Applied Intelligence 13, 247–258, 2000

A Coevolutionary Algorithm for Balancing and Sequencing in Mixed Model Assembly Lines

YEO KEUN KIM AND JAE YUN KIM
Department of Industrial Engineering, Chonnam National University, 300 yongbongdong, Pukku, Kwangju 500-757, Republic of Korea
kimyk@chonnam.chonnam.ac.kr
kimjy@chonnam.chonnam.ac.kr

YEONGHO KIM
Department of Industrial Engineering, Seoul National University, Seoul 151-742, Republic of Korea
ykim@cybernet.snu.ac.kr

Abstract. A mixed model assembly line is a production line where a variety of product models are produced. Line balancing and model sequencing problems are important for an efficient use of such lines. Although the two problems are tightly interrelated with each other, prior researches have considered them separately or sequentially. This paper presents a new method using a coevolutionary algorithm that can solve the two problems at the same time. In the algorithm, it is important to promote population diversity and search efficiency. We adopt a localized interaction within and between populations, and develop methods of selecting symbiotic partners and evaluating fitness. Efficient genetic representations and operator schemes are also provided. When designing the schemes, we take into account the features specific to the problems. Also presented are the experimental results that demonstrate the proposed algorithm is superior to existing approaches.

Keywords: coevolutionary algorithm, genetic representation and operators, line balancing, model sequencing, mixed model assembly lines

1. Introduction

A mixed model assembly line (MMAL) is a production line where a variety of product models that have similar characteristics are assembled. The products produced in MMALs usually have differences in the amount of production, work contents, and assembly time depending on the models. For an efficient use of such a line, it is widely conceived that line balancing and model sequencing are important [1]. Line balancing is a problem of assigning various assembly tasks to workstations on the line, whereas model sequencing is a problem of determining a production sequence of models. We propose a new method that can solve the two problems simultaneously.

The above two problems are very tightly interwoven with each other. The optimality of model sequence depends on the results of task assignment, which is affected by the model sequence in the other way. However, in almost all the existing researches on MMALs, the problems have been considered separately. Note that both problems are known as an NP-hard class of combinatorial optimization problems [2, 3]. Several researchers have studied MMAL balancing problems: Thomopoulos [4], Macaskill [5], Chakravarty and Shtub [6], and so on. These previous researches however have ignored the model sequence. For MMAL sequencing problems, after the first investigation by Kilbridge and Wester [7], a large number of researches have been carried out including Okamura

and Yamashina [8], Yano and Rachamadugu [9], Bard et al. [10], and Tsai [3]. These authors have considered several different objectives in MMAL sequencing problems, such as minimizing the total utility work, the overall line length, and keeping a constant rate of part usage. However, they assume that the line balance is predetermined.

As far as we know, Thomopoulos [1] and Dar-El and Nadivi [11] have studied the contemporary consideration of balancing and sequencing problems in MMALs. They have proposed a hierarchical approach to the aggregated problem. The problems are treated sequentially in such a manner that one of them is solved first, and then the other is considered under the constraint of the first solution. As is well known, such a hierarchical approach to an aggregated problem combining multiple sub-problems has a limitation to exploring the solution space.

Artificial coevolution offers a promising alternative to solving difficult and dynamic problems [12]. Our new methodology adopts a coevolutionary algorithm. The algorithm is a search technique simulating the evolutionary process in nature where one species has influences on and at the same time is influenced by other species [13]. While a species is evolving, it is interacting with and adapting to other species. We conceive that this is similar to the situation where several problems are related with each other in such a manner that a change to the solution of one problem can have an effect on those of other problems. We propose a new coevolutionary algorithm that can simultaneously deal with both balancing and sequencing problems in MMAL. This approach, as contrasted with the hierarchical one, is characterized by its ability to effectively search the solution space formed by the two problems at the same time.

2. Balancing and Sequencing in MMAL

2.1. Mixed Model Assembly Line

It is assumed that the MMALs considered in this paper operate under the following conditions. First, the conveyor moves at a constant speed (v_c), and each product is launched into the conveyor at a fixed rate (γ). Second, it is assumed that the workstations on the line are all closed types, and an early start schedule is used. For an early start schedule a workstation has a reference point, and the worker working in the workstation is not allowed to work beyond the reference point. Third, tasks that are not completed within a workstation are handed over to utility workers. Fourth, the line takes the strategy of cyclic production. Due to the space limitation, only the cyclic production with some notational conventions is further described. Readers may refer to Bard et al. [10] for other aspects of the operational decisions. Let D_m ($m = 1, 2, \ldots, M$) be the demand of model m during the planning horizon, and h be the greatest common divisor of D_1, D_2, $\ldots$, D_M. Then, the vector, $\boldsymbol{d} = (d_1, \ldots, d_M)$, where $d_m = D_m/h$, is the model mix, which is called minimum part set (MPS) [10]. All the sequences that appear in this paper are based on MPS. Obviously, h times repetition of the MPS sequence meets the total demand.

