A Fast Heuristic for a Lot Splitting and Scheduling Problem of a Textile Industry

Pimentel, Carina*. Alvelos, Filipe**
Carvalho, J. M. Valério *. Duarte, António *****

*Centro de Investigação Algoritmi, Universidade doMinho, Braga, Portugal
(Tel:+351 253 604 756; e-mail: carina@dps.uminho.pt).
** Centro de Investigação Algoritmi/Departamento de Produção e Sistemas, Universidade doMinho, Braga, Portugal
(Tel:+351 253 604 751; e-mail: falvelos@dps.uminho.pt).
*** Centro de Investigação Algoritmi/Departamento de Produção e Sistemas, Universidade doMinho, Braga, Portugal
(Tel:+351 253 604 744; e-mail: vc@dps.uminho.pt).
***** Centro de Investigação Algoritmi, Universidade doMinho, Braga, Portugal /Departamento de Gestão Industrial, Instituto Politécnico de Bragança, Bragança, Portugal
(Tel:+351 273 303 143; e-mail: aduarte@ipb.pt).

Abstract: In this paper we address a lot splitting and scheduling problem of a Textile factory that produces garment pieces. Each garment piece is made of a set of components that are produced on the knitting section of the company. The problem consists of finding a weekly production plan for the knitting section, establishing the quantities to produce of each component (organized in one or several lots), and where and when (starting/completion times) to produce them. The main contribution of this work is the development of a constructive heuristic that generates automated knitting scheduling plans. The heuristic produces solutions very fast for a set of randomly generated instances based on real world data.

1. INTRODUCTION

In this paper we present a procedure for a practical lot splitting and scheduling problem of a Textile company. The company produces fine knitted goods, such as cardigans, pants, dresses, sweaters and scarves. Each product, defined by a piece of cloth and size, is made up of a set of components, which are knitted in a group of identical parallel machines. Every Friday, a set of production orders are sent to the knitting manager. The production orders contain the set of garment pieces to be considered in the next knitting scheduling plan, as well as the associated set of components. Additionally, the production orders have information about the quantity ordered by the customer and the due date of each product.

The knitting manager is responsible for the development of a weekly production plan for the knitting section, taking into account all the production orders. Nowadays, these plans are developed manually, based on common sense rules and on the several years of experience of the knitting manager. The main contribution of this work is the development of a fast algorithm that generates automated knitting scheduling plans. The developed algorithm solves a lot splitting problem and an assignment and scheduling problem simultaneously. In the lot splitting problem, the number of components requested is split into lots of different sizes as a way of speeding up the production process. In the assignment and scheduling problem each of those lots is assigned to a given machine and its starting/completion times are determined. Two or more lots of a given component can be produced independently in more than one machine, at the same time or in different times, but a given machine can only process one lot at a time. Moreover, two or more lots of the same component may be produced in the same machine, with lots of other components or with empty intervals between them. As the lot splitting decisions are taken at the same level and in coordination with the assignment and scheduling decisions, the quality of the solutions is increased.

The knitting section has three groups of identical parallel machines. The characteristic that defines a group is a gauge. The gauge is associated with the thickness of the yarns and with the type of needles existent in the machine. There is a unique relationship between the gauge and the yarn. A given product will then be associated with only a gauge. The factory has three gauges, so three scheduling plans must be prepared. The machines are identical, since they take the same amount of time to produce a unit of a given component. There is a compatibility matrix between the machines of a given gauge and the components of that gauge. This compatibility matrix is needed because of technical characteristics of the components and of the machines. In addition, each machine has a given release date.

One important objective is to develop scheduling plans that minimize work-in-process inventory. Besides, on-time delivery of products is very important. Being so, we use the following two measures to evaluate a scheduling solution: (1) total tardiness of products and (2) total deviations occurred during production of each product. The total deviation of a given product is the sum of all the absolute deviations of each component lot completion time and the completion time of the last component lot. Moreover, the completion time of the last component lot is the product completion time.

