{cases} (S + \alpha P 2)\mu & if\ 0 < n < N \\ N\mu & if\ n \geq N \end{cases} \quad (9)$$

When a job is processed in states where $n \leq N$, a state transition occurs, depending on n and P. When $2 \leq n \leq N$ and $P = 0$, a transition occurs, depending on whether the floater or a normal worker completes a job. The floater completes a job with the probability of $1/n$ and goes to help another worker. In this case the number of active single workers, S, decreases by two and P is set to one, i.e. the system moves from the state (n, S, 0) to the state (n-1, S-2, 1). Otherwise a normal worker completes a job and becomes idle and only S decreases by one, i.e. the system moves from the state (n, S, 0) to the state (n-1, S-1, 0). When $2 \leq n < N$ and $P = 1$, the transition always occurs from the state (n, S, 1) to the state (n-1, S-1, 1), regardless of whether the pair or a single worker completes the job.

If new jobs arrive, they are primarily assigned to idle normal workers (randomly selected). When $n = 0$, a job is assigned to such a worker and the floater starts helping him. When $1 < n < N$, a job is assigned to an idle normal worker alone. When $n = N$-1 and $P = 1$, the floater stops helping and takes care of the job that has arrived. Otherwise, if $n \geq N$, the length of the queue (Q) is increased by 1. Figure 3 presents the transition rates in a Markov state diagram for six parallel stations ($N = 6$) with the *floater* policy.

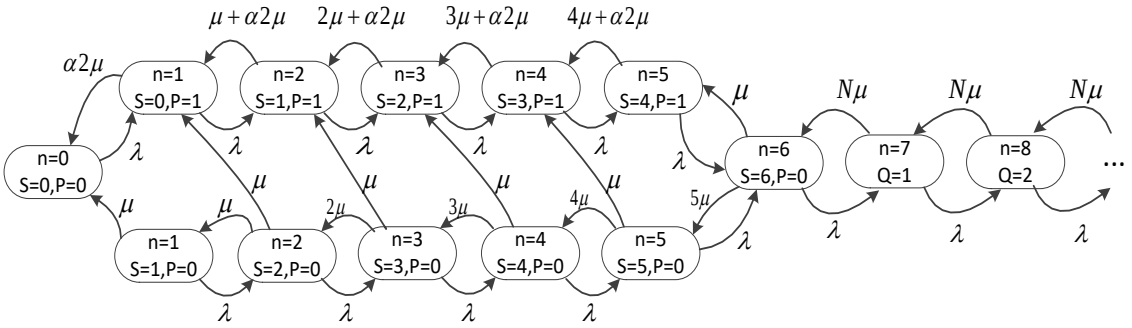


Figure 3. Markov state diagram for six parallel stations with the *floater* policy.

Using the above model, first, the case of a given set of W jobs is examined. If $W < N$, it is clearly most efficient if the floater helps all the time (i.e. $P = 1$). Then the average time spent processing the next job when there are n jobs is $1/((n\text{-}1+2\alpha)\mu)$. The average CT is calculated by weighting the previous average time by the number of jobs and dividing it by W.

If $W \geq N$, the floater does not help initially, but he moves to help when he completes his job and the queue is empty. In this case, for each number of unfinished jobs n, when $1 \leq n < N$, whether the floater is helping or is not has to be taken into account. If the floater is not helping ($P = 0$) the probability that the floater completes his job in a state is $1/n$. Thus, for n jobs, the probability that $P = 0$ is n/N and the probability that $P = 1$ is $(N\text{-}n)/N$. If $P = 0$, the time spent processing the next job is $1/n\mu$ and if $P = 1$, the time spent processing the next job is $1/((n\text{-}1+2\alpha)\mu)$. Then the average time spent processing the next job when there are n jobs is calculated by multiplying the probability by the time spent. The average CT is calculated by weighting the previous average time by the number of jobs and dividing it by W. Thus, the average CT's for different W are calculated as follows:

$$CT = \frac{\sum_{n=1}^{W} (1/\mu_{(n,S,P)})\, n}{W}$$

$$= \begin{cases} \left(\sum_{n=1}^{W} n/((n-1+2\alpha)\mu) \right)/W & if\ 0 < W < N \\ \left(\sum_{n=1}^{N-1} \left(\frac{1}{N\mu} + \frac{N-n}{(n-1+2\alpha)N\mu} \right) n \right. & \\ \left. +(1/(2N\mu))\,(W-N+1)(\mathrm{N}+W) \right)/W & if\ W \geq N \end{cases} \quad (10)$$

In the case of the *continuous arrival of jobs* the steady-state flow-balance equations for each state can be determined by applying the method (presented e.g. by Hillier and Lieberman, 1995) that for any state of the system the total rate entering equals the total rate leaving. Applying this procedure we have such numbers of variables and equations that $p_{(n,S,P)}$, a steady-state probability for each valid state (n, S, P), can be solved. Since

the procedure leads to a large number of equations, the equations, as well as the steady-state probability formulae, are omitted for this policy. If needed, each probability $p_{(n,S,P)}$ can be solved numerically from the transition rate matrix, as was done in the numerical experiments of this paper. After that, Equations (1) and (2) can be used to calculate the average CT for the *floater* policy.

