Minimisation of Energy Consumption Variance for Multi-Process Manufacturing Lines Through Genetic Algorithm Manipulation of Production Schedule

C. Duerden, L.-K. Shark, G. Hall, J. Howe

Abstract – **Typical manufacturing scheduling algorithms do not consider the energy consumption of each job, or its variance, when they generate a production schedule. This can become problematic for manufacturers when local infrastructure has limited energy distribution capabilities. In this paper, a genetic algorithm based schedule modification algorithm is presented. By referencing energy consumption models for each job, adjustments are made to the original schedule so that it produces a minimal variance in the total energy consumption in a multi-process manufacturing production line, all while operating within the constraints of the manufacturing line and individual processes. Empirical results show a significant reduction in energy consumption variance can be achieved on schedules containing multiple concurrent jobs.**

Index terms – **Energy consumption optimisation, Genetic algorithms, Peak energy, Schedule optimisation**

I. INTRODUCTION

Scheduling manufacturing jobs and ensuring that they operate within the capabilities of the manufacturing production line is fundamental in mass production and high volume manufacturing. The goal of a traditional scheduling algorithm [1] is to allocate limited machinery and equipment to manufacturing jobs without taking into account how this will affect the energy consumption at the production line level, and how it will vary as the schedule is being executed. This can potentially limit the availability of a manufacturing production line as the infrastructure can only deliver a certain amount of energy at any given time. Ideally, for manufacturers an optimal schedule is one which completes all required jobs in the least amount of time and consumes the minimal amount of energy at any given instance, thus increasing the productivity and availability of the production line whilst reducing costs. However this is hindered by many scheduling problems being NP-hard and therefore cannot be practically optimised by schedulers that are based on polynomial-time algorithms. Despite this, work has been undertaken to include additional objectives for traditional schedulers to work towards. Fang et al and Pechmann et al. both present methodologies for production schedules which aim to also minimise peak energy consumption [2]-[4]. As does the commercial scheduling software E-PPS by Transfact [5]. While the work of Fang et al and Pechmann et al. shows promising results, it is concluded by Fang et al that finding the optimal schedule is difficult due to the complexity of the problem and its NP-hard nature. Electrical energy is undoubtedly one of the most valuable resources available to manufacturers. In 2012 UK industry consumed approximately 97.82 TWh (8411 ktoe) [6]. Recently there have been numerous works on intelligent scheduling which aims to reduce manufacturing energy consumption by reducing the idling times of machines by putting them into energy saving modes or shut them down entirely [7]-[9]. The main purpose of leaving a machine idling is to ensure it is ready to run when the next job arrives. In their proposed systems, by intelligently deciding when to shut down a machine or put it into an energy saving mode, the total energy consumption for the production line can be reduced. While all these show promising results, the problem with generating energy optimised schedules has received little attention and appears to be plagued by its NP-hard nature. The use of artificial intelligence in the generation of manufacturing schedules has shown some promising results. Genetic algorithms appear to be a popular choice for solving scheduling optimisation which can include multi-objective [10], [11] and multi-project [12] problems.

Although there have been investigations into multi-objective scheduling algorithms which produce a valid schedule while aiming to reduce the peak, or variance in the energy consumption [10], the optimal solution is difficult to find when using traditional methods and algorithms due to a very large search space. While artificial intelligence has been shown to be capable of solving schedule optimisation problems in an efficient manner [10]-[12], it is noted that the schedule optimisation systems are designed to perform the entire scheduling process. This may present manufactures with a disadvantage if a set of jobs needs to be completed as quickly as possible with no concern for energy consumption. An example of which is a product order with a short lead time. To the authors knowledge it is unknown how well

This work is financially supported by BAE Systems (Operations) Ltd and the Engineering and Physical Sciences Research Council (EPSRC) as part of a CASE studentship.

C. Duerden and G. Hall are with the Advanced Digital Manufacturing Research Centre, University of Central Lancashire, Burnley Campus, Burnley, Lancashire, BB12 0EQ, UK (phone: 01772 896093; email: cjduerden@uclan.ac.uk, ghall5@uclan.ac.uk).

L.-K. Shark is head of the Advanced Digital Manufacturing Research Centre, University of Central Lancashire, Preston Campus, Preston, Lancashire, PR1 2HE, UK (phone: 01772 893253; email: lshark@uclan.ac.uk).

J. Howe is the Director of the Thornton Energy Institute, University of Chester, Thornton Science Park, Pool Lane, Ince, Cheshire, CH2 4NU (phone: 01244 513956; email: j.howe@chester.ac.uk).

these systems can be adjusted to produce a schedule suited to the manufacturers changing preferences.

Promising work is shown by Bruzzone et al who aims to reduce peak power consumption by implementing a form of energy aware scheduling embedded into an advanced planning and scheduling system [13]. The manufacturer is able to specify a maximum allowable peak power consumption and the energy aware scheduler modifies the original schedule to accommodate the manufactures restrictions. While producing encouraging results, the optimiser references fixed average power values for each job and as such, does not account for how the energy consumption changes as the job is executed. Additionally it is allowed to increase the tardiness. While tardiness minimisation is one of the optimiser's objectives, the manufacturers may prefer that it not be increased beyond that what is specified by the original scheduler.

While the research discussed above focuses on schedule optimisation from an infrastructure viewpoint, similar research has also been conducted for energy demand-side management. This focuses on modifying the production schedule so that the variation in production line energy consumption relates to the changing cost of energy. The work of Shrouf et al. proves that through the use of a genetic algorithm the production schedule can be modified in such a way that it results in significant savings in production costs [14]. Like Bruzzone et al. however, the algorithm references fixed energy consumptions during job processing. Very similar work has also been applied for matching production line energy consumption to the varying generation rate of renewable energy sources [15].

