Transfer batch scheduling using genetic algorithms

George Christopher Vosniakos, Vasilis Millas

Transfer batch scheduling using genetic algorithms

Journal:	*International Journal of Production Research*
Manuscript ID:	TPRS-2005-IJPR-0006.R4
Manuscript Type:	Original Manuscript
Date Submitted by the Author:	19-Jul-2006
Complete List of Authors:	Vosniakos, George; National Technical University of Athens, Mechanical Engineering Millas, Vasilis; NTUA, Mech.Eng.
Keywords:	GENETIC ALGORITHMS, BATCH SCHEDULING, HEURISTICS
Keywords (user):	

Transfer batch scheduling using genetic algorithms.

V. S. Millas and G.-C. Vosniakos[1]

Manufacturing Technology Division,

School of Mechanical Engineering,

National Technical University of Athens,

Heroon Polytehneiou 9, GR-15780, Athens, Greece.

tel. +30 210 772 1457

fax +30 210 772 1197

e-mail : vosniak@central.ntua.gr

[1] Corresponding author

Abstract

In this paper scheduling in a manufacturing system with transfer batches is examined. Transfer batches are considered as different batches although they stem from the same job. Genetic Algorithms determine the size of the transfer batches for each job and the final schedule with a makespan criterion. A novelty of the Genetic Algorithm developed is the twin chromosome encoding, the first chromosome representing the relative size (participation ratio) of each transfer batch with respect to the whole batch and the second chromosome applying in effect a dynamic heuristic dispatching rule representation for resolving operation antagonism. New crossover and mutation operators were employed for the first chromosome and standard operators for the second. The genetic algorithms were coded in C++ for better control. A 20 job X 8 machine shop is used as a test-case. Results favour Genetic Algorithms over heuristic procedures, but the latter close the gap with increase of the number of transfer batches. Design of Experiments was used to focus on the most promising Genetic Algorithm parameter value combinations.

Keywords:

genetic algorithms, heuristics, batch scheduling, transfer batches.

Wordcount :

Main text : 5948 words + figures and tables

1. Introduction

Job-shop scheduling can be regarded as an optimization process by which limited resources, most commonly machines, are allocated over time among operations. A job is characterized by processing time for each of its operations on respective machines, by its due date and by the sequence of the machines on which its operations must be conducted. An operation cannot be interrupted (no pre-emption), each machine can process only one job at a time and that there are no precedence constraints among operations of different jobs. In fact, the notion of jobshop adopted in this work coincides with the classical definition by Baker (1974). The goal of scheduling is the determination of the job (operations) sequence as well as their release time on the appropriate machines, respecting the job constraints. Schedule quality is often judged by makespan.(Sule 1996).

This paper examines primarily genetic algorithm (GA) based scheduling and, for comparison, heuristic scheduling. However, even the genetic algorithms themselves make use of heuristic dispatching rules.

Furthermore, the paper focuses on batch-processing job-shops involving transfer batches. The trasfer batch problem in job shops refers to the number and size of the sub-batches into which each batch is split, and, subsequently, to scheduling of these sub-batches, According to a survey by Potts and Kovalvov (2000) on scheduling with batching and a more recent survey by Chang and Chiu (2005) on lot streaming, it turns out that for job shops the transfer batch problem has been tackled in the framework of Integer Programming, Mixed Integer Programming and the shifting Bottleneck heuristic, without use of meta-

heuristics, such as GAs. Wherever GAs were considered in scheduling of batches, batch splitting was not the main issue.

Genetic scheduling flourished in the last decade, see Ponnambalam *et al* (2001a) for a comparison of several approaches using makespan as performance measure, and Cheng *et al* (1999) for a survey of hybrid genetic algorithms for job-shop scheduling. In general, there are several points of view from which research on genetic scheduling of job-shops can be observed.

One point of view concerns the detailed application, namely the variation of the basic job shop scheduling problem. Alternative route choices with schedule revision, as in dynamic scheduling, are reported in Jawahar *et al* (1998). Simultaneous scheduling of machines and automated guided vehicles in flexible manufacturing systems is described in Abdelmaguid *et al* (2004). Earliness / tardiness scheduling including lot sizing and capacity in multi-product production environment is discussed in Ip *et al* (2000) and Keung *et al (2001)*. Fuzzy scheduling, where processing time is supposed to vary, is the focus of Kubota and Fukuda (1999).

A second point of view concerns novelty. In Baek and Yoon (2002) each machine is assigned an appropriate dispatching rule 'in harmony' with the rules used in neighbouring machines, according to the notion of 'derivative contribution feedback'; in the latter an individual rule for a machine takes responsibility for the first-order change of the system performance. In Al-Hakim (2001) a new encoding scheme is proposed, based on an electrical analogue of the job-shop model. In Piramuthu *et al* (2000) information obtained from snapshots of the system at various points in time is used to tailor

the dispatching rule to be used at any instant. In Zhiming and Chunwei (2000) jobs are organised into groups using Group Technology and scheduling of a group is treated like a flow shop scheduling problem. Jordon (1998) refers directly to batch sizing and sequencing, decomposing the problem into two phases, i.e. batching and scheduling, but restricting it to single machine and two-machine flow-shops.

A third point of view examines performance improvement of genetic algorithms by 'tweaking' genetic operators. Esquivel *et al* (2002) investigate multi-re-combinative approaches, i.e. multiple crossovers per parent pair and multiple crossovers on multiple parents. Wang and Brunn (2000) use a simple heuristic rule to ensure solution feasibility and describe selection, sequence-extracting crossover and neighbour-swap mutation. Ponnambalam *et al* (2001b) propose a multi-objective genetic algorithm (makespan, machine idle times and tardiness) with randomly assigned weights to derive the optimal dispatching rules without entrapment in local minima. In a complementary work, the number of generations, the probability of crossover and the probability of mutation are optimised relating to the size of the scheduling problem (Ponnambalam *et al* 2002).

Genetic algorithms have been combined with other techniques into hybrid solutions, too. Dominic *et al* (2004) compare (based on analysis of variance) schedules resulting from genetic algorithms, simulated annealing and hybrid simulated annealing. Dagli and Shierholt (1997) evaluate schedules with a neural network trained using the knowledge of scheduling experts. Mesghouni *et al* (1999) use constraint logic programming to generate a first population and

subsequent multi-criteria decision making to allow for flexibility. Su and Shiue (2003) integrate genetic algorithms to search the space of candidate scheduling situation features and decision trees, generated algorithmically for a given feature subset. Simulation was used in Lee *et al* (1997) to generate empirical results for feeding machine learning. Jahangirian and Conroy (2000) address machine learning scheduling strategies using a simulation technique and a genetic algorithm that drives the learning module. Chen *et al* (2003) propose a genetic algorithm based on a Petri net model in order to find near optimal dispatching rules under specific performance measures and restrictions. Wang and Zheng (2002) replaced the classical mutation operator by the metropolis sample process of simulated annealing with a probabilistic jumping property, to enhance the neighbourhood search, to avoid premature convergence and to bypass choice of the mutation rate.

Based on the above (within the variety of issues that are still open and being researched concerning meta-heuristic methods) a three-fold approach was decided. First, to focus on transfer batches which have received negligible attention to date. Second, due to the problem nature (batch splitting and sub-batch scheduling), to try a twin chromosome GA formulation and compare results to heuristics performance. Third, to enhance GA performance by systematically selecting few alternative combinations of GA parameter values. In what follows, first, a brief introduction on genetic algorithms is given, followed by description of the approach. Next, implementation issues are presented, followed by results and their discussion and conclusions.

2. Genetic Algorithms

A genetic algorithm is an optimization process by which a population of candidate solutions evolves systematically, with the objective of reaching the best solution, by using evolutionary computational processes inspired by genetic variation and natural selection. The basic idea is the survival of the fittest by progressively accepting better solutions to the problem. Genetic algorithms were developed by John Holland at the University of Michigan when trying to abstract and rigorously explain the adaptive processes of natural systems and to design artificial systems software that retains the important mechanisms of natural systems (Holland 1972).

Each solution of the problem is first encoded as a string of symbols called chromosome, and is associated with a measure of adaptation, the fitness, often related to the objective function. Starting from an initial population, new solutions are generated by selecting some parents randomly, but with a probability growing with fitness, and by applying genetic operators such as crossover (an exchange of substrings of the parents chromosome) and mutation (a random perturbation of a chromosome). Some existing solutions are then selected at random and replaced by some of the offspring, to keep a constant population size. The process is repeated until a satisfactory (optimum or near-optimum) solution is found (Goldberg 1989).

3. Job-shop scheduling

3.1 Definitions

A job shop (class I) with 20 jobs and 8 machines has been defined as a test case, without loss of generality. Each job consists of a number of operations,. Each operation is executed on a particular machine and has some processing time (per part), as defined in Table 1. To keep things simple - in accordance with the Baker jobshop definition - machine numbers are given in Table 1 to denote operations. The operations of each job have to be executed in a particular sequence, which is defined in Table 2. The production program is defined by a batch size for each job (number of parts to be processed) and by the corresponding due date, as defined in Table 3. Note that processing times and changeover times are expressed in terms of 'standard' time units (which may be seconds, minutes, hours etc.). Due dates are expressed as time duration with reference to time 0.