3.3.3 *The pairs policy*

In the *pairs* policy, workers form a fixed pair with the neighbouring station. In this paper, we assume that the number of stations, N, is even and thus there is a pair partner for every worker. According to this policy, a worker who does not have a job will help his pair partner. Similarly to the *floater* policy, the number of unfinished jobs in the system, n, is not enough to define the states of the system. The sum of active single workers, S, and pairs, P, is always the minimum from n and N, but there may be several possible combinations of S and P. Thus, a single state can be defined as a set of (n, S, P), where $S+P = \min(n, N)$. The average processing rate $\mu_{(n,S,P)}$ in a state is the sum of the processing rates of single workers ($S\mu$) and pairs ($\alpha P2\mu$) as follows:

$$\mu_{(n,S,P)} = \begin{cases} (S + \alpha P2)\mu & if\ 0 < n < N \\ N\mu & if\ n \geq N \end{cases} \qquad (11)$$

When a job is processed in the states where $n \leq N$ the transitions depend on the actual numbers of S and P in the following way. With the probability of $S/(S+\alpha P2)$ a single worker completes the job and moves to help his pair partner. In this case, the number of single workers, S, decreases by two and the number of pairs, P, increases by one, i.e. the system moves from the state (n, S, P) to the state (n-1, S-2, P+1). Further, with the probability of $\alpha P2/(S+\alpha P2)$ a pair completes the job. In this case, the number of pairs decreases by one, i.e. the system moves from the state (n, S, P) to the state (n-1, S, P-1).

If new jobs arrive, they are primarily assigned to idle pairs (randomly selected) if at least one idle pair exists. If this is the case, the system moves from the state (n, S, P) to the state (n+1, S, P+1). Otherwise, a worker from a busy pair takes care of the job that has arrived, i.e. the system moves from the state (n, S, P) to the state (n+1, S+2, P-1), and if all the workers are already working as singles (i.e. $S = N$) the length of the queue (Q) is increased by 1. Figure 4 presents the transition rates in a Markov state diagram for six parallel stations ($N = 6$) with the *pairs* policy.

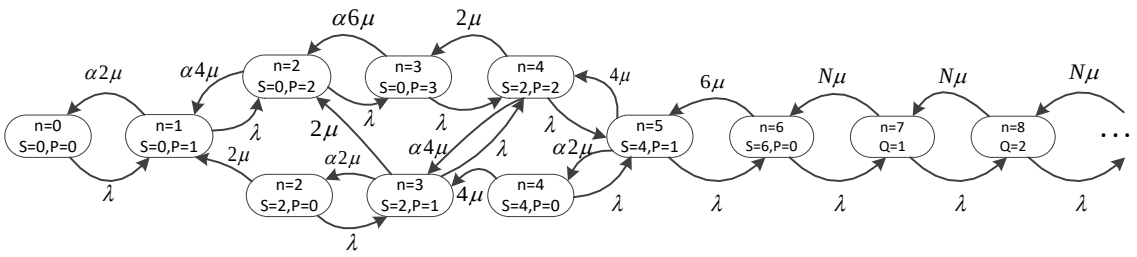


Figure 4. Markov state diagram for six parallel stations with the *pairs* policy.

Using the above model, first, the case of a given set of *W* jobs is examined. In this case, since each pair has the same average processing speed, and the pairs are independent of each other, it is sufficient to study one pair. The processing time of one pair can be thought of as being divided into two states. First, there is the time both workers take to process their jobs as single workers until the completion of either of their jobs. Second, there is the time both workers take to process the remaining job as a pair. The total time for one pair to complete two jobs is divided by two to get the average cycle time, CT, for one job. In the case of $W \leq N$, when the maximum number of workers is used initially and since *S*+*P* equals *W*, the number of initial single workers (*S*) is the maximum from (2*W*-*N*, 0) and the number of initial pairs (*P*) is the minimum from (*N*-*W*, *W*). On the basis of the study of one pair described above, the initial *S* and *P* are divided so that the average processing time of a single worker $1/\mu$ is multiplied by (*S*/2) and the average processing time of a pair $1/\alpha 2\mu$ by (*S*/2+*P)*. The average CT is the sum of the above processing times divided by *W*. After that the case with $W > N$ can be applied. Thus, the average CT in both cases is calculated as follows:

$$CT = \begin{cases} \dfrac{S/(2\mu) + ((S/2) + P)/(\alpha 2\mu)}{W} & if\ W \leq N, S = \max(2W - N, 0) \\ & \qquad P = \min(N - W, W) \\ ((N-2)/(2\mu) + ((N-2)/2 + 1)/(\alpha 2\mu) & \\ +(1/(2N\mu))\,(W - N + 1)(N + W))/\mathrm{W} & if\ W > N \end{cases} \quad (12)$$

In the case of the *continuous arrival of jobs* the steady-state flow-balance equations can be determined by applying the same method as with the *floater* policy, that for any state of the system the total rate entering equals to the total rate leaving. Because of the complex and *N*-dependent transitions between the states, the balance equations are tedious to construct with the *pairs* policy. Therefore, the balance equations and steady-state probability formulae are omitted for this policy. If needed, a steady-state probability $p_{(n,S,P)}$, for each (*n*, *S*, *P*), can be solved numerically from the transition rate matrix, as was done in the numerical experiments of this paper. After that, Equations (1) and (2) can be used to calculate the average CT for the *pairs* policy.

3.3.4 *The complete helping policy*

According to the *complete helping* policy, when a worker completes his job, he is allowed

to help others without any restrictions. The worker to be helped is a randomly selected one that has work left, but is not yet being helped. With the *complete helping* policy, when $n < N$, there will be a minimum of (N-n, n) workers working as pairs. Thus, the average processing rate μ_n in each state n can be written as follows:

$$\mu_n = \begin{cases} \big((2\alpha - 1)\min(N-n, n) + n\big)\mu & if\ 0 < n < N \\ N\mu & if\ n \geq N \end{cases} \quad (13)$$

Equation (13) also defines the transition rate from the state n to the state n-1 when the completion of a job occurs. If new jobs arrive, they are primarily assigned to idle pairs (randomly selected) if at least one idle pair exists. Otherwise, a worker from a busy pair takes care of the job that has arrived and if all the workers are already working as singles (i.e. $S = N$), the length of the queue (Q) is increased by 1. Figure 5 presents the transition rates in a Markov state diagram for six parallel stations ($N = 6$) with the *complete helping* policy.