2.2. The Objective

Various objectives can be used for balancing and sequencing problems in MMAL. In this research, both of the balancing and sequencing problems aim at minimizing utility work. Utility work is defined as the amount of work that is not completed within the given length of workstation. The utility work is typically handled through the use of utility workers who assist the regular workers during work overload. Minimizing the utility work contributes to reducing not only labor cost but the risk of stopping the conveyor and the required line length [9]. For a given task assignment and model sequence, the total amount of utility work during one MPS cycle can be computed as Eq. (1). Worker's moving time is ignored in this formulation.

$$U = \sum_{j=1}^{J} \left\{ \sum_{s=1}^{S} \max\left[0, \left(Z_{sj} + v_c T_{jm_s} - L_j\right)/v_c\right] + Z_{(S+1)j}/v_c \right\}, \quad (1)$$

where J is the number of workstations, S ($= \sum_{m=1}^{m} d_m$) is the number of products that are produced in one MPS cycle, and m_s represents the model of the s-th product in the production sequence. Z_{sj} is the starting position of the assembly work on the s-th product at the j-th workstation, T_{jm_s} is the amount of assembly time required for model m_s at the j-th workstation, and L_j is the line length of the j-th workstation. Since $v_c T_{jm_s}$, is the line length required to assemble the product, $Z_{sj} + v_c T_{jm_s}$ is the ending position of the assembly work. If this is larger than L_j, utility work is incurred. Otherwise, no utility work is required. Therefore, the

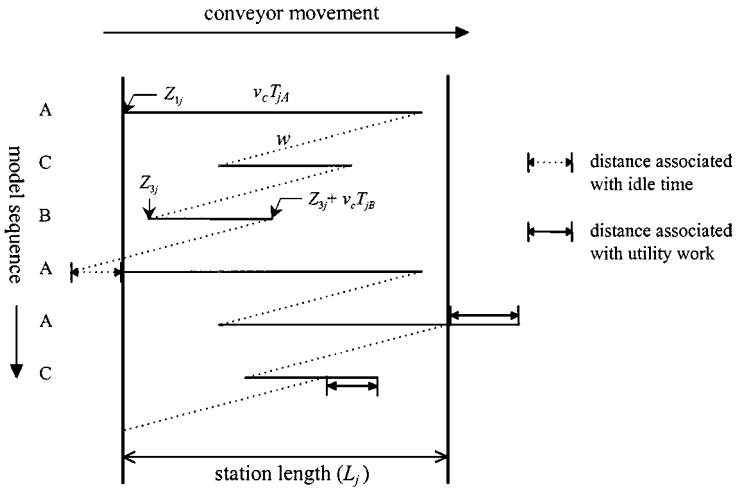


Figure 1. Operation of closed workstation.

first whole term represents the total amount of utility work required for one MPS cycle at the j-th workstation. Now, consider the second term. We assume that the first product of every cycle must begin at the reference point, that is, $Z_{1j} = 0, \forall j$. In general, however, this cannot be satisfied. The second term is introduced to take into account this consideration. With an early start schedule, the starting position of assembly work at the j-th workstation can be computed as follows.

$$Z_{(s+1)j} = \max\left[0, \min\left(Z_{sj} + v_c T_{jm_s} - w, L_j - w\right)\right], \quad s = 1, 2, \ldots, S \tag{2}$$

where w $(=\gamma v_c)$ is the distance of conveyor movement during the launch interval.

For example, consider a closed workstation following an early start schedule shown in Fig. 1. Suppose tasks that should be performed in the workstation are already determined, and the models are sequenced as (A C B A A C). On completion of work on a product, the worker moves to the opposite direction of the conveyor movement. A dotted line indicates the distance between two consecutive products. Since the launch rate is fixed, all the dotted lines are parallel. If the succeeding product is within the workstation boundary, the worker can begin to work on it. Otherwise, the worker has to wait until the product arrives at the reference location. Notice that in the example, utility work is required for the fifth and sixth models.

3. A Coevolutionary Algorithm for Balancing and Sequencing in MMAL

3.1. Proposed Coevolutionary Algorithm

Coevolutionary algorithms are search algorithms that imitate the biological coevolution that is a series of reciprocal changes in two or more interacting species. Although there are many variants of coevolutionary algorithms, they are typically classified into two main forms: cooperative coevolutionary algorithm and competitive coevolutionary algorithm, which respectively imitate symbiosis and parasitism. In cooperative algorithms [12, 14, 15], individuals of several different species are combined before they are evaluated with respect to an evolutionary target. The algorithms aim to measure how good an individual from a population can adapt the expectations of individuals from another population. In competitive algorithms [16, 17], candidate solutions compete with each other in game-like tournaments. The purpose is to measure how good an individual strategy can stand when played against various strategies by the opponent.

In this paper a cooperative coevolutionary algorithm, also called a symbiotic evolutionary algorithm, is used as the underlying framework. A common hypothesis for coevolutionary algorithms is that several parallel searches for different pieces of the solution are more efficient than single search for the entire solution. Note that in the problem considered here, the amount of

utility work can be assessed when information on both problems is given. Each of the balancing and sequencing problems is considered as one species, and is characterized by genes which are adequately designed to describe the features specific to the problem.

The coevolution in the proposed algorithm is based on the localized interactions which we expect can promote the diversity of population and search efficiency. We will show this effect later in the experiment. The species coevolves while interacting within and between populations as follows. Each of the two populations, that is, balancing population (Pop-B) and sequencing population (Pop-S), forms a two-dimensional structure of toroidal grids. A structure of 3×3 neighborhood is used here for the localized coevolution. NB_{ij} and NS_{ij} denote the neighborhood including individual (i, j) and its eight neighbors in Pop-B and Pop-S, respectively. The size and shape of neighbor sets can of course be defined in many different ways. While one population evolves, it interacts with the other population. One of the two species calls the other a symbiotic species. Figure 2 shows the general idea of the proposed coevolutionary algorithm. The overall procedure of the algorithm is presented below, and the crucial operations are further detailed later in this paper.

Step 1 (Initialization). Generate two sets of initial populations, Pop-B and Pop-S, which contain individuals representing balancing and sequencing, respectively.

Step 2 (Fitness evaluation). Initial fitness is evaluated for every pair of matching individuals (which are those two located at the same position in Pop-B and Pop-S), and set f_{best} to the best fitness value.

Step 3 (Neighborhood setting). Select an arbitrary location (i, j), and set up the evolving neighborhood $\mathrm{EN} = \mathrm{NB}_{ij}$, and the symbiotic neighborhood $\mathrm{SN} = \mathrm{NS}_{ij}$. Set $F = 0$.

