Contents lists available at SciVerse ScienceDirect

CIRP Annals Manufacturing Technology

Journal homepage: www.elsevier.com/locate/cirp

Scheduling with alternative routings in CNC workshops

Youichi Nonaka (3)[a], Gábor Erdős[b], Tamás Kis[b], Takahiro Nakano (3)[a], József Váncza (1)[b,c]

[a] *Hitachi, Ltd., Yokohama Research Laboratory, Yokohama, Japan*
[b] *Computer and Automation Research Institute, Hungarian Academy of Sciences, Budapest, Hungary*
[c] *Dept. of Manufacturing Science and Technology, Budapest University of Technology and Economics, Budapest, Hungary*

In workshops of CNC machines, jobs may have alternative sequences of operations, where each operation must be performed on one of a pre-specified subset of machines. The key to solving to this extremely hard scheduling problem is balancing the load on machines of a flexible job shop. The proposed method combines mathematical programming for selecting the best routing alternatives and tabu search for finding the best assignment of machines to operations along with the routings. Experiments in an industrial case study refer to the primary role of optimized load balancing that proved to be computationally tractable on large-scale problem instances.

Scheduling, Optimization, Load balancing

1. Introduction

The objective of this research was to open new avenues for production engineering by investigating how the efficiency of production systems can be increased by the integration of traditionally separated process planning and production scheduling functions. The scope of investigations has been set to complex engineer-to-order and make-to-order production, where individual products are manufactured by means of advanced Computerized Numerical Control (CNC) machining technology. Typically, 5-axis CNC machining is used when parts with complex shapes and high precision surfaces – like engine blocks, gear parts, turbine blades, etc. – have to be manufactured. Furthermore, 5-axis CNC machines are extremely *flexible*: by changing their tool type, they can easily be applied in various processes like turning, drilling, and milling. However, for all such capabilities one has to pay a relatively high price when installing advanced CNC machinery in a workshop. In fact, *efficiency* of CNC machines is the key performance indicator for factory return on investment (ROI) in many industries including automobile engine fabrication, machine gear system production, turbine manufacturing for the power industry, to name a few.

In the practice computer-aided process planning (CAPP) and manufacturing (CAM) generate typically a unique *process plan* and corresponding NC code for each product which determine a fixed *routing* along some resources whenever an order is released as a job for production. This process plan is typically the one that is judged the most efficient in the hypothetical situation that no resource conflict arises with other orders. In real production, this is rarely the case. However, workshops equipped even with the most advanced technology have usually mixed resources, and many operations assigned to 5-axis machines could also be executed on less sophisticated turning, drilling, 3-axis milling or other cutting machines. While any switch to these resources would result in definitely higher changeover, processing and transportation times, the workshop as a whole, thanks to a balanced load of its capacities, could be able to fulfill its complete set of orders even in shorter time. This potential can be exploited if (1) one is able to prepare alternative process plans with different resource assignments for the same products, and (2) can cope with the drastically increased number of decision alternatives during scheduling. This research was aimed at making advance in both aspects. Since generating alternative process plans for complex machined parts and workshops with mixed CNC technologies is still a work underway, this paper presents a novel solution method of the *scheduling* problem.

As for the actual industrial problem at hand, a workshop of CNC machines with different but partly overlapping capabilities is modeled. The workshop processes a set of jobs at a time, where each job is aimed at manufacturing a particular product. For each product, manufacturing engineering generates alternative process plans that specify the sequence of operations together with sets of such machines that are capable to execute them. Operation and setup times are known a priori, but may vary per machine. Since products are of high quality and value, process plan alternatives have to be validated prior to scheduling. Hence, machine allocation for each operation can be changed provided scheduling has enough time before its execution. The overall goal is to maximize the workshop's efficiency in terms of its throughput (that correlates well with its ROI). The problem is of *large scale*: jobs to be executed are released in the order of hundred, and each complex product goes through dozens of operations. As a consequence, manufacturing lead times take two months in average, requiring a production scheduling horizon of about half a year, with a resolution of daily units. The workshop consists of tens of individual resources. The flexibility of machines allows for up to ten process plan variants per part, and within a plan variant in average ten machine alternatives for each operation.

While the above conditions can be considered normal on an industrial scale, they define a solution space that is far too large for state-of-the-art optimization methods tailored to this problem type. A new solution method had to be developed that is able to find a trade-off between the great number of alternatives and acceptable solution time. The heart of the problem is finding a *balanced load* of machines that are available and eligible for processing the jobs. After surveying related works (Section 2) and defining the problem formally (Section 3), a detailed account of the solution method is provided that combines advanced mathematical programming techniques in a novel way (Section 4). The paper presents results of computational experiments on a case study taken from the target industry (Section 5) and points both to future research directions and application potential (Section 6).