In summary, our lot splitting and scheduling problem has the following characteristics: identical parallel machines, arbitrary demands and due dates, associated with products, a

compatibility matrix between machines and components, unit production times associated with components and machine release dates. To the best of our knowledge, no research has ever been published dealing with this problem when the objective is to minimize total deviations occurred during production of each product.

Our problem is to some extent related with the classical parallel machine scheduling problem (PMSP), in which there are *n* jobs to schedule in *m* machines aiming at optimizing a certain performance measure, but there are two important differences: (1) in our problem a given job (component) can be split into several lots of smaller size and processed in more than one machine simultaneously, while in PMSP no splitting or preemption of jobs can occur; (2) in our problem, a job (product) is divided into several sub-jobs (components) that are linked/related to each other because the job completion time depends of the completion times of all the sub-jobs, while in PMSP jobs are independent of each other. Cheng and Sin (1990) and Mokotoff (2001) survey the research contributions to the PMSP, both for enumerative algorithms and for approximate algorithms.

Xing and Zhang (2000) show that the identical parallel machine scheduling problem with jobs splitting, without setup times and with objective to minimize total tardiness (problem P/split/ΣT_j according to the three-field classification $\alpha/\beta/\gamma$ introduced by Graham et al. (1979)) is NP-hard. As our problem is an extension of the previous one, it is also NP-hard.

Yalaoui and Chu (2003) and Tahar et al. (2006) developed a two step heuristic algorithm for the identical parallel machine scheduling problem with job splitting and with sequence dependent setup times, aiming at minimizing the makespan. In the first step the problem is reduced into a single machine scheduling problem with sequence dependent setups and transformed into a travelling salesman problem that they solve using Little's method. In the second step Yalaoui and Chu (2003) try to improve the solution obtained in step one using a step by step procedure, taking into account setup times and job splitting, while Tahar et al. (2006) use a linear program to determine the size of the lots. The main differences between our problem and the one studied by Yalaoui and Chu (2003) and Tahar et al. (2006) are that: i) they minimize the makespan, while we consider the minimization of a function that involves total tardiness and the deviation between the completion time of a product and the completion times of all the component lots of that product; ii) they consider sequence dependent setups, while we do not and iii) they consider that all the machines can process all the jobs, while we restrict job assignments to specific machines.

Sheen and Liao (2007) present a network flow technique to solve a preemptive scheduling problem with identical parallel machines with availability constraints. Their goal is to minimize the maximum lateness. In their problem, each job can only be processed in specific machines. They solve this problem using a series of maximum flow problems. They propose a polynomial time two-phase binary search algorithm to verify the feasibility of the problem and to solve the scheduling problem optimally if a feasible schedule exists. This problem is related to ours, but there are two important differences: in our problem, a job can be split into several lots, while in their problem a job can be preempted (but can not be processed at the same time in different machines); and the objectives are different.

The remainder of this paper is structured as follows: in Section 2 a list scheduling constructive heuristic, which explores the specific characteristics of the practical problem, is developed, and in Section 3 an illustrative example is presented. In Section 4, the computational experiments are presented and finally, in Section 5, the main conclusions of this work are summarized.

2. LIST SCHEDULING ALGORITHM

In this section a list scheduling algorithm for the lot splitting and scheduling problem defined in Section 1 is presented. A list scheduling (LS) algorithm is a constructive heuristic that determines a schedule for a given ordering of jobs (Hurink and Knust, 2001). In a LS algorithm, a schedule is obtained in two steps. In the first step an ordered list of jobs is created according to some pre-defined priorities. After that, in a second step, the jobs of the ordered list are iteratively selected one by one, and assigned and scheduled in a given machine. The machine is selected from the set of available parallel machines, using pre-defined criteria. Our LS heuristic performs three steps. In step 1 an ordered list of products is created. In step 2, an ordered list of components is created based on the list defined in step 1. Finally, in step 3, the components are selected one by one and for each component one or more machines are selected to schedule the component under analysis, following the order defined in step 2. A detailed description of the LS algorithm is presented below.