Step 4 (Coevolution).

Step 4.1: Two parents are selected from EN based on fitness, and two offspring are created by applying a crossover operation.

Step 4.2: Two individuals which have a low fitness in EN are replaced with the offspring newly created in Step 4.1.

Step 4.3: A mutation operation, if turned on, is applied to the individuals in EN.

Step 4.4: The fitness is evaluated for the individuals newly produced, where the individual with the best fitness in SN is selected as the symbiotic partner. If the fitness of the best individual is higher than f_{best}, then f_{best} is updated with the new best fitness.

Step 4.5: If the termination criteria for EN are met, then go to Step 4.6. Otherwise, go to Step 4.1.

Step 4.6: If $F = 0$, then $F = 1$, $\mathrm{EN} = \mathrm{NS}_{ij}$, $\mathrm{SN} = \mathrm{NB}_{ij}$, and go to Step 4.1. Otherwise, go to Step 5.

Step 5 (Termination criteria). If the termination criteria are satisfied, then stop. Otherwise, go to Step 3.

Step 4 unfolds the evolution process for NB_{ij} and NS_{ij}. An arbitrary location (i, j) is selected, where the localized evolution process is centered. Then, the evolution proceeds within NB_{ij}, wherein NS_{ij} becomes the symbiotic partners of NB_{ij}. After repeating the evolution until some termination criteria are satisfied, the

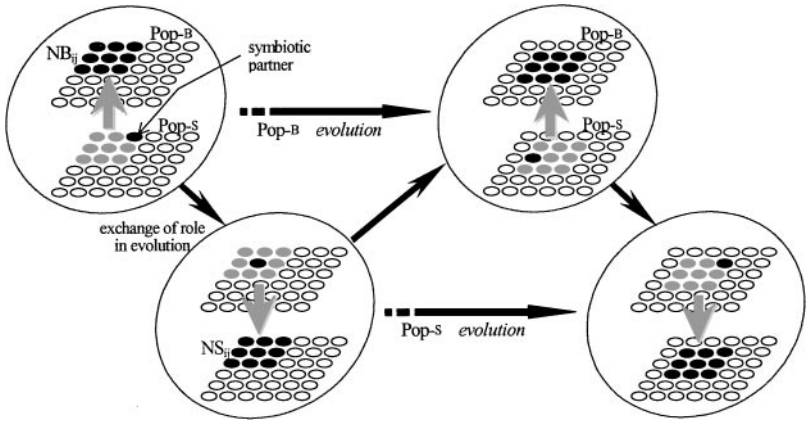


Figure 2. Proposed coevolutionary algorithm.

evolving neighborhood and the symbiotic neighborhood switch their roles. The evolution of Pop-B and Pop-S in Step 4 alternates, while varying the neighborhood.

There are several factors that influence the performance of coevolutionary algorithms. This includes symbiotic partners and fitness evaluation, representation of solutions, genetic operation, and other genetic parameters. Among these, methods of selecting symbiotic partners and evaluating fitness are mostly distinguished from traditional genetic algorithms (GAs), and thus are explained in more detail in the next section. Since Pop-B and Pop-S are concerned with different problems having quite distinct features, we need a method that can handle those features effectively. The genetic representation and operations for Pop-B and Pop-S are respectively addressed in Sections 4 and 5. Other genetic parameters including the sizes of populations, the rates of mutation, and termination criteria are usually determined through a number of training experiments. These are mentioned in Section 6 with experimental results.

3.2. *Selection of Symbiotic Partners and Fitness Evaluation*

Maher and Poon [18] proposed two methods for selecting symbiotic partners. Consider two populations, Pop-1 and Pop-2. One method is that, for Pop-1(i, j), the (i, j)-th individual of Pop-1, its symbiotic partner is fixed to Pop-2(i, j), the (i, j)-th individual of Pop-2. Note that in this method each individual has a different symbiotic partner, and the position of symbiont does not change over evolutionary time. This model is called tightly-coupled coevolution, and can be viewed as a model having a single population where one individual has two chromosomes. The other method is that the symbiotic partner of Pop-1(i, j) is the individual that gives the highest fitness in Pop-2. Therefore, all the individuals share the same symbiotic partner and the symbiont does change for every generation. This model is called loosely-coupled coevolution. In this research, a variant of this second model is used in Step 4.4. A symbiont is selected from its neighbors, not from all the individuals of a population.

In Steps 4.1 and 4.2, the selection of parents and individuals for replacement is based on fitness. A roulette wheel method is used for the selection. For the (s, t)-th individual of NB_{ij}, its fitness, $f_{\mathrm{B}}(s, t)$, is evaluated using Eq. (3). The fitness has a value in the range of (0, 1].

$$f_{\mathrm{B}}(s,t) = \frac{u_{\mathrm{SP}}(s,t) - \{\max_{(p,q)\in \mathrm{NB}_{ij}} u_{\mathrm{SP}}(p,q) + 1\}}{\min_{(p,q)\in \mathrm{NB}_{ij}} u_{\mathrm{SP}}(p,q) - \{\max_{(p,q)\in \mathrm{NB}_{ij}} u_{\mathrm{SP}}(p,q) + 1\}}, \tag{3}$$

where $u_{\mathrm{SP}}(s, t)$ is the amount of utility work computed by Eq. (1) when individual Pop-B(s, t) is associated with its symbiotic partner SP in NS_{ij}. The equation rescales the objective function value of Eq. (1) so that it can make the selection effective and deal with the minimization problem. The fitness of an individual in NS_{ij}, $f_{\mathrm{S}}(s, t)$, can be similarly computed by switching the roles of NB_{ij} and NS_{ij}.