2. Related works

Scheduling with routing alternatives is a form of the *integration of process planning and scheduling*. As early as in the 1960s efforts were made to integrate the decisions about alternative plans into the scheduling phase of project planning [1]. Linking the product and production oriented aspects of production engineering at higher levels of abstraction and decision levels has been persistently in the forefront of research for more than thirty years [2][3][4][5]. The main motivations have been twofold. Firstly, research has been aimed at achieving *responsiveness* which involves ongoing mapping of projections of the future (i.e., plans, schedules) to the reality of production [6]. Responsiveness calls for process plans that match the actual shop floor conditions and machine availability, as well as schedules that can be adapted to changing circumstances of production. The *agent-based* approach proved to meet these requirements in particular since it provided an appropriate design metaphor to structure domain knowledge and responsibilities – and system design, accordingly – around entities like products, orders, jobs, and resources. Further on, agent-based modeling was especially suitable for *simulating* the behavior of complex manufacturing systems embedded in dynamic execution environments [7]. The interplay and coordination of these autonomous agents were facilitated by systematic biological [8] or market principles [9][10] that fit into Ueda's broader engineering concept of *emergent synthesis* [11]. The distributed system concept has been manifested in a recent *reconfigurable* manufacturing system that adapted to changing conditions by closed-loop dynamic scheduling so as to improve machine utilization rate [12]. Nonetheless, as a comparison of centralized and autonomous control regimes in a dynamic flexible flow shop suggests, central scheduling can be superior in terms of utilization rate when there are many resource alternatives and a relatively stable and long horizon for decision making [13].

In fact, the other main motivation behind the integration of process planning and scheduling is just *increasing performance*: balanced load of available resources, better utilization, higher throughput and improved service level [14][15]. Whatever the actual priority of these objectives may be, they are positively coupled: a balanced load that does not create a bottleneck machine facilitates improving other criteria, too. However, it was also understood that in an industrial setting process planning and scheduling cannot be tackled in a monolithic way because of the different responsibilities and decision making horizons, as well as of the hard computational tasks involved by both functions apiece. One can either (1) anticipate the expected shop-floor performance of candidate process plans early during the planning phase [16], or, (2) generate alternative routings and let the scheduler choose the one that is most suitable under actual shop-floor conditions.

Circumstances of the study discussed here fit the second approach, when typically predictive *job shop scheduling* problems (JSP) with extensions of alternative resources for operations (routing flexibility) and alternative routings for schedules (process plan flexibility) have to be solved [14][17]. Due to its high practical relevance and challenging complexity JSP with routing flexibility – also referred to as *flexible job shop scheduling* (FJSP) – has been the subject of a number of investigations. The results refer to the dominance of local search methods using a sophisticated neighborhood structure [18][19]. As for handling process plan flexibility, the representation of plan alternatives embraces enumerative and various graph-based models; for an overview, see Kis [15]. Since due to their inherent complexity the exact solution of such problems is computationally prohibitive, meta-heuristics like evolutionary algorithms [15], ant-colony optimization [20][21], or tabu search [15][18] are typically applied. A recent exception presents a mixed-integer program (MIP) model that is novel in terms of its parsimonious use of variables [17]. However, according to computational results on quite small-scale test examples its standard solution technique is inefficient.

For solving problems of realistic size, one has to apply heuristic methods, and primarily, the principle of *hierarchical decomposition*. Such an approach is presented by Brandimarte who solves first the load balancing problem by selecting the best routing alternatives, and next sequences operations on the machines [14]. Machine loading is solved by a method called genetic descent: local search picking up critical machines prepares the population for genetic algorithm based optimization. A generalized large-scale job shop scheduling problem with routing alternatives that arose in the lighting industry has been earlier investigated by the authors in [22]. Here, a two-phase solution method has been proposed, where an initial solution is computed using mixed-integer linear programming, and next this solution is improved by means of tabu search.

Alternative routings involve, however, not only opportunities buts also risks for schedule optimization. As Usher has shown in a series of experimental studies [23], common sense choice from among alternatives routings (e.g., selection of routings with minimum processing times, or requiring the least utilized resources) may result in adversary effect and degraded overall performance. Hence, the approach presented below does not make early commitments when selecting routings (and resources), facilitates revising such decisions, and keeps, at the same time, the combinatorial complexity of the solution process at bay. This last point on tractability is of special concern here, because the size of the industrial problem instances to be tackled highly exceeds the size of successfully solved test problems reported in the literature so far [14][17].