Each batch has the same operation processing times for every member part as well as a due date common to all parts. Processing time of the whole batch is proportional to batch size.

Changeover time is allocated to each job (batch), e.g. for tool set changing, tool offset registration, CNC program download, etc., when production switches from job A to job B, or if a job has to be produced on an idle machine. Changeover times are given in an incidence table, see Table 4.

insert Table 1 about here.

insert Table 2 about here.

insert Table 3 about here.

insert Table 4 about here.

When scheduling with transfer batches, the batch is split into sub-batches. The number of sub-batches is denoted by the variable 'transfer'. If transfer=n and the original number of jobs is m, then m*n jobs result, each one corresponding to a transfer batch. Each one of the n new jobs stemming from splitting the same original job has, obviously, the same operations/machine sequence with the latter and the same due dates, but different total processing time, as determined by its batch size compared to the batch size of the 'mother' job. This ratio of batch sizes is termed ‘participation ratio’.

For example, consider three jobs with corresponding batch sizes : Job1:30, Job2:20, Job3:50. Suppose that 3 sub-batches are formed (i.e. transfer=3), these being for Job1:15-7-8, for Job2:4-10-6 and for Job3:25-15-10. The number of jobs, after transfer batch splitting, becomes 9. The three new jobs stemming, for instance, from Job1, named as Job1-1, Job1-2, Job1-3 , will have operation processing times 15, 7 and 8 times the operation processing time defined in Table 1. Participation ratios for the nine new jobs corresponding to the three original jobs are : 0.5, 0.23, 027, 0.2, 0.5, 0.3, 0.5, 0.3, 0.2 respectively.

Next, four different heuristic scheduling rules are considered first, and then, a genetic algorithm for transfer batches is presented as a means of improving the schedule.

3.2 Heuristic formulation

In general, dispatching rules are distinguished into static, where job priority is independent from the state of the schedule, and dynamic, where job priorities may change from one scheduling step to the next according to prevailing conditions. Some dispatching rules may be used both as static and as dynamic, the latter usually resulting in better schedules (Sule 1996). Four different heuristics were tried as follows, see Anderson 1994 for more comprehensive definition amd discussion.

SPT (Shortest Processing Time) heuristic refers to the selection of the job with the least processing time. Since processing times remain the same throughout the scheduling duration, the heuristic is a static one.

EDD (Earliest Due Date) heuristic chooses the job that has the earliest delivery date and forces the schedule to respect those dates. This is again a static rule, since due dates do not change.

MST (Minimum Slack Time) heuristic is more complex than the previous two. Slack Time is defined as the time interval starting at the end of the one but last operation of a job and ending at the due date, provided that the job has not been delayed. If at time point t_o the first k of the total q operations comprising job A have been completed and the operations k+1,k+2,...,q still remain to be executed, having processing times equal to $p_{k+1}^{(A)}$, $p_{k+2}^{(A)}$,...., $p_q^{(A)}$ respectively, and the due date of job A is $d^{(A)}$, then slack time is simply calculated as :

$$Slack\_time = d^{(A)} - t_o - \sum_{i=k+1}^{q} p_i^{(A)}$$

The rationale of the heuristic is that least slack time corresponds to high probability of delay. This is a dynamic heuristic, because slack time changes after each operation allocation.

EOD (Earliest Operation Due Date) heuristic is dynamic, too. According to this heuristic, operations with the earliest due date are promoted. When considering a schedule at some point in time t_o, the due date of an operation k of a job A is the point in time (expressed with respect to t_o) when this operation would be completed if the overall job were completed on its due date and waiting time in the input queue of the respective machine for each operation was proportional to the processing time on this machine. Keeping the same notation as for slack time, operation due date is calculated as:

$$d_k^{(A)} = t_o + (d^{(A)} - t_o) \cdot \frac{\sum_{i=1}^{k} p_i^{(A)}}{\sum_{i=1}^{q} p_i^{(A)}}$$

3.3 GA formulation

Modelling of transfer batch scheduling with genetic algorithms can be considered as a two stage process : first, given the number of transfer batches which applies to all jobs (batches), the size of each transfer batch has to be determined, and, second, start times have to be established for all operations of each transfer batch for constructing a schedule. This logic is coded using two chromosomes.

The first chromosome codes the participation ratios of each transfer batch for all jobs available, as defined in 3.1. For a total of m jobs, the chromosome

(chromo1) will contain m*transfer number of genes, i.e. one for each transfer batch created. Gene chromo1([i+j*transfer]) where i=1:m and j=0:(transfer-1) corresponds to a certain transfer batch which is considered a 'new' job, i.e resulting from splitting of the original jobs, and takes a value in the interval [0,1]. This represents the participation ratio of the corresponding new job within the original (non-split) job. Therefore, $\sum_{j=0}^{transfer-1} chromo1(i + j * transfer) = 1$

where chromo1(i) denotes the value of gene i in chromosome 1.

Referring to the example stated in section 3.1, where transfer=3, chromosome 1 will be as follows :

Gene	1	2	3	4	5	6	7	8	9
Value (phenotype)	0.50	0.20	0.50	0.23	0.50	0.30	0.27	0.30	0.20
Job	1	2	3	1	2	3	1	2	3
Transfer batch	1	1	1	2	2	2	3	3	3

In this way, participation ratios and hence corresponding processing times for each operation corresponding to each transfer batch are suggested by the GA.

The second chromosome works with the new jobs (each corresponding to a transfer batch of size suggested by the first chromosome), expanding them into operations. It implicitly codes the actual schedule, i.e. every operation of each job. Therefore, each chromosome consists of as many genes as the total number of operations of all jobs. Note that each –new- job consists of a number of operations that is different to that of other jobs that do not stem from splitting

of the same original job. By contrast, all new jobs stemming from the same original job consist of exactly the same operations, but have different processing time per operation, because they refer to different numbers of parts (different participation ratio in the original job).

Coding of the second chromosome (chromo2) is based on Dorndorf and Pesch idea, which is well-established now (Dorndorf and Pesch 1995). Gene g of the second chromosome takes an integer value between 1 and the total number of heuristics available, i.e. 4 in this case, denoting the scheduling rule according to which the g-*th* operation in the schedule will be selected. A different set of heuristics is used in this work compared to Dorndorf and Pesch (1995), i.e. SPT, EDD, MST and EOD rules as defined in section 4.1, corresponding to integers 1,2,3 and 4 respectively.

As an example, a possible instance of chromosome 2 for the example given above is the following :

Chromo2=[1 2 3 3 1 1 4 4 2 1 3 3 2 1 2 3. 4 3 …]

For this particular individual, the first operation to be scheduled will be selected from the set of all the first operations of all jobs (transfer batches) according to the SPT rule. To schedule the second operation, all operations that are available for scheduling when the first scheduled operation starts will be considered and the EDD rule will be applied to chose one of them. The third operation will be chosen (from the ones that can be scheduled next) by applying the MST rule etc. If just one operation is available at an instant, this is scheduled next without a need to use dispatching procedures. Each operation to be scheduled has an earliest starting time established by a procedure identical to that of the Giffler

and Thompson (1960) algorithm for generating active schedules (i.e. schedules in which no operation can start earlier without causing delay to another operation which would have started otherwise).

The initial population for the second chromosome is created randomly, using as gene value an integer from 1 to 4 determined with equal probability through the random number generator.

In summary, two populations are considered simultaneously represented by chromosomes chromo1 and chromo2. The individuals of the first chromosome suggest to the individuals of the second chromosome processing times for the transfer batches, through participation ratios. The individuals of the second chromosome try to find the best heuristic rule by which the successive operations are chosen, thereby building the schedule.

In each generation, different crossover and mutation operators and potentially different selection processes are used for each chromosome type in order to form the next generations corresponding to the two chromosomes.

For the first chromosome, a new crossover operator, peculiar to the nature of the problem is considered, namely a random number of jobs is selected for the two parent chromosomes, and the genes corresponding to all transfer batches into which these jobs are split are simply swapped between the parents, see 4.2, too. In this way, the transfer batch size mix is changed every time crossover is applied, keeping at the same time the sum of participation ratios for each job equal to 1. This technique is also favoured in Holland (1972), who argues that in this way the order of good 'schemas' (i.e. the number of genes whose values

need to be kept unaltered in each population change) is preserved, whilst new ones, sufficiently different, are generated, too.

As far as mutation is concerned for the first chromosome, chromo1, a job is again selected at random and new participation ratios for the respective transfer batches are selected at random, in order to expand solution search into combinations that have not been examined.

As far as the second chromosome is concerned, multi-point crossover (MX), see section 4.2, and simple 'move'-mutation were used.

Parent selection was implemented using rank selection in combination with elitism, in order to give a fairer chance of selection to chromosomes with fitness values way below those of the best individuals. Otherwise, e.g. if straightforward roulette wheel selection were implemented, any change in transfer batch participation ratio and in sequence might relatively easily destroy good solutions.

4. Implementation

Both heuristic results and GA results are obtained by running the same GA. Heuristic results corresponding to the SPT, EDD, MST and EOD rules are obtained by creating an initial population with just 1, 2, 3, or 4 respectively as the single value of all genes. The genetic algorithm was coded in C++ in order to have total control on it. Standard GA programming tools, by contrast, make it either impossible at all or, in any case, not easy to use non-standard concepts such as parallel twin chromosome encoding, special crossover and mutation

operators and an improved number generator. Some implementation features are explained below.