This paper presents a genetic algorithm based schedule optimisation system which modifies the timings of a schedule produced by a traditional scheduling algorithm, in order to minimise the variance in production line energy consumption without exceeding the overall production deadline [16]. The technique used is inspired by load-shifting, a traditional energy optimisation method in which energy intensive jobs are scheduled to run during times of low energy tariffs [17]. Following the methodology described in section II, experiments and results are presented in section III to demonstrate the level of potential reduction in energy consumption variance. Finally the performance, limitations and potential improvements for the current system are discussed in section IV.

II. Methodology

In the proposed system, a schedule for a series of manufacturing jobs is initially generated using traditional scheduling algorithms. This takes in a list of jobs to be processed and produces a schedule which ensures that:

a. All jobs are processed in the correct order;
b. The total makespan for each process and their child jobs do not exceed the process deadline;
c. At no point does the total resource utilisation exceed the resource constraints of the manufacturing production line.

A schedule produced within these constraints will be valid and could be executed on the proposed manufacturing production line, with job order and resource allocation already assigned. The genetic algorithm is then used to optimise that schedule with a goal to minimise the variance in energy consumption. This is achieved by adjusting the start time for each job and referencing job specific empirical energy models to predict the energy variance generated by the proposed schedule.

A. Gene Representation and Genome Generation

As the goal of the genetic algorithm is to optimise when a job starts, the genomes, representing possible schedules, utilise value encoding [18] with the value of each gene representing the start time for a particular job. In order to maintain the sequential order of a schedule the following two rules are specified:

(i) *Processes are independent and can be executed in parallel.*
(ii) *The jobs of a process are order dependent and must be executed sequentially.*

During the generation of the initial population of candidate solutions, to ensure that all genomes comply with these rules and to ensure job order is maintained, the string of genes representing the individual jobs are grouped according to their parent process and job order. A relation seed of equal length to the gene string is then generated based on job order. This consists of a number which increments with every gene and resets back to zero when a process ends. An example of a two process gene string can be seen in Fig. 1. In this example, the relation seed is used by the random number generator to ensure the random numbers comply with the job order.

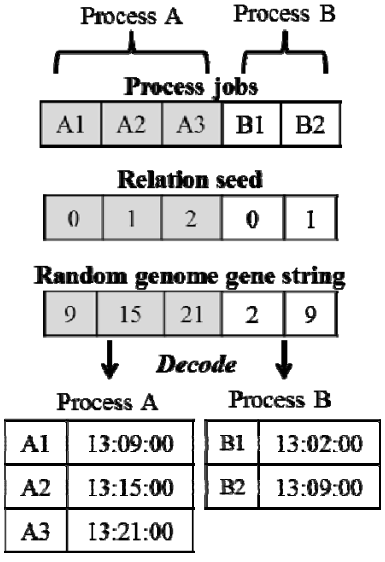

Fig. 1. Example of a relation seed generated based on job order and parent process order. In this example, T = 00:01:00 and s_e = 13:00:00.

Let $R = \{r_0, \ldots, r_{N-1}\}$ denote the relation seed with N denoting the total number of jobs, $S = \{s_0, \ldots, s_{N-1}\}$ the job start time, $G = \{g_0, \ldots, g_{N-1}\}$ the genome representation of S, and $D = \{d_1, \ldots, d_L\}$ the process deadline time with L denoting the total number of processes. To reduce the computation time and to increase the probability of generating a valid schedule, the search of optimum G is

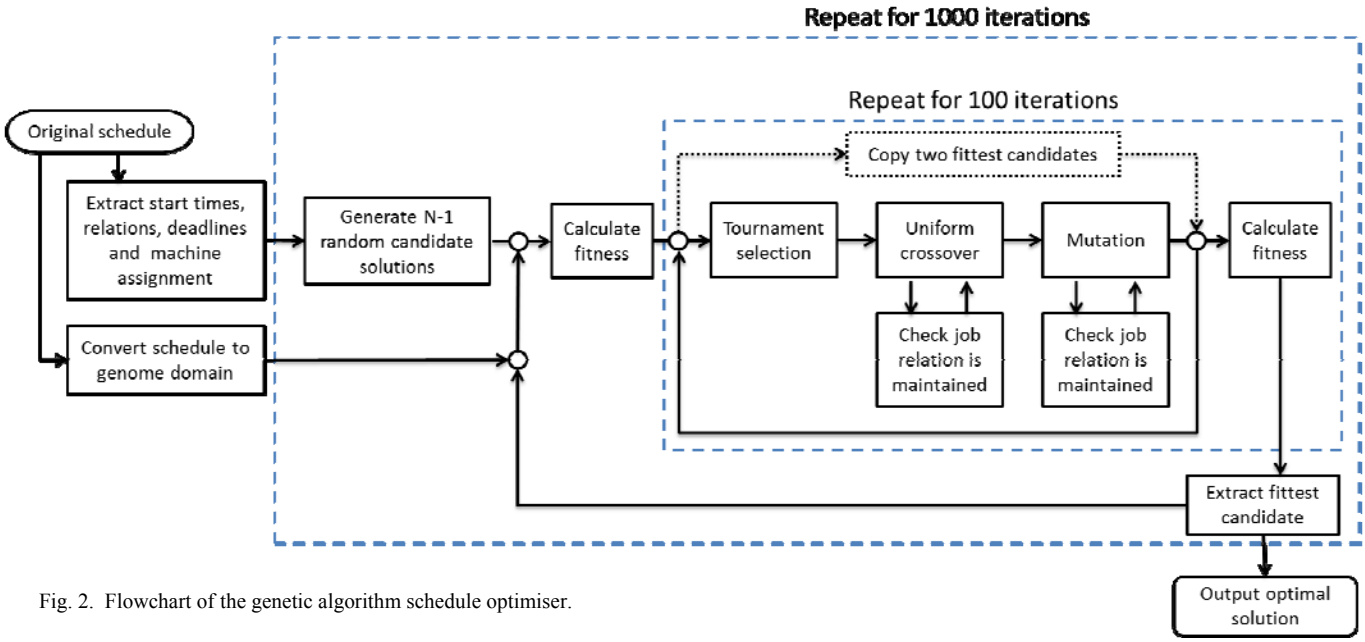

Fig. 2. Flowchart of the genetic algorithm schedule optimiser.

