CIRP 2nd International Conference
on Reconfigurable Manufacturing
Ann Arbor, MI, USA
August 20-21, 2003

Concurrent Line-Balancing, Equipment Selection And Throughput Analysis For Multi-Part Optimal Line Design

Li Tang[1], Derek M. Yip-Hoi[2], Wencai Wang[1], Yoram Koren[1]

[1] NSF Engineering Research Center for Reconfigurable Manufacturing Systems
University of Michigan
Ann Arbor, MI 48109, USA

[2] Department of Mechanical Engineering
University of British Columbia
Vancouver, B.C., V6T 1Z4

Abstract
Optimal line design seeks to identify the best configuration of resources and allocation of tasks to satisfy criteria such as maximum throughput or minimum cost. Coupling several levels of the problem together provides a more comprehensive solution but can be difficult because of problem formulation and computational complexities. In this paper we present an approach to coupling line-balancing, machine selection (including buffer) and throughput analysis for manufacturing lines that produce multiple parts. We utilize a Genetic Algorithm formulation to capture in string form the configuration and task allocation for a multiple parts line (MPL). Minimal ratio of cost to throughput is used as the criterion for the fitness function. An analytical throughput analysis engine is called during the evaluation of each solution to size and locate buffers, and to consider the effects of machine breakdown. This method is effectively used during the initial stages of line design to guide manufacturing engineering in evaluating different line design options.

Keywords:
Design, System, Multi-part

1 INTRODUCTION

Designing a manufacturing line is a very complicated problem because it involves a number of issues that are coupled with each other and increase complexity of the problem. These issues include line balancing, machine and buffer selection and throughput analysis. It becomes more intertwined when the line is for multiple parts due to interactions and interventions among different part types.

With growing trends towards great product variety and fluctuations in market demand, building of a manufacturing system with customized flexibility (reconfigurability) oriented at part family has gained increasing attention [1]. The manufacturing line studied in this paper is exactly this kind of system, called multiple parts line (MPL), which can produce a number of parts in a part family. As shown in Figure 1, a MPL consists of several serial stages and each stage contains a number of identical parallel machines that perform the same set of tasks. Between every two stages a finite-size buffer is located to deal with machine failures. The use of MPLs yields great benefits for manufacturing enterprises. Manufacturers can convert to the other parts of the part family quickly to meet stringent due times. Diverse demands from customers are also satisfied while less initial capital investment is required compared with constructing distinctive lines for each part because the MPL shares toolings and fixtures which need considerable expenses.

2 LITERATURE REVIEW

Several key issues need to be considered when an MPL is designed. First, a manufacturing line must be well balanced such that the line attains the highest throughput. Secondly, the machine type to be chosen for each stage also greatly affects the effectiveness of the line for long runs since different machines will run with different speeds and reliability. The third issue is throughput analysis for estimating how many parts the line can yield during a specified time period considering machine parameters and buffers allocation.

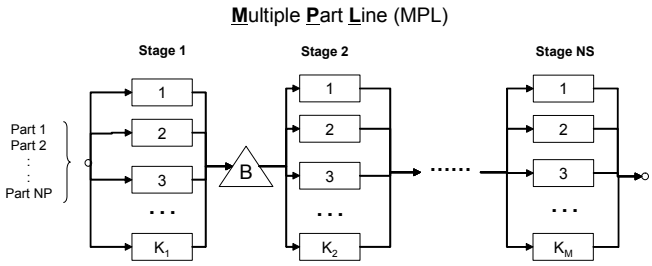

Figure 1 Multiple Parts Line

To date, researchers have investigated each of the above issues as an independent problem. In 1955 Salveson[2] was the first researcher who constructed a mathematical model for the line-balancing problem and provided a solution procedure. Since then numerous optimum-seeking algorithms and heuristic procedures that attempt to solve different line-balancing problems for single product or multiple products have been developed. Basically these methods seek an optimal task allocation scheme that minimizes the number of stations for a fixed cycle time (Type I problem), or maximize the line throughput for a fixed number of stations (Type II problem)[3]. Shen and Tsai[4] propose a graph matching approach to search for the optimal task allocation. Ercal [5] use a simulated annealing method to find a solution. Others, such as Yip-Hoi[6], Aoyagi[7] and Hadj-Alouane[8], use genetic algorithm (GA) to explore the best allocation scheme.