Equation (4) determines the probability that the individual at position (s, t) in NB_{ij} is replaced with offspring. The equation gives a higher probability when the individual requires more utility work. The probability for individuals in NS_{ij} is similarly calculated.

$$P_{\mathrm{del}}(s,t) = \frac{f'_{\mathrm{B}}(s,t)}{\sum_{(m,n)\in \mathrm{NB}_{ij}} f'_{\mathrm{B}}(m,n)}, \tag{4}$$

where

$$f'_{\mathrm{B}}(s,t) = \frac{\{1 + u_{\mathrm{SP}}(s,t)\} - \min_{(p,q)\in \mathrm{NB}_{ij}} u_{\mathrm{SP}}(p,q)}{\{1 + \max_{(p,q)\in \mathrm{NB}_{ij}} u_{\mathrm{SP}}(p,q)\} - \min_{(p,q)\in \mathrm{NB}_{ij}} u_{\mathrm{SP}}(p,q)}.$$

4. Genetic Representation and Operation for MMAL Balancing

4.1. *Genetic Representation*

It is desirable that genetic representation of potential solutions be natural and contain minimal alphabets. It is also needed that with the representation important information is easily extracted by genetic operators and transferred to offspring. In this paper a *group-number* representation scheme is used for the balancing problem. An individual is a string of length I (the number of tasks), each element of which is an integer between 1 and J (the number of workstations). If the i-th element is j, it represents that task i is assigned to workstation j. For example, consider the assignment that three sets of

tasks, {1, 2, 3}, {4, 6, 7}, and {5, 8, 9, 10} are assigned to workstations 1, 2, and 3, respectively. The assignment is encoded as (1 1 1 2 3 2 2 3 3 3). The advantage of this representation is that decoding is straightforward and it is easy to take advantage of the information on task assignment.

4.2. Genetic Operators

In this research, a structural crossover operator [19] is adapted. The original operator is modified to efficiently pull out the characteristics of balancing problems and transmit it to offspring. Also considered is maintaining the feasibility of the reproduced individuals. The procedure of crossover operation is described as follows.

Step 1: Set r to an integer which is randomly generated from [1, J].
Step 2: The genes representing workstations 1 to r in parent P1 are passed on to the same positions in offspring O1.
Step 3: The remaining positions in O1 are copied from those of $r+1$ to J of parent P2.
Step 4: To assign the empty positions in offspring O1, use the *alteration* procedure explained below in this section.

Another offspring, O2, is produced by reversing the roles of parents P1 and P2.

In evolutionary algorithms, mutation is also important in that it increases the exploration capability of the algorithms, which means it enables the algorithms to search various solutions in the search space. For the balancing problem, the mutation is applied to each gene of individuals with a certain rate of mutation. Actual mutation is performed by invoking the alteration procedure below.

The above crossover and mutation operations cause some genes to be empty, which means the corresponding tasks are not assigned to any workstation. A workstation must be newly determined for every empty gene. This is accomplished by the alteration procedure. In this procedure, a care is exercised in balancing workload as well as satisfying precedence relations. To begin with the procedure, we first describe the notation used as follows.

t_{im} the assembly time of task i of model m ($i = 1, 2, \ldots, I$, $m = 1, 2, \ldots, M$).
$\hat{t}_i$ the MPS assembly time of the i-th task ($= \sum_{m=1}^{M} d_m t_{im}$).
$\bar{T}$ the average of workstation workloads for one MPS cycle ($= (1/J) \sum_{i=1}^{I} \hat{t}_i$).
R the set of tasks that are not assigned yet or need to be reassigned.
$IP(i)$ the set of tasks immediately preceding task i.
$S(i)$ the set of tasks following task i.
$NR(i)$ $= R \cap IP(i)$ (i.e., the set of tasks immediately preceding task i among the elements of R).
E_i the latest workstation among those which any task in *IP(i)* is assigned to. If $IP(i) = \emptyset$, then $E_i = 1$.
L_i the earliest workstation among those which any task in $S(i) - R$ is assigned to. If $S(i) - R = \emptyset$, then L_i is set to the last workstation.
$AW(i)$ $= \{E_i, E_i + 1, \ldots, L_i\}$ (i.e., the set of workstations to which task i can be assigned).
T'_j the sum of MPS assembly times of tasks already assigned to workstation j.

Step 1: Construct the sets R and $NR(i)$, and compute T'_j for $j = 1, 2, \ldots, J$.
Step 2: Select the task that has the largest MPS assembly time among such tasks that $i \in R$ and $NR(i) = \emptyset$. Let i^* be the selected task.
Step 3: Compute E_{i^*} and L_{i^*}, and identify $AW(i^*)$.
Step 4: If there exist workstations satisfying $T'_j + \hat{t}_{i^*} \leq \bar{T}$ and $j \in AW(i^*)$, then j^* is set to the workstation having the smallest workstation number (i.e., the left most). Otherwise, j^* is set to the workstation having the smallest T'_j. A tie here is broken by selecting the smallest workstation number.
Step 5: Assign task i^* to workstation j^*, and remove i^* from R. If R is empty, then stop; Otherwise, update $NR(i)$ and T'_j, $j = 1, 2, \ldots, J$, and go to Step 2.

The basic idea of the above procedure comes from the best-fit-decreasing (BFD) heuristic rule [20]. It is known that the BFD rule, a type of greedy heuristic, provides good results for group partitioning problems. The rule is modified suitable for balancing problem, so that it can not only balance the workloads, but reduce the utility work. The alteration procedure has a tendency to make an even distribution of the overall

workload among workers. Even assignment of tasks tends to decrease the amount of utility work required for the tasks.