4.1 Random number generator

Linear congruential generators, despite being fast, exhibit significant correlation of resulting numbers in repeated use. In addition, looking at the binary form of the resulting integers, the least significant bits are 'less random' than those of highest significance. In this work a random number generator was coded according to the formula $I_{j+1} = aI_j \pmod s$ with α = 7^5 = 16807, s= 2^{31}-1 = 2147483647

To avoid 32-integer overflow for the product of α και (m -1) m is factorised as proposed by Schrage, i.e. $m = a \cdot q + r$ with $q = \lfloor m/a \rfloor$ and $r = m \bmod a$ with q=127773 and r=2836 (Knuth 1981).

Performance of the random number generator was improved further to avoid correlations by shuffling the output, e.g. value j, and using it as input, e.g. in iteration j+32. The seed does not change in the program if all random numbers have to belong to the same distribution.

4.2 Crossover operators

For the first chromosome, a 'swap-type' crossover as presented in principle in 3.3 was implemented. For instance, for 4 jobs and transfer=3, swaping of the first and third job data between the parents :

[0.23 0.34 0.78 0.87 0.66 0.02 0.10 0.11 0.11 0.64 0.12 0.02]

[0.55 0.42 0.21 0.69 0.34 0.32 0.22 0.1 0.11 0.26 0.57 0.21]

will yield as offspring :

[0.55 0.34 0.21 0.87 0.34 0.02 0.22 0.11 0.11 0.64 0.57 0.02]

[0.23 0.42 0.78 0.69 0.66 0.32 0.1 0.1 0.11 0.26 0.12 0.21]

For the second chromosome, the standard crossover operators cycle, uniform, single point and multi-point crossover (Goldberg 1989) have been implemented in the GA-code created, however multi-point crossover was employed. This involves random selection of g genes defining g+1 gene strings. The offspring result by alternate swapping of these strings.

For instance, for g=4 and parents (for 12 operations):

[1 4 | 2 2 | 3 1 1 4 | 3 2 3 | 1]

[3 2 | 4 1 | 2 1 3 2 | 4 2 4 | 3]

possible offsping are :

[1 4 | 4 1 | 3 1 1 4 | 4 2 4 | 1]

[3 2 | 2 2 | 2 1 3 2 | 3 2 3 | 3]

4.3. GA Parameter Definition

The parameters of a GA that were considered most influential are : population size (p_s), crossover probability (p_c) and mutation probability (p_m).

Different values of those parameters normally result in different solutions. However, the exact values that result in optimum solution are usually a matter of trial and error. A more intelligent methodology in determining the near-best

combination of parameter values is Taguchi's Design of Experiments (DoE) (Ross 1996) and this was employed in this work. The three GA parameters are considered as quality factors of the 'experiment', i.e. the execution of the GA.

DoE is run in a series of steps (Ross 1996): stating the problem, stating the objectives of the experiment, selecting the quality characteristics and the measurement systems, selecting the factors that may influence the quality characteristics, selecting levels for the factors, selecting the appropriate Orthogonal Arrays (OAs), selecting the interactions that may influence the quality characteristic, assigning factors to OAs and locating interactions, conducting the experiment runs, analysing the results, and conducting a confirmation experiment.

There are 3 types of OAs dealing with two-level, three-level and mixed level factors. OAs are symbolised with a latin L followed by a number that defines the number of the array's lines, e.g. L4, L8, L16, L32 for OAs with 2 level factors and L9, L18, L27 with 3 level factors.

Each OA type is accompanied by an interaction table and a linear graph that depict the pattern of interaction columns in relation to factor columns. The criteria for chosing the appropriate OA for a DoE are : number of factors and their interactions, number of levels for each factor and desired experiment resolution, which varies from 1 (lowest) to 4 (highest).

Results are acquired for each experiment (OA line). If the influence of each factor needs to be determined, Analysis of Variance (ANOVA) is run.

In this work, the respective value ranges of the three factors are known, namely p_s between 10 and 60, p_c between 0.65 and 0.95 and p_m between 0.005 and

0.15. There is no interaction between p_c και p_m, but there is an inter-relation between p_s και p_c, because these determine the population members that will mate. There is also an interaction between p_s και p_m, because these determine the population members that will undergo mutation. The appropriate orthogonal array is selected, in this case L9, which possesses four columns. The way in which L9 is populated is shown in Table 5.

insert Table 5 about here.

In column 4 both factor and factor interactions are placed, therefore experiment resolution is 2. The final array is shown in Table 6.

insert Table 6 about here.

In this way, just nine experiments are conducted, by contrast to the full factorial experiment involving 27 iterations. No statistical analysis was carried out subsequently on the result obtained by the genetic algorithm, because the aim of the work was not to rank GA parameter combinations, but to select the combination that performs best. DoE was therefore a means to reduce the number of GA executions in a meaningful way.

5. Results and discussion

Three batch-splitting cases were studied. First, the batch is taken in full (transfer=1), secondly, it is split into two transfer-batches (transfer=2) and thirdly, it is split into three transfer batches (transfer=3). The size of each transfer batch is to be determined by the genetic algorithm, but it has to be non-zero. In essence, the job-shop scheduling problem is solved for 8 machines and

20, 40 or 60 jobs, scores of which have the same characteristics (due dates, changeover times, operation sequence), but their processing time differs depending on the respective participation ratio of each job. Makespan is used as the objective function of the problem.

In addition, the quality of the schedule is judged using, apart from makespan, mean tardiness of each job, the number of late jobs and the total changeover time. Mean tardiness was used instead of total tardiness, as it is more representative, due to the variation of number of jobs.

Indicatively, for an experimental run of 50 generations on a 1.4 GHz Pentium processor the algorithm takes about 15 minutes.

The results obtained for transfer equal to 1,2 and 3 are shown in Table 7. Results representing GA scheduling refer to all 9 experiments performed with different combinations of GA parameters as designed in the respective L9 orthogonal array. These are complemented by results for the four heuristic scheduling rules employed.

insert Table 7 about here..

It has to be noted that the number of crossover points in MX for chromosome 1 was different for each transfer case. In fact, the initial values of genes are doubled and tripled for transfer equal to 2 and 3 respectively. For this reason, the number of crossover points was increased to 5 and 7 respectively. These values were determined using trial and error in order to achieve a satisfactory on and off-line performance of the GA. By contrast, for chromosome 2 the number of pairs that undergo mutation was kept the same (3) for all cases.

GA results are invariably better than heuristic results. For transfer=1 the best makespan solution is obtained in the 7th experiment, but the 8th experiment, see Table 7 seems more balanced, because it corresponds to the third best makespan value (with least difference from the best two) and minimum tardy jobs and second least lateness compared to the best of MST and the second best changeover time. The evolution of the solution of the 7th experiment is shown in Figure 1.

insert Figure 1 about here.

For transfer=2 the difference in makespan between the best heuristic and any genetic algorithm run is higher than 499 units, whilst the difference between the best heuristic and the best GA run is 1389 units. Least makespan is 37354 units for the 8th run. The best balanced solution seems to be that of the 6th run, its evolution being shown in Figure 2.

insert Figure 2 about here.

For transfers=3 the difference in makespan between the best heuristic and any genetic algorithm run is higher than 616 units, whilst the difference between the best heuristic and the best GA run is 1121 units. The GA gives invariably zero tardiness, which is something that only EDD and MST heuristics achieve, since these work with due date as scheduling criterion. The best solution (1st run) combines both lowest makespan and second best changeover time, its evolution being shown in Figure 3. The best schedule in terms of makespan for each of the three cases are shown in Figure 4.

insert Figure 3 about here.

insert Figure 4 about here.

The general trend observed is that the more the transfer batches the lower the makespan, but at the same time the larger the proportion of the makespan that changeover accounts for. This is exactly as expected.

Comparatively, for increased transfer batches (transfer-1,2 and 3) makespan decreases from 39550 to 37354 and to 36966 time units. This is due to the different critical path, shifted to the left, and the reduced idle times. Left shift of the critical path is due to the exploitation of machine availability created by reduced batch sizes. A characteristic example refers to job 6 which completes the schedule in all three cases. In the first case (transfers=1), see Fig. 5 (a), machine 6 waits for job 6 to finish on machine 1. By contrast, in Fig. 5(b) job 6 has been split into two, i.e. 6 and 26, thereby avoiding the waiting. The fact that the two transfer batches are processed sequentially on machines 6 and 1 results from the optimisation process. The critical path in the first case is defined by all jobs in machine 5 up to 6, then by 6 on machine 1 and 6 on machine 6. In the second case, the critical path starts from machine 5 up to job 40, goes on with 1 up to job 6 and finishes with job 6 on machine 6. In the third case, it starts with machine 5 up to job 46, proceeds on machine 1 up to job 6 and closes in machine 6. Critical path reduction due to transfer batches is obvious, i.e. the latter tend to create a non-delay schedule.

insert Figure 5 about here.