Step 0. Initialization: consider the set of products *N*, the set of components *J*, the set of machines *M* and the set of components that belong to product *n*, *S(n)*. Let D_n be the demand of product *n*, d_n the due date of product *n*, r_m the ready time of machine *m*, a_j the unit production time of component *j*, b_{jm} a compatibility indicator that takes value 1 if component *j* can be processed in machine *m* and takes value 0 otherwise and f_{jn} the number of units of component *j* required to produce one unit of product *n* ($n \epsilon S(n)$).

Step 1. Build ordered list of products: sort the set of products, *N*, in increasing order of due date d_n. To break ties, choose the product *n* with the lowest total number of compatible machines. The total number of compatible machines is given by the sum of compatible machines of each component *j* that belongs to product *n* ($\sum_{j \in J | j \in S(n)} \sum_{m \in M} b_{jm}$). To break ties, choose the product *n* with higher total unit production time. The total unit production time of a product *n* is the sum of unit production times of all the components that belong to that product ($\sum_{j \in J | j \in S(n)} a_j$). To break ties, select arbitrarily a product *n*.

Step 2. Build ordered list of components: for each product *n* of the ordered list defined in step 1, do:

sort the components *j* that belongs to product *n* in increasing order of number of compatible machines. The number of compatible machines of a given component *j* is given by $\sum_{m \in M} b_{jm}$. To break ties, choose the component *j* with higher unit production time. To break ties, select arbitrarily one of the components *j* that belongs to product *n*.

Step 3. Assignment and scheduling of components: for each component *j* of the ordered list defined in step 2, do:

repeat while total unscheduled production time of component *j* (given by $D_n \times a_j \times f_{jn} \,|\, j \in S(n)$) is greater than zero:

Assignment: select the machine compatible with component *j* that allows scheduling it closest to its objective date. If *j* is the first component lot of product *n* to be scheduled, its objective date is equal to the due date of product *n*, to which component *j* belongs. If *j* is the first component of product *n* to be scheduled, but one or more lots of component *j* are already scheduled or if *j* is not the first component of product *n* to be scheduled, its objective date will be equal to the last completion time (considering all the lots of product *n* already scheduled). Component *j* will be scheduled in the selected machine in the free interval closest to the objective date. To break ties, *i. e.*, if in more than one compatible machine the free interval closest to the objective date ends at the same time, choose the machine with more idle time. The idle time of a machine is the sum of all its free intervals, from its release date, r_m, until the completion time of the free interval that is closest to the objective date. To break ties, select arbitrarily one of the machines.

Scheduling: schedule component *j* in the selected machine. If the length of the free interval closest to the objective date (in the selected machine) is smaller than the total unscheduled production time of component *j*, schedule component *j* in that interval, fully occupying the interval, and update the unscheduled production time of component *j*. However, if the length of the free interval closest to the objective date (in the selected machine) is greater than or equal to the total unscheduled production time of component *j*, schedule the total unscheduled production time of component *j* in that interval, and update the total unscheduled production time of component *j* to zero.

If it is not possible to schedule component *j* in any of the compatible machines before its objective date, meaning that all the compatible machines with component *j* are fully occupied until the objective date, that component will be late. In that case, divide the unscheduled production time of component *j* by the number of compatible machines with component *j* (getting a number of lots equal to the number of compatible machines) and schedule each of the lots in each of the compatible machines, closest to the objective date. In this case the objective date is equal to the due date of product *n*, to which component *j* belongs.

The worst-case computational complexity of LS algorithm is determined in step 3, and is O (JK^2M), where J is the number of components, K is the maximum number of lots of a machine, and M is the number of machines. Steps 1 and 2, corresponds to sorting two lists.

3. ILLUSTRATIVE EXAMPLE

Consider a problem with five products that must be scheduled, at most, in five machines, in the next 48 hours. The data associated with this example is presented in Table 1 and in Table 2.