5. Genetic Representation and Operation for MMAL Sequencing

5.1. Genetic Representation

An individual in Pop-S is represented by a sequential list that simply enumerates all the models following the launch sequence for an MPS cycle. Consider for example that the MPS, $\boldsymbol{d} = (d_A, d_B, d_C) = (2, 3, 4)$, which means each of the models should be produced d_m, $m =$ A, B, C, times during the cycle. Now, the launch sequence of A, C, B, C, B, A, C, C, B is simply represented by (A C B C B A C C B). This is very natural in that the information on model sequence can be exactly preserved in every position of gene.

5.2. Genetic Operators

In this research, immediate successor relation crossover (ISRX) operator and inversion operator are used. For MMAL sequencing problems, it has been shown experimentally that a combined use of these two is superior to other possible operators [21].

ISRX is a crossover operator that uses the information of immediate successor relations in parents. An immediate successor of a gene A is the gene immediately following A. The immediate successor of the last gene is the first one, by definition. This relation can be summarized in a tabular form as shown in Table 1, which maintains all the immediate successors for every gene type and the ratio of the immediate successors, called gene ratio. The overall procedure of creating offspring with the ISRX operator is as follows.

Step 1: Construct the initial ISRX table, set $i = 1$, and select an initial gene type (e_i) randomly.

Step 2: Inherit e_i at the i-th position in the offspring. If a complete offspring is built, then stop; otherwise, go to the next step.

Step 3: Update the immediate successors by removing two randomly chosen elements among those having the gene type of e_i, and also update the gene ratio.

Step 4: Increase i by 1, and set e_i to the gene type such that the number of elements having the type is the largest in the row of e_{i-1}. Ties are broken by selecting the one with the smallest gene ratio. When this also ties, one gene type is randomly selected. Return to Step 2.

For example, consider two parent strings: (A A B B B C C C C) and (A B C A B C B C C). The ISRX operator begins with setting up the initial ISRX table, as shown in Table 1 (a). The first four iterations of the procedure are provided in Table 1. Repeating the procedure can produce an offspring of (B C A B C C B C A). The other offspring can be obtained by starting with a different initial gene type. A more detailed explanation can be found in [21].

Inversion is a mutation operator. The inversion operator first chooses two random cut points in a parent. The elements between the cut points are then reversed. An example of the inversion operation is presented in Fig. 3. P and O denote parent and offspring, respectively.

Figure 3. Inversion operator.

Table 1. ISRX table.

	(a)		(b)		(c)		(d)	
Gene type	Immediate successor	Gene ratio	Immediate successor	Gene ratio	Immediate successor	Gene ratio	Immediate successor	Gene ratio
A	ABBB	2	ABB	2	ABB	2	ABB	1
B	BBCCCC	3	BCCCC	2	BCCC	2	BCCC	2
C	CCCAABCA	4	CCCAABCA	4	CCAABCA	3	CCBCA	3
Selected gene type	B (randomly selected)		C		A		B	

6. Experimental Design and Results

6.1. Test-bed Problems and Parameter Setting

The performance of the proposed coevolutionary algorithm is analyzed through a set of computer experiments. The experiments are carried out with the 19-task problem formulated by Thomopoulos [4] and the 111-task problem by Arcus [22, Appendix X-3]. Another problem composed of 61 tasks is a practical one obtained from an automobile company while the authors are carrying out an industry project with the company. With these three base problems, the test-bed problems are constructed with several different numbers of workstations and MPS as shown in Table 2. For Arcus's problem, the assembly time of task 95 is changed from 334.91 to 66.15. This allows us to use a larger number of workstations.

The performance of the proposed coevolutionary algorithm (PCoA) is compared with that of the hierarchical approach, the tightly-coupled coevolutionary algorithm (TCoA), and loosely-coupled coevolutionary algorithm (LCoA).

Hierarchical approaches have been widely used to solve aggregated problems which combine several sub-problems that are inter-linked. In general, for MMAL solving a balancing problem is followed by solving a sequencing problem. For the balancing problem, it is tried to minimize the deviation of workload. The solution to the balancing problem obtained becomes a parameter for the sequencing problem. The objective of the second problem is to minimize the total utility work. Since GA has proven to be powerful in solving each problem of line balancing [23] and model sequencing [21], the GAs explained in Sections 4 and 5 are employed, respectively.

The schemes of TCoA and LCoA are designed as follows. The fitness of population is assessed as described in Section 3.2. Based on the fitness values, 50% of population are selected for crossover operation, and each of the resulting offspring is replaced for an individual with a low fitness value. All the selection is carried out probabilistically. And then, some of the individuals are mutated. On completion of evolving one population, the other population is evolved. This is repeated until a termination criterion is satisfied. More detailed procedures for TCoA and LCoA can be found in Maher and Poon [18].

The algorithms are coded in C++, and implemented on an IBM-PC with 166 MHz Pentium CPU. For all

Table 2. Test-bed problems.

Problems	Number of tasks	Number of models	Number of stations	MPS
Thom1	19	3	3	1 1 1
Thom2	19	3	3	3 2 1
Kim1	61	4	6	1 1 1 1
Kim2	61	4	6	1 3 4 5
Kim3	61	4	6	6 4 2 1
Kim4	61	4	12	1 1 1 1
Kim5	61	4	12	1 3 4 5
Kim6	61	4	12	6 4 2 1
Arcus1	111	5	12	1 1 1 1 1
Arcus2	111	5	12	5 3 2 1 1
Arcus3	111	5	12	1 2 4 5 8
Arcus4	111	5	12	1 4 8 3 1
Arcus5	111	5	15	1 1 1 1 1
Arcus6	111	5	15	5 3 2 1 1
Arcus7	111	5	15	1 2 4 5 8
Arcus8	111	5	15	1 4 8 3 1
Arcus9	111	5	27	1 1 1 1 1
Arcus10	111	5	27	5 3 2 1 1
Arcus11	111	5	27	1 2 4 5 8
Arcus12	111	5	27	1 4 8 3 1

the problems, the line length is set to 1.5 times the distance of conveyor movement during launch interval. For convenience, the speed of conveyor is fixed to 1.