concentrated in a smaller sub-space by limiting the range of random time generation for each job start. If a job is denoted by i and belongs to a process denoted by u with deadline d_u, then the candidate job start time represented is given by (1).

$$s_i = \begin{cases} 0 \le random\ number \le d_u & if\ r_i = 0 \\ s_{i-1} \le random\ number \le d_u & if\ r_i > 0 \end{cases} \quad (1)$$

This ensures that, within a process group, the next randomly generated gene (job start time) will be larger than the previous. It also permits the jobs belonging to different processes to potentially run concurrently as demonstrated in Fig. 1 by jobs A1 and B2 starting at the same time.

In order to convert between genome representation G and a list of job start times S, an encoding / decoding function is used. In the original schedule, the earliest start time s_e is used as a reference point for encoding and decoding job start time s_i to or from gene g_i.

Encoding:

$$g_i = {(s_i - s_e)}/{T} \quad (2)$$

Decoding:

$$s_i = s_e + (g_i \times T) \quad (3)$$

where T is the minimal time shift that can be applied to the job start times.

B. Algorithm Overview

To ensure the algorithm finds a solution as closely to the global optimum as possible, there are two separate loops as can be seen in Fig 2. The inner loop is used to simulate multiple generations of a population within the genetic algorithm. The outer loop is used to repeat the entire genetic algorithm process a predefined number of times. For every iteration of the outer loop, a population P of N_p genomes is generated. The fitness of each genome is calculated before tournament selection, which emulates survival of the fittest while maintaining diversity is used to build a population for reproducing. In tournament selection, a subset of the main population is randomly selected and the fittest genome in that subset is placed in an interim population. This is repeated until the interim population is the same size as the original population. Tournament selection was chosen due to its implementation simplicity, which directly affects computational time. Additionally the selection diversity can be easily altered by adjusting the subset population size. Uniform crossover and mutation is then applied to the interim population. During these operations, there is a potential for the job order to be invalidated. To decrease the likelihood of this occurring additional checks are performed within the operators. After crossover has been applied the first gene in the exchanged section is checked; if it is smaller than the one before it, and they are of the same parent process, its value will be modified according to (4),

$$g_i = (g_{i-1} + C_{i-1} + 1) \quad (4)$$

where C_{i-1} is the makespan of the job whose start time is represented by g_{i-1}. Similarly within the mutation operator, the new randomly generated gene is limited to a range allowed by the job order. It is important to note that these additional checks and constraints do not guarantee that the job order will be maintained and that the potential schedule will not be invalidated by these operations. Post-crossover only a single gene is checked. Altering this gene to fit with the previous gene relation may very well invalidate it relative to the next gene. With the mutation operator, depending on how many times the gene string has been mutated, the potential range for the mutating gene may not be sufficiently large enough to accommodate the associated jobs makespan. These problems could be solved by performing these checks on the entire gene string. While this would increase the probability of a candidate schedule being valid, it would increase the computational time as each gene would need to be checked sequentially. As such the current implementation is seen as an efficient balance between the two.

Different rates of crossover probability were investigated to determine the effect on the final outcome. This is

discussed further in section III. Mutation probability remained fixed at 2%. Once crossover and mutation have been applied the fitness is recalculated. The algorithm will then reiterate with the newly generated population until the inner loop is completed. Parallel to this elitism is used, the two fittest genomes are copied and its clones bypass the crossover and mutation process. This ensures that if the optimal value is generated in an early iteration, their clone will survive unchanged until the end while the original will reproduce to see if a fitter solution can be generated.

Once the inner loop is completed, the fittest genome generated is added to a list of fittest genomes. The outer loop then iterates and a new population is generated. This consists of $N_p - 1$ random genomes generated using the relation seed, and a copy of the best genome from the list of fittest genomes. For the very first iteration, the fittest genome is considered to be a direct encoding of the original schedule. All this is done for several reasons:

a. The original schedule may produce the optimal energy variance already or may contain optimal components which could be extracted during crossover;
b. The optimal genome from a previous iteration may be further optimised;
c. It ensures that a valid genome is always returned from the inner loop.

The entire system is not designed to stop once a particular fitness has been reached. The fitness itself is the predicted energy consumption variance, constructed from the job start times proposed in the genomes gene string and the job energy models. As the minimal variance for a manufacturing schedule cannot be known beforehand, the system is allowed to run for 1000 outer loop iterations. At this point it returns the global fittest genome. The configuration used in these experiments consists of 100 iterations of the inner loop and 1000 iterations of the outer loop. During testing it was found that as the number of jobs increased, the number of outer loop iterations also had to be raised in order to increase the likelihood of a genome fitter than the original being generated. This is due, in part, to the conditions a genome must meet to be classified as a valid schedule.