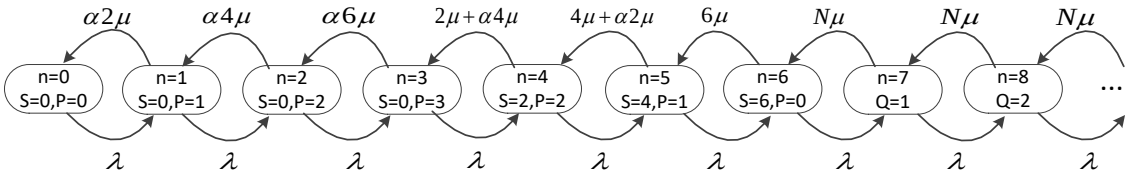


Figure 5. Markov state diagram for six parallel stations with the *complete helping* policy.

First, the case of a given set of W jobs is examined. The average job cycle time, CT, can be calculated with a similar principle as with *no helping* and *floater*, as follows:

$$CT = \frac{\sum_{n=1}^{W}(1/\mu_n)\,n}{W}$$

$$= \begin{cases} 1/(\alpha 2\mu) & if\ 0 < W \leq N/2 \\ \Big((N/2)/(\alpha 2\mu) + \sum_{n=N/2+1}^{W} n/\Big(\big((2\alpha-1)(N-n)+n\big)\mu\Big)\Big)/\mathrm{W} & if\ N/2 < W < N \\ ((N/2)/(\alpha 2\mu) + (\sum_{n=N/2+1}^{N-1} n/(((2\alpha-1)(N-n)+n)\mu)) \\ +\,(1/(2N\mu))\,(W-N+1)(N+W))/W & if\ W \geq N \end{cases} \quad (14)$$

In Equation (14), if $N/2 < W < N$ or if $W \geq N$, the first term in the equation sums the processing times up to $n = N/2$.

Second, the case of the *continuous arrival of jobs* is examined. The steady-state flow-balance equations for each state n can be written by the cut partitioning method (as with *no-helping*). Further, each steady-state probability p_n can be solved and presented in terms of p_0 as follows:

$$p_n = \begin{cases} \dfrac{\rho^n}{\alpha^n 2^n (n!)} p_0 & if\ 0 < n \leq N/2 \\ \dfrac{\rho^n}{\prod_{i=N/2+1}^{n} (2i - N) + \alpha 2(N - i)} \left(\dfrac{1}{\alpha^{(N/2)} 2^{(N/2)} (N/2)!} \right) p_0 & if\ N/2 < n < N \\ \dfrac{\rho^n}{N^{n-N+1}} \left(\dfrac{1}{\prod_{i=N/2+1}^{N-1} (2i - N) + \alpha 2(N - i)} \right) \left(\dfrac{1}{\alpha^{(N/2)} 2^{(N/2)} (N/2)!} \right) p_0 & if\ n \geq N \end{cases} \quad (15)$$

where $\rho = \lambda/\mu$. Since the sum of all the probabilities is 1 (Equation 6), the actual value for p_0 can be calculated from the following:

$$p_0 = 1/\left(1 + \sum_{n=1}^{N/2} \frac{\rho^n}{\alpha^n 2^n (n!)} + \sum_{n=N/2+1}^{N-1} \left(\frac{\rho^n}{\prod_{i=N/2+1}^{n} (2i - N) + \alpha 2(N - i)} \right) \left(\frac{1}{\alpha^{(N/2)} 2^{(N/2)} (N/2)!} \right) + \left(\frac{N^{N-1}}{\prod_{i=N/2+1}^{N-1} (2i - N) + \alpha 2(N - i)} \right) \left(\frac{1}{\alpha^{(N/2)} 2^{(N/2)} (N/2)!} \right) \left(\frac{(\rho/N)^N}{1 - (\rho/N)} \right) \right) \quad (16)$$

The value p_0 from Equation (16) is used in Equation (15) to calculate the values for each probability p_n. This can again be combined with Equations (1) and (2) in order to calculate the average CT.

4. Numerical results

This section studies numerically the models introduced in the previous section. First, the parameter values that are used for calculating the performances of worker coordination policies are presented. Then the results for a *given set of jobs* and *the continuous arrival of jobs* are presented and analysed. The results for a *given set of jobs* are calculated by the equations described in Section 3.3. The results for the *continuous arrival of jobs* are generated from the transition rate matrices by using the Monte Carlo simulation method and they are validated with the equations and methods presented in Section 3.3. The results, the performances of the policies, are compared at the end of this section.

4.1 *Parameters*

The following parameter values are used. The number of parallel stations goes from N = 2 up to N = 8. The default parameter value in the experiments is N = 8 as it enables a relatively large number of workers to collaborate, which is relevant for the purpose of this study. The workers are assumed to be identical and they all have the same Poisson-

distributed processing rate, with a mean of $\mu = 1$. This is kept fixed in all the experiments. In the case of a *given set of jobs*, the number of jobs in each set is fixed to $W = N$ except for the default $N = 8$ stations, in which the number of jobs W is varied from 1 to 16. In the case of the *continuous arrival of jobs*, the system utilisation rate is fixed to $u = 0.75$, except for $N = 8$, where it is varied with steps of 0.125 from (almost) zero up to 0.875.

