A Hybrid Genetic Algorithm for Parallel Machine Scheduling at Semiconductor Back-End Production

J. Adan,[1,2] **A. Akcay,**[1] **J. Stokkermans,**[2] **R. Van den Dobbelsteen**[2]
[1]Department of Industrial Engineering, Eindhoven University of Technology
De Zaale, 5600 MB, Eindhoven, The Netherlands
[2]Industrial Technology and Engineering Centre (ITEC), Nexperia
Jonkerbosplein 52, 6534 AB, Nijmegen, The Netherlands

Abstract

This paper addresses batch scheduling at a back-end semiconductor plant of Nexperia. This complex manufacturing environment is characterized by a large product and batch size variety, numerous parallel machines with large capacity differences, sequence and machine dependent setup times and machine eligibility constraints. A hybrid genetic algorithm is proposed to improve the scheduling process, the main features of which are a local search enhanced crossover mechanism, two additional fast local search procedures and a user-controlled multi-objective fitness function. Testing with real-life production data shows that this multi-objective approach can strike the desired balance between production time, setup time and tardiness, yielding high-quality practically feasible production schedules.

Introduction

Production scheduling is one of the most active areas of operational research, which is largely due to the rich variety of problems within the field. The first to recognize the parallel machine scheduling problem was (McNaughton 1959). In the last 50 years this problem has been extensively studied. This research includes literally hundreds of publications on exact techniques, heuristics and meta-heuristic algorithms for the parallel machine scheduling problem and some of its variants (Allahverdi 2015). Nonetheless, there is a noticeable gap between the theory and the application of the developed methods. The proposed methods often neglect important aspects of real-world manufacturing environments. On the other side, scheduling is becoming an increasingly complex task, as technological advancement leads to companies with ever growing product portfolios.

Although there is a recent trend towards more realistic formulations of scheduling problems, there are yet not many research efforts that consider the full complexity of realistic manufacturing environments. Most literature addresses the case with identical machines. However, less research focuses on environments where the process time of a job depends on the machine on which it is processed, i.e. the machines are unrelated (Cheng, Ding, and Lin 2004). The number of works that also address sequence- and machine-dependent setup times are even fewer. Recent works that apply meta-heuristics for such environments can be found in (Vallada and Ruiz 2011; Ying, Lee, and Lin 2012; Avalos-Rosales, Angel-Bello, and Alvarez 2015). Furthermore, some jobs may only be processed on a certain subset of the available machines. Environments with such a heavy machine dependency are common in industry due to the presence of multiple machines from different ages and with various technologies.

Motivated by the back-end production of integrated circuits (ICs) at an Assembly and Testing facility of Nexperia, this paper addresses an unrelated parallel machine scheduling problem with sequence- and machine-dependent setup times, machine eligibility and due dates. A drawback of the current way of scheduling is that planners can only oversee a limited number of machines over a limited time span, and furthermore that the quality of schedules relies largely on the planner's experience. Through increased scheduling automation, these drawbacks can be overcome. To solve this NP-hard (Pinedo 1995) scheduling problem, a novel hybrid genetic algorithm (HGA) metaheuristic is proposed. Besides two fast local search methods, this genetic algorithm includes a local search enhanced cross-over mechanism. The most studied optimization criterion for scheduling problems is the minimization of the maximum completion time, a measure that is also known as the makespan or $C_{\max}$. It is foreseeable that the minimization of $C_{\max}$ will attempt to fully utilize all available machines upto $C_{\max}$, possibly at the expense of additional setup time (i.e. $C_{\max}$ is a regular criterion). However, setups are highly undesired in practice as they cost capacity and operator time. The fitness function of the genetic algorithm contains a multi-objective criterion that attempts to strike a user-controlled balance between the setup time, process time and tardiness.

In the next section an overview of the back-end production process of ICs at a Assembly and Testing facility of Nexperia is given.

Back-end Production Process

At the back-end production of electronics manufacturing at Nexperia, semi-finished ICs are encapsulated in a supporting case to prevent physical damage and corrosion. The case, which is commonly known as a "package", supports the electrical contacts which connect the device to a circuit board. This packaging process is done on assembly lines.