For the machine selection problem when designing a manufacturing system, ElMaraghy[9] has proposed the use of an Alternative Process Plan (APP) to describe all the

optional machines mapped to one task. Kiritsis and Porchet[10] utilize a simple and safe Petri net model to represent an APP. By reachability analysis on the Petri net, all possible process plans can be generated and the appropriate machines for each task can be identified.

In addition to task allocation and machine selection, the system performance, especially the system throughput under a long run period is of interest to designers. Gershwin[11] applied a Markov model to a three-stage-flow-line and obtained the exact analytical solution. For the flow line with more than three stages, both aggregation and decomposition techniques are employed to approximately analyze the throughput. Yang[12] developed an analytical model based on decomposition technique to estimate the throughput, which is used in the research work of this paper. Apart from analytical methods, many simulation-based methods are used to evaluate the performance of a manufacturing line, e.g. the commercial software *Witness*@ [13] has been widely used in the manufacturing industry.

Although great progress has been achieved in each of the above areas, it is inadequate to build a good line configuration if only one or two of them is concerned. For example, a well-balanced task allocation scheme may not yield the highest throughput if machine reliability in one stage is much lower than those in other stages. Since these factors are coupled together, a concurrent method is presented to provide a more comprehensive solution.

3 NOMENCLATURE

NP: Total number of part models produced by MPL

NT: Maximum number of manufacturing tasks of a part

NS: Total number of production stages of MPL

NR: Total number of available machine types

K: Maximum number of alternative machines being assigned to a manufacturing task

NC: Maximum number of stage capabilities

p : Index for part, p = 1, …, NP

i,j : Index for manufacturing task, i, j = 1, …, NT

s : Index for production stage in MPL, s = 1, …, NS

r : Index for available machine type, r = 1, …, NR

k : Index for alternative machine type, k = 1, …,K

4 PROBLEM FORMULATION

Given a part family with NP parts to be produced and each part possessing up to NT manufacturing tasks (hereinafter "tasks"), a MPL will be built to perform all these tasks. The sequence of performing tasks of each part must conform to basic process planning rules as well as user-specific requirements, which are documented in a task-precedence-graph. Each task can be performed on K alternative machines chosen from the machine library which contains NR candidate machines. All tasks must be finished in NS serial production stages (hereinafter "stages"), where identical machines are arranged in parallel in each stage with finite-size buffers (the size can also be zero) between stages. With cost constraints, machine quantity limit and demands, the system designer wishes to configure a MPL with maximum ratio of throughput to cost. The goal is achieved by answering the following questions:

1. How are the tasks assigned to each stage without violating precedence constraints?
2. What machine type is used and how many machines are required in each stage?
3. How large is the buffer size between each stage?
4. What is the estimated throughput for each part?

4.1 Assumptions

The following assumptions are made in this work:

1. The number of stages, NS, is specified by designers in advance.
2. Each part in the family must visit all stages in the system.
3. The machines in the same stage perform the same tasks.
4. Each part is produced in a batch and the batch is large enough so that the throughput for each part can be estimated separately.

4.2 Inputs

Task-Precedence-Graphs

Each part in the part family has its own task-precedence-graph that defines sequential constraints between tasks. A three-dimension binary matrix Pre[1..NP][1..NT][1..NT] has been used to represent all precedence graphs.

$$Pre[p][i][j] = \begin{cases} 1 & \text{if task i must be performed} \\ & \text{before task j of part p;} \\ 0 & \text{otherwise.} \end{cases}$$

Machine Library

All available machine types comprise the machine library from which alternative machines are specified for each task. It is represented by a one-dimension array *ML[1..NR]*.

$$ML[r] = \text{Unique ID of the rth Machine}$$

Machine Cost Array

An array *CE[1..NR]* records costs of all machines in the machine library.

$$CE[r] = \text{Cost of the rth machine}$$

Machine Reliability Martix

A matrix *REL[1..NR][2]* records mean-time-between failure (MTBF) and mean-time-to-repair (MTTR) of each machine in the machine library.

$$REL[r][0] = \text{MTBF of the rth machine;}$$
$$REL[r][1] = \text{MTTR of the rth machine.}$$

Alternative Machine Matrix and Alternative Time Matrix

Each task can be performed on alternative machines and that may entail different working times. The Alternative Machine Matrix *AMM[1..NP][1..NT][1..K]* stores the alternative machines for every task of the part family. The

respective working times are stored in the Alternative Time Matrix *ATM[1..NP][1..NT][1..K]*.