Extensive preliminary experiments have been performed to fine-tune the other parameters. First, a 10×10 grid structure is established for every problem, and thus the size of population is 100. Second, the evolution of each neighborhood in Step 4 is repeated two or three times with the same probability. Third, a mutation operation is applied to each gene for the balancing problem, whereas it is carried out on each individual for the sequencing problem; The rates used are 0.05 and 0.1, respectively. Finally, the total number of reproduced individuals in both populations is used for the termination of algorithm. When this number comes up to 5,000, 15,000, and 30,000, the algorithm is terminated for Thomopoulos's, Kim's, and Arcus's problem, respectively. Since the size of solution space depends on the number of tasks and models, the criterion is differently set for each base problem. All the parameter values and genetic operators used in TCoA and LCoA are the same with those in PCoA.

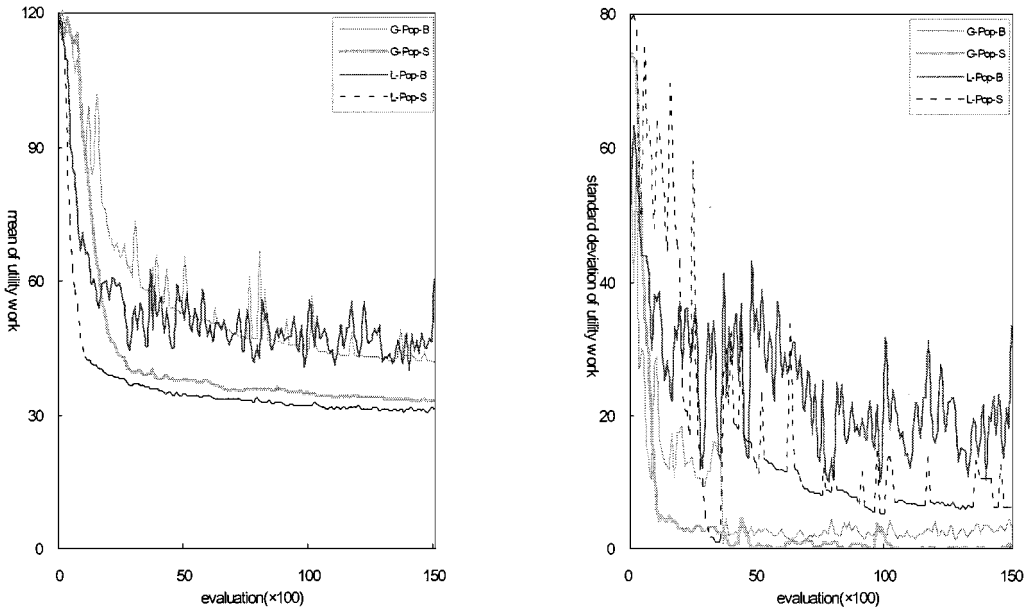
6.2. Population Diversity and Niching

The first experiment has been performed to examine how the proposed algorithm maintains the diversity of populations. The diversity is necessary for the long-term success of evolutionary systems. This is due to that genetic diversity helps a population adapt quickly to changes in the environment, and it allows the population to continue searching for productive niches, thus avoiding becoming trapped at local optima [24, 25]. It is expected that the localized evolution proposed in this paper leads to an efficient search of the solution space.

When a non-binary representation, like those in this paper, is used, it is generally difficult to measure the diversity directly from the genotype representation. Therefore, in this research, the diversity of population is measured using the phenotype representation, that is, the amount of utility work. Figure 4 shows the mean and the standard deviation of utility work for the problem Kim 6. In the legend of the figures, G and L stand for global evolution and localized evolution, respectively. For the global evolution, LCoA is used.

Figure 4(a), the graph of the mean, reveals that both evolution strategies drive the populations to converge. One observation we should notice here is that L-Pop-B and L-Pop-S are lower than G-Pop-B and G-Pop-S. Since we deal with a minimization problem, this indicates that our localized evolution provides higher quality solutions than the other.

An interesting observation is made in Fig. 4(b). The graph demonstrates that the localized evolution can



(a) Mean of utility work

(b) Standard deviation of utility work

Figure 4. Mean and standard deviation of phenotype.

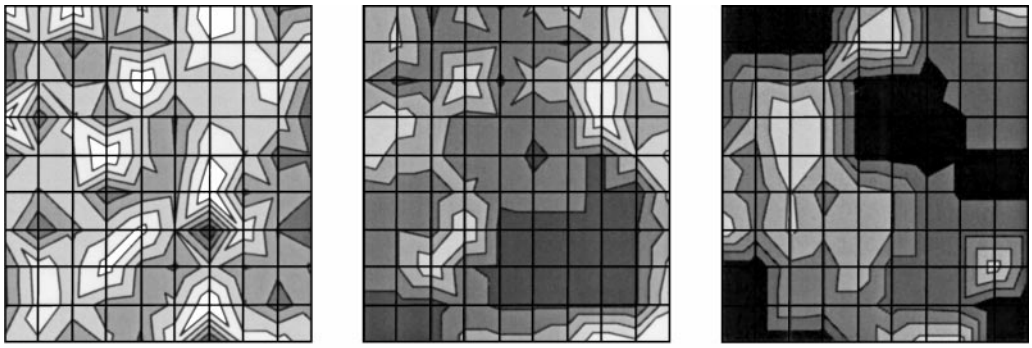


Figure 5. Contour map of the phenotype distribution in the grid.

maintain a larger standard deviation of individuals. Although the standard deviation drops drastically around 3,500 evaluations (i.e., reproduced individuals), a certain amount (28%–43% of average utility work) of diversity is kept up with the localized evolution. On the other hand, the global evolution strategy shows little phenotypical diversity after 4,000 evaluations. This assures that in maintaining the diversity of population the proposed algorithm is better than the global evolution.