Each assembly line consists of multiple workstations, each of which can only perform one specific operation: die bonding, wire bonding or molding. The semi-finished ICs, so-called dies, are small blocks of semiconducting material on which a certain IC is fabricated. Die bonding refers to the bonding of dies to a (metal) support structure, the so-called lead frame. Wire bonding is the process of making interconnections between the die and a specific location on the lead frame. Molding refers to the encapsulation of the device by a molding compound to protect it from the outside environment. Each assembly line is capable to perform all three operations, i.e. it is composed of at least one die bonding, one wire bonding and one molding station. However, some assembly lines contain more than three workstations, which simply means that some can process products at higher velocity. Since the lead frame flows through each workstation of an assembly line, and thereby couples all workstations, each line is treated as a single machine.

Each product requires a certain set of process parameters to be set accordingly on each workstation of the assembly line. This set of parameters is referred to as the product's recipe. When two different products, each having a different recipe, are subsequently processed on a certain assembly line, this requires reconfiguration of the workstations, the extent of which depends on (1) the difference between the recipes and (2) the number of workstations that require reconfiguration. Thus, the setup time depends on both the production sequence and the machine. Furthermore, not all machines are able to accommodate all recipes, which translates to machine eligibility constraints.

Summarizing, this scheduling problem is described by: the set of jobs $N = \{1, \dots, n\}$ and the set of machines $M = \{1, \dots, m\}$. The due date of job i is d_i, its process time on machine k is $p_{i,k}$, the setup time between the subsequent processing of job i and j on machine k is $s_{i,j,k}$.

Genetic Algorithm

In a genetic algorithm a population of candidate solutions within the search space, called individuals, evolves toward better solutions through an iterative evolutionary process. The population in each iteration is referred to as a generation. In each generation, the fitness of the individuals is evaluated, which is usually the value of the objective function in the optimization problem. Then, a subset of the individuals is selected (the parents) and a cross-over mechanism is applied to obtain a new generation of individuals (the offspring). Moreover, a mutation operator is applied to maintain genetic diversity.

The main features of the proposed algorithm are a local search enhanced cross-over mechanism, two fast local search procedures and a user-controlled multi-objective fitness function. A description of the components of this hybrid genetic algorithm is presented in the next sections.

Representation and Fitness

The genome of an individual is represented by arrays of jobs for each machine that represent the process sequence of the jobs assigned to a certain machine. The population consists of P_{size} individuals, where the genome of each individual consists of m arrays of jobs, one for each machine. This way, each array can be viewed as a chromosome constituent of the individual's genome.

The fitness F of an individual is the weighted sum of the process time P, setup time S and tardiness T, i.e.

$$F = \alpha P + \beta S + \gamma T$$

where α, β and γ are non-negative weights. These weights can be set by the user and allow the user to put more or less emphasis on each of the output parameters. Thus, the lower the fitness F, the fitter the individual. The fitness of a chromosome is defined accordingly and the fitness of an individual is the sum of the fitness of each of its chromosomes.

Initialization and Selection

Although it is common to randomly generate the initial population in a GA, there is a recent trend to include some good individuals in the initial population that are generated by some heuristic. In this GA, the initial population is generated by first creating a random permutation of the list of jobs to be assigned through a Fisher-Yates shuffle mechanism (1948). Then, all jobs in the list are scheduled as follows: for each job all possible insertion positions are evaluated. Finally, the most favourable in terms of the fitness is selected. All P_{size} individuals are generated according to this heuristic.

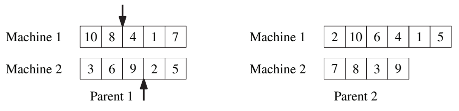
In the selection stage of the GA, parents are chosen from the population for the next generation of offspring. Various selection operators that give preference to the fitter individuals in the population are reported in literature. Here, completely random selection is applied.

