The 7th International Symposium on Operations Research and Its Applications (ISORA'08)
Lijiang, China, October 31–Novemver 3, 2008

A Hybrid Metaheuristic for the Lot Streaming Problem in Job Shops

Udo Buscher* Liji Shen†

Chair of Industrial Management, Department of Business Administration and Economics
Dresden University of Technology, Germany

Abstract This paper presents a hybrid metaheuristic for solving the lot streaming problem in job shop production systems. In contrast with conventional tabu search implementations, where the disjunctive graph is adopted to represent the scheduling problem, our tabu search approach is based on permutation with repetition. Moreover, a specific procedure inspired by genetic algorithms is incorporated to seek better sublot sizes for a given schedule. By combining two metaheuristics, good solutions can be found in reasonable computing times and experimental results confirm the remarkable advantage of lot streaming.

1 Introduction

This paper concentrates on developing a hybrid metaheuristic to solve the lot streaming problem in job shop scheduling systems. The *job shop* problem can be briefly described as follows: A set of jobs and a set of machines are given. Each job consists of a sequence of operations, which need to be processed during an uninterrupted time period of a given length on a given machine. A *schedule* is an allocation of the operations to time intervals on the machines. The objective is to find a schedule of minimum length, which is referred to as *makespan*. This class of problems is not only NP-hard but also belongs to the most difficult combinatorial optimization problems.

By applying *lot streaming*, a job can be regarded as a *production lot* containing identical items. Lot streaming actually represents the concept of dividing a lot into multiple smaller sublots, so that they can be transferred to the next stage immediately upon their completion. As a result of operation overlapping, both idling time of machines and makespan can be substantially reduced [6].

In this paper, we develop a hybrid metaheuristic by combining tabu search and genetic algorithms. The remainder of the paper is organized as follows: In the subsequent section, we describe the specific representation based on permutation with repetition. Section 3 addresses the determination of schedules with tabu search techniques. In section 4, sublot sizes are further varied by employing genetic algorithms. Section 5 provides a detailed analysis of computational results. Brief conclusions are summarized in section 6.

*buscher@rcs.urz.tu-dresden.de
†liji.shen@mailbox.tu-dresden.de

2 Problem Representation

It is assumed that n jobs and m machines are given. Each job is divided into s sublots and consists of m operations. The total number of operations can thus be expressed as $n \cdot m \cdot s$. For each operation v_{ijk} there is a job J_i to which it pertains, a sublot S_{ij} with which it is associated, a machine M_k on which it must be processed and a unit processing time $p^u_{ijk} \in \mathbb{N}$.

The representation of the lot streaming problem is based on *permutation with repetition* (PwR), which is introduced by [2] and originally applied to standard job shop problems. In a permutation, each element represents an individual operation. All operations v_{ijk} belonging to the same sublot S_{ij} are denoted by the same symbol A_{ij} and therefore, A_{ij} appears exactly m times. Instead of indicating a concrete operation, each A_{ij} is interpreted according to the order of occurrence in the associated permutation. Due to the specific assignment of operations, infeasible schedules are automatically excluded.

As an illustration, we consider an example with 2 jobs and 2 machines. Each job is composed of 2 operations and divided into 2 sublots. Job J_1 is to be processed on machine M_1 first and Job J_2 on M_2. Suppose the following permutation with repetition is given:

$$[A_{21} \quad A_{11} \quad A_{22} \quad A_{12} \quad A_{12} \quad A_{11} \quad A_{22} \quad A_{21}].$$

Since each sublot contains 2 operations, there are 4 different elements in the permutation and each of them emerges exactly twice. For instance, the first A_{21} corresponds to the operation of the first sublot of job J_2 (v_{212}), which is to be scheduled on machine M_2, whereas the second A_{21} represents operation v_{211}. Overall, the interpretation of the permutation is illustrated as follows:

PwR:	A_{21}	A_{11}	A_{22}	A_{12}	A_{12}	A_{11}	A_{22}	A_{21}
	↓	↓	↓	↓	↓	↓	↓	↓
Operation:	v_{212}	v_{111}	v_{222}	v_{121}	v_{122}	v_{112}	v_{221}	v_{211}
	↓	↓	↓	↓	↓	↓	↓	↓
Machine:		M_1		M_1			M_1	M_1
	M_2		M_2		M_2	M_2		

The associated schedule is depicted in figure 1.