Increasing of the number of transfer batches beyond a certain point is expected to induce marginal improvement in the schedule. This becomes quite pronounced when the critical path is confined to one machine, see for instance

Fig. 6, where machine 1 becomes critical and a further increase in transfer batches will not improve completion time on this machine, but will increase changeover time.

insert Figure 6 about here.

In parallel to makespan reduction, tardiness reduction is achieved, in general, too. In the first case (transfer=1 7th run) jobs 3, 4 και 16 are tardy, and in the other two cases (transfer-2 and 3, 6th and first runs respectively) no jobs are tardy for the same reasons for which makespan decreases.

Transfer batch participation ratio was also optimised, achieving sensible values (not too low and not too high either), see Table 8.

insert Table 8 about here..

6. Conclusions

The attempt to combine two chromosomes in one genetic algorithm proved successful, because each scheduling solution results from different participation ratios, i.e. trying different independent combinations of participation ratio and sequence becomes possible. Besides, using the two chromosomes helped in computing the value of one single fitness function, which means that the best pairs can be selected using the very same criterion.

The genetic algorithm using the well-proven priority rule based encoding for the second (main) chromosome, gave better results than those achieved by the individual heuristics which were employed in the encoding scheme.

In addition, the influence of the number of transfer batches into which each original batch had to be split, was made clear, a larger number resulting normally to better exploitation of machine idle time, but just marginally so after a certain point. Although just three values were examined (one, two and three transfer batches), it is obvious that the methodology followed is generally applicable to any number of values that would have made practical sense.

Finally, Taguchi DoE proved quite effective, because the results achieved by each combination of essential parameter values of the genetic algorithm exhibited significant differences in terms of makespan, tardiness, tardy jobs and total changeover time. This proved also that exploration of such combinations is a necessary ingredient in the genetic recipe, particularly when not just good but as close to optimum as possible solutions are sought. Note that DoE enables insight into the significance of each GA parameter, but this was deemed to be outside the scope of this work.

Note that the approach was demonstrated with just one example. However, this does not harm generality because of the use of random numbers in assigning processing times etc. and because of the non-favourable nature of the job-shop (i.e. prone to bottlenecks). Experimenting with the size of the job-shop (number of machines and number of jobs) is a possible research continuation direction.

Further future work involves a more general representation, where the number of transfer batches will generally vary for each original batch, all criteria used to assess the solution will be used in the fitness function with a Pareto front technique and additional heuristics will be examined, too.

References

Al-Hakim, L., An analogue genetic algorithm for solving job shop scheduling problems, Int Jnl Production Research, 2001, 39 (7), 1537-1548.

Abdelmaguid, T.F., Nassef, A.O., Kamal, B.A. and Hassan, M.F., A hybrid GA/heuristic approach to the simultaneous scheduling of machines and automated guided vehicles, Int Jnl Production Research, 2004, 42 (2), 267-281.

Anderson, E.J., *The management of manufacturing – models and analysis*, 1994, Addison Wesley.

Baek, D.H. and Yoon, W.C., Co-evolutionary genetic algorithm for multi-machine scheduling: coping with high performance variability, Int Jnl Production Research, 2002, 40 (1), 239-254.

Baker, K., *Introduction to Sequencing and Scheduling*, 1974, John Wiley & Sons.

Candido, M.A.B., Khator, S.K. and Barcia, P.M., A genetic algorithm based procedure for more realistic job shop scheduling problems, Int Jnl Production Research, 1998, 36 (12), 3437-3457.

Chang, J.H. and Chiu, H.N., A comprehensive review of lot streaming. International Journal of Production Research, 2005, 43(8), 1515–1536.

Cheng, R., Gen, M. and Tsujimura, Y., A tutorial survey of job shop scheduling problems using genetic algorithms: Part 2 hybrid genetic search strategies, Computers & Industrial Engineering, 1999, 37 (1-2), 51-55.

Chen, R.-S., Lin, H.-C. and Chang, C.C., Improve the efficiency of traditional scheduling system with GA-Petri net model, Int Jnl Computer Applications in Technology , 2003, 17 (1), 16-30.

Dagli, C.H. and Schierholt, K., Evaluating the performance of the genetic neuro scheduler using constant as well as changing crossover and mutation rates, Computers & Industrial Engineering, 1997, 33 (3-4), 253-256.

Dominic, P.D.D., Kaliyamoorthy, S. and Murugan, R., A conflict-based priority dispatching rule and operation-based approaches to job shops, Int Jnl Advanced Manufacturing Technology, 2004, 24 (1-2), 76-81.

Dorndorf U. and Pesch E., Evaluation Based Learning in a Jobshop Scheduling Environment, Computers and Operations Research, 1995, 22, 25-40.

Esquivel, S., Ferrero, S., Gallard, R., Salto, C., Alfonso, H. and Schütz, M., Enhanced evolutionary algorithms for single and multiobjective optimisation in the job shop scheduling problem, Knowledge-Based Systems, 2002, 15 (1-2), 13-25.

Giffler B. and Thomson G., Algorithms for Solving Production Scheduling Problems, Operations Research, 1960, 8(4), 487-503.

Goldberg G. E., *Genetic Algorithms in Search, Optimization and Machine Learning*, 1989, Addison-Wiley.

Ip, W.H., Li, Y., Man K.F. and Tang, K. S., Multi-product planning and scheduling using genetic algorithm approach, Computers & Industrial Engineering, 2000, 38 (2), 283-296.

Jahangirian, M. and Conroy, G.V., Intelligent dynamic scheduling system: the application of genetic algorithms, Integrated Manufacturing Systems, 2000, 11 (4), 247-257.

Jawahar, N., Aravindan, P. and Ponnambalam, S.G., A genetic algorithm for scheduling flexible manufacturing systems, Int Jnl Advanced Manufacturing Technology, 1998, 14 (8), 588-607.

Jordon, C., A two-phase genetic algorithm to solve variants of the batch sequencing problem, Int Jnl Production Research, 1998, 36 (3), 745-760.

Holland J.H., *Adaptation in Natural and Artificial systems: An introductory Analysis with Application in Biology, Control and Artificial Intelligence*, 1972, MIT Press, Cambridge.

Keung, K.W., Ip, W.H. and Chan, C.Y., An enhanced MPS solution for FMS using GAs, Integrated Manufacturing Systems, 2001, 12 (5), 351-359.

Knuth D.E., *The art of computing programming*, 1981, Addison-Wesley.

Kubota, N. and Fukuda, T., Structured intelligence for self-organising manufacturing systems, Jnl Intelligent Manufacturing, 1999, 10 (2), 121-133.

Lee, C.-Y., Piramuthu, S. and Tsai, Y.-K., Job shop scheduling with a genetic algorithm and machine learning, Int Jnl Production Research, 1997, 35 (4), 1171-1191.

Mesghouni, K. , Pesin, P., Trentesaux, D., Hammadi, S., Tahon, C. and Borne, P., Hybrid approach to decision-making for job shop scheduling, Production Planning & Control, 1999, 10 (7), 690-706.

Piramuthu, S., Shaw, M. and Fulkerson, B., Information-based dynamic manufacturing system scheduling, Int Jnl Flexible Manufacturing Systems, 2000, 12 (2/3), 219-234.

Ponnambalam, S.G., Aravindan, P. and Rao, P.S., Comparative evaluation of genetic algorithms for job shop scheduling, Production Planning & Control, 2001, 12 (6), 560-574.

Ponnambalam, S.G., Ramkumar, V. and Jawahar, N., A multiobjective genetic algorithm for job shop scheduling, Production Planning & Control, 2001, 12 (8), 764-774.

Ponnambalam, S.G., Jawahar, N. and Kumar, B.S., Estimation of optimum genetic control parameters for job shop scheduling, Int Jnl Advanced Manufacturing Technology, 2002, 19 (3), 224-234.

Potts, C.N. and Kovalyov, M.Y., Scheduling with batching: a review, European Jnl Operational Research, 2000, 120 (2), 228-249.

Ross, J.P., *Taguchi techniques for quality engineering*, 1996, McGraw-Hill.

Su, C.-T. and Shiue, Y.-R., Intelligent scheduling controller for shop floor control systems: a hybrid genetic algorithm/decision tree learning approach, Int Jnl Production Research , 2003, 41 (12), 2619-2641.

Sule D. R., *Industrial Scheduling*, 1996, PWS Publishing Company.

Wang, W. and Brunn, P. , An effective genetic algorithm for job shop scheduling, Proc Institution Mechanical Engineers-B, 2000, 214 (B4), 293-300.

Wang, L. and Zheng, D.-Z., A modified genetic algorithm for job shop scheduling, Int Jnl Manufacturing Technology, 2002, 20 (1), 72-76.

Zhiming, W. and Chunwei, Z., Genetic algorithm approach to job shop scheduling and its use in real-time cases, Int Jnl Computer Integrated Manufacturing, 2000, 13 (5), 422-429.

Table 1. Processing time (in standard time units per part processed) per machine (operation) per job.