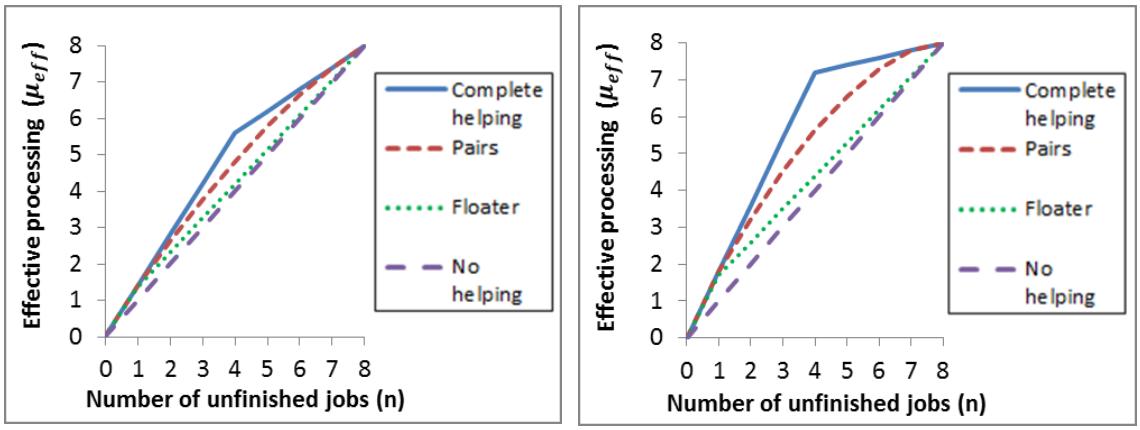
The collaborative efficiency factor in pair working, α, varies from 0.5 up to 1. The fixed comparative values are set to $\alpha = 0.7$, indicating major collaborative inefficiency (a 30% loss in the processing rate of one pair, $\alpha 2\mu = 1.4$) and $\alpha = 0.9$, indicating only minor inefficiency (a 10% loss in the processing rate of one pair, $\alpha 2\mu = 1.8$), respectively.

4.2 *Results for given set of jobs*

In the case of a *given set of jobs*, W jobs have to be processed. The processing of the given jobs is started with the maximum possible number of workers. The flexibility of movement that is allowed, determined by the worker coordination policy that is used, affects the effective processing rate μ_{eff} in each $n < N$. To clarify this, μ_{eff} with different policies is the sum of the processing rates of single workers (S) and pairs (P) in each state n as follows:

$$\mu_{eff} = (S + \alpha P2)\mu \qquad (17)$$

Figure 6 presents the dependence of the effective processing rate on the number of unfinished jobs n in the case of a given set of $W = N = 8$ jobs and with (a) major ($\alpha = 0.7$) and (b) minor ($\alpha = 0.9$) collaborative inefficiency conditions. [Equations (4), (9), (11) and (13)]



(a) $\alpha = 0.7$ (b) $\alpha = 0.9$

Figure 6. *Given set of jobs*: dependence of effective processing rate on number of unfinished jobs (W = $N = 8$, $\mu = 1$)

As the figure shows, *complete helping* is the most effective, followed by *pairs*, *floater* and *no helping*. The non-linearity of the processing rate of *pairs* is due to the more complex transitions in the *pairs* policy.

Figure 7 presents the average job cycle time (CT) as a function of the collaborative efficiency factor α [Equations (5), (10), (12) and (14)]. In the figure, the maximum collaborative efficiency (α = 1) indicates the situation in which no inefficiency occurs in pair working. On the other hand, α = 0.5 efficiency indicates the situation in which helping others is completely useless.

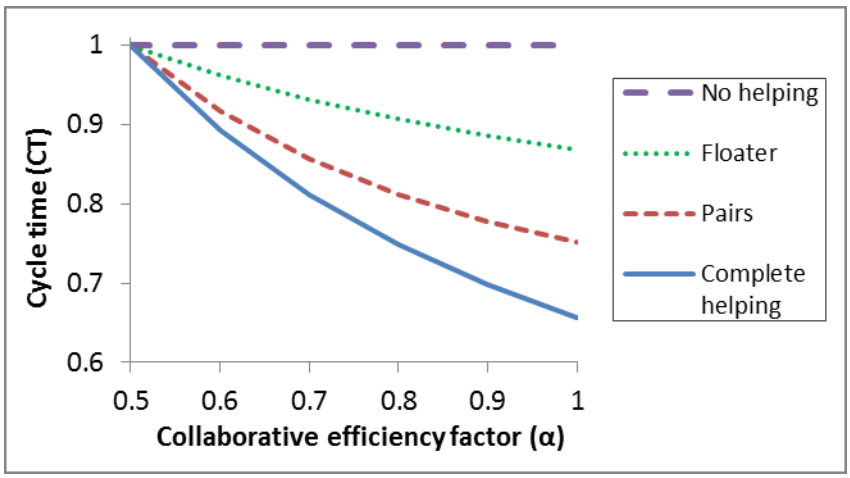
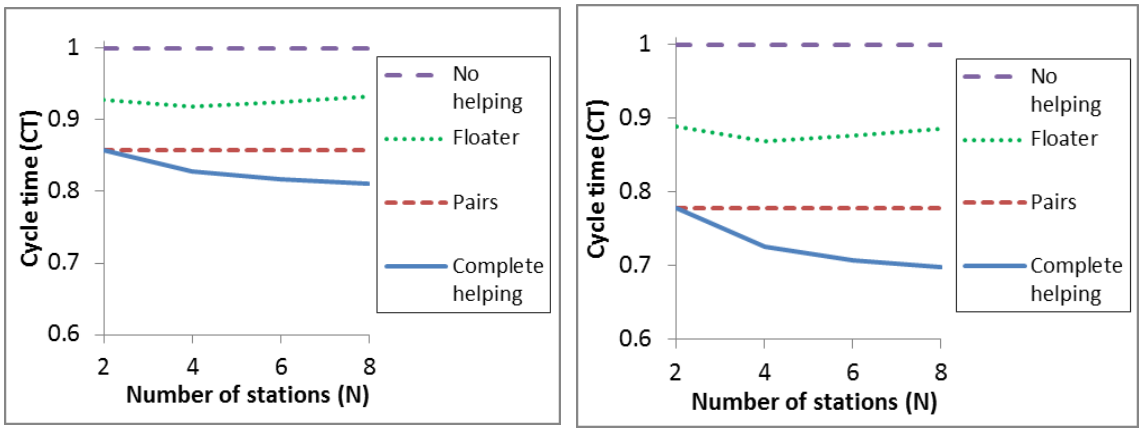