3. Statement of the job shop scheduling problem

This section gives a formal definition of the scheduling problem at hand. There is a finite set of machines M, and a set of jobs J. Each job j has a finite set A_j of *routing alternatives*, $S_{j,1}$ through $S_{j,a(j)}$, each $S_{j,k}$ being a sequence of $s(j,k)$ operations $(o_{j,k,1}, \ldots, o_{j,k,s(j,k)})$. The sequences have no operations in common. Associated with each operation o, there is a subset of *eligible machines* M_o; any machine from this set is capable to perform the operation. The processing time of the operation depends on the selected machine, i.e., $p_m(o)$ is the processing time of operation o if processed on machine m from M_o. Setup times are part of the processing times. Buffers before the machines are unlimited. The processing of operations cannot be interrupted, and once they are started on one of the machines, they have to be fully processed on the same machine. Moreover, there can be finish-to-start precedence constraints between jobs, i.e., jobs may constitute chains, or more generally, assembly trees (see Figure 1).

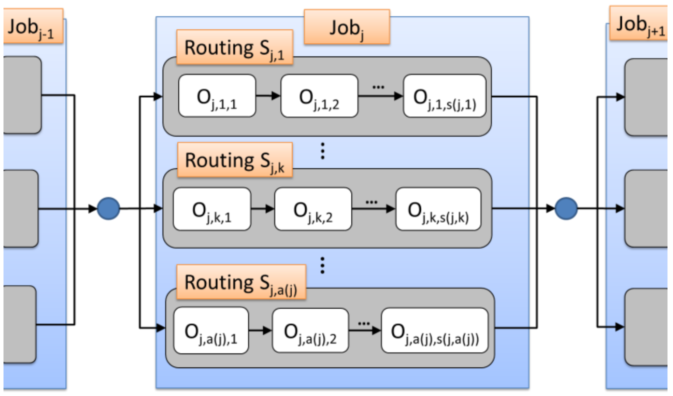

Figure 1. Structure of jobs with alternative routings.

A schedule consists of a selection of routing alternatives, and machines for the selected operations, as well as *starting times* for these operations. More formally, a schedule is a triple (σ,μ,τ), where σ is the selection of routing alternatives, i.e., σ selects a sequence from A_j for each job j, μ is a selection of machines, i.e., μ assigns a machine $\mu(o)$ from M_o to each operation o of the sequences selected by σ, and τ specifies a starting time $\tau(o)$ for those operations. A schedule is feasible, if the starting times of operations respect the precedence constraints induced by the operation sequences of jobs, by the finish-to-start precedence constraints between jobs, and if all the operations assigned to the same machine are performed in disjoint time intervals. The objective is to find a feasible schedule of minimum length or *makespan*.

4. Solution approach

The above scheduling problem is harder than job shop scheduling, due to the selection of routing alternatives for jobs, and alternative machines for the operations. Since job shop scheduling is a notoriously difficult scheduling problem for which several techniques have been developed over the past decades, the main challenge is to cope with the increased complexity of the problem. To this end, the problem is hierarchically decomposed into (1) *load balancing*, and (2) *machine selection* and *sequencing*. In the first phase the goal of load balancing is to choose a routing for each job from the set of alternatives. Once the routings have been chosen, the algorithm proceeds with the second phase, in which machines are assigned to operations and the operations get sequenced on the selected machines. In this second phase, the job routings are not modified.

4.1. Load balancing by selecting routing alternatives

In a feasible schedule there is a routing selected for each job, and a machine selected for each operation of the selected routings. In the *load balancing* problem, the precedence and sequencing constraints of the original scheduling problem are relaxed, and an optimal solution for this relaxed problem is sought. The load balancing problem is modeled by a mathematical program. Recall that each job j has a set of routing alternatives, where each routing alternative is a sequence of operations, and each operation has a set of eligible machines. Let $\Omega_{j,k}$ denote the set of all possible *machine assignments* to the operations of routing alternative $S_{j,k}$ of job j. Namely, any member ω of $\Omega_{j,k}$ specifies an eligible machine $\omega(o)$ for each operation o of $S_{j,k}$. With each routing alternative $S_{j,k}$ of each job j, and each machine assignment ω from $\Omega_{j,k}$, there is associated an $|M|$ dimensional vector $\pi(j,k,\omega) = (p_1(j,k,\omega),\dots, p_{|M|}(j,k,\omega))$, where $p_m(j,k,\omega)$ is the total processing time of those operations in the routing alternative $S_{j,k}$ with $\omega(o)=m$. Formally, $p_m(j,k,\omega)=\text{sum}(p_o(m)$: for all operation o in $S_{j,k}$ with $\omega(o)=m)$. For an example of an eligible assignment, see Figure 2.

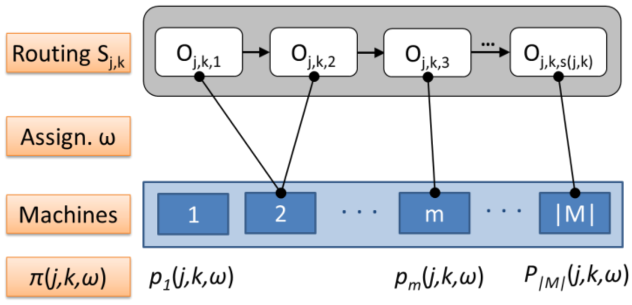

Figure 2. An eligible assignment ω of machines to operations in a selected routing $S_{j,k}$.