Crossover Mechanism and Mutation

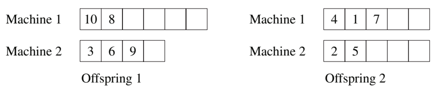
Once the parents are selected from the population a crossover mechanism is applied. The aim of the crossover mechanism is to produce fitter offspring. Many cross-over mechanisms are reported in literature. Here, a local search enhanced crossover (LSEC) mechanism is applied, cf. (Vallada and Ruiz 2011). This crossover mechanism is similar to the classical one-point crossover mechanism, except that it is enhanced with a local search procedure. In Figure 1 an example for a situation with two machines and ten jobs is depicted.

After the selection of two parents, one point is randomly determined in the job sequence of each machine of parent 1. The jobs to the left hand side of this point are copied to offspring 1 and the jobs on the right hand side are copied to offspring 2. Then, for each machine of parent 2, the jobs that are not yet assigned to parent 1 are inserted. At this moment the local search procedure comes into play: when a missing job is inserted into parent 1, it may be assigned to any eligible machine and any position in the job sequence. It is inserted at the best position in terms of fitness.

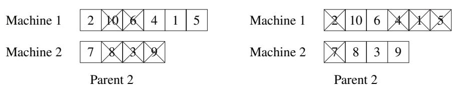
As mentioned, the aim of the crossover mechanism is to produce fitter offspring. If the offspring is indeed fitter, it replaces its parents. Otherwise, the parents remain in the population. To maintain genetic diversity in the population and to prevent the algorithm from getting trapped in a local



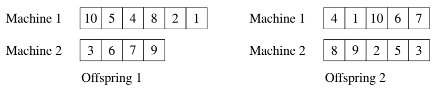
(a) Selected parents.



(b) Copied jobs from parent 1.



(c) Missing jobs.



(d) The offspring.

Figure 1: One point cross-over mechanism enhanced with local search.

optimum, individuals are subjected to mutation. Here, the well-known shift mutation mechanism is applied. A job is randomly selected and removed from its current position. It is reinserted at a random position of the job sequence of another random machine. The mutation operator is applied after cross-over and before local search with probability P_{mut}.

Local Search

The genetic algorithm is combined with two fast local search methods to speed up search towards the optimum in the region of convergence. The first exploits the insertion neighborhood while the second evaluates the swap neighborhood. These neighborhoods have been proven powerful for the parallel machine problem considered (Barnes and Laguna 1993). An insertion move removes one job from one machine and inserts it into another. A swap move chooses two jobs and switches their machine assignments. Moved jobs are always inserted at the best possible position in terms of fitness and machine eligibility is taken into account. The size of the insertion neighborhood is $O(n(m-1))$ and the size of the swap neighborhood is $O(n^2)$, where n is the number of jobs and m the number of machines. Therefore, to limit the size of the neighborhood the moves are only applied to the chromosomes with the lowest fitness. This portion is always known since the fitness of the individual's genome as well as of its constituent chromosomes is continuously monitored and ranked. Furthermore, for a given job, its optimal insertion location on any machine can be identified with limited computational effort. After all moves have been evaluated, the move yielding the highest improvement in terms of the fitness is applied, i.e. steepest descent. These search procedures are applied to all offspring during every evolution as follows: with activation probability P_{ls} the insertion neighborhood is exploited first. When activated, the swap neighborhood is exploited subsequently. Both neighborhoods are searched until no move in the neighborhood can be identified that improves the fitness.

Application

The HGA described in the previous section is now applied to the scheduling of a single package at the Assembly and Testing facility of Nexperia. In the next sections the test scenario is described in detail, followed by the presentation of the obtained results.

Test Scenario

The proposed HGA is tested for the scheduling of one package of Nexperia's extensive product portfolio. This package contains more than 600 unique products, each of which requires a different recipe. Therefore, it can be considered as one of the more complex packages being assembled at this particular facility. The assembly of these products is carried out on 17 machines, the production velocity (i.e. capacity) of which differs at most a factor three. This means that the machine on which a batch of products is assembled has a significant effect on the required process time.