$AMM[p][i][k] = r$: r is an index of the ML array, standing for the k[th] alternative machine for task i of part p $(1 \le r \le \text{NR})$.

$ATM[p][i][k] =$ The working time needed by task i of part p, if it is assigned to the k[th] alternative machine.

Stage Key Characteristic Matrix

Each manufacturing stage usually has limited capabilities, which are reflected by a group of key characteristics of the stage. When a set of tasks are assigned to a stage, the necessary capabilities must fall in the key characteristics of the stage. Otherwise the task allocation is invalid. For example, the machining stage only has several possible cutting-tool access directions and thus the tasks assigned to this stage must be performed on those directions.

Assuming the number of the key characteristics of each stage is NC, a capability matrix *SKC[1..NS][1..NC]* stores all possible key characteristics of each stage.

$SKC[s][j] = d$: d is the jth key characteristic of stage s $(1 \le s \le \text{NS}, 1 \le j \le \text{NC})$.

Task Key Characteristic Matrix

Respectively, each manufacturing task corresponds to a key characteristic. A task key characteristic matrix *TKC[1..NP][1..NT]* stores the key characteristic of each task.

$TKC[p][i] = d$: d is the key characteristic of task i in part p $(1 \le p \le \text{NP}, 1 \le i \le \text{NT})$.

Throughput Demands Array

The MPL must be able to meet throughput demands of all parts which are stored in the array *TH[1..NP]*.

$TH[p] =$ Throughput demand of part p

Cost bound

Maximal allowable investment on the MPL is *MaxInvest*.

Machine Quantity Limit

Total number of machines cannot exceed *MaxNMC*

Buffer Cost per Unit Size:

The average cost of each unit buffer size is *BufCost*.

4.3 Decision Variables

MPL Type Array

$MPL[s] =$ r : r is an index of ML, representing the machine type to be used in stage s $(1 \le r \le NR)$.

Machine Number Array

$NMC[s] =$ Numbe of machine being used in stage s.

Buffer Size Array

$BUF[s] = B$uffer size between stage s and $s+1$.

BUF[NS] = 0 since there is no buffer after the last stage.

Task Allocation Matrix

A two-dimensional matrix TAM[1..NP][1..NT] records the allocation of tasks to stages of the MPL.

$TAM[p][i] = s$: s is an index of stage to which the task i of part p is assigned $(1 \le s \le NS)$.

4.4 Throughput Function

F_{TH}(p, MPL,NMC, BUF, TAM, REL)

The function is used to estimate the throughput of part *p* under the line configuration specified by arrays MPL[], NMC[], BUF[] and matrices TAM[][], REL[][]. A throughput analysis engine – Performance Analysis of Manufacturing Systems (PAMS), based on Yang's work [12], is called to calculate the throughput for each part.

4.5 Mathematical Model

A mathematical model has been formulated to find the best line configuration and task allocation scheme. It is described in the following:

$$Minimize \quad \frac{\sum_{s=1}^{NS} NMC[s] * CE[MPL[s]] + \sum_{s=1}^{NS-1} BUF[s] * BufCost}{\sum_{p=1}^{NP} F_{TH}(p, MPL, NMC, BUF, TAM)} \quad (1)$$

The objective expressed in (1) is to minimize the ratio of the total investment in machines and buffers to the total throughput of all parts.

Subject to

1. Precedence constraints:

$$Pre[p][i][j] * TAM[p][i] \le TAM[p][j] \quad (1 \le p \le NP, 1 \le i \le NT, 1 \le j \le NT) \quad (2)$$

For two tasks i and j in part p, if task i must be performed before j (Pre[p][i][j] = 1), it will be assigned to the same or a preceding stage as j.

2. Functionality constraints:

$$\prod_{k=1}^{K} (MPL[TAM[p][i]] - AMM[p][i][k]) = 0 \quad (1 \le p \le NP, 1 \le i \le NT) \quad (3)$$

All tasks assigned to the same stage must use the same machine type.

3. Key characteristic constraints:

$$\prod_{j=1}^{NC} (SKC[TAM[p][i]][j] - TKC[p][i]) = 0 \quad (1 \le p \le NP, 1 \le i \le NT) \quad (4)$$

If the task i is assigned to the stage s, its key characteristic must be one of those for the stage s.