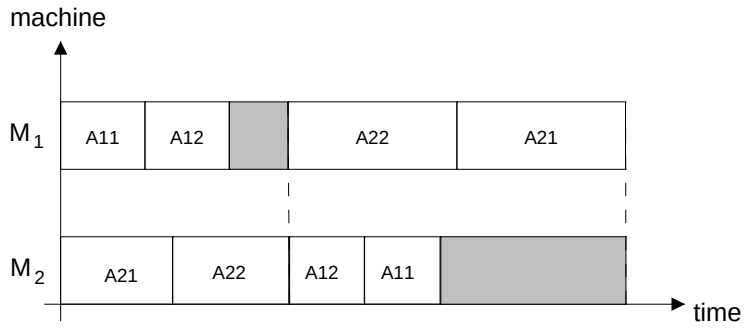
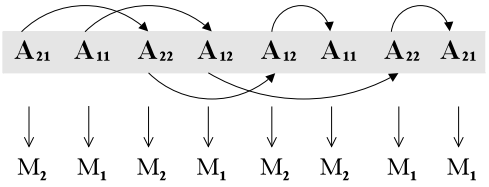


Figure 1: The corresponding GANTT-Diagram of PwR

3 Determining Schedules Based on Tabu Search

The determination of schedules is based on tabu search techniques. Subsequently, we present our implementation of the fundamental elements as well as some enhancements of the basic version of tabu search. In the study of [5], a move is defined as adjacent pairwise interchange. However, a close examination of the resulting neighbours reveals that a vast majority of unfruitful moves are embedded in this neighbourhood structure. Therefore, the move in our tabu search implementation is specified as interchanging 2 elements of closest distance in the permutation, which are to be scheduled on the same machine. For example, moves of the given permutation are illustrated as follows:



Note that the total number of moves for a permutation amounts to $(n \cdot m \cdot s - m)$ at each iteration. As a result, the neighbours to be evaluated disproportionally increase with the problem size. In order to reduce the computational burden, only a random sample of 40% neighbours is investigated.

Moreover, it is of importance to point out that different neighbours often provide the same smallest makespan, especially when the problem size increases. In this case, a function is invoked to choose the ultimate neighbour at random. This procedure for selecting the best neighbour avoids unfavourable concentration during the search process.

At each iteration the two elements involved in the performed move are memorized in the tabu list, regardless of their previous precedence relation.

In order to properly determine the tabu tenure, *Reactive tabu search* is incorporated into our algorithm. This concept was first introduced by [1] to solve the quadratic assignment problem. As an extension to the standard tabu search procedure, a simple mechanism is integrated to adapt the tabu tenure in accordance with the evolution of the search process. Compared to the original version of reactive tabu search, modifications and refinements are conducted regarding the characteristics of lot streaming problems.

If solutions are frequently repeated during the search, which implies the existence of cycles, reactive tabu search is activated. After a certain number (REP) of repetitions of a solution, the tabu tenure is increased by INC and this particular solution together with the iteration number are stored in the set CYC. If this solution emerges again, the tabu tenure is again increased by INC and the iteration number is saved. Moreover, the mean value of the stored iteration numbers of the same solution is calculated, after which, if no repetition takes place, the tabu tenure is decreased by DEC. If the size of CYC or the tabu tenure exceeds a prescribed number (MaxC or MaxT), diversification is initiated.

In order to escape from local optima and achieve sufficient diversity, we implement an efficient method of *diversification*. When diversification is activated, elements of the current permutation are interchanged with a randomly chosen partner. After adequate iterations (*nms*), a considerable distance to local optima can be reached, which complies exactly with the purpose of diversification.

The search will be terminated if a prescribed number of iterations (MaxIte) are executed.

The best permutation and its makespan value are then transferred to the next stage of the algorithm.