To test the performance of the proposed HGA three consecutive data sets are available. Each data set contains (1) the product batches to be scheduled within a certain time span and (2) the current schedule for these jobs. The number of jobs varies between 121 and 134. For each of these data sets, a comparison is made between the automatic (HGA based) and current schedules on the basis of the fitness function with equal emphasis on each of its constituents, i.e. $\alpha = \beta = \gamma = 1$. Additionally, the variability of the outcome of the HGA is examined by performing several runs for the same set of data. Furthermore, the tunability of the multi-objective fitness function is illustrated by comparing schedules obtained with:

- $\alpha = \beta = \gamma = 1$, i.e. equal emphasis on all constituents of the fitness function.
- $\alpha = 0$ and $\beta = \gamma = 1$, i.e. only emphasis on the setup time and tardiness.
- $\alpha = \beta = 0$ and $\gamma = 1$, i.e. only emphasis on the tardiness.

Although the HGA is capable of handling job specific due dates, in this scenario the due dates for all jobs are equal. Therefore, the latter is similar to the common $C_{\max}$ optimization criterion.

Results

The HGA is coded in C# 6.0 and run on a computer with an Intel Core i7 processor running at 2.60 GHz and 8 GB of RAM. The performance of the HGA depends largely on the setting of certain parameters. In this case, the population size P_{size} is set to 20, the crossover probability P_{cross} is 0.75, the activated probability for the mutation operator P_{mut} is 0.25, and local search is activated with a probability P_{ls} of 0.75.

It is well-known that genetic algorithms are effective to determine the region in which the global optimum exists. However, they can take relative long time to determine the exact local optimum in this region. Application of local search techniques within a GA can substantially improve

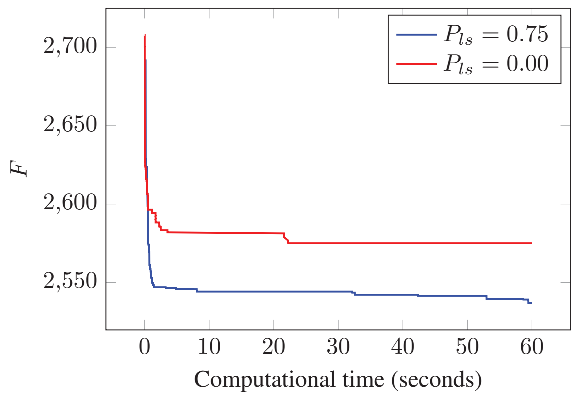


Figure 2: Convergence of the fitness F with (blue) and without (red) the local search operator for data set 3 (i.e. $P_{ls} = 0.75$ and $P_{ls} = 0$ respectively).

its exploiting ability (de Jong 2005). This is demonstrated in Figure 2 where the convergence of the genetic algorithm with and without local search operator (i.e. $P_{ls} = 0.75$ and $P_{ls} = 0$ respectively) is shown for one of the data sets. Similar behavior is observed for the other data sets.

In Table 1 below, the results obtained for all three data sets after one minute are shown. Current and automatic schedules of data set 1 are depicted in Figure 3. A visualisation of the schedules corresponding to data set 2 and 3 are included in the appendix. To gain insight in the variability of the outcome, the HGA is run five times. The computational time required to complete a single run is approximately one minute. The sample mean and standard deviation over these five runs are shown in the utmost right column. The standard deviation on average represents only 0.2% of the sample mean, which indicates that the search produces stable results. For the automatic schedule, the overall fitness function as well as its constituents are shown corresponding to the run yielding the lowest fitness. It can be seen that for each of the three data sets, the fitness significantly improves with respect to the current schedule. This can largely be appointed to the decrease of the process time P. As mentioned, the production speed of the machines varies significantly. Therefore, only through the smarter assignment of jobs to machines a major improvement can be achieved. Additionally, the total setup time and tardiness are significantly improved as well. The former is mainly achieved through the smarter sequencing of jobs and a reduction is highly desired because time spent on setups represents lost production capacity and operator time.