4. Throughput demands:

$$F_{TH}(p, MPL, NMC, BUF, TAM, REL) \ge TH[p] \quad (1 \le p \le NP) \quad (5)$$

Each part must achieve its predefined throughput under the optimal system configuration.

5. Cost bound:

$$\sum_{s=1}^{NS} NMC[s] * CE[MPL[s]] + \sum_{s=1}^{NS-1} BUF[s] * BufCost \le MaxInvest \quad (6)$$

The total cost of machines and buffers being used cannot exceed the budget bound.

6. Machine quantity limit:

$$\sum_{s=1}^{NS} NMC[s] \le MaxNMC \quad (7)$$

The total number of machines cannot exceed a specified value because of shop floor area.

The above formulation can be viewed as a Mixed Integer Programming problem. However it contains a few complex functions some of which do not have explicit forms like the throughput function $F_{TH}()$. Furthermore, previous studies have shown that it is an NP-hard problem, i.e., as the problem size increases combinatorial explosion will happen. Therefore conventional optimization techniques, such as gradient method and SQP, cannot find the solution in a reasonable time.

Due to the above reasons a heuristic method utilizing a Genetic Algorithm is used to solve this optimization problem.

5 GENETIC ALGORITHM METHOD

Genetic algorithms (GA) have been developed as a powerful and broadly applicable heuristic search and optimization technique since the 1970s. For mixed integer problems with complex functions and combinatorial explosion on feasible space, like MPL design problem, GA is a good and efficient way to solve them according to Aoyagi, Hadj-Alouane, Yip-Hoi, Cheng and Gen's research experiences [6-8, 14].

The basic idea of a GA is to maintain a population of strings, say *P*(t), for generation t. Each string represents an encoding of a potential solution to the problem at hand and will be evaluated to provide some measure of its fitness. Some strings undergo stochastic transformations by means of genetic operations. There are two types of transformations: *crossover*, which creates new strings by combining parts from two parent strings, and *mutation*, which creates new strings by making changes in a single one. New strings, called *offspring C*(t), are then evaluated. By selecting the more fit strings from the parent and offspring populations a new population is formed. After a number of generations, the algorithm converges to the best individual, which represents a near-optimal solution to the problem. Due to the stochastic nature of a GA, it is not guaranteed that a global optimum can be attained.

5.1 String Representation

The first step of applying GA is to represent a solution of the problem by the format of a string, i.e. encoding. An MPL design scheme is encoded by an integer double string which consists of 5 portions, corresponding to (1) task sequencing, (2) alternative machine selection, (3) task allocation, (4) buffer size and (5) number of machines, respectively. Figure 2 depicts the string that encodes an MPL producing two parts L and R.

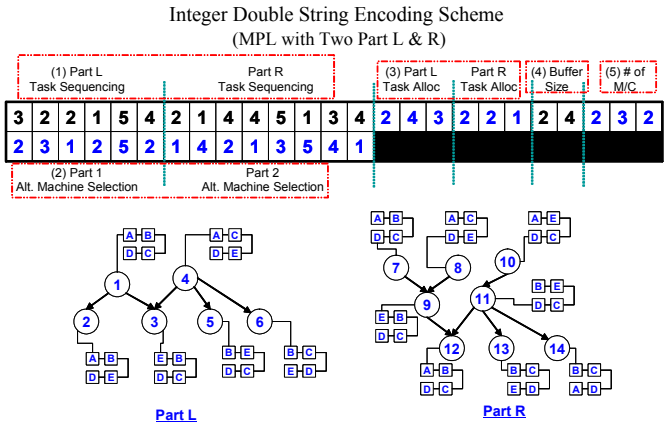

Figure 2 String Representation

5.2 Decoding

After an offspring is obtained by genetic operators, it has to be translated to a corresponding solution for evaluation. This is called decoding. The 5 portions of the string mentioned in the preceding paragraph will be decoded to their respective part of an MPL design solution. That is described in the following sections.

Decoding the task sequence

Since task precedence relations are stored in a precedence graph, a partial precedence graph sorting technique [14] is used to translate the string to a sequence of tasks without violating precedence constraints.