Job#	Machine#							
	1	2	3	4	5	6	7	8
1	0	6	4	5	0	9	2	0
2	0	0	2	9	6	3	1	2
3	1	2	9	3	3	2	8	0
4	9	1	8	5	9	4	2	3
5	9	4	0	0	0	4	7	0
6	7	1	5	7	8	3	8	2
7	6	2	0	5	8	0	9	6
8	0	1	4	9	6	7	9	0
9	3	8	3	5	8	3	4	8
10	5	2	9	4	4	5	0	8
11	2	2	4	3	0	1	2	6
12	8	5	9	1	2	8	5	5
13	3	5	1	4	9	9	3	3
14	4	4	0	7	2	9	7	2
15	8	7	6	8	3	7	3	2
16	1	2	0	1	7	7	0	0
17	4	5	8	5	9	8	5	9
18	2	0	0	5	7	6	7	4
19	2	1	6	7	7	1	4	4
20	9	8	0	3	5	6	4	2

Table 2. Machine sequence per job.

Job #	Machine							
1	2	3	6	4	7			
2	3	7	6	5	4	8		
3	5	6	3	4	1	7	2	
4	7	4	8	2	1	5	3	6
5	1	2	7	6				
6	8	7	3	2	4	5	1	6
7	5	7	4	1	8	2		
8	4	5	7	2	6	3		
9	5	4	6	3	7	2	1	8
10	8	6	5	4	3	1	2	
11	2	8	1	7	6	4	3	
12	4	3	5	1	2	6	7	8
13	5	2	7	3	8	4	1	6
14	1	4	6	2	7	5	8	
15	5	1	2	6	8	7	3	4
16	5	4	2	6	1			
17	3	6	5	2	7	4	1	8
18	4	7	5	6	8	1		
19	1	7	2	8	3	6	5	4
20	2	6	8	5	1	7	4	

Table 3. Batch size and due date (in standard time units) per job

Job #	Batch size	Due Date
1	300	27750
2	200	19460
3	500	20300
4	400	21840
5	200	17640
6	500	87150
7	200	23800
8	400	31920
9	300	49350
10	200	24080
11	200	23800
12	500	39200
13	300	56070
14	400	65520
15	200	33180
16	500	32900
17	400	75600
18	300	43050
19	200	35560
20	400	51800

Table 4. Changeover times (in standard time units) from job to job, applicable to all operations within a job. Figures on the diagonal denote job setup time when machine has been idle.

TO / FROM	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	09	02	16	11	19	08	14	02	15	14	08	13	18	11	14	05	06	16	16	19
2	02	06	18	06	11	08	05	10	17	17	01	20	14	16	02	08	10	19	19	19
3	07	15	01	08	06	20	06	13	04	15	09	20	15	05	02	11	10	02	10	11
4	09	07	19	16	11	11	12	14	19	04	18	17	11	12	01	06	15	11	18	15
5	05	01	17	12	17	02	01	14	01	05	12	13	09	10	09	14	11	01	07	03
6	07	17	02	17	12	03	12	14	20	08	18	20	10	17	09	08	19	16	04	03
7	17	15	04	20	14	06	04	05	06	09	13	19	03	05	18	02	03	04	03	03
8	10	08	03	07	10	10	05	19	01	12	12	10	04	19	10	13	04	19	16	13
9	20	06	09	03	13	18	08	14	06	06	09	20	15	08	17	10	10	14	10	17
10	15	15	02	06	20	03	18	20	09	15	11	15	18	14	15	10	19	09	05	17
11	07	20	04	11	18	11	09	18	05	17	18	13	03	13	16	17	18	01	18	03
12	14	10	07	09	02	15	20	10	01	05	13	14	11	20	05	20	10	16	09	04
13	15	08	18	17	14	17	05	02	02	13	19	16	09	06	12	11	08	11	05	08
14	04	06	05	16	01	08	14	01	12	17	03	04	16	20	14	07	02	04	16	02
15	19	04	10	02	04	15	05	12	06	15	15	06	09	06	08	05	13	07	03	11
16	14	08	13	02	03	06	10	07	08	11	14	17	16	12	03	13	15	07	14	14
17	11	11	01	14	14	17	09	08	20	07	07	14	11	11	06	10	16	18	20	17
18	12	20	16	13	03	04	19	12	01	13	14	19	09	15	17	06	15	12	14	11
19	04	11	04	13	16	09	09	01	19	14	15	07	10	08	11	14	01	14	02	15
20	03	01	04	15	05	12	08	05	06	02	06	11	08	17	11	02	17	11	02	18

Table 5 . Positioning of the experiment elements in L9 array.

Element	Column number
p_s	1
p_c	2
Interaction p_s - p_c	3
p_m | interaction p_s – p_m	4

Table 6- L9 array configuration

α/α	1	2	4
1	1	1	1
2	1	2	2
3	1	3	3
4	2	1	3
5	2	2	1
6	2	3	2
7	3	1	2
8	3	2	3
9	3	3	1

1	2	4
30	0.75	0.05
30	0.85	0.10
30	0.95	0.15
40	0.75	0.15
40	0.85	0.05
40	0.95	0.10
50	0.75	0.10
50	0.85	0.15
50	0.95	0.05

Table 7. Genetic algorithm and heuristic scheduling results. MS:makespan, TJ: tardy jobs, MT: mean tardiness, CT:changeover time.

Transfer	1				2				3			
Run	MS	TJ	MT	CT	MS	TJ	MT	CT	MS	TJ	MT	CT
L9-1	39557	3	1026	1421	38244	0	0	2982	36966	0	0	4373
L9-2	40387	3	1207	1563	38180	0	0	3072	37429	0	0	4506
L9-3	40397	2	803	1439	38071	2	32	2859	37471	0	0	4430
L9-4	40167	3	1365	1464	38093	0	0	3009	37098	0	0	4455
L9-5	40106	2	985	1493	37766	0	0	2897	37367	0	0	4387
L9-6	40393	4	1163	1571	37914	0	0	2783	37332	0	0	4452
L9-7	39550	3	838	1438	37761	0	0	3062	37356	0	0	4350
L9-8	39601	2	706	1438	37354	2	112	2873	37124	0	0	4488
L9-9	40598	4	1456	1520	37845	0	0	3037	37088	0	0	4556
SPT	44064	4	2420	1480	38743	4	771	2739	38087	7	718	4423
EDD	47356	1	407	1453	40294	0	0	2959	39751	0	0	4402
MST	45289	1	93	1539	40466	0	0	3022	38973	0	0	4196
EOD	46168	5	1934	1470	40873	5	756	2932	39871	6	388	4309

Table 8 : Participation ratio results

Job #	Batch Size	Transfer=2		Transfer-3		
		Transfer Batch 1	Transfer Batch 2	Transfer Batch 1	Transfer Batch 2	Transfer Batch 3
1	300	210	90	117	109	74
2	200	75	125	59	58	83
3	500	169	331	8	174	318
4	400	136	264	88	124	188
5	200	99	101	85	9	106
6	500	169	331	127	139	234
7	200	68	132	33	96	71
8	400	136	264	81	93	226
9	300	101	199	89	87	124
10	200	110	90	51	55	94
11	200	102	98	85	9	106
12	500	256	244	127	139	234
13	300	165	135	76	83	141
14	400	223	177	189	42	169
15	200	110	90	94	21	85
16	500	169	331	19	225	256
17	400	136	264	81	93	226
18	300	165	135	50	144	106
19	200	46	154	33	96	71
20	400	92	308	160	88	152

Figure captions

Figure 1. GA results for transfer=1, 7th run (a) Makespan (MS), (b) Tardy jobs (TJ), (c) Tardiness (T), (d) Changeover time (CT).

Figure 2 GA results for transfer=2, 7th run (a) Makespan (MS), (b) Tardy jobs (TJ), (c) Tardiness (T), (d) Changeover time (CT).

Figure 3. GA results for transfer=3, 7th run (a) Makespan (MS), (b) Tardy jobs (TJ), (c) Tardiness (T), (d) Changeover time (CT).

Figure 4. Best Schedules on Gantt charts for (a) transfer=1, (b) transfer=2, (c) transfer=3.

Figure 5. Gannt chart excerpts showing reduction of idle time in machine 6 from (a) transfer=1 to (b) transfer=2.

Figure 6. Gannt chart excerpts showing reduction of idle time in machine 1, from (a) transfer=1 to (b) transfer=2.