In Figure 3, the current schedule and the ones obtained by varying the weight factors of the fitness function are visualized for the first set of data.

It can immediately be seen that the current schedule (Figure 3a) can be improved on all aspects. The overall fitness as well as all of its constituents are highest compared to the other schedules shown. For the schedule obtained with equal emphasis on all constituents of the fitness function (Figure 3b), especially the production time P and the setup

	Current schedule (Time units)					
	P	S	T	F		
Data set 1	2777	218	286	3281		
Data set 2	3043	215	504	3761		
Data set 3	2551	156	272	2980		
	Automatic schedule (Time units)					
	P	S	T	F	$\%\Delta F$	$\overline{F}$
Data set 1	2395	156	245	2796	14	2807±10
Data set 2	2559	151	426	3136	16	3152±10
Data set 3	2244	121	188	2528	15	2533±7

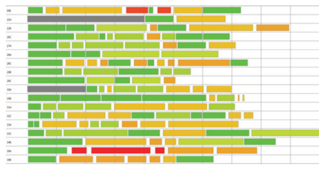
Table 1: For each of the three data sets: the fitness function and its constituents, the proportional difference of the fitness function between the current and (best) automatic schedule, and the sample mean and standard deviation for five consecutive runs of the HGA. Recall P is the total process time, S is the total setup time and T is the total tardiness.

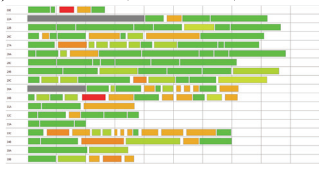
time S are significantly reduced. This is clearly reflected in the images since, respectively, the total area of the schedule is greatly reduced and the schedule appears to be much brighter green. However, downsides of this schedule are that the tardiness is relatively high and that numerous machines are unoccupied for a large period of time. This is due to the large capacity difference of the machines: the HGA has a tendency to utilize the high capacity machines as much as possible, in favor of the production time P and at the expense of the tardiness T. This tendency diminishes if the weight of the production time is set to zero (Figure 3c). As can be seen the load is more balanced, i.e. the utilization of the low capacity machines is increased. Overall this results in a higher production time (still lower than the current schedule) and a much lower tardiness. At last, the schedule obtained with only the tardiness represented in the fitness function is shown (Figure 3d). As mentioned earlier, this situation is very similar to the widely studied $C_{\max}$ optimization criterion. Although the tardiness is very low, the production and setup time are much higher and approach the current schedule, both of which are highly undesired in practice.

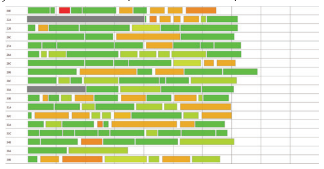
Conclusions and Recommendations

It has been demonstrated that the proposed HGA produces stable high-quality solutions within reasonable computational time. A comparison between several current schedules and ones proposed by the HGA indicates that the potential of increased scheduling automation at Nexperia is high. Furthermore, it has been shown that the widely studied optimization criterion $C_{\max}$ is not suitable for the complex manufacturing environment studied here. As an alternative, a user-controllable multi-objective fitness function has been introduced. Through this approach the output can be tuned towards more practically feasible production schedules.

The parameters of the proposed HGA are set through manual experimentation. The performance may be further enhanced through systematic calibration of key parameters, e.g. by means of a Design of Experiments approach. It would also be interesting to rigorously examine the quality of the solutions generated by the proposed HGA by making a nu-


(a) $P = 2777$, $S = 218$, $T = 286$


(b) $P = 2407$, $S = 165$, $T = 241$


(c) $P = 2651$, $S = 164$, $T = 57$

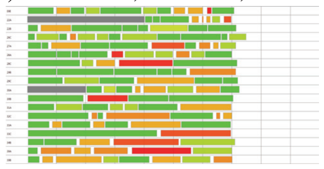

(d) $P = 2724$, $S = 190$, $T = 39$

Figure 3: Visualisation of (a) current schedule (b) auto schedule with $\alpha = \beta = \gamma = 1$ (c) auto schedule with $\alpha = 0$ and $\beta = \gamma = 1$ (d) auto schedule with $\alpha = \beta = 0$ and $\gamma = 1$ for data set 1. Colors range between green and red. Bright green indicates that the setup time prior to the job is relatively small and sharp red indicates it is relatively large. Gray blocks indicate scheduled machine down-time. Recall P is the total process time, S is the total setup time and T is the total tardiness.