C. Fitness Function

The fitness function serves two purposes: a) determine if genome G represents a valid schedule that can be executed on the proposed manufacturing line, and if it is valid, b) to decode the gene string, build a predicted energy consumption profile, and calculate the variance. The validity conditions for each genome are as follows:

For a set of job start time genes g_i with a makespan of C_i, belonging to the same parent process:

$$g_i > (g_{i-1} + C_{i-1}) \tag{5}$$

For each job with a parent process deadline d_j:

$$d_j > (g_i + C_i) \tag{6}$$

At any time t, usage on a machine of type M_k, $M_{k,Usage}$ with a maximum availability of $M_{k,X}$:

$$M_{k,Usage}(t) =< M_{k,X} \tag{7}$$

Equation (5) ensures for jobs within the same parent process, the next job does not begin until the previous one is finished. Equation (6) ensures all jobs do not exceed their parent process deadline. Finally, equation (7) ensures that usage of each machine type at any one point in time never exceeds the total amount of that machine. If a particular genome fails any of these conditions it is classified as invalid and is assigned the maximum fitness, in practice this is the maximum value of a double precision number in C#. In this implementation the fittest genome is defined as the one with the minimum fitness value, and therefore minimum energy consumption variance. If a genome meets all conditions, the energy consumption variance for that schedule is calculated by generating a predicted energy consumption profile.

Initially an energy consumption profile is created for each machine m in the production line, E_m. This spans from s_e to D_{Max} with time spacing T and is initially populated with the idling value for that particular machine m_{Idle}. Then in chronological order, for every job j using that machine j_m, their associated job specific energy consumption profile, $j_{Profile}$ is copied to E_m, beginning at the start time denoted by the appropriate gene j_g. This process overwrites the values currently assigned to the associated elements of the profile. Additionally the recorded idling consumption from that particular job j_{Idle} is subsequently assigned to the remainder of the profile. This is because the idling energy consumption could have changed if the machine is now in a different position or configuration due to the previous job. This process is repeated until the energy profiles of all jobs running on that machine have been merged. The process described above is detailed in pseudo-code below.

Input: List of jobs each with a representative gene specifying its start time g_i. The energy consumption profiles and idling energy consumptions for each job $j_{Profile}$ and j_{Idle}. A list of machines and their standard idling energy consumptions, $M \in \{m_1,..,m_X\}$ and $m_{,Idle}$
Output: Energy profile for machine E_m.
For each machine *m*
  $E_M(s_e)$ *to* $(D_{Max}) = M_{,Idle}$
  For every job *j* which uses *m*, in chronological order.
  Copy $j_{Profile}$ *to* $E_{m,}$ *beginning at* $g_{i.}$
  Copy j_{Idle} *to* E_m *from* $(g_i + C_i)$ *to* D_{Max}
  End
End

The system assumes that when not in operation, each machine is left idling. Once the predicted energy consumption profile is generated for each machine, the total predicted energy consumption profile for a total of X machines can be calculated using (8).

$$E_{Total}(t) = \sum_{m=1}^{X} E_m(t) \tag{8}$$

From (8), the variance of the predicted total energy consumption profile is given by:

$$E_{Var} = \frac{T}{D_{Max} - S_e} \sum_{t=s_e}^{D_{Max}} \left[E_{Total} - \overline{E_{Total}} \right]^2 \qquad (9)$$

where $D_{Max} = max\{d_1, ..., d_L\}$ and $\overline{E_{Total}}$ denotes the average energy consumption. Once calculated the variance is assigned as the genomes fitness value.

As can be seen, the algorithm has a single objective to optimise – the energy consumption variance. This is in contrast to many other systems reviewed in the literature which also attempt to minimise production time. This decision was made as the methods for optimising energy consumption variance are directly opposite to the methods for production time minimisation. The former aims to distribute the jobs as widely as possible while the latter wishes to compact the jobs as tightly as possible. The algorithm ensures that the optimised schedule operates within the process start time and the deadline specified by the original scheduler.

III. Experiments and Results

The proposed genetic algorithm was tested with multiple simulated schedules of increasing complexity. These schedules were devised based on real schedules and their job specific energy consumption profiles incorporate waveform characteristics such as inrush currents and transients. Each schedule file contains details of the processes and jobs, the energy profiles for each job, and the idling power consumptions for each machine. A separate file contains data related to the specification of the individual manufacturing lines.

The performance of the proposed genetic algorithm was evaluated by modifying low complexity schedules and comparing against the calculated actual optimal result. This actual optimal result was determined by generating all possible combinations of time steps and selecting the one which produced the lowest energy consumption variance. For N jobs to be completed by D_{Max}, the number of possible gene combinations is D_{max}^{N}. Among the first set of experiments, a schedule containing five jobs with a maximum deadlines of 25 time steps was the most complex schedule considered, where every possible combination was generated with the corresponding energy consumption profile. This proved that the algorithm was successful in finding the optimum solution among 25^5 = 9,765,635 possible solutions.

For schedules where $N \leq 5$, the global optimal solution, which can be calculated using a traditional iterative approach, is in most cases returned after only a few outer loop iterations. However as N increases the amount of outer loop iterations before a proposed optimum is returned diverges. With a schedule of $N = 10$, this has been observed to range from one to 323 iterations.

Furthermore the proposed genetic algorithm was applied to more complex schedules with $N > 5$. This showed that schedules with a large amount of downtime, the optimal solution appears to be generated quicker. This is likely due to the fact that while more downtime increases D_{Max}, it also increases the probability of a proposed schedule being valid in accordance with fitness function condition (6). Additionally, the optimal solution generated by repeatedly running the algorithm ten times with the same schedule is not concise. This is to be expected as genetic algorithms may not find the most optimal solution in the time allotted to them. However the optimal results produced by each only vary slightly. With only a small range of returned values, it can be assumed that the proposed schedule that produced it is a near optimal solution.

Table I
Average number of outer loop iterations until optimal value generated with differing crossover rate and tournament selection size.