The mathematical program has binary decision variables $x_{j,k,\omega}$, where j indexes the jobs, k the routing alternatives of job j, and ω the eligible machine assignments to the operations of the routing alternative k of job j. In addition, there is one more variable, L, to express the maximum load of the machines. The program can be expressed shortly as

Minimize L
subject to
$$\text{sum}(p_m(j,k,\omega)\ x_{j,k,\omega} : \text{for all } (j,k,\omega)\ ) \le L, \text{ for all machines } m \tag{1}$$
$$\text{sum}(x_{j,k,\omega} : \text{for all } (k,\omega)) = 1, \text{ for all jobs } j \tag{2}$$
$$x_{j,k,\omega} \text{ is a binary variable, for all } (j,k,\omega). \tag{3}$$

Note that when referring to all (j,k,ω), all jobs, all of their routing alternatives, and all the eligible machine assignments to the operations of the routing alternatives are involved. The objective function expresses that the maximum load of the machines is to be minimized. The first constraint relates L to the load of every machine with respect to the selected routing alternatives and machine assignments. The second constraint expresses that for each job precisely one routing alternative and machine assignment must be chosen. In what follows this integer linear program is referred to as MIP-OLB.

An optimal solution of this mathematical program constitutes an optimal solution of the load balancing problem. The only drawback of the above problem formulation is that it may have exponentially many variables, since the size of the set $\Omega_{j,k}$ (the set of machine assignments to the operations of routing alternative $S_{j,k}$ of job j) may be exponential in the number of jobs and machines. Fortunately, there is an advanced technique of integer linear programming that enables solving mathematical programs even with billions of variables. Such a technique is called *column generation* that, instead of storing the column vectors of coefficients associated with every variable, generates them as needed in the course of optimization (this is where the name column generation stems from). Every linear program (LP) has a dual obtained by exchanging the roles of variables and constraints. Such a pair is called *primal-dual pair.* Moreover, by LP duality, a primal solution is optimal if and only if there is a dual solution of the same objective function value.

By exploiting this duality, column generation proceeds as follows. Firstly, an initial set of variables is selected, and the linear program restricted to these variables is solved to optimality. Then it has to be checked whether the current primal solution is optimal for the complete linear program. To this end, a column missing from the restricted primal, and having a negative reduced cost is sought. By duality, such a column corresponds to a violated dual constraint. The problem of finding a column with negative reduced cost is called the *pricing problem*. If such a column is found, it is added to the restricted primal program, which is then re-optimized. Otherwise, the current primal solution is optimal for the complete linear program. Columns which are inactive, i.e., not in the primal basis for several re-optimization steps, may be deleted to reduce the computational burden of the method. When there are also integer variables, column generation can be embedded in a branch-and-bound method: after finding an (sub)optimal solution of the linear relaxation of the integer linear program, branching occurs on integer variables or sets of integer variables taking fractional values. Notice that in the nodes of the branch-and-bound tree, in order to solve the node LP to optimality, new columns may have to be generated. Such a method is called *branch-and-price* [24].

To solve the LP relaxation of MIP-OLB, an initial set of columns is needed, as well as a method for solving the pricing problem. For each job j, the restricted linear program initially contains $a(j)$ columns, one for each routing alternative. Namely, for each routing alternative the machine assignment ω is chosen that gives the smallest total processing time. As for the pricing problem, each column of MIP-OLB consists of two parts: a vector $\pi(j,k,\omega)$ and the n-dimensional unit vector e_j which has a 1 in the j^{th} coordinate, and 0 elsewhere. The vector $\pi(j,k,\omega)$ corresponds to the coefficients of variable $x_{j,k,\omega}$ in the first $|M|$ constraints, and the unit vector e_j to that of the second set of $|J|$ constraints. When generating new columns for the restricted primal LP, for each job j, a set of at most $a(j)$ new columns are added, one for each routing alternative with negative reduced cost. The reduced cost $rc(j,k,\omega)$ of a column corresponding to machine assignment ω is w_j+sum$(-\pi(j,k,\omega)_m\ v_m : m=1,...,|M|)$, where v and w are the dual variables associated with constraints (1) and (2), respectively. The pricing problem for job j and routing alternative k is

PRICE(j,k): min{sum$(-\pi(j,k,\omega)_m\ v_m : m=1,...,|M|)$: over all ω}.