Figure 7. *Given set of jobs*: cycle time as a function of collaborative efficiency factor ($W = N = 8$, $\mu = 1$)

As shown, the *no-helping* CT is 1.000, regardless of the efficiency factor, because there is no collaboration between workers. With major collaborative inefficiency (α = 0.7), the *floater* CT is 0.932, that of *pairs* 0.857 and that for *complete helping* 0.811. With minor inefficiency (α = 0.9) the respective values are 0.886, 0.778 and 0.698. With no inefficiency (α = 1), the CT's are 0.868, 0.750 and 0.656 respectively. In the comparison, *pairs* perform relatively better, with α = 0.7 (*pairs* results in a 5.7% higher CT than *complete helping*) than with α = 0.9 (*pairs* results in an 11.5% higher CT than *complete helping*).

Figure 8 presents the average job cycle time (CT) as a function of the number of stations N with (a) major (α = 0.7) and (b) minor (α = 0.9) inefficiency conditions [Equations (5), (10), (12) and (14)].

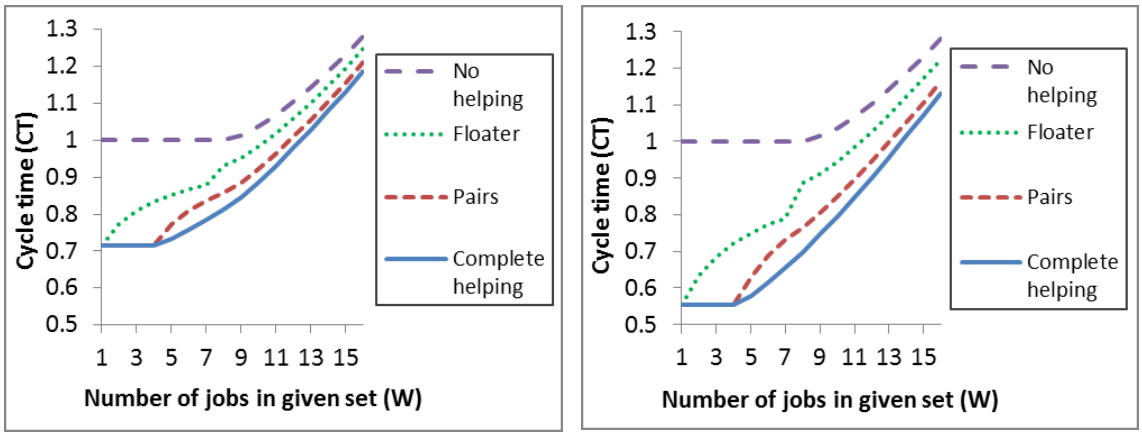


(a) $\alpha = 0.7$ (b) $\alpha = 0.9$

Figure 8. *Given set of jobs*: cycle time as a function of number of stations ($\mu = 1$, $W = N$)

The following observations can be made from the figure. First, the absolute difference in CT between *complete helping* and other policies increases up to $N = 8$ stations. Second, the CT of *pairs* is constant and independent of N. This is due to each pair having the same average processing speed, as discussed in the previous section. Third, *floater* results in its best CT with $N = 4$.

Figure 9 presents the average job cycle time (CT) as a function of the number of jobs in a given set W with (a) major ($\alpha = 0.7$) and (b) minor ($\alpha = 0.9$) collaborative inefficiency conditions [Equations (5), (10), (12) and (14)]. The number of parallel stations $N = 8$.



(a) $\alpha = 0.7$ (b) $\alpha = 0.9$

Figure 9. *Given set of jobs*: cycle time as a function of number of jobs in given set ($N = 8$, $\mu = 1$)

The figure shows that CT in *complete helping* and the *pairs* policy is equal from $W = 4$ down to 1. This is due to the similar use of workers ($S = 0$ and $P = W$) in the initial state in which the maximum possible number of workers is used. Collaborative efficiency conditions do not make a clear difference in relative performance between *pairs* and *complete helping* in this experiment. A rapid increase in CT from $W = 7$ to $W = 8$ with the *floater* policy is due to the fact that with $W = 7$ a floater is initially helping whereas

with $W = 8$ a floater initially has his own job which has to be completed before he can start helping. The cycle time curves of the different policies approach each other as W increases and the influence of balancing effectiveness decreases.

4.3 *Results for continuous arrival of jobs*

In the case of the *continuous arrival of jobs*, the jobs arrive into the system with a Poisson-distributed rate, λ, after which jobs are assigned from a single queue to possible idle workers so that the maximum number of workers is always used within the limits of the worker coordination policy that is in use.

Figure 10 presents the average job cycle time (CT) as a function of the collaborative efficiency factor [Equations (1) and (2)]. In the figure, a collaborative efficiency factor of 0.5 indicates a situation in which helping is completely useless and, on the contrary, an efficiency factor of 1 indicates a situation in which no inefficiency in helping occurs.