Another experiment has been carried out to investigate how the proposed algorithm forms proper niches while maintaining the diversity of population. We can draw a map showing the contour of utility work on the population grid structure. The results for Pop-S of Arcus 1 after 5,000, 15,000, and 25,000 evaluations are presented in Fig. 5. A region having high fitness is darkly painted.

The contour map of Fig. 5(a) shows the island formation at earlier stages of generations, though in general the distribution of phenotypic value is still quite disordered. After 15,000 evaluations have been performed, islands of similar individuals, that are phenotypically flat regions, have been formed as shown in Fig. 5(b). It seems that these results are similar to those obtained by Davidor [26] that analyzes the behavior of a standard GA. After 25,000 evaluations in Fig. 5(c), such islands are still found. However, comparing Fig. 5(b) and (c), the flat regions have been changed. In a standard GA the fitness function is static in that it does not usually change over evolutionary time so that the population of solutions can be seen as traversing a fitness landscape and gradually converging to one of its peaks. Therefore, it tends to preserve the overall shape of contour map and the location of niches, as is observed by Davidor [26]. In contrast, coevolutionary algorithm happens in a dynamic fitness landscape where individuals are constantly adapting to their symbiotic species [27]. This results in a different niching pattern. Since the fitness of individuals changes dynamically according to the symbiotic partners, a rugged fitness landscape is formed and the niche regions shift over evolutionary time. The difference between Fig. 5(b) and (c) supports such assertions.

6.3. Performance Analysis of Coevolutionary Algorithms

Table 3 compares the experimental results obtained from the hierarchical approach, TCoA, LCoA, and PCoA. The first column indicates the problem numbers. The second column shows the total amount of utility work obtained from the hierarchical combination of two GAs. This is compared with those of three cooperative coevolutionary algorithms from column three to five. The experiment is repeated 10 times for every problem instance, and the average of those is reported in the table. The last column shows the improvement rate computed as $\{(\text{Hierarchical approach} - \text{PCoA})/\text{Hierarchical approach}\} \times 100(\%)$.

The experimental results show that, for every problem instance, all the coevolutionary algorithms provide better outcomes than the hierarchical approach. This makes it clear that the traditional approach to aggregated problems is less effective in exploring the solution space. Even though the individual GA is known to be powerful for each sub-problem, simply stacking

Table 3. Comparison of PCoA and other algorithms.

Problems	Hierarchical approach	Coevolutionary algorithms			Improvement rate (%)
		TCoA	LCoA	PCoA	
Thom1	0.00	0.00	0.00	0.00	0.00
Thom2	0.40	0.24	0.17	0.00	100.00
Kim1	1.65	1.25	1.10	0.82	50.30
Kim2	5.51	4.45	3.26	2.26	58.98
Kim3	4.76	3.64	2.74	1.53	67.86
Kim4	6.45	5.01	4.58	3.60	44.19
Kim5	23.42	18.71	15.14	10.76	54.06
Kim6	39.71	27.17	23.91	15.13	61.90
Arcus1	2784.42	1212.75	1181.52	950.57	65.86
Arcus2	7787.11	4957.09	4846.33	3002.16	61.45
Arcus3	2448.67	1166.53	1046.73	968.54	60.45
Arcus4	9377.36	4792.10	4472.61	3714.65	60.39
Arcus5	6477.27	2965.27	2845.91	1845.20	71.51
Arcus6	15804.22	10536.33	9097.72	6533.00	58.66
Arcus7	4256.50	2634.81	2239.33	1677.44	60.59
Arcus8	17362.18	11838.32	10989.59	8934.15	48.54
Arcus9	14825.63	7848.08	6939.79	6013.67	59.44
Arcus10	46890.72	38971.69	37935.44	26996.73	42.43
Arcus11	14013.29	8349.37	7429.33	7343.30	47.60
Arcus12	92997.84	48556.41	43722.57	39309.59	57.73

up the results cannot guarantee good solutions to the aggregated problem.

Among the coevolutionary algorithms, the proposed one (PCoA) outperforms the other two for all the problem instances. With TCoA the association between an individual and its symbiotic partner is fixed over all the generations. Such strong restrictions can prevent individuals from interacting with various symbiotic partners. In contrast with TCoA, LCoA allows the symbiotic partners of an individual can change over evolutionary time. However, all individuals in a population adapt to only one symbiotic partner that gives the highest fitness. This also discourages maintaining population diversity. To overcome this shortcoming, we choose a strategy of localized evolution in PCoA. As the results shown in Section 6.2, this new interaction arrangement leads to maintaining global diversity, and thereby enhances the performance of the optimization process. This conclusion is in agreement with that in Davidor [26].

Furthermore, the improved performance can be achieved with a little additional computational burden. For PCoA, the average computation time for Thomopoulos's, Kim's, and Arcus's problems are 15, 150, and 380 seconds, respectively. The computational burden for LCoA is about the same as ours, and TCoA requires 30% less computation time.

7. Concluding Remarks

We propose a new approach that can solve balancing and sequencing problems in MMALs at the same time. A cooperative coevolutionary algorithm is employed as the underlying framework. To enhance population diversity and search efficiency, a localized evolution strategy is adapted and methods of selecting symbiotic partners and evaluating fitness are developed. Efficient genetic representations and operator schemes are also provided. While designing the schemes, we take into account the features specific to each of the two problems.