Now, the machine assignment ω is sought that gives the smallest sum$(-\pi(j,k,\omega)_m\ v_m : m=1,...,|M|)$ value. Such an assignment can be found by choosing for each operation o in the sequence $S_{j,k}$ the machine m giving the smallest $p_m(o)v_m$ value. Therefore, solving the pricing problem *PRICE*(j,k) takes $O(|M| \times \max |S_{j,k}|)$ time. The output is a vector $\pi(j,k,\omega^*)$ which corresponds to an optimal solution ω^*, and which can be added as a new column, along with the unit vector e_j to the restricted primal LP, provided $rc(j,k,\omega^*)$ is negative. For the structure of the matrix and examples of deleting and adding columns, see Figure 3.

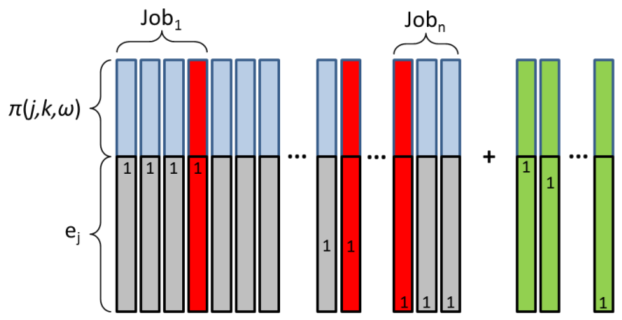

Figure 3. One phase of column generation: deleting unused columns (red) and adding new columns (green) of coefficients.

Having solved the LP relaxation of MIP-OLB by column generation, one gets a linear program containing a subset of columns of MIP-OLB, and an optimal solution for the LP relaxation in which some of the variables $x_{j,k,\omega}$ may take fractional values. Let *LP-OLB*$_{opt}$ denote the value of the optimal solution of the LP relaxation. Instead of proceeding with a full branch-and-price, which may be very time consuming due to the expensive column generation phase in search-tree nodes, the MIP-OLB is solved to optimality restricted only to those columns that have already been generated for solving the linear relaxation LP of MIP-OLB. For this purpose a standard MIP solver is used. At the end of this procedure the solution is optimal for the restricted MIP-OLB, but suboptimal for the complete MIP-OLB.

Let *MIP-OLB*$_{opt}$, and *R-MIP-OLB*$_{opt}$ denote the optimum value of MIP-OLB, and that of MIP-OLB restricted to the set of columns at the end of the column generation phase. The following relations apply between these values: *LP-OLB*$_{opt}$ ≤ *MIP-OLB*$_{opt}$ ≤ *R-MIP-OLB*$_{opt}$, and usually both inequalities are strict.

4.2. Resource assignment and sequencing by tabu search

The load balancing phase returns a routing and a machine assignment for each job. Subsequently, the routing alternatives cannot be changed, but the machine assignments are still subject to change in the course of resource assignment and sequencing, which is discussed in this section. Note that with fixed routing alternatives, the problem becomes the well-known flexible job shop scheduling problem, for which many methods have been published in the literature (see Section 2). Hence, the machine assignment of the load balancing problem will be used as the initial solution, but then it will be revised by the most successful techniques for solving FJSP.

Firstly, an initial schedule is built by inserting the jobs one-by-one into a growing schedule. The operations of a job are inserted following the sequence of the selected routing alternative on the machine assigned (temporarily) in the solution of the load balancing problem. The best position for an operation is chosen by evaluating the objective function (makespan) in all the feasible insertion points and choosing the most favorable one.

Once an initial schedule has been built, it is iteratively improved by the *tabu search* method of Mastrolilli and Gambardella [19]. This method reduces the *neighborhood* of a solution in a way that the reduced set still contains an optimal neighbor. In the course of tabu search, both the machine assignment and the operation sequences on the machines are subject to change. In every iteration, one operation is moved to a new position, either on the same, or on a new machine. The crux of the method is a quick evaluation technique for finding the best position on a machine, which allows performing more iterations than other methods using the same computing time (for more details, see [19]).

4.3. The complete procedure

To summarize, the main steps of the method are as follows:

1. Solve the LP relaxation of MIP-OLB by column generation. Let R-MIP-OLB denote MIP-OLB restricted to the set of columns at the end of the column generation procedure.
2. Solve R-MIP-OLB by a MIP solver.
3. Build an initial schedule.
4. Improve the initial schedule by tabu search.

In step 2, the solution of R-MIP-OLB may be terminated before finding an optimal solution, in which case one has a feasible, but not provably optimal solution for R-MIP-OLB. The computational experiments show that the gap between the best solution found for R-MIP-OLB with early termination, and *LP-OLB*$_{opt}$ is small (below 5%), and thus the procedure delivers a solution close to the optimum value *MIP-OLB*$_{opt}$.