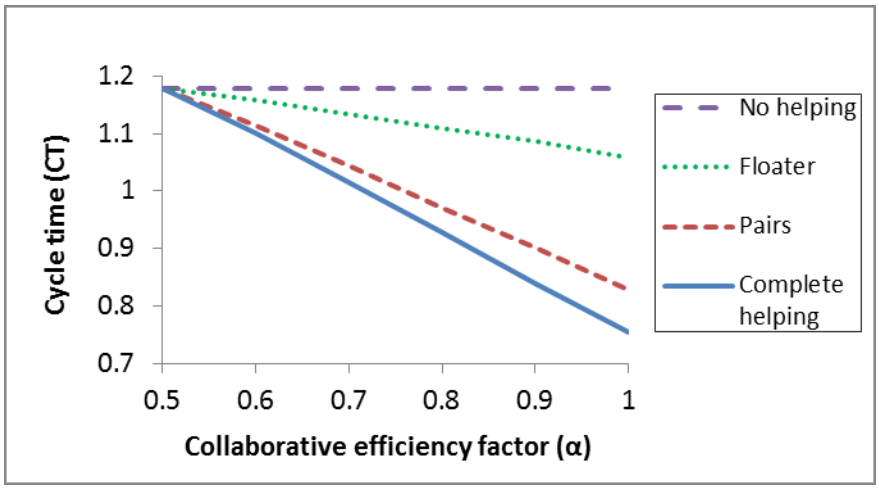
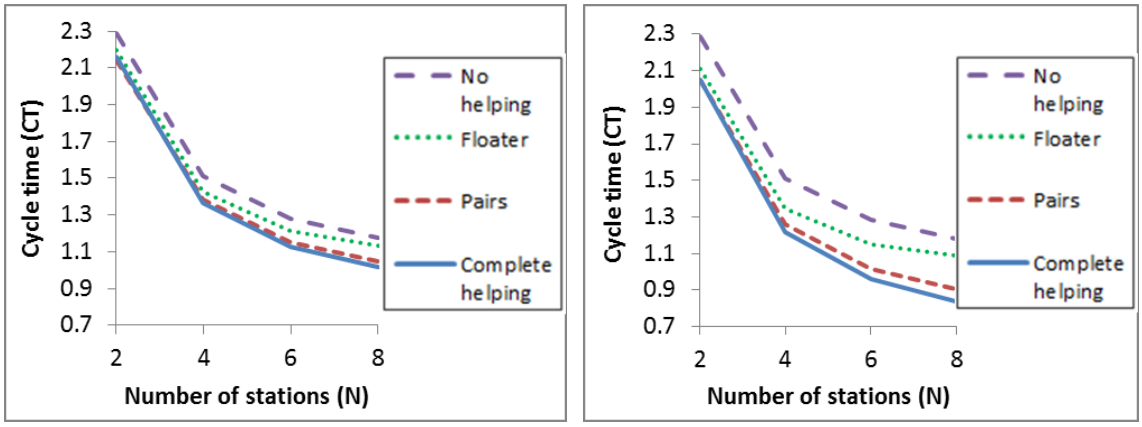


Figure 10. *Continuous arrival of jobs*: cycle time as a function of collaborative efficiency factor ($N = 8$, $\lambda = 6$, $\mu = 1$, $u = 0.75$)

Figure 10 shows that, first, *no helping* results in a constant CT of 1.178. Second, with major ($\alpha = 0.7$), minor ($\alpha = 0.9$) and no ($\alpha = 1$) inefficiency, *floater* results in CT's of 1.134, 1.088 and 1.058. The corresponding values for *pairs* are 1.044, 0.902 and 0.829 and for *complete helping* 1.016, 0.840 and 0.755. The results show that if the collaborative inefficiency is major ($\alpha = 0.7$), the CT with *pairs* is only 2.8% worse than with *complete helping*. With minor inefficiency ($\alpha = 0.9$) the difference is 7.4%.

Figure 11 presents the average job cycle time (CT) as a function of the number of stations N with (a) major ($\alpha = 0.7$) and (b) minor ($\alpha = 0.9$) collaborative inefficiency conditions [Equations (1) and (2)]. In this experiment, the utilisation rate $u = \lambda/(N\mu) = 0.75$. Since the processing rate $\mu = 1$, the arrival rate λ and the system capacity are changed when the number of stations N is changed.

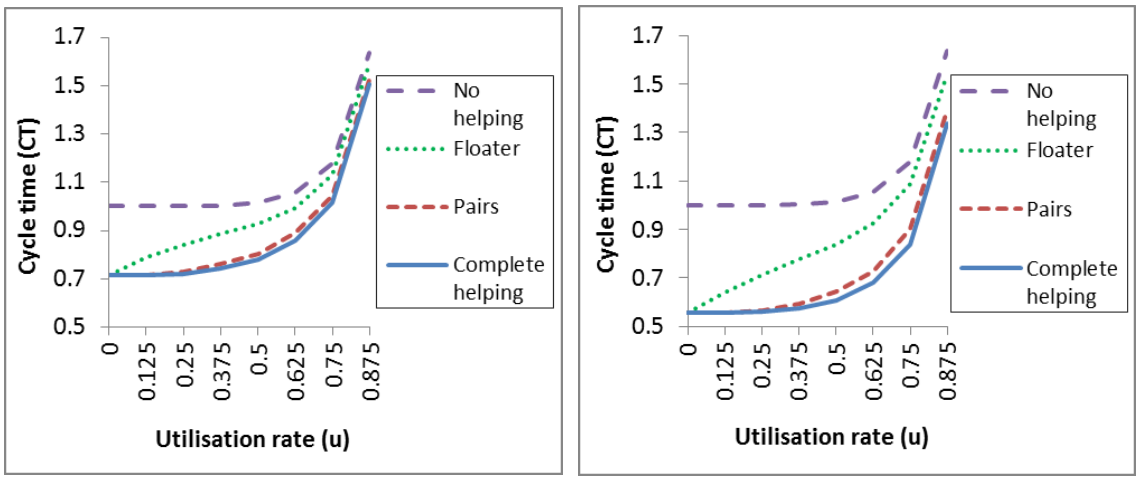


(a) $\alpha = 0.7$ (b) $\alpha = 0.9$

Figure 11. *Continuous arrival of jobs*: cycle time as a function of number of stations ($\mu = 1$, $u = 0.75$)

As the figure shows, *complete helping* performs relatively slightly better than *pairs* when the number of stations increases. However, this absolute superiority decreases when the collaborative inefficiency increases.