The experimental results described in Section 6.2 supports that the localized evolution encourages niching and sharing while discouraging convergence

on sub-optimal solutions. It turns out that the proposed coevolutionary algorithm is much better for the problems considered than not only a traditional hierarchical approach but also the other two cooperative coevolutionary algorithms.

The underlying scheme of the proposed algorithm can be applied to problems involving two interrelated sub-problems that need to be solved simultaneously. Several conditions in order for the system we envision to be put to use are extended from Potter [14]: 1) a species represents a partial solution to the problem being solved; 2) a complete solution is obtained by assembling a member of each of the species present so that the fitness of an individual is determined when all the species are combined; and 3) each species can evolve by itself. Some examples include capacity assignment and machine scheduling in production planning, the determination of network structure and weights in neural network design, forming machine cells and part groups in cellular manufacturing system, determining facility location and resource allocation, and so on. Considering the high compliance of evolutionary algorithms, the proposed algorithm can be easily suited to those problems.

Acknowledgments

This research has been supported by the Korea Science and Engineering Foundation under Grant Number KOSEF 98-0200-09-01-3.

References

1. N.T. Thomopoulos, "Line balancing-sequencing for mixed-model assembly," *Management Science*, vol. 14, pp. 59–75, 1967.
2. A.L. Gutjahr and G.L. Nemhauser, "An algorithm for the line balancing problem," *Management Science*, vol. 11, pp. 308–315, 1964.
3. L. Tsai, "Mixed model sequencing to minimize utility work and the risk of conveyor stoppage," *Management Science*, vol. 41, pp. 485–495, 1995.
4. N.T. Thomopoulos, "Mixed model line balancing with smoothed station assignment," *Management Science*, vol. 16, pp. 593–603, 1970.
5. J.L.C. Macaskill, "Production-line balances for mixed-model lines," *Management Science*, vol. 19, pp. 423–434, 1972.
6. A.K. Chakravarty and A. Shtub, "Balancing mixed model lines with in-process inventory," *Management Science*, vol. 31, pp. 1161–1174, 1985.
7. M. Kilbridge and L. Wester, "The assembly line model-mix sequencing problem," in *Proceeding of the 3rd International Conference on Operations Research*, Oslo, Paris, 1963, pp. 247–260.
8. K. Okamura and H. Yamashina, "A heuristic algorithm for the assembly line model-mix sequencing problem to minimize the risk of stopping the conveyor," *International Journal of Production Research*, vol. 17, pp. 233–247, 1979.
9. C.A. Yano and R. Rachamadugu, "Sequencing to minimize work overload in assembly lines with product options," *Management Science*, vol. 37, pp. 572–586, 1991.
10. J.F. Bard, E.M. Dar-El, and A. Shtub, "An analytic framework for sequencing mixed model assembly lines," *International Journal of Production Research*, vol. 30, pp. 35–48, 1992.
11. E.M. Dar-El and A. Navidi, "A mixed-model sequencing application," *International Journal of Production Research*, vol. 19, pp. 69–84, 1981.
12. D.E. Moriarty and R. Miikkulainen, "Forming neural networks through efficient and adaptive coevolution," *Evolutionary Computation*, vol. 5, pp. 373–399, 1997.
13. J.R. Koza, *Genetic Programming*, The MIT Press: Cambridge, Massachusetts, 1992.
14. M.A. Potter, "The design and analysis of a computational model of cooperative coevolution," Ph.D. Dissertation, George Mason University, 1997.
15. L. Bull and T.C. Fogarty, "Artificial symbiogenesis," *Artificial Life*, vol. 2, pp. 269–292, 1995.
16. W.D. Hillis, "Co-evolving parasites improve simulated evolution as an optimization procedure," *Physica D*, vol. 42, pp. 228–234, 1990.
17. C.D. Rosin and R.K. Belew, "New methods for competitive coevolution," *Evolutionary Computation*, vol. 5, pp. 1–29, 1997.
18. M.L. Maher and J. Poon, "Modelling design exploration as coevolution," *Microcomputers in Civil Engineering*, vol. 11, pp. 195–210, 1996.
19. G. von Laszewski, "Intelligent structural operators for *k*-way group partitioning problem," in *Proceedings 4th International on Conference Genetic Algorithms*, San Mateo, CA, 1991, pp. 45–52.
20. E. Falkenauer, "A new representation and operators for genetic algorithms applied to grouping problems," *Evolutionary Computation*, vol. 2, pp. 123–144, 1994.
21. Y.K. Kim, C.J. Hyun, and Y. Kim, "Sequencing in mixed model assembly lines: A genetic algorithm approach," *Computers & Operations Research*, vol. 23, pp. 1131–1145, 1996.
22. A.L. Arcus, "An analysis of a computer method of sequencing assembly line operations," Ph.D. Dissertation, University of California, 1963.
23. Y.K. Kim, Y.J. Kim, and Y. Kim, "Genetic algorithms for assembly line balancing with various objectives," *Computers & Industrial Engineering*, vol. 30, pp. 397–409, 1996.
24. R.E. Smith, S. Forrest, and A.S. Perelson, "Searching for diverse, cooperative populations with genetic algorithms," *Evolutionary Computation*, vol. 1, pp. 127–149, 1993.
25. S. Forrest, B. Javornik, R.E. Smith, and A.S. Perelson, "Using genetic algorithms to explore pattern recognition in the immune system," *Evolutionary Computation*, vol. 1, pp. 191–211, 1993.
26. Y. Davidor, "A naturally occurring niche and species phenomenon: The model and first results," in *Proceedings 4th International on Conference Genetic Algorithms*, San Mateo, CA, 1991, pp. 257–263.
27. H.J.C. Barbosa, "A coevolutionary genetic algorithm for a game approach to structural optimization," in *Proceedings 7th International on Conference Genetic Algorithms*, East Lansing, MI, 1997, pp. 545–552.