		$P_{Crossover}$		
		0.65	0.75	0.85
$N_{Tournament}$	$N_p/8$	161.875	128.5	165
	$N_p/6$	203.25	218.375	242
	$N_p/4$	101.625	123	186.625

In the first set of experiments, different values for crossover probability and tournament selection size were also investigated to determine the optimal values. For $P_{Crossover}$, 0.65, 0.75, and 0.85 were selected. This range was chosen as a probability any higher than 85% would cause too much disruption to a population. This may result in a possible optimal solution being lost before it can be identified. This has been concluded by other authors [19]. A value less than 65% would not allow a population to sufficiently reproduce. For $N_{Tournament}$ values, $N_p/8$, $N_p/6$ and $N_p/4$ were investigated. Each combination was tested ten times to determine the number of outer loop iterations required to return a near optimal solution.

The number of iterations presented in table I demonstrate that the algorithm works most efficiently at low rates of crossover probability with a high tournament selection size.

Further to this, a more in-depth experiment was carried out to determine how the probabilities of the algorithms internal operators affect the final result. The algorithm was repeatedly tested 10 times on a predetermined set of combinations of these probabilities. The order of combination was that $P_{Mutation}$ increased first. Increasing from 0.01 to 0.1 in 0.01 intervals, and then increasing to 1 in 0.1 intervals. Following this, $N_{Tournament}$ increased in $N_p/2$ steps from $N_p/2$ to $N_p/8$. Finally $P_{Crossover}$ would increase in 0.05 steps from 0.5 to 1. In these experiments, the performance measure was the global optimal variance (GOV) found after 1000 outer loop iterations. The experiment was conducted on two schedules containing N=8 and N=50 jobs. In both cases the average GOV was calculated for each combination and this was used to determine the efficiency of that combination.

The results, as seen in Fig. 3, show that the rate of crossover has very little influence on the overall performance with a very similar pattern occurring over all values of $P_{Crossover}$. Conversely, there was no constant pattern formed as the values of $N_{Tournament}$ were changed. This indicates that the tournament selection size also has little influence on the performance, however its influence may be combined or enhanced by the values of the final

variable. Ultimately the variable that has the most influence is $P_{Mutation}$, however the results from the two schedules are quite different from one another. For the N=8 schedule, between the values of 0.01 to 0.5 the GOV gradually decreases with an approximate low point of 70 between 0.3 and 0.5. As $P_{Mutation}$ continues to increase, the GOV begins to sharply increase until $P_{Crossover}$ is updated and the pattern restarts, at which point the GOV has increase to ~90. For the N=50 schedule, the changes are significantly more immediate and wide spread. A low point is consistently found within a $P_{Mutation}$ range of 0.01 to 0.1, where the GOV is between 50 and 30. Similar to the previous schedule, the GOV begins an exponential increase once it has left this range. However, unlike the previous schedule the GOV quickly reaches the maximum possible variance. In this schedule this is the variance of the original none-optimised schedule. As such between the $P_{Mutation}$ range of 0.4 to 1 the algorithm fails to locate a schedule better than the original throughout its entire run. This demonstrates that the probabilities of the internal operators do not only just govern the quality of the algorithm, but can directly affect its usefulness.

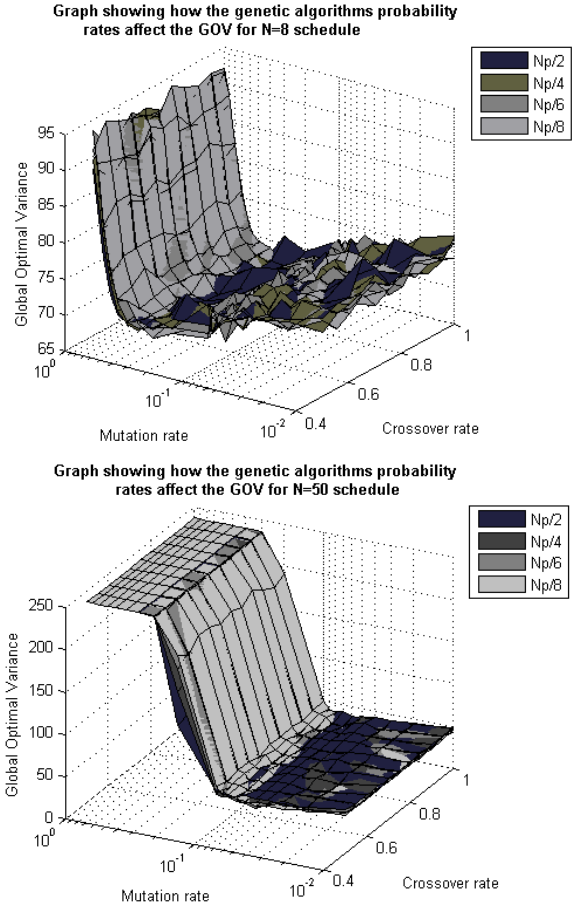

Fig 3. - Graphs showing how the probabilities of the genetic algorithms operators affect the GOV for N=12 and N=50 schedules.

Ultimately values of 0.5, 4 and 0.1 were selected for $P_{Crossover}$, $N_{Tournament}$ and $P_{Mutation}$ respectively. This combination was located inside the 'low points' of both schedule experiment results, however some sacrifice in performance was required to ensure universal compatibility with the two schedules. Unfortunately without significant experimentation there is no way to confirm that this combination is universally located within a near-optimal region of the operators probability space. Furthermore, it cannot be confirmed that the algorithm with this combination will return a better solution than the original for every possible schedule.

Table II demonstrates the reduction in variance that can be achieved, compared to the original energy consumption variance. Fig. 4 shows the effect on the energy consumption profile as an example with the associated original and optimised schedules shown in Fig. 5. It can be seen in Fig. 4 that by redistributing the jobs the energy consumption profile can alter significantly. This can potentially result in a large percentage reduction in the overall variance with only slight changes to the actual schedule. However, this level of reduction will be dependent on the optimisation of the original schedule, the available downtime, and the individual job specific energy consumption profiles. If for example, a schedule consists of jobs whose energy consumption changes very little over time, and there is very little downtime before the deadline, it may not be possible to optimise this schedule significantly.