5. Industrial case study and evaluation

5.1. Problem description

In this case study, component manufacturing for the energy industry has been investigated. In addition to the general features of this problem presented in Section 1, it can be characterized as follows: the factory has turning, drilling, and 3-axis milling machines, as well as advanced 5-axis machining centers and other cutting machines. The factory makes products of complex geometry in response of a fluctuating demand. Scheduling receives from production planning monthly cumulative target numbers of products and generates production schedules by using dispatching rules. However, jobs typically stack up in front of advanced machines that often become bottlenecks. This reduces manufacturing efficiency significantly. Therefore, the purpose of this study was to reveal the effectiveness of the proposed method by computational experiments in general, and to investigate the effect of machine load balancing on the production schedule in particular.

Figure 4 shows the shop floor image of the actual case study where two types of parts of a product are made from a cylindrical shape raw material. The workshop is segmented into a turning and a milling shop, with CNC machines with partly overlapping capabilities in each shop. Accordingly, the process plans of the parts are segmented into turning and milling type operations. The

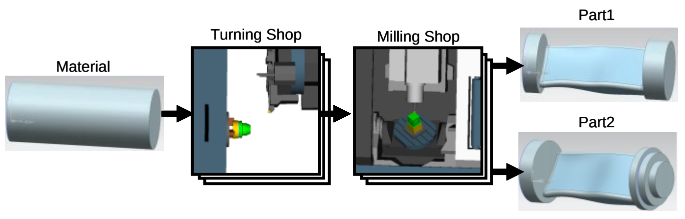

Figure 4. Production of a sample product.

plan alternatives consist of various operation sequences and assign multiple machines to the operations (see Table 1). All in all, the release of a production order of a product implies four jobs that are linked pairwise by finish-to-start precedence relations.

Table 1
Characteristics of the test instances

Part type	Process type	# of routings	# of operations
Part1	Turning	3,6,9,10	18-21
	Milling	3,6,9,10	30-32
Part2	Turning	3,6,9,10	18-21
	Milling	3,6,9,10	30-32

5.2. Computational experiments

The complete scheduling method was implemented in the C++ programming language, with the application of the Coin Branch and Cut (CBC) solver of COIN-OR [25] for solving the load balancing problem. The tests were run on a workstation with Debian Linux, and Intel XEON CPU, 2.4 GHz clock rate, and 2 GB memory.

Below, two kinds of solutions are compared. As a baseline solution, for each job the routing alternative with the shortest total processing time has been chosen, when for each operation the machine yielding the shortest processing time is assigned. This heuristic solution that complies with the actual practice in the factory under study is referred to as HLB. In the other variant called OLB the load balancing problem was solved for (semi)optimality using the method in Section 4.1. In the column generation phase the number of iterations (column generation + re-optimization) was set to the total number of routing alternatives of all jobs, and the subsequent MIP phase was stopped after 300 seconds. After the load balancing phase both methods proceeded with the schedule construction and improvement technique of Section 4.2. In both methods 20000 iterations and at most 600 seconds were allowed for the tabu search procedure. In total 80 test instances each with 120 jobs have been solved. Jobs had 3, 6, 9, or 10 routing alternatives together with the sets of ca. ten eligible machines per operations. The routings consisted of sequences of 18 to 32 operations. The sample workshop included 25 machines with various capabilities.

5.3. Evaluation

Experiments on each problem instance have confirmed that the novel method consistently outperforms the actual factory scheduling method. This was to be expected since it has a longer horizon and much broader scope for optimization than dispatching heuristics. The results unanimously improved as the number of routing alternatives increased.

The experiments have also pointed out the effectiveness, and in general, the key role of load balancing in the hierarchical solution process. Firstly, the novel column generation method was able to cope with large-scale instances, too. As an illustrative example, Figure 5 shows how in a particular test instance the value of maximum load was reduced as the column generation method advanced step by step. Note that the bound L decreased very rapidly at the beginning, and tailed off in the end of the column generation process. Figure 6 presents the final result of the load balancing stage for the same test case.

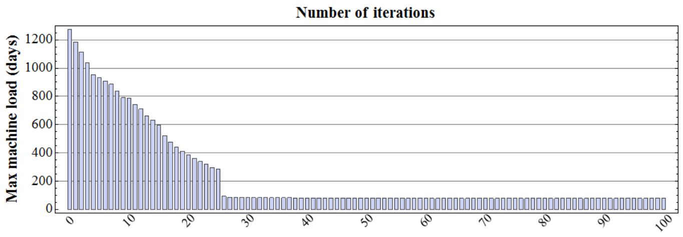

Figure 5. Decreasing value of L during column generation.

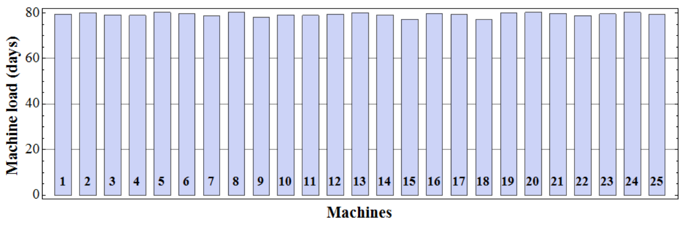

Figure 6. Loads of machines in a final integer solution.