4 Varying Sublot Sizes with Genetic Algorithms

Starting with equal-sized sublots, an approach inspired by genetic algorithms for further varying sublot sizes is incorporated. In order to constitute the initial generation, we implement a specific procedure for calculating sublot sizes. To each sublot S_{ij} a factor a_{ij} is assigned, which is chosen randomly among $Rand(1,\ldots,1000)$. Sublot sizes are then determined according to:

$$X_{ij} = \frac{a_{ij}}{\sum_{j=1}^{s} a_{ij}} \cdot D_i \quad \forall i = 1,\ldots,n, j = 1,\ldots,s, \tag{1}$$

where D_i denotes the given demand of job J_i. As shown in figure 2, each string actually represents a complete set of sublot sizes. Members of a population can thus be generated by performing this procedure repeatedly. A prescribed number (P) of pairs of parents

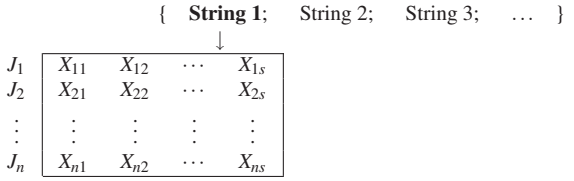


Figure 2: Constituting Initial Generation

are then arbitrarily selected to reproduce offspring. It should also be pointed out that the adopted *2-cut-point crossover* ensures the feasibility of new solutions.

As illustrated by the example with 4 jobs and 16 sublots in table 1, sublot sizes of

Table 1: Varying Sublot Sizes with 2-cut-point crossover

Parent 1

Job J_i	X_{i1}	X_{i2}	X_{i3}	X_{i4}
J_1	3	3	3	3
J_2	* 3	3	3	3
J_3	3	3	3	3 *
J_4	3	3	3	3

Parent 2

Job J_i	X_{i1}	X_{i2}	X_{i3}	X_{i4}
J_1	1	4	6	1
J_2	* 2	2	6	2
J_3	1	4	3	4 *
J_4	2	7	2	1

Offspring 1

Job J_i	X_{i1}	X_{i2}	X_{i3}	X_{i4}
J_1	3	3	3	3
J_2	2	2	6	2
J_3	1	4	3	4
J_4	3	3	3	3

Offspring 2

Job J_i	X_{i1}	X_{i2}	X_{i3}	X_{i4}
J_1	1	4	6	1
J_2	3	3	3	3
J_3	3	3	3	3
J_4	2	7	2	1

* denotes the cut points.

job J_2 and J_3 are switched after performing 2-cut-point crossover. Note that this type

of combination also enables interchanges of sublot sizes concerning different number of jobs. However, in order to ensure the fulfilment of demands, cut points must not be inserted between sublots of the same job. Therefore, the number N of generated offspring can be written as:

$$N = P(n^2 + n - 2).$$

Based on the schedule determined by tabu search, each individual (set of sublot sizes) is then evaluated in terms of the newly calculated makespan. In consequence, parents with inferior values are replaced by better offspring and the population remains constant.

5 Computational Results

The proposed algorithm is coded in C++ and runs on an AMD Athlon64 2450 MHz PC with 4GB memory. Table 2 presents the determination of relevant parameters, most of which are specified regarding the associated problem size. The tested instances are the well-known job shop benchmark problems from [4], which range from 10 jobs on 5 machines to 30 jobs on 10 machines.

To evaluate the solution quality, we adopt the measurement *LB* introduced in [3]. The de-

Table 2: Parameter Settings

Tabu tenure	MaxI	P	MaxPop	MaxIte
$\lceil (nm)^{\frac{1}{2}} \rceil$	$10,000$	ns	ns	$(nms)^2$
Parameter related to reactive tabu search				
REP	MaxC	MaxT	INC	DEC
3	$\lceil (nms)^{\frac{1}{2}} \rceil$	$\lceil 4(nm)^{\frac{1}{2}} \rceil$	$2(s+1)$	$2(s-1)$

termination of *LB* implies that all operations are scheduled regardless of their precedence constraints and all machines are fully occupied. Therefore, This value represents a lower bound of makespan.

Due to the integration of random elements, all tested instances are run 4 times. As presented in table 3, there are some instances, for which optimal solutions of the standard job shop problem are equal to their lower bounds. In these cases, despite the complex interaction between sublots and machines, our algorithm is still able to reach the lower bound while adopting different number of sublots. On the other hand, the benefits of lot streaming is clearly shown by the remaining instances. Obviously, makespan is considerably improved with the application of 2 sublots and is further reduced as the number of sublots increases. Moreover, table 4 provides the average computing time required for solving the corresponding problem set. In general, computing time increases in proportion to the problem size. Nevertheless, the largest problem class with 1200 operations can still be solved in reasonable computing time.