Table 1. Illustrative example data

Product	Due date (hours)	Component	Compatible machines	Unit processing time (minutes)	Total processing time (hours)
CM1	24	CM1F	0,1,2,3,4	1	40
		CM1C	0,1,2,3,4	0.9	36
CM2	48	CM2F	0,3,4	1	33.33
		CM2C	0,3,4	0.9	30
CM3	24	CM3F	0,1,2,3	1	8.33
		CM3C	0,1,2,3	0.9	7.5
		CM3M	0,1,2,3,4	0.6	10
CS1	24	CS1F	0,2,3	1	6.67
		CS1C	0,2,3	0.9	6
		CS1M	0,1,2,3,4	0.6	8
CS2	48	CS2F	0,1,2,3,4	1	10
		CS2C	0,1,2,3,4	0.9	9
		CS2M	0,1,2,3,4	0.6	12

Table 2. Release times of machines

Machine	Release time (hours)
0	0
1	1
2	0
3	1
4	2

The products ordered list of step 1 of the LS algorithm, for the example is CM1, CS1, CM3, CM2, CS2, and the components ordered list of step 2 is CM1F, CM1C, CS1F, CS1C, CS1M, CM3F, CM3C, CM3M, CM2F, CM2C, CS2F, CS2C, CS2M. In Table 3 the auxiliary information used during step 1 is presented.

Table 3. Information used during step 1

Product	Due date	Total number of compatible machines	Total unit production time
CM1	24	10	1.9
CM2	48	6	1.9
CM3	24	13	2.5
CS1	24	11	2.5
CS2	48	15	2.5

Figure 1 presents the Gantt chart of the schedule obtained in step 3 of the LS algorithm. The total tardiness of this schedule is 2.63 hours, due to product CM3, the total deviation is 78.4 hours and the average machine utilization is equal to 92%. The machine utilization of a given machine M is given by: $\frac{Total\ occupied\ time\ of\ M}{Time\ Horizon - Release\ date\ of\ M} \times 100$.