Table 2 provides a detailed comparison of the relative strength of the methods. Let I_{OLB}, I_{HLB}, denote the makespan of the initial schedule obtained by the OLB, and HLB procedure, respectively. Likewise, let C_{OLB}, C_{HLB} denote the makespan of the best schedule in the end of the tabu search of the OLB, and the HLB procedure, respectively. The averages are taken over the 80 test instances. The first row of Table 2 shows that on average the makespan of the initial schedule obtained by OLB is only 3% worse than the best schedule obtained after tabu search by the HLB method. Moreover, the best schedule obtained by OLB is 23% better than that of the HLB procedure. Yet, the HLB procedure reduces the initial schedule length by 51%, whereas the OLB procedure improves the initial schedule length only by 26% during tabu search. The results suggest that the success of the OLB procedure is primarily determined by the load balancing algorithm.

Table 2
Comparison of solution methods

Comparative performance measure	Result
Average (I_{OLB}–C_{HLB})/ C_{HLB} [%]	3
Average (C_{HLB}–C_{OLB})/ C_{OLB} [%]	23
Average (I_{HLB}–C_{HLB})/ C_{HLB} [%]	51
Average (I_{OLB}–C_{OLB})/ C_{OLB} [%]	26

In fact, load balancing prepares the ground for the subsequent local search that can efficiently squeeze the set of jobs into as small a timeframe as possible provided it can work with properly selected routings. As for the continuation of the illustrative example, see the Gantt charts on Figure 7 and Figure 8 (horizontal axis indicates manufacturing timeline, vertical axis the machines, operations of the same job have the same color). The first chart shows the final result of tabu search after the HLB procedure (i.e., with the application of the heuristic load balancing method), while the next chart presents solution of the same problem with the application of the optimized load balancing method. The particular schedules are compared in Table 3 .

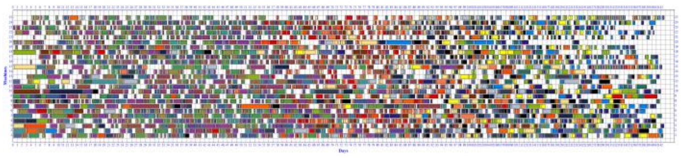

Figure 7. Schedule generated with heuristic load balancing.

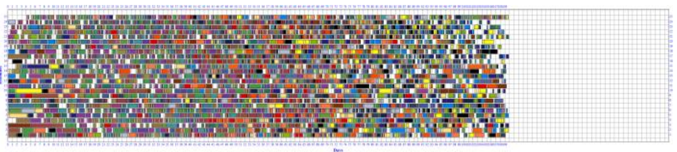
Figure 8. Schedule generated with optimized load balancing.

Table 3
Performance comparison of the illustrative test example

Performance measure		**Result**
Makespan [days]	with heuristic load balancing	143
	with optimized load balancing	110
Average utilization rate [%]	with heuristic load balancing	70
	with optimized load balancing	86

Finally, reports on the solution of similar problems are scarce in the literature. The latest results are presented in [17] where tests have been run on a much smaller scale (5 x 5, 10 x 10, 20 x 5 jobs and machines, with 2 or 3 routing alternatives) but higher time limit (3600s). Results obtained indicate that a single MIP alone is not amenable for solving the problems on an industrial scale.

6. Conclusions

This research focused on optimizing the manufacturing efficiency of factories which have many types of machines with overlapping capabilities. The key idea was exploiting the potential of alternative routings provided by flexible CNC machines. However, in scenarios of real-life relevance, this opportunity creates extremely large decision problems. Hence, a new solution method has been developed that applies well-proven principles of problem decomposition, relaxation and iterative improvement. The crux of the problem, load balancing, could be solved by a new column generation technique that offered a good selection of routing alternatives for the jobs. The effectiveness of the method has been proved in a real-life case study. The results suggest that mathematical programming has powerful principles and modern techniques to face combinatorial complexity involved by scheduling problems of industrial scale. A special concern was to trade off computation time with solution quality. The method gives initial solutions in a short time, and generates an improving series of solutions as time passes. The response times facilitate interactive decision making, hence the method can be incorporated into a new, high-performance manufacturing execution system (MES) of the company. This work will be complemented by an automated process planner still under development. Departing from the design models of the parts and the description of available processes and resources, it will generate executable process plan alternatives optimized for traditional engineering criteria (like minimal setups and processing times) and maximal variety.

Acknowledgements

The Hungarian authors acknowledge the support of the OTKA grant K76810, the NKTH grant OMFB-01638/2009, as well as the TÁMOP-4.2.1/B-09/1/KMR-2010-0002 grant.

References

[1] Crowston W, Thompson GL (1967) Decision CPM: A Method for Simultaneous Planning, Scheduling, and Control of Projects. *Operations Research* 15:407–426.