Each digit in the first portion of the string stands for an element of the eligible task set in which no tasks have any precedent task. The first eligible set is constructed based on the original precedence graph and the first digit indicates which task will be chosen as the first one in the sequence. Then a new eligible set is generated by removing the first task from the graph and the second digit identifies the second task. The detailed procedure is explained in [14]. Finally a sequence of tasks is generated with all precedence constraints satisfied.

Decoding the task allocation

The second portion of the string determines how tasks are assigned to each stage of the MPL. Each digit corresponds to a stage and stands for the percentage ratio of the number of tasks in the stage to the total number of tasks of the part. For example, by the string 2-4-3 it is determined that stage 1 contains 2/(2+4+3)=2/9 of all tasks. According to the ratios the task sequence is divided and assigned to N stages.

Decoding the alternative machine selection

Each task corresponds to a number of alternative machines which comprise a machine set. As shown in Figure 2, task 1 has A, B, C, D (4 machine types) associated with it. Each digit in the third portion corresponds to the respective task in the sequence

obtained in (1) and a machine is chosen by the value of the digit. In the example of Figure 2, if task 1 is put in the second place, 3 is the code for alternative machine selection and machine C is chosen.

Since tasks assigned to the same stage (by decoding the second portion) may select different machine types, the code in the third portion needs to be repaired, i.e. making the tasks in the same stage use identical machine. A simple heuristics is employed such that the most frequently used machine type of each stage becomes the dominant one and all tasks in the stage must use this one. If some tasks cannot be performed on this machine, a penalty value is imposed.

Decoding the buffer size

The n^{th} number stands for the size of buffer following stage n. The buffer following the last stage is assumed to be infinity.

Decoding the number of machines

The value of the n^{th} digit is the number of identical machines used in the stage n.

5.3 Fitness (Objective) Function

This is a function which measures the relative worth of the solution when applied to a decoded string. To this problem, the fitness function evaluates the ratio of the total cost to the total throughput. Through the comparison of string fitness values, the GA moves toward the optimal solution.

5.4 Other Constraints

The decoding method above has satisfied precedence constraints (2) and tested functionality constraints (3). However, the remaining three constraints: key characteristic constraints (4), throughput demands (5) and cost bounds (6), still need to be evaluated.

To evaluate these constraints, the penalty method is used. A penalty function is developed for each type of constraint. If a decoded string violates any constraints, its fitness will be added on the respective penalty value.

For each violation of key characteristic constraint, 1000 penalty value is added to the final fitness.

If the actual throughput TH_a (parts/hour) is less than the throughput demand TH_d, a penalty value $10000*(TH_d - TH_a)$ will be applied.

When the total system cost C_t exceeds the cost bound C_b, a penalty value $10000*(C_t - C_b)$ will be applied.

5.5 Genetic Operators

The following three mechanisms are used to create successive generations of strings from the parent strings.

1. Reproduction: This operator copies the individual strings with better fitness value into the next generation.
2. Crossover: The "genetic material" within parent strings is broken down and then reconstituted into new solutions. It is the primary method to introduce new solutions into the population.
3. Mutation: This is another way to add new "genetic material" to a population. Randomly selected strings undergo the altercation according to the mutation rate. The purpose of doing so is to help the GA avoid getting trapped at a local optimum.

6 CASE STUDY

The above genetic algorithm is applied to the machining domain by using a case study from the automotive industry to demonstrate the MPL design approach.

6.1 Machining Domain Background

The 'task' in the machining domain refers to a series of cutting operations that use the same cutting tool in the same setup. The operations like drilling several mounting holes on the same surface are considered as a single task. According to the requirements of the machining process planning, between tasks exist sequential constraints which are represented by a task precedence graph. The key characteristic of the task is defined as the cutting-tool access direction (TAD). Each 'stage' contains identical CNC machines or Reconfigurable Machine Tools (RMT) that normally possess 2 or 3 possible TADs. Thus the SKC and TKC matrix will record TADs.

6.2 Inputs

Two cylinder heads, called "left-hand" and "right-hand" respectively (shown in Figure 3), comprise the part family to be produced. 84 manufacturing tasks are required for a left-hand cylinder head while 80 tasks for a right-hand one. Both parts are supposed to visit 8 production stages (this is an input and can also be 7 or 9).