6 Conclusion

In this paper, we present a hybrid metaheuristic for solving the lot streaming problem in job shop scheduling systems. Whereas tabu search is employed to determine schedules,

Table 3: Test Results of Lot Streaming

$n \cdot m$	Instance	C_{max}^{opt}	LB	ΔLB (%)			
				$s=1$	$s=2$	$s=3$	$s=4$
$10 \cdot 5$	la01	666	666	0,00	0,00	0,00	0,00
	la03	597	588	3,06	0,00	0,00	0,00
	la04	590	537	9,87	2,79	1,86	1,40
$15 \cdot 15$	la06	926	926	0,00	0,00	0,00	0,00
	la09	951	951	0,00	0,00	0,00	0,00
	la10	958	958	0,00	0,00	0,00	0,00
$20 \cdot 5$	la11	1222	1222	0,00	0,00	0,00	0,00
	la12	1039	1039	0,00	0,00	0,00	0,00
	la14	1292	1292	0,00	0,00	0,00	0,00
$10 \cdot 10$	la16	945	660	43,18	16,21	6,82	5,30
	la18	848	623	36,12	8,19	1,93	0,44
	la19	842	685	22,92	0,00	0,00	0,00
$15 \cdot 10$	la23	1032	1032	0,00	0,00	0,00	0,00
	la24	935	857	4,28	0,00	0,00	0,00
	la25	977	864	13,08	3,94	0,96	0,23
$30 \cdot 10$	la31	1784	1784	0,00	0,00	0,00	0,00
	la32	1850	1850	0,00	0,00	0,00	0,00
	la34	1721	1721	0,00	0,00	0,00	0,00
$15 \cdot 15$	la36	1268	1028	23,35	10,36	4,64	3,23
	la39	1233	1012	24,90	6,82	4,11	3,09
	la40	1222	1027	18,99	3,65	1,36	1,02

sublot sizes are calculated with the application of genetic algorithms. Due to the adoption of an operation-based representation, our tabu search implementation differs distinctively from traditional tabu search approaches proposed in the literature. Furthermore, the determination of sublot sizes inspired by genetic algorithms assists in increasing the solution quality. As a result of combining two metaheuristics, good solutions can be found in reasonable computing times and experimental results confirm the remarkable benefits of lot streaming.

Based on permutation with repetition, not only infeasible solutions are automatically excluded, the calculation of makespan is also accelerated. Therefore, further researches are encouraged to explore other properties and advantages resulted from this specific representation.

Table 4: Computing Time

$n \cdot m$	Computing time (sec.)			
	$s=1$	$s=2$	$s=3$	$s=4$
10·5	6	31	63	98
15·5	11	82	151	240
10·10	19	100	211	360
20·5	19	169	308	481
15·10	18	118	453	824
15·15	112	475	1311	2633
30·10	166	1448	2691	5113

On the other hand, it is also desirable to develop more sophisticated procedures based on genetic algorithms to enable a more efficient variation of sublot sizes.

References

[1] Battiti R., Giampietro T. The reactive tabu search. ORSA Journal on Computing 1994;6; 126-140.

[2] Bierwirth C., Mattfeld D., Kopfer H. On permutation representations for scheduling problems. In: Voigt H. et al. (Eds), Parallel problem solving from nature – PPSN IV. Springer: Berlin; 1996. p.310-318.

[3] Dauzère-Pérès S., Lasserre J. Lot streaming in job-shop scheduling. Operations Research 1997;45; 584-595.

[4] Lawrence S. Resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques (supplement). Graduate School of Industrial Administration, Garnegie-Mellon University, Pittsburgh, PA, 1984 – Technical Report.

[5] Ponnambalam S., Aravindan P., Rajesh S. A tabu search algorithm for job shop scheduling. International Journal of Advanced Manufacturing Technology 2000;16; 765-771.

[6] Glass C., Gupta J., Potts C.N. Lot streaming in three-stage production processes. European Journal of Operational Research 1994;75; 378-394.