[2] Iwata K, Murotsu Y, Oba F, Okamur K (1980) Solution of Large-Scale Scheduling Problems for Job-Shop Type Machining Systems with Alternative Machine Tools. *CIRP Annals - Manufacturing Technology* 29/1:335–338.

[3] Chryssolouris G, Chan S (1985) An Integrated Approach to Process Planning and Scheduling. *CIRP Annals - Manufacturing Technology* 34/1: 413–417.

[4] Larsen NE, Alting L (1992) Dynamic Planning Enriches Concurrent Process and Production Planning. *International Journal of Production Research* 30/8:1861–1876.

[5] Phanden RK, Jain A, Verma R (2011) Integration of Process Planning and Scheduling: A State-of-the-Art Review. *International Journal of Computer Integrated Manufacturing* 24/6:517–534.

[6] Váncza J, Monostori L, Lutters D, Kumara SRT, Tseng M, Valckenaers P, Van Brussel H (2011) Cooperative and Responsive Manufacturing Enterprises. *CIRP Annals - Manufacturing Technology* 60/2:797-820.

[7] Monostori L, Váncza J, Kumara SRT (2006) Agent-Based Systems for Manufacturing. *CIRP Annals - Manufacturing Technology* 55/2:697-720.

[8] Ueda K, Vaario J, Ohkura KH (1997) Modelling of Biological Manufacturing Systems for Dynamic Reconfiguration. *CIRP Annals - Manufacturing Technology* 46/1:343–346.

[9] Márkus A, Kis T, Váncza J, Monostori L (1996) A Market Approach to Holonic Manufacturing. *CIRP Annals - Manufacturing Technology* 45/1:433–436.

[10] Tseng MM, Lei M, Su C, Merchant ME (1997) A Collaborative Control System for Mass Customization Manufacturing. *CIRP Annals - Manufacturing Technology* 46/1:373–376.

[11] Ueda K, Márkus A, Monostori L, Kals HJJ, Arai T (2001) Emergent Synthesis Methodologies for Manufacturing. *CIRP Annals – Manufacturing Technology* 50/2: 535–551.

[12] Valente A, Carpanzano E (2011) Development of Multi-Level Adaptive Control and Scheduling Solutions for Shop-Floor Automation in Reconfigurable Manufacturing Systems. *CIRP Annals - Manufacturing Technology* 60/1:449–452.

[13] Scholz-Reiter B, Rekersbrink H, Görges M (2010) Dynamic Flexible Flow Shop Problems – Scheduling Heuristics vs. Autonomous Control. *CIRP Annals - Manufacturing Technology* 59/1:465–468.

[14] Brandimarte P (1999) Exploiting Process Plan Flexibility in Production Scheduling: A Multi-Objective Approach. *European Journal of Operational Research* 114/1:59–71.

[15] Kis T (2002) Job-shop Scheduling with Processing Alternatives. *European Journal of Operational Research* 151:307–332.

[16] Valckenaers P, Van Brussel H (2005) Holonic Manufacturing Execution Systems. *CIRP Annals - Manufacturing Technology* 54/1:427–432.

[17] Özgüven C, Özbakır L, Yavuz Y (2010) Mathematical Models for Job-Shop Scheduling Problems with Routing and Process Plan Flexibility. *Applied Mathematical Modelling* 34:1539–1548.

[18] Brandimarte P (1993) Routing and Scheduling in a Flexible Job Shop by Tabu Search. *Annals of Operations Research* 41:157–183.

[19] Mastrolilli M, Gambardella LM (2000) Effective Neighborhood Function for the Flexible Job Shop Problem. *Journal of Scheduling* 3:3–20.

[20] Rossi A, Dini G (2007) Flexible Job-Shop Scheduling with Routing Flexibility and Separable Setup Times Using Ant Colony Optimisation Method. *Robotics and Computer-Integrated Manufacturing* 23:503–516.

[21] Yu X, Ram B (2006) Bio-Inspired Scheduling for Dynamic Job Shops With Flexible Routing and Sequence-Dependent Setups. *International Journal of Production Research* 44/22:4793–4813.

[22] Drótos M, Erdős G, Kis T (2009) Computing Lower and Upper Bounds for a Large-Scale Industrial Job Shop Scheduling Problem. *European Journal of Operational Research* 197/1:296–306.

[23] Usher JM (2003) Evaluating the Impact of Alternative Plans on Manufacturing Performance. *Computers & Industrial Engineering* 45:585–596.

[24] Barnhart C, Johnson EL, Nemhauser GL, Savelsbergh MWP, Vance PH (1998) Branch-and-Price: Column Generation for Solving Huge Integer Programs. *Operations Research* 46/3:316–329.

[25] The Computational Infrastructure for Operations Research (COIN-OR). http://www.coin-or.org/, accessed on 30.11.2011.