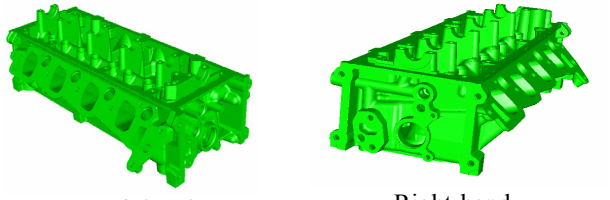

Figure 3 Cylinder Head Part Family

The precedence graphs for both parts are shown in Figure 4. The left-hand part has 106 constraints (edges) and the right-hand part has 99 constraints.

Each task has 3 alternative machines with different task completion time on each. In this case all alternative machines use equal time. The TAD of the task is also documented in the Table 1.

Task ID	M/C 1	Time 1	M/C 2	Time 2	M/C 3	Time 3	TAD
101001	101	18.32	102	18.32	103	18.32	EF
101002	101	15.50	102	15.50	103	15.50	EF
…	…	…	…	…	…	…	

Table 1 Tasks with Alternative Machines

The available machines (machine library) are listed in Table 2.

M/C ID	Name	Cost(K$)	MTBF(min)	MTTR(min)
101	A	200	1500	8
102	B	160	1200	10
103	C	180	1300	12
104	D	225	1850	10
105	E	140	850	20

Table 2 Machine Library

It is required that both throughputs must exceed 35 parts per hour (PPH), total cost is limited under 8400 K$ and at most 43 machines can be used in the line.

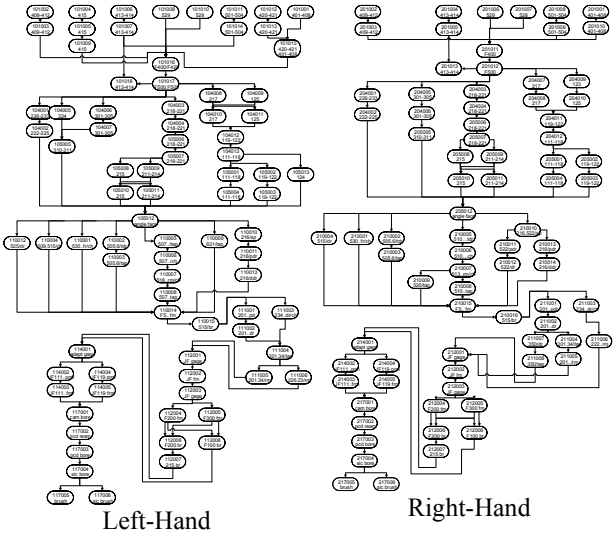

Figure 4 Precedence Graphs

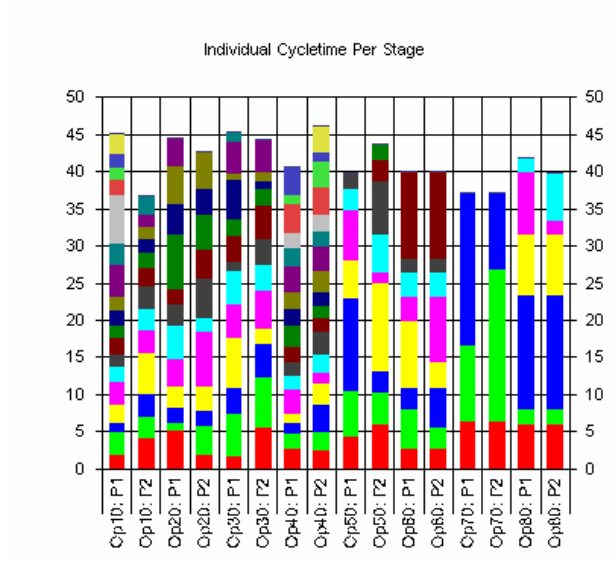

Figure 5 Cycletime Distributions Per Stage of Each Part

The stage key characteristics, i.e., the possible TADs for each stage are listed in Table 3.

Stage	1	2	3	4	5	6	7	8
TAD 1	FF	CF	CF	FF	CF	CF	CF	CF
TAD 2	EF	IF	IF	RF	IF	JF	JF	FF
TAD 3	n/a	JF	JF	CF	JF	n/a	n/a	n/a

Table 3 Stage Key Characteristic (TAD)

6.3 Results

The optimal ratio of system cost to throughput is 7.4431 where the total cost is 7,310 K$ and total throughput is 71.328 PPH. There are in total 43 machines and 14 unit buffers being used in the line.