Figure 12 presents the average job cycle time (CT) as a function of the system utilisation rate u with (a) major ($\alpha = 0.7$) and (b) minor ($\alpha = 0.9$) collaborative inefficiency conditions [Equations (1), (2) and (3)]. The system utilisation u rate describes the fraction of the time all the workers are busy when no collaboration is taken into account. It is calculated by dividing the job arrival rate by the processing rate of the full capacity of N workers. In this experiment, the number of stations, i.e. workers, is $N = 8$.



(a) $\alpha = 0.7$ (b) $\alpha = 0.9$

Figure 12. *Continuous arrival of jobs*: cycle time as a function of system utilisation rate ($N = 8$, $\mu = 1$)

In the figure, it is shown that while the system utilisation rate decreases from $u = 0.5$, the resulting reduction in CT is only minor, except for *floater*. With low u there are few arrivals and processing can usually be started immediately by the existing idle workers

when a job arrives. By contrast, a higher u means more situations in which worker capacity is fully utilised and the arriving jobs have to be put into the queue, which increases the cycle time respectively.

The relative difference in CT between *complete helping* and *pairs* is the largest when u = 0.625 (*pairs* results in a 3.6% higher CT) in major collaborative inefficiency (α = 0.7) conditions and when u = 0.75 (*pairs* results in a 7.4% higher CT) in minor inefficiency (α = 0.9) conditions.

4.4 *Comparison and discussion between policies and models*

The results show that the *complete helping* policy clearly outperforms the other policies when all three of the following conditions are fulfilled: (1) the number of parallel stations is not minimal ($N \geq 4$) (Figures 8 and 11); (2) helping is useful ($\alpha > 0.5$) (Figures 7 and 10) and (3) the number of jobs in a given set is more than half of the number of stations (Figure 9). The results also clearly show that *pairs* performs the second-best, as well as that *no helping* shows the worst results, followed by *floater* (Figures 6-12).

In addition to the numerical results, the policies can also be analytically compared for all values of n and general $\alpha > 0.5$. For example, for n = 3 and N = 6, it can be simply proved that *no helping* (NH) is worse than *floater* (F) is worse than *pairs* (P) is worse than *complete helping* (CH) for $\alpha > 0.5$, since using Figures 2, 3, 4 and 5:

$$\mu_3(NH) = \mu_{(3,3,0)}(F) < \mu_{(3,2,1)}(F) = \mu_{(3,2,1)}(P) < \mu_{(3,0,3)}(P) = \mu_3(CH)$$

since $3\mu = 3\mu < 2\mu + \alpha 2\mu = 2\mu + \alpha 2\mu < \alpha 6\mu = \alpha 6\mu$.

A similar comparison with equal results can easily be made for other values of $n < N$. However, in the cases of n = 1 and $n = N$-1, the performances of *pairs* (P) are equal to the performances of *complete helping* (CH). This ranking can also be seen in effective processing rates in Figure 6.

On the basis of numerical results (Figures 7 and 10) a more detailed comparison is made between *complete helping* and *pairs*. As stated earlier, a comparison is made in different job release and collaborative inefficiency conditions as well as with such a number of parallel stations (N = 8) as enables a relatively large number of workers to collaborate. In general, the higher the inefficiency, the better the relative cycle time of *pairs* in relation to *complete helping* is, as a result of the larger numbers of pairs working in the latter. When different job release conditions are taken into account, minor collaborative inefficiency conditions (α = 0.9) with a *given set of jobs* result in an 11.5% higher CT with *pairs* in relation to *complete helping* (Figure 7), which can be characterised as a significant difference. In contrast, major inefficiency conditions (α = 0.7) with the *continuous arrival of jobs* result in only a 2.8% higher CT with *pairs* compared to *complete helping* (Figure 10), which can be characterised as a small difference.

In addition to comparing the performance in terms of the average job cycle times, CT's, as was done above, the worker coordination policy that is used should be feasible in terms of the nature of the job release type that is used as well. In the case of a *given set*

of jobs, workers have a collective target of completing the jobs they are given. Therefore, their motivation to help others may be on a high level, which again favours the use of *complete helping*.

In the case of the *continuous arrival of jobs*, the workload varies, which is realised as changes in the idle and busy times of workers. Therefore, the helping principle should be simple so as to ensure clear control of the system, and thus a simple *pairs* policy might be a reasonable alternative to *complete helping*. Using the *pairs* policy is also supported by the fact that with it the most effective pairs can be formed or at least the workers in a pair get used to working together, which could reduce the collaborative inefficiency. As an example of that, assuming that while with *complete helping* inefficiency stays at a major level ($\alpha = 0.7$) as a result of there being a relatively large number of different pair combinations, with *pairs*, collaborative efficiency could improve to a level in which $\alpha = 0.8$, thus resulting in a 4.4% reduction in CT in relation to *complete helping* (CT = 1.016 with *complete helping* and $\alpha = 0.7$; CT = 0.971 with *pairs* and $\alpha = 0.8$). In general, attention should also be paid to the fact that full flexibility among workers (as with *complete helping*) has more complex and ambiguous psychological and social consequences, as reviewed by Slomp and Molleman (2002). Therefore, in practice, with *complete helping* in the long term workers may seek to help a worker with whom they prefer to collaborate and the efficiency of these pairs may improve. This again favours considering a simple *pairs* policy.