Table II
Comparison of energy consumption variance in the original and optimised schedules

N	Energy variance in original schedule	Energy variance in optimised schedule	Reduction %
8	143.958	52.511	63.523
10	215.111	67.185	68.767
12	237.396	69.090	70.897
15	151.928	72.077	52.558
20	76.960	26.530	65.528
30	144.673	36.395	74.843
50	236.233	42.464	82.025

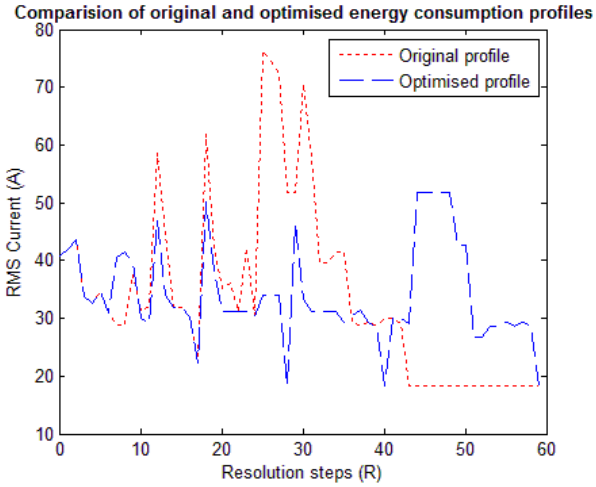

Fig 4. Comparison of energy consumption profiles produced by the original and optimised schedules. N_s = 12. Original variance = 237.396, optimised variance = 69.09

In Fig. 5 it can be seen that the optimisation algorithm distributes the jobs more evenly throughout the entire time available. Ideally if sufficient time was available, the algorithm would attempt to organise the jobs in such a way that only one job is ever being executed at any one time. Such a schedule would guarantee the lowest achievable variance. In reality however this is not possible given the competitive nature of the manufacturing industry. Additionally the algorithm also naturally attempts to prevent

any two jobs from starting at the same time. As the simulated job energy profiles are designed to represent the energy consumption of actual manufacturing operations, they typically contain an initial inrush current. These can result in large peaks in the energy consumption if two or more occur simultaneously. By preventing jobs from starting at the same time, the likelihood of this occurring is reduced.

IV. Performance and Limitations

The overall performance of the entire system is heavily influenced by the schedule it is given to optimise. As discussed in section III, depending on the schedule, the best solution found can range from the potential global optimum to no progress being made at all. This is related to the value of the internal operator's probabilities and there is currently no known method for determining if the algorithm, with these values, is capable of improving all schedules. However, during testing the algorithm was able to successfully reduce the consumption variance of a number of different schedules in its current configuration.

In addition to this the runtime of the system is directly affected by the number of jobs in the schedule. A larger number of jobs results in a larger gene string. This means more time is spent generating the random candidates (see Fig. 2), it takes longer to perform the mutation operator as each gene is checked in sequence, and it takes longer to perform the schedule validity checks in the fitness function. As such the execution time for these individual operations can be considered $O(N)$. However this cannot be consistently applied to the entire system as certain operations, such as uniform crossover are not influenced by the length of the gene string.

A significantly more influential factor in the runtime is the temporal granularity of the job specific energy profiles. As detailed in section II the fitness function is responsible for constructing the predicted production line energy profile from the individual job energy profiles. While there is very little data processing in this part there is a large amount of data manipulation as data is copied and moved around to construct the production line energy profile. In order to accurately represent the energy consumption for each job, the data must be recorded at a suitably high rate. For schedules which last many hours this can potentially result in very large datasets. This, combined with the fact that this operation must be repeated several thousand times means that the energy profiles granularity holds the primary influence over the overall systems runtime.

During initial development and testing, the temporal granularity of the energy profiles was set to T. This was supported by the relatively slow reporting rates of industry capable energy and power monitoring devices. At this level the processing time for assembling the production line profile was minimal. While the issues surrounding the poor reporting rates of many energy monitoring systems is not considered within the scope of this paper, the system was tested with simulated profiles with a 4000% increase in granularity. As expected, this results in a significant increase in the runtime. On a 3.3GHz Intel i5 machine with 8GB RAM, runtime increased from approximately 4 minutes to 5 hours. Although it should be noted that these experiments were performed with the system set to perform 1000 outer loop iterations. As demonstrated in table I, this number can be significantly reduced. One of the main current areas of