Figure 5 shows how tasks from each part are allocated to 8 stages. The cycle-times of both parts per stage are shown in Figure 6. The individual cycletime distribution is unbalanced for two reasons: one is that constraints imposed on tasks prevent them from being performed in any stage; the other is that some tradeoff must be made so that the total cycletimes are more evenly distributed. As shown in Figure 6, the total cycletime per stage is around 80 seconds.

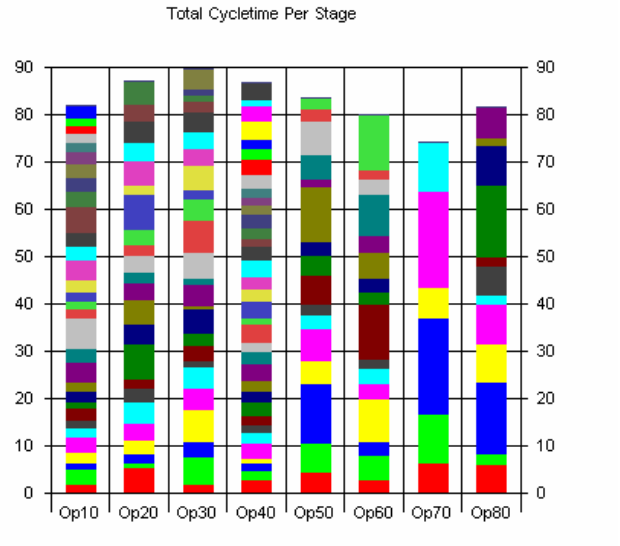

Figure 6 Total Cycletime Distributions per Stage

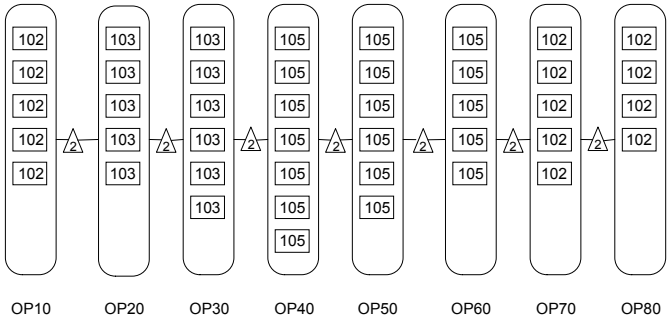

Figure 7 MPL System Configuration

The type and quantity of machines used in each stage are shown in Figure 7 as well as buffer sizes.

Figure 8 gives the GA convergence curves using different control parameters. It shows the best solution of 7.4431 was found in 600 generations (curve 2). Most of the runs converge to close to the same value with the similar solution. The computation time required was around 66 minutes on a Pentium 4 2.0GHz PC with 512MB memory. Compared to the large solution space, $8^{5}\cdot(84+80)^{8}\cdot 7^{10} = 4.844\text{x}10^{30}$ (obtained from the following facts: each stage can choose 5 machine types, each task can perform at any of 8 stages, each buffer size can vary from 1 to 10), the GA algorithm is much more efficient.

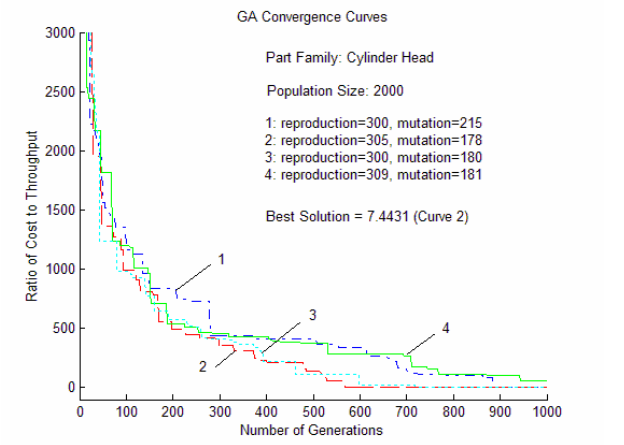

Figure 8 GA Run Convergence Curves

7 DISCUSSIONS AND FUTURE WORK

The results obtained in the case study show a good match compared to the actual task allocation and machine quantity distribution determined by the experts in the industry.