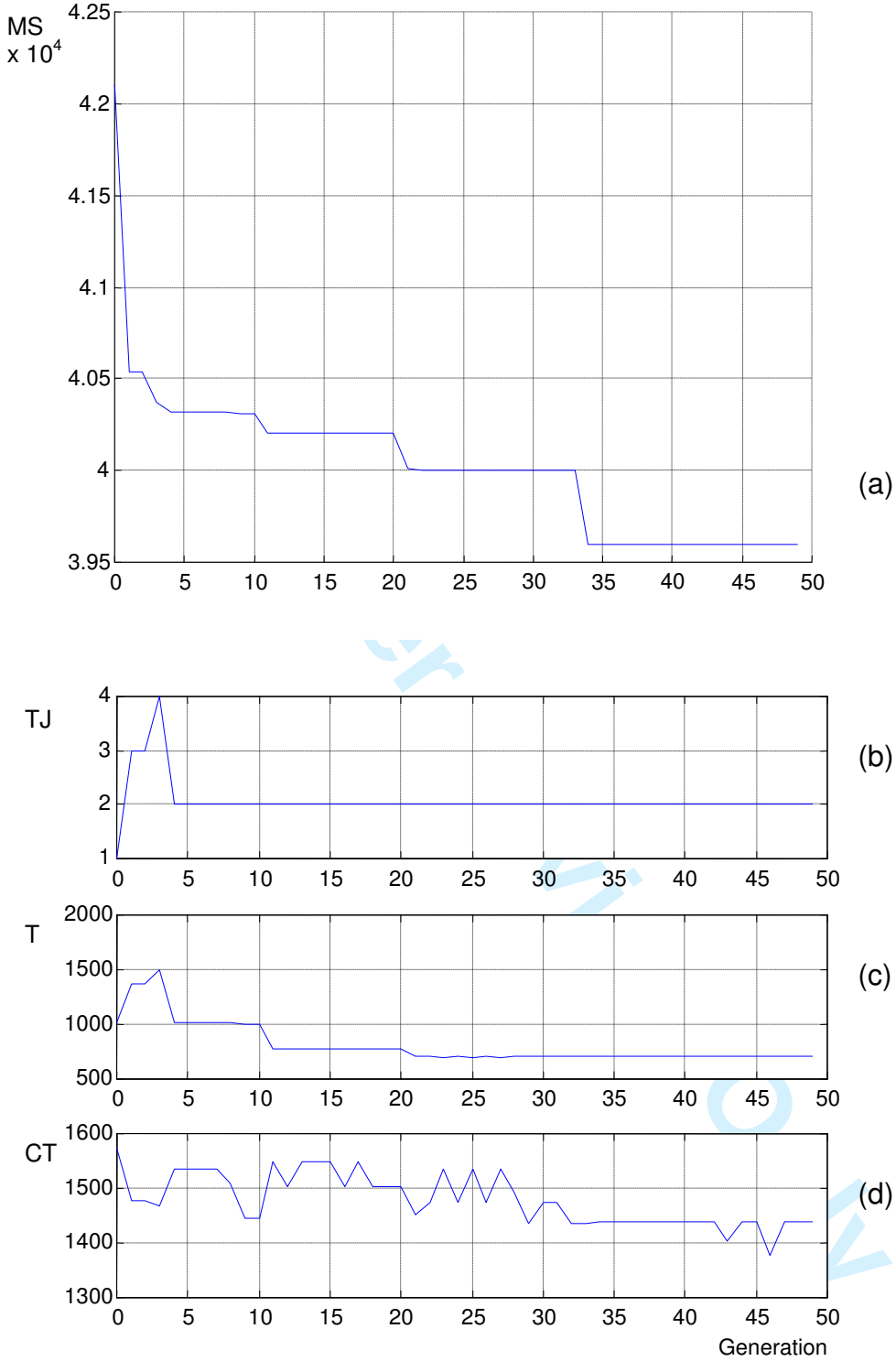


Figure 1. GA results for transfer=1, 7th run (a) Makespan (MS), (b) Tardy jobs (TJ), (c) Tardiness (T), (d) Changeover time (CT).

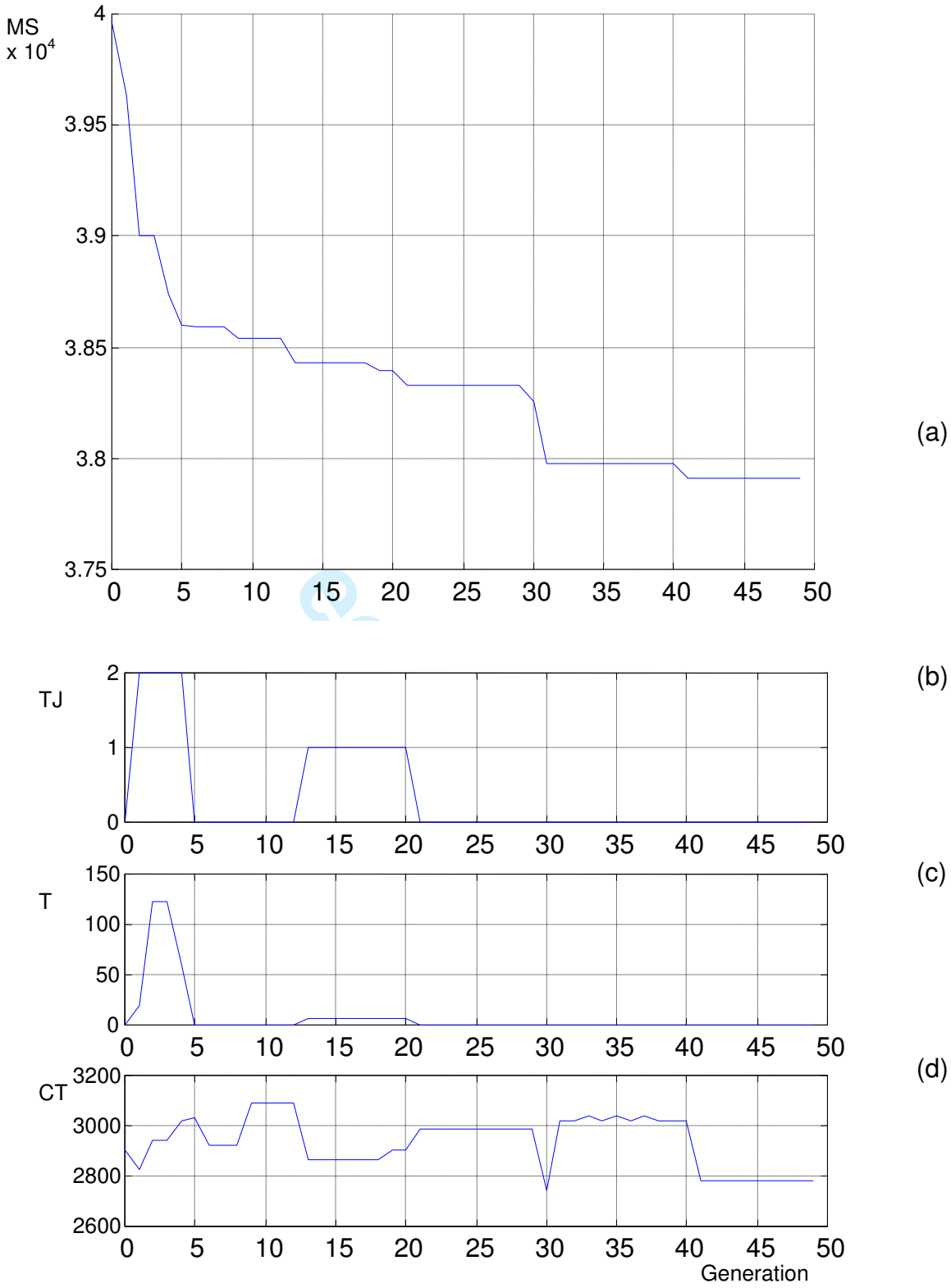


Figure 2 GA results for transfer=2, 6th run (a) Makespan (MS), (b) Tardy jobs (TJ), (c) Tardiness (T), (d) Changeover time (CT).