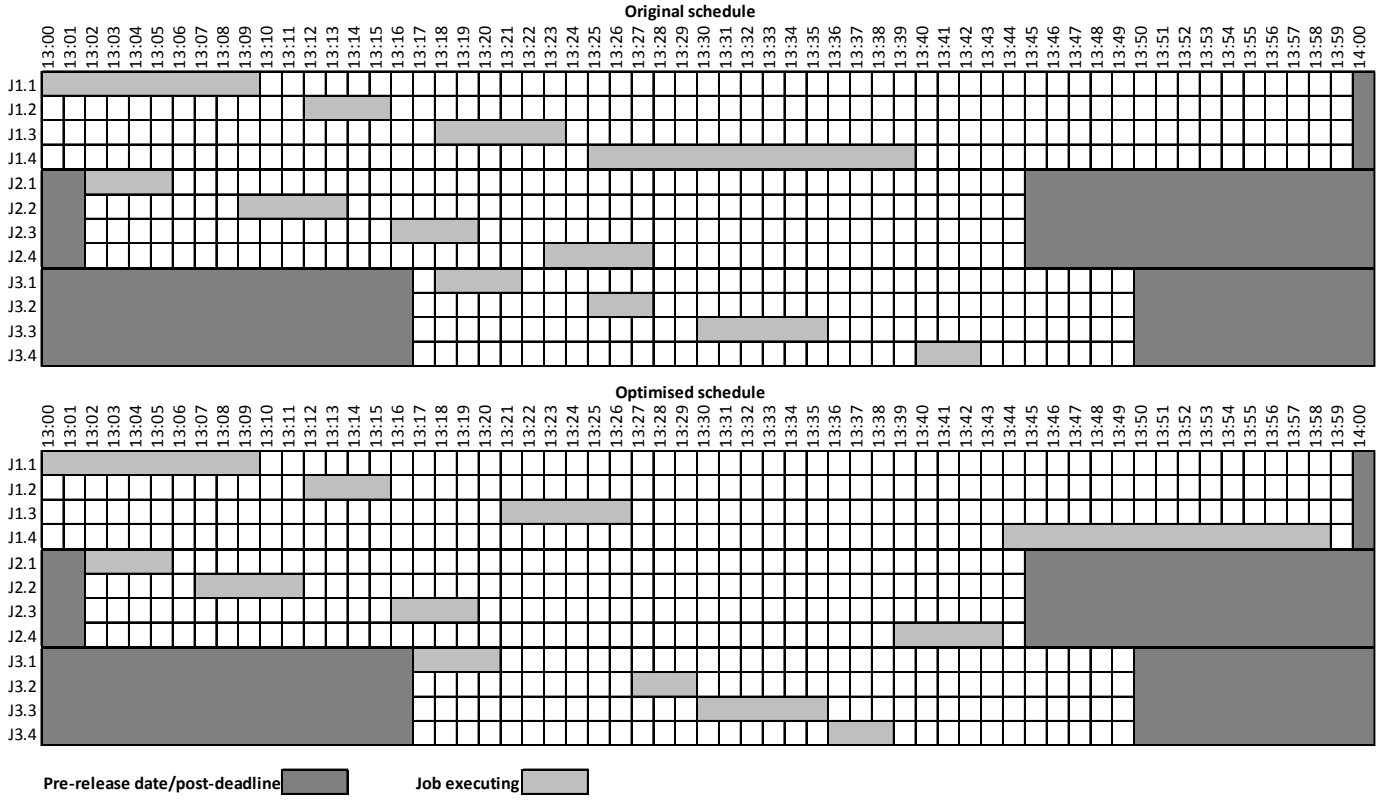

Fig 5. Comparison of the original and optimised 12 jobs 3 process schedule

focus is to reduce the overall runtime through increasing the efficiency of the actual software.

The system described in this paper is still under development and being improved with expanded capabilities. While it is able to reduce the energy consumption variance by a significant percentage it does currently have a number of limitations, listed below, which can restrict its effectiveness in a real-world situation.

1. Currently the algorithm references fixed energy profiles for each job and uses these to construct a production line energy profile. While these models may be accurate at the time of recording, factors such as the age of the machine, tool sharpness, machine uptime, etc. will have a definite effect on the profile. As such the actual production line energy consumption profile may be very different to the predicted one. This could potentially be solved by modelling how the machines status affects job specific energy profiles, or by selecting the most appropriate profile from a historical library based on the predicted machine status. With low cost computer storage and the emergence of Big Data, manufacturers will likely retain historical energy data, making the latter solution more promising.
2. As part of the validity checks, the fitness function ensures that the candidate schedule does not over-utilise the machinery available. While a candidate schedule may pass this check, jobs may have to be allocated to different machines of the same type. In the systems current configuration, the algorithm does not report any reassignments, nor is there any candid way to prevent this should it be undesirable.
3. Finally while the system accounts for the common constraints in which a manufacturing schedule must operate within, it does not consider the storage limitations or preferences of the manufacturer. This means that while executing an optimised schedule, the amount of sub-assemblies requiring storage and/or custom storage jigs may exceed availability.

V. CONCLUSION

This paper presents a methodology for modifying a manufacturing production schedule with a goal to minimise the variance in the production line energy consumption. The use of a genetic algorithm for individual job start time manipulation is detailed and the value of the algorithms internal operator probabilities are evaluated and optimised based on experimental data. For a series of scheduled processes, the algorithm is found to be successful in reducing the energy consumption variance to its near minimum, while ensuring that manufacturing resource limitations and process deadlines are never exceeded. While the global optimum solution is not guaranteed to be returned, a solution near the global optimum is always produced. For each potential solution, a predicted energy consumption profile is generated based on job specific energy models. The variance of the predicted energy profile is then calculated. At the end of the algorithm, the solution which holds the minimal variance is concluded to be the most optimal. The accuracy of the system is entirely dependent on the accuracy and resolution of the energy models and the minimal time shift that can be applied to each job, *T*. Usage wear and periodic maintenance on the machines can result in the job energy profiles changing over time which holds the potential to increase the error in the predicted production line energy profile. Part of the future work aims to improve this through dynamic profile selection based on the current machines status. For the work presented in this paper, mock energy models based on real job consumption profiles were utilised. For increased accuracy and granularity in real implementations, it is recommended that the resolution of the models be significantly higher than *T*. Unfortunately in testing, this resulted in a significant increase in system runtime. As mentioned before, the future work aims to improve this. However, results have shown that with suitably accurate models a significant reduction in energy consumption variance can be achieved regardless of the amount of jobs in the schedule. Through empirical testing an average reduction percentage of approximately 70% has been achieved with the schedules tested. It is also seen that the reduction percentage is independent of the schedule job count with a range between 8 to 50 jobs tested. Analysis of the experimentation results show that the potential variance reduction is based on several factors including the amount of down time in the schedule and the landscapes of the individual energy profiles. It may be possible to predict the potential amount of variance reduction ahead of time based on these factors. In a real implementation this may allow a manufacturer to preview the results before committing to a potentially timely optimisation process, saving time if the previewed results do not pass a predetermined threshold.