merical comparison with a MILP benchmark and methods reported in (Vallada and Ruiz 2011; Avalos-Rosales, Angel-Bello, and Alvarez 2015). Additionally, several important practical aspects have still been neglected. For example, in practice the number of setups that can be performed simultaneously is limited due to the constrained number of available technicians. Also, in the manufacturing situation considered there is a large variation in machine capacity. Consequently, setups on high capacity machines are more "expensive", but this aspect is neglected by treating the setup time at each machine as equally important to the fitness. However, this aspect can simply be incorporated in the current framework by considering "weighted" setup times relative to the machine capacity.

References

Allahverdi, A. 2015. The third comprehensive survey on scheduling problems with setup times/costs. *European Journal of Operational Research* 246(2):345–378.

Avalos-Rosales, O.; Angel-Bello, F.; and Alvarez, A. 2015. Efficient metaheuristic algorithm and re-formulations for the unrelated parallel machine scheduling problem with sequence and machine-dependent setup times. *The International Journal of Advanced Manufacturing Technology* 76(9):1705–1718.

Barnes, J. W., and Laguna, M. 1993. Solving the multiple-machine weighted flow time problem using tabu search. *IIE Transactions* 25:121–128.

Cheng, T.; Ding, Q.; and Lin, B. 2004. A concise survey of scheduling with time-dependent processing times. *European Journal of Operational Research* 152(1):1–13.

de Jong, K. 2005. Genetic algorithms: a 30 year perspective. In Booker, L.; Forrest, S.; Mitchell, M.; and Riolo, R., eds., *Perspectives on Adaption in Natural and Artifical Systems*, volume 1. New York: Oxford University Press.

Fisher, R. A., and Yates, F. 1948. *Statistical Tables for Biological, Agricultural and Medical Research*. London: Oliver & Boyd, 3rd edition.

McNaughton, R. 1959. Scheduling with deadlines and loss functions. *Management Science* 6:1–12.

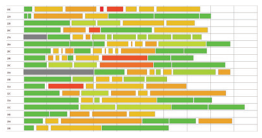
Pinedo, M. L. 1995. *Scheduling: Theory, Algorithms, and Systems*. Upper Saddle River: Prentice Hall, 2nd edition.

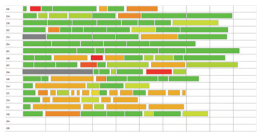
Vallada, E., and Ruiz, R. 2011. A genetic algorithm for the unrelated parallel machine scheduling problem with sequence dependent setup times. *European Journal of Operational Research* 211:612–622.

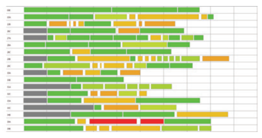
Ying, K.; Lee, Z.; and Lin, S. 2012. Makespan minimization for scheduling unrelated parallel machines with setup times. *Journal of Intelligent Manufacturing* 23(5):1795–1803.

Appendix

This appendix includes the visualisation of the other data sets, see Figure 4 below.


(a)


(b)


(c)

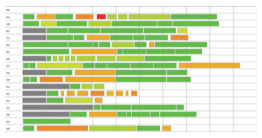

(d)

Figure 4: Visualisation of (a) current schedule for data set 2 ($P = 3043$, $S = 215$, $T = 504$), (b) auto schedule with $\alpha = \beta = \gamma = 1$ for data set 2 ($P = 2662$, $S = 159$, $T = 352$), (c) current schedule for data set 3 ($P = 2551$, $S = 156$, $T = 272$), (d) auto schedule with $\alpha = \beta = \gamma = 1$ for data set 3 ($P = 2238$, $S = 123$, $T = 176$).