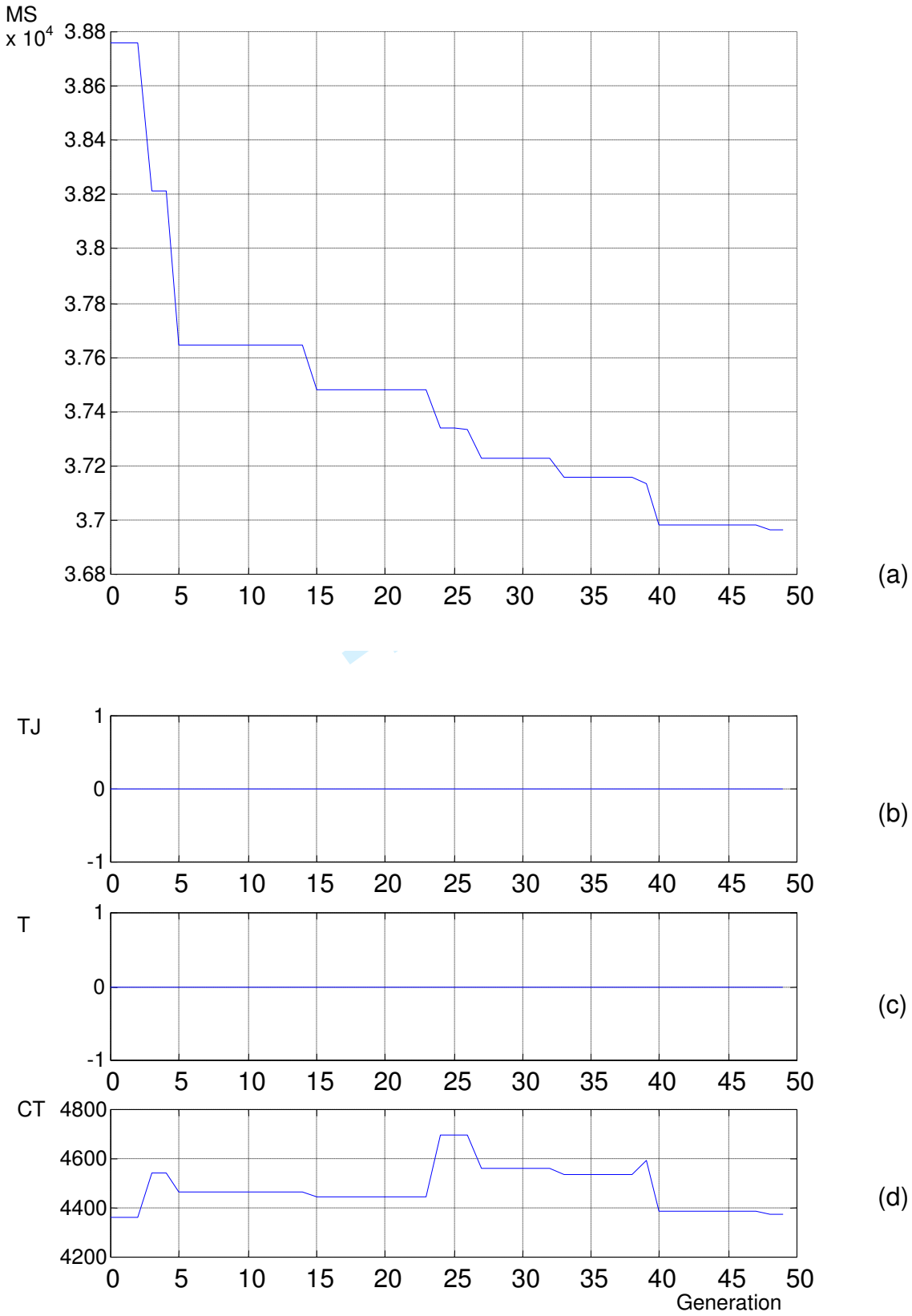


Figure 3. GA results for transfer=3, 1st run (a) Makespan (MS), (b) Tardy jobs (TJ), (c) Tardiness (T), (d) Changeover time (CT).