5. Conclusions and future research

In this paper, four worker coordination policies utilising moving workers to avoid their idleness and improve efficiency in a parallel station system were studied. The comparison of these policies was performed in order to find out which policy should be used in order to achieve the best performance in different circumstances. The policies were tested with two cases of releasing jobs. With the first case, a given number of jobs have to be completed by the entire system. With the second case, new jobs arrive with a Poisson-distributed rate. In both cases, jobs are first released to a single queue from which they are centrally assigned to a station when there is a need for jobs. The study assumed that the system has as many workers as stations and those workers are identical but that the customisation of products affects the exponentially distributed processing times of the jobs. As the worker coordination policies involve different amounts of worker collaboration, in terms of helping others, the collaborative inefficiency, i.e. loss of productivity, in helping was taken into account in the models.

The paper presented the systematics of worker coordination policies as Markov models. Using the models, mathematical equations were formed to calculate the performances of the policies. The results from the experiments show the performances of the policies in terms of the average job cycle time. To summarise the numerical results in this paper, the following two main conclusions are drawn.

(1) The *complete helping* policy is suggested for use in conditions where there is minor collaborative inefficiency, especially when there is a given set of jobs to be completed.
(2) The *pairs* policy is a reasonable alternative in conditions where there is major collaborative inefficiency, when jobs arrive continuously into the system.

The other two policies that were tested, *floater* and *no helping,* clearly perform worse as a result of their major limitations on worker movement.

In further research, it would be relevant to include the learning effect and improving collaborative efficiency in the Markov models. However, if this is done it requires the addition of the levels of skills gained and the collaborative efficiency of each worker to the mathematical models, which is laborious. Additionally, in order to gain greater certainty of the parameter values for assembly processes, empirical case studies are needed. And finally, the worker coordination policies presented in this paper should be tested in practice in order to see how well they fit in with real assembly systems.

References

Andradottir, S., Ayhan, H., and Down, D. G., 2001. Server assignment policies for maximizing the steady-state throughput of finite queueing systems. *Management Science, 47*(10), 1421-1439.

Bokhorst, J. A. C., and Gaalman, G. J. C., 2009. Cross-training workers in Dual Resource Constrained systems with heterogeneous processing times. *International Journal of Production Research, 47*(22), 6333-6356.

Bokhorst, J. A. C., Slomp, J., and Gaalman, G. J. C., 2004. On the who-rule in Dual Resource Constrained (DRC) manufacturing systems. *International Journal of Production Research, 42*(23), 5049-5074.

Curry, G. L., and Feldman, R. M., 2011. *Manufacturing Systems Modeling and Analysis.* Springer-Verlag.

Easton, F. F., 2011. Cross-training performance in flexible labor scheduling environments. *IIE Transactions*, *43*(8), 589-603.

Graves, S. C., 2008. Flexibility principles. *In*: *Building Intuition*, 33-49. Springer US.

Gunduz, M., 2004. A quantitative approach for evaluation of negative impact of overmanning on electrical and mechanical projects. *Building and Environment, 39*(5), 581-587.

Hanna, A. S., Chang, C. K., Lackney, J. A., and Sullivan, K. T., 2007. Impact of overmanning on mechanical and sheet metal labor productivity. *Journal of Construction Engineering and Management, 133(1),* 22-28.

Hillier, F. S., and Lieberman, G. J., 1995. Introduction to operations research. McGraw-Hill.

Hopp, W. J., and Oyen, M. P., 2004. Agile workforce evaluation: A framework for cross-training and coordination. *IIE Transactions, 36*(10), 919-940.

Hopp, W. J., Tekin, E., and Van Oyen, M. P., 2004. Benefits of skill chaining in serial production lines with cross-trained workers. *Management Science, 50*(1), 83-98.

Hytönen, J., Niemi, E., and Pérez, S., 2010. The effect of worker skill distribution and overmanning on moving worker assembly lines. *In: Proceedings of International Conference on Competitive Manufacturing,* 3-5 February 2010, Stellenbosch, South Africa, 171-176.

Inman, R. R., Jordan, W. C., and Blumenfeld, D. E., 2004. Chained cross-training of assembly line workers. *International Journal of Production Research, 42*(10), 1899-1910.

Jordan, W. C., and Graves, S. C., 1995. Principles on the Benefits of Manufacturing Process Flexibility. *Management Science, 41*(4), 577-594.

Little, J. D., 1961. A Proof for the Queuing Formula: $L = \lambda W$. *Operations Research, 9*(3), 383-387.

Nembhard, D. A., and Bentefouet, F., 2012. Parallel system scheduling with general worker learning and forgetting. *International Journal of Production Economics, 139*(2), 533-542.

Nembhard, D. A., and Bentefouet, F., 2014. Selection policies for a multifunctional workforce. *International Journal of Production Research,* (ahead-of-print), 1-18.

Peltokorpi, J., Tokola, H., and Niemi, E., 2012. Comparison of Balancing Policies in Multi-Item Assembly. *In: International Conference on Flexible Automation and Intelligent Manufacturing,* 10-13 June 2012, Helsinki, Finland.

Saadat, M., Tan, M. C., Owliya, M., and Jules, G., 2013. Challenges and trends in the allocation of the workforce in manufacturing shop floors. *International Journal of Production Research, 51*(4), 1024-1036.

Sengupta, K., and Jacobs, F. R., 2004. Impact of work teams: a comparison study of assembly cells and assembly line for a variety of operating environments. *International Journal of Production Research, 42*(19), 4173-4193.

Sennott, L., Van Oyen, M., and Iravani, S., 2006. Optimal dynamic assignment of a flexible worker on an open production line with specialists. *European Journal of Operational Research, 170*(2), 541-566.

Slomp, J., and Molleman, E., 2002. Cross-training policies and team performance. *International Journal of Production Research, 40*(5), 1193-1219.

Yang, K. K., 2007. A comparison of cross-training policies in different job shops. *International Journal of Production Research, 45*(6), 1279-1295.