Besides system cost, throughput and task precedence issues mentioned in this paper, other optimization goals and constraints concerned by users (if any) can be easily integrated into the proposed approach. Only minor changes to the genetic representation and evaluation are needed.

There are a number of improvements that can be made in future. The first is about the number of stages in the MPL, which is specified by the designer in the current approach. The MPL is divided into several stages because tasks have different TADs that may need different setups and fixtures. Apart from the effects of process plans, material handling devices impose some limitations that restrict the maximum number of stages and machines in each stage of the line. We are currently looking at ways to include this issue in our GA algorithm.

A second improvement focuses on routing issues in MPL. In the current approach, it is assumed that every part will visit every stage and every machine in each stage undertakes identical tasks. The assumption may not be reasonable in some manufacturing practices like assembly where a part does not go through all stages and machines in the same stage may perform different tasks. To cop with this situation we need to revise our gene representation scheme to adapt to a flexible-routing-MPL.

The third one considers how to reconfigure the MPL when new parts are introduced into the part family to replace existing ones. In addition to all requirements mentioned before, we have to generate a reconfiguration plan with minimal expense.

Some other topics such as reducing computation time, improving algorithm efficiency and providing user-interactive capabilities are also under investigation.

8 CONCLUSIONS

In this paper, we have studied how to design a multiple parts line with integration of line balancing, equipment selection and throughput analysis. It is formulated by a mixed integer programming model and solved by a genetic-algorithm based approach which enables us to cope with the intrinsic complexity of configuring a manufacturing line. Furthermore, a case study from a real-world manufacturing application has been investigated by our approach and a satisfying result was obtained within a short duration. Compared to many trial-and-error processes currently used in industry, this saves a lot of time for line designers to come up with a final decision.

9 ACKNOWLEDGMENTS

The authors are pleased to acknowledge support of this research by the National Science Foundation Engineering Research Center for Reconfigurable Manufacturing Systems under Grant # EEC-9529125.

10 REFERENCES

[1] Koren, Y., Heisel, U., Jovane, F., Moriwaki, T., Pritchow, G., Van Brussel, H., Ulsoy, A.G., 1999, Reconfigurable Manufacturing Systems, Annals of the CIRP, 48/2 (keynote paper).

[2] Salveson, M., 1955, The assembly line balancing problem, Journal of Industrial Engineering, 6/3:18-25

[3] Ghosh, S. and Gagnon, R., 1989, A comprehensive literature review and analysis of the design, balancing and scheduling of assembly systems, Int. J. Prod. Res., 27/4:637-670.

[4] Shen, C. C. and Tsai, W. H., 1985, A graph matching approach to optimal task assignment in

distributed computing systems using a minimax criterion, IEEE Trans. on Computers, 34/3:197--203.

[5] Ercal, F., Sadayappan, P. and Ramanujam, J., 1988, Task allocation by simulated annealing, Proceeding of International Conference on Supercomputing, Boston, MA, 475--497,.

[6] Yip-Hoi, D., Dutta, D., 1996, A Genetic Algorithm Application For Sequencing Operations In Process Planning For Parallel Machining, IIE Transactions, 28/1:55-68

[7] Aoyagi, Y., Uehara, M., Mori, H. and Sato A., 1999, GA-based Task Allocation by Throughput Prediction, IPSJ Journal, 40/12

[8] Hadj-Alouane, A., J. Bean and K. Murty, 1999, A Hybrid Genetic/Optimization Algorithm for a Task Allocation Problem, Journal of Scheduling, Vol. 2:189-201.

[9] ElMaraghy, H.A., 1993, Evolution and Future Perspectives of CAPP, Annals of the CIRP, Vol.42.

[10] Kiritsis D. and Porchet M., 1996, A generic Petri net model for dynamic process planning and sequence optimization, Advances in Engineering Software, Vol.25:61-71.

[11] Gershwin S.B., 1994, Manufacturing Systems Engineering, Prentice-Hall, Englewood Cliffs, NJ,

[12] Yang S., Wu C., Hu S.J., 2000, Modeling and analysis of multi-stage transfer lines with unreliable machines and finite buffers, Annals of Oper. Res., Vol.93:405-421.

[13] Learning Witness, 1998, Lanner Group, Inc.

[14] Gen M., Cheng R., 2000, Genetic algorithms and engineering optimization, Wiley, New York.