Figure 4. Best Schedules on Gantt charts for (a) transfer=1

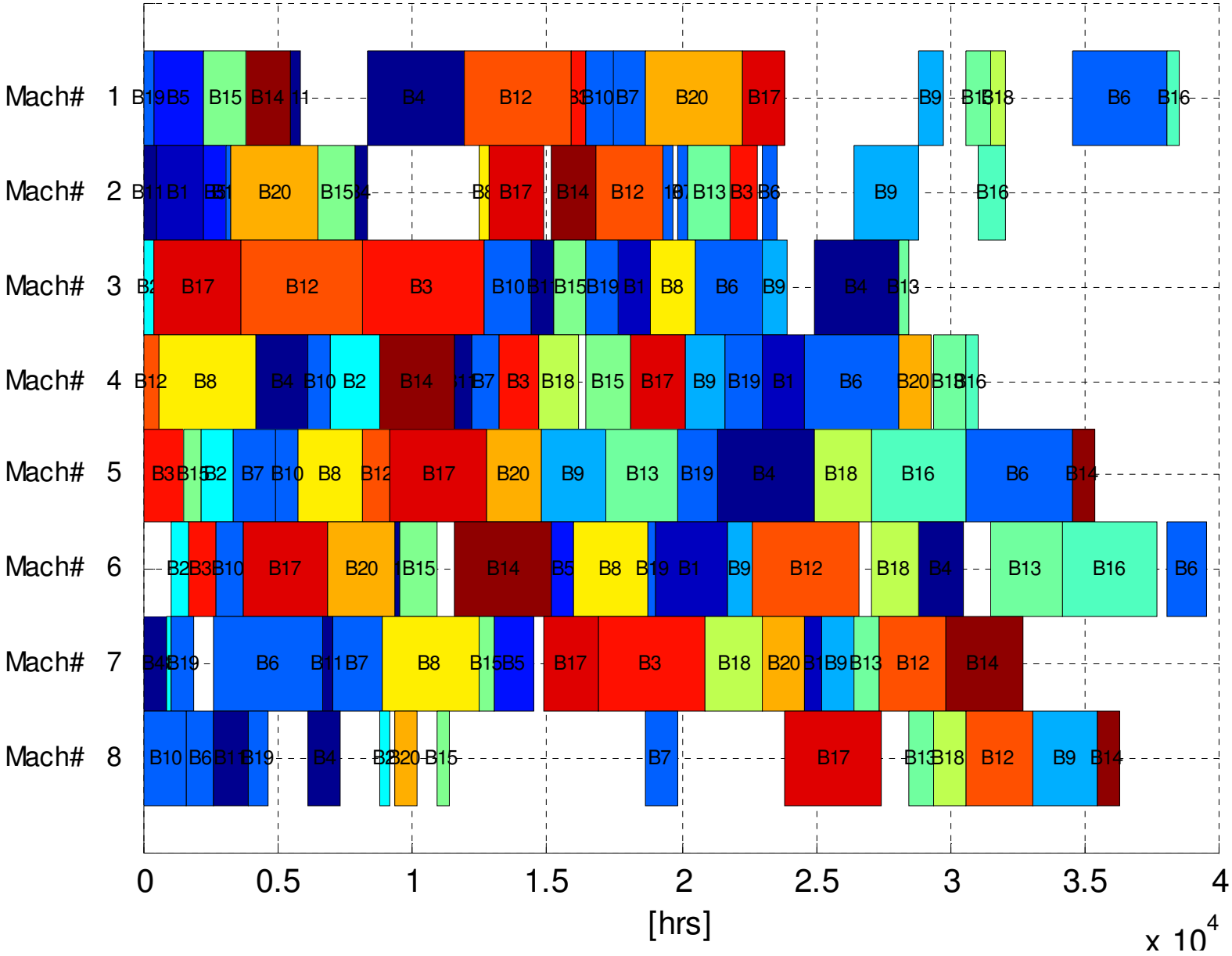


Figure 4. Best Schedules on Gantt charts for (b) transfer=2

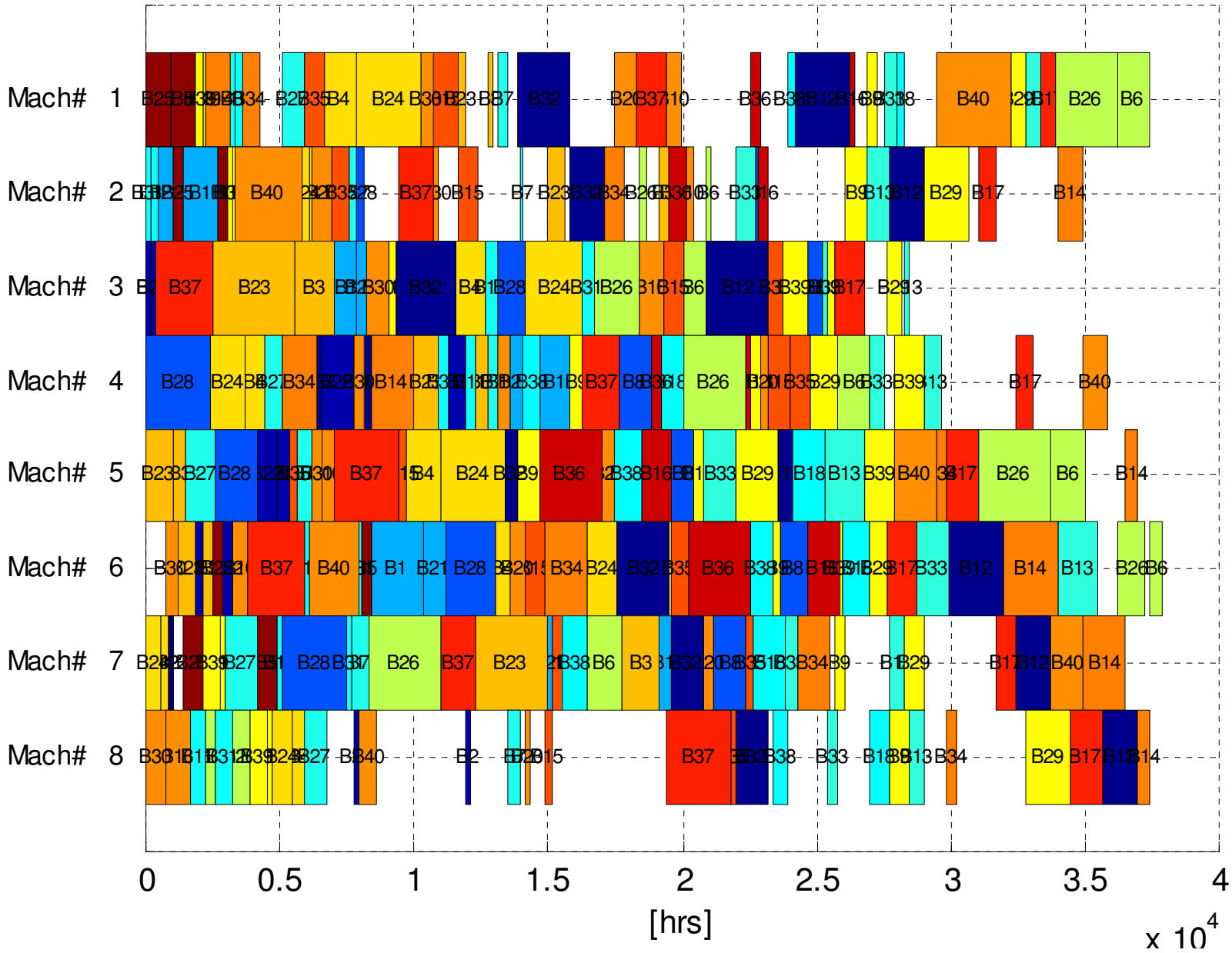
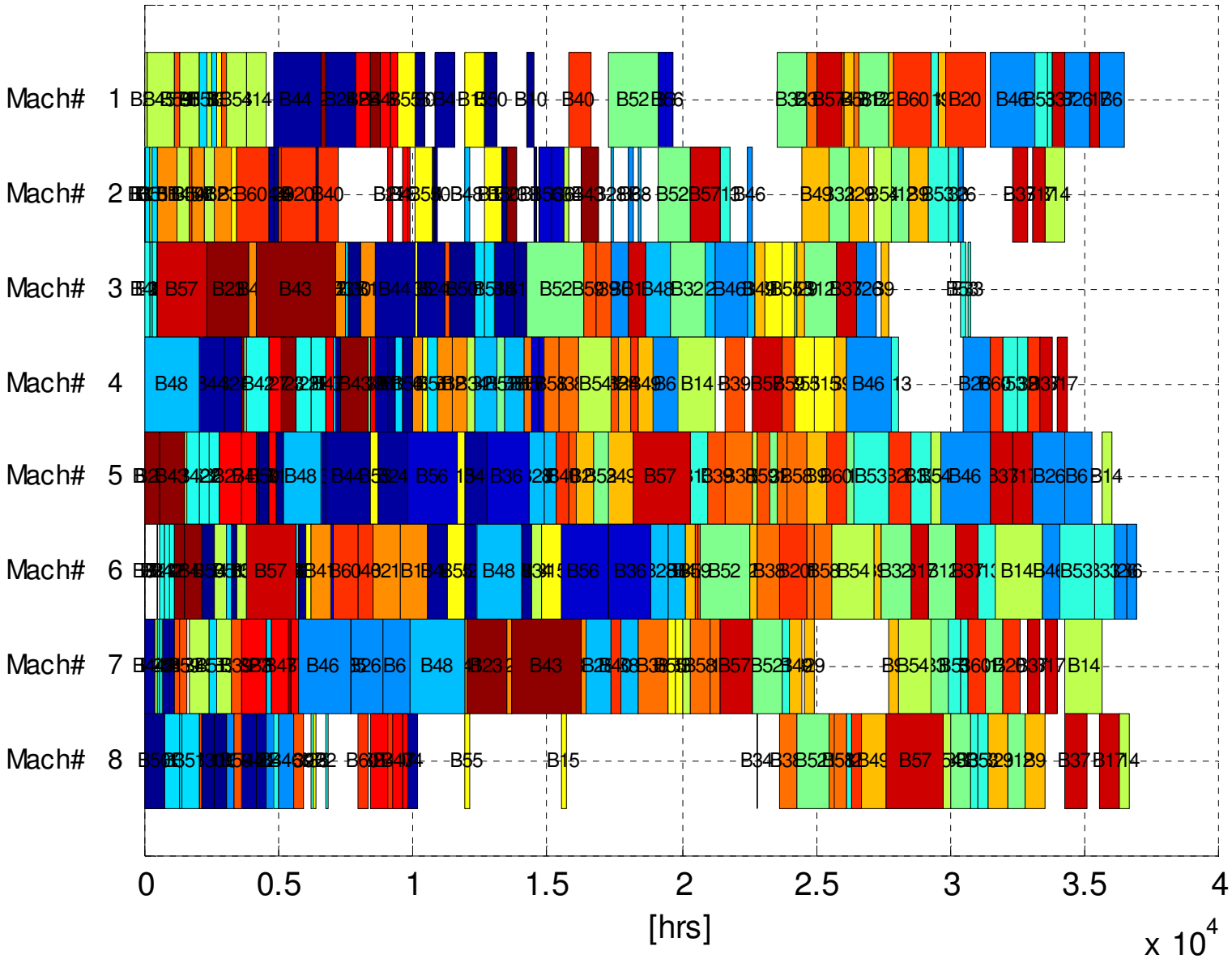


Figure 4. Best Schedules on Gantt charts for (c) transfer=3.



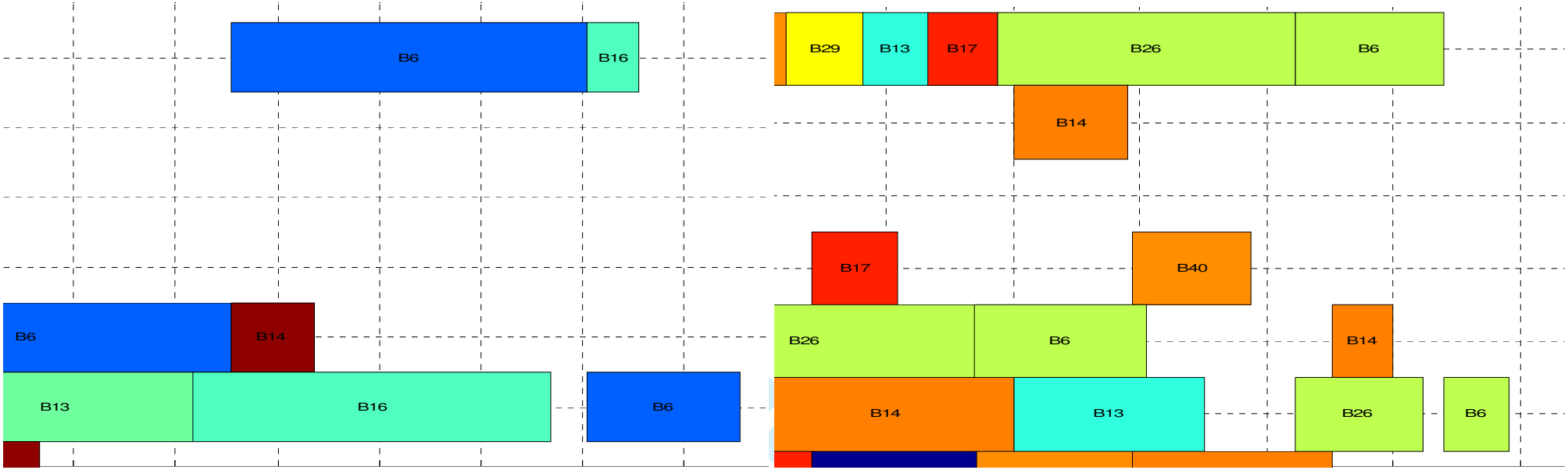


Figure 5. Gannt chart excerpts showing reduction of idle time in machine 6 from (a) transfer=1 to (b) transfer=2

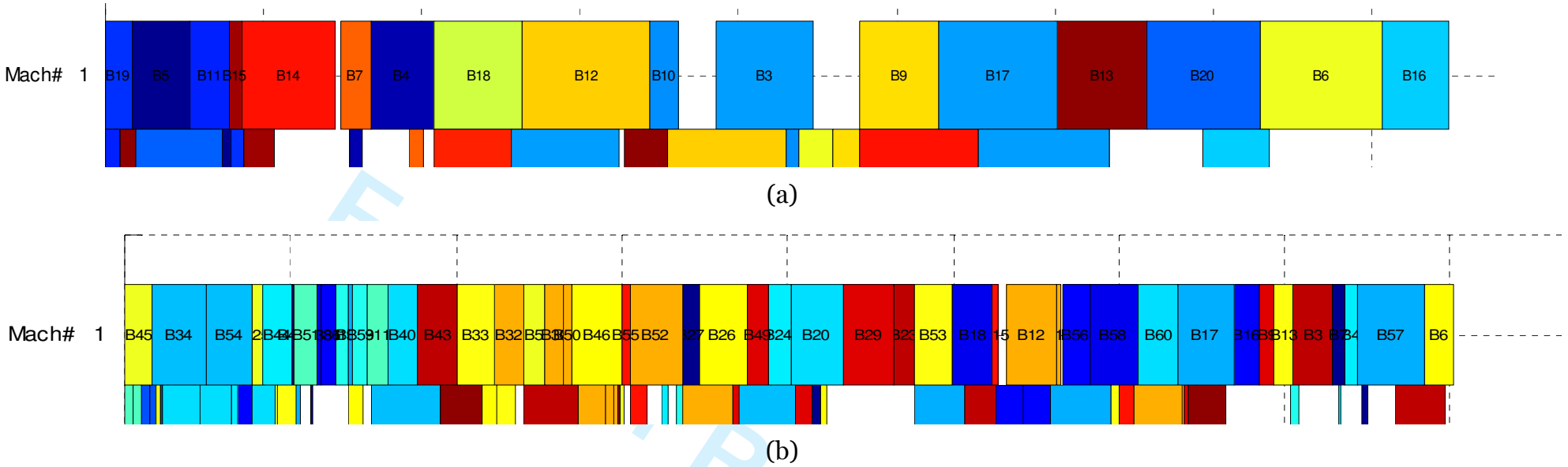


Figure 6. Gannt chart excerpts showing reduction of idle time in machine 1 from (a) transfer=1 to (b) transfer=2.