If the methodologies described in this paper were implemented in a real manufacturing production line with struggling power distribution capabilities, the reduced variance could potentially allow for another process, with a suitable optimised energy consumption variance, to run without needing to reinforce the local infrastructure. These methodologies could also allow a manufacturing production line to be powered entirely from limited supply resources, such as renewable energy sources.

ACKNOWLEDGEMENT

The authors would like to thank the industrial supervisors Damian Adams and Stuart Barker of BAE Systems (Operations) Ltd. Their insights and dedication have proven invaluable to the success of this work.

REFERENCES

[1] D. Karger, C. Stein, J. Wein, "Scheduling Algorithms," *Algorithms and Theory of Computation Handbook*, 2nd ed. Florida, USA: Chapman & Hall/CRC, 2009, pp 20-1 – 20-34.

[2] K. Fang, N. Uhan, F. Zhao, J.W. Sutherland, "Flow Shop Scheduling with Peak Power Consumption Constraints," *Annals of Operations Research*, vol. 206, issue 1, pp 115 – 145, Jul. 2013

[3] A. Pechmann, I. Schöler, "Optimizing Energy Costs by Intelligent Production Scheduling," *Glocalized Solutions for Sustainability in Manufacturing*, pp 293 - 298, May 2011

[4] K. Fang, N. Uhan, F. Zhao, J.W. Sutherland, "A New Shop Scheduling Approach in support of Sustainable Manufacturing," *Glocalized Solutions for Sustainability in Manufacturing*, pp 305 – 310, May 2011

[5] Transfact. Energy in Production Planning and Scheduling (E-PPS) [Online]. Available: http://www.transfact.de/EN/

[6] Department of Energy & Climate Change, "Energy Consumption in the UK – Industrial Factsheet," ch. 4, pp 3 [Online]. Jul. 2013 Available: https://www.gov.uk/

[7] G. Mouzon, M.B. Yildirim, J. Twomey, "Operational Methods for Minimization of Energy Consumption of Manufacturing Equipment," *International Journal of Production Research,* vol. 45, issue 18-19, Dec 2010

[8] P. Eberspächera, A. Verla, "Realizing Energy Reduction of Machine Tools Through a Control-integrated Consumption Graph-based Optimization Method," *Forty Sixth CIRP Conference on Manufacturing Systems 2013,* vol. 7, pp 640 – 645, 2013

[9] G.D. Orio, G. Candido, J. Barata, J.L. Bittencourt, R. Bonefeld, "Energy Efficiency in Machine Tools – A Self-Learning Approach," *2013 IEEE International Conference on Systems, Man, and Cybernetics*, pp 4878 – 4883, Oct 2013

[10] M.B. Yildirim, G. Mouzon, "Single-Machine Sustainable Production Planning to Minimize Total Energy Consumption and Total Completion Time Using a Multiple Objective Genetic Algorithm," *IEEE Transactions on Engineering Management*, vol. 59, issue 4, pp. 585 – 597, Nov. 2012

[11] S. Yassa, J. Sublime, R. Chelouah, H. Kadima, G. -S. Jo, B. Granado, "A Genetic Algorithm for Multi-Objective Optimisation in Workflow Scheduling with Hard Constraints," *International Journal of Metaheuristics,* vol. 2, no. 4, pp. 415 – 433, 2013

[12] J.F. Gonçalves, J.J.M. Mendes, M.G.C. Resende, "A Genetic Algorithm for the Resource Constrained Multi-Project Scheduling Problem," *European Journal of Operational Research*, vol. 189, issue 3, pp 1171 – 1190, Sep. 2008

[13] A.A.G. Bruzzone, D. Anghinoff, M. Paolucci, F. Tonello, "Energy-Aware Scheduling for Improving Manufacturing Process Sustainability: A Mathematical Model for Flexible Flow Shops," *CIRP Annals – Manufacturing Technology*, vol. 61, issue 1, pp 459 – 462, 2012

[14] F. Shrouf, J. Ordieres-Mere, A. Garcia-Sanches, M. Ortega-Mier, "Optimizing the Production Scheduling of a Single Machine to Minimize Total Energy Consumption Costs" *Journal of Clearner Production,* vol. 67 pp. 197 – 207, Mar. 2014

[15] S. Emec, M. Kuschke, M. Chemintz, K. Strunz, "Potential for Demand Side Management in Automotive Manufacturing," *4th IEEE PES Innovative Smart Grid Technologies Europe*, pp. 1 – 5, Oct. 2013

[16] C. Duerden, L.-K. Shark, G. Hall, J. Howe, "Genetic Algorithm based Modification of Production Schedule for Variance Minimisation of Energy Consumption," *Lecture Notes in Engineering and Computer Science: Proceedings of the World Congress on Engineering and Computer Science 2014,* WCECS 2014, 22 – 24 October, 2014, San Francisco, USA, pp 370 – 375.

[17] N. Brown, R. Greenough, K. Vikhorev, S. Khattak, "Precursors to using Energy Data as a Manufacturing Process Variable," *6th IEEE International Conference on Digital Ecosystems Technologies*, pp 1 – 6, Jun. 2012

[18] F. Herrera, M. Lozano, J.L. Verdegay, "Tackling Real-Coded Genetic Algorithms: Operators and Tools for Behavioural Analysis," *Artificial Intelligence Review*, vol. 12, issue 4, pp. 265 – 319, Aug. 1998

[19] W.-Y. Lin, W.-Y. Lee, T.-P. Hong, "Adapting Crossover and Mutation Rates in Genetic Algorithms," *Journal of Information Science and Engineering*, vol. 19, pp. 889 – 903, Jan. 2003