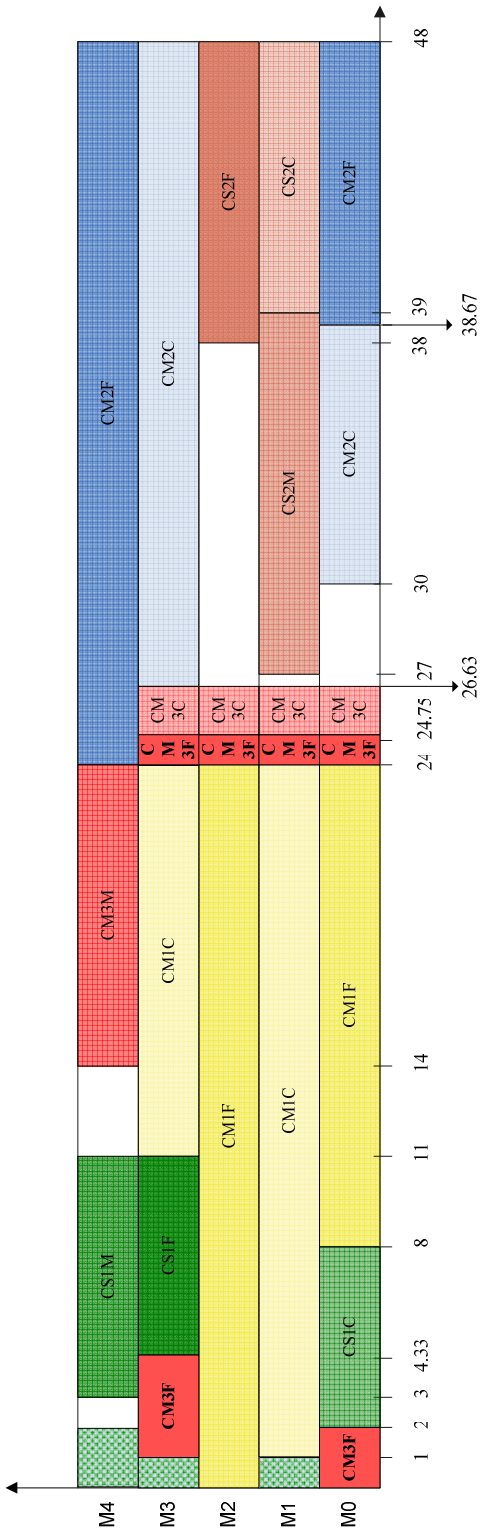

Figure1. Gantt chart for the solution of example

4. COMPUTATIONAL EXPERIMENTS

In this section the computational results for the list scheduling algorithm are presented. The test instances were randomly generated taking into account data obtained at the company. For example, the number of machines of each gauge is the same as in the company and the demands of different product types and the processing times were randomly generated within intervals defined by data provided by the company. The instance set is made up of 54 instances, grouped by gauge (18 instances of gauge 21, 18 of gauge 24 and 18 of gauge 27). The instances size is presented in Table 4. The first 18 instances belong to gauge 21, the next 18 to gauge 27, and the last 18 to gauge 24.

Table 4. Instances size

Instance	Number of products	Number of components	Number of machines
Inst20T1.1.G21	8	18	5
Inst20T1.2.G21	9	25	5
Inst20T1.3.G21	9	17	5
Inst30T1.1.G21	18	50	5
Inst30T1.2.G21	20	56	5
Inst30T1.3.G21	20	60	5
Inst40T1.1.G21	20	58	5
Inst40T1.2.G21	24	69	5
Inst40T1.3.G21	27	74	5
Inst50T1.1.G21	29	74	5
Inst50T1.2.G21	30	74	5
Inst50T1.3.G21	33	99	5
Inst60T1.1.G21	26	70	5
Inst60T1.2.G21	30	77	5
Inst60T1.3.G21	29	77	5
Inst70T1.1.G21	30	90	5
Inst70T1.2.G21	33	83	5
Inst70T1.3.G21	39	116	5
Inst20T1.1.G27	31	89	11
Inst20T1.2.G27	32	84	11
Inst20T1.3.G27	29	84	11
Inst30T1.1.G27	41	107	11
Inst30T1.2.G27	38	103	11
Inst30T1.3.G27	44	128	11
Inst40T1.1.G27	53	142	11
Inst40T1.2.G27	42	112	11
Inst40T1.3.G27	47	125	11
Inst50T1.1.G27	43	120	11
Inst50T1.2.G27	65	174	11
Inst50T1.3.G27	60	154	11
Inst60T1.1.G27	71	197	11
Inst60T1.2.G27	76	210	11
Inst60T1.3.G27	67	181	11
Inst70T1.1.G27	70	182	11
Inst70T1.2.G27	81	221	11
Inst70T1.3.G27	67	187	11
Inst20T1.1.G24	34	94	13
Inst20T1.2.G24	37	108	13
Inst20T1.3.G24	34	98	13
Inst30T1.1.G24	51	139	13
Inst30T1.2.G24	49	135	13
Inst30T1.3.G24	38	90	13
Inst40T1.1.G24	57	152	13
Inst40T1.2.G24	64	174	13
Inst40T1.3.G24	57	157	13
Inst50T1.1.G24	55	152	13
Inst50T1.2.G24	70	197	13
Inst50T1.3.G24	70	184	13
Inst60T1.1.G24	81	216	13
Inst60T1.2.G24	69	188	13
Inst60T1.3.G24	81	232	13
Inst70T1.1.G24	82	226	13
Inst70T1.2.G24	94	254	13
Inst70T1.3.G24	108	277	13

We coded the list scheduling heuristic in visual C++, and the tests were run in a personal computer with a Pentium 4 processor, with 1 GB of RAM. In our implementation we set K, the maximum number of lots of a machine, to: number of components × planning horizon in days. In Table 5 we present the results for the instance set. The performance measures considered were: total tardiness (column 2), number of products late (column 3), total deviation (column 4), number of lots (column 6), average number of lots per component (column 7), average deviation per product (column 8) and average deviation per lot (column 9).

Table 5. Results

Instance	Total tardiness (hours)	Number of products late	Total deviation (hours)	Average machine utilization (%)	Number of lots	Average number of lots by component	Average deviation by product	Average deviation by lot
Inst20T1.1.G21	5.60	1	254.36	46.27	23	1.28	31.80	11.06
Inst20T1.2.G21	3.38	1	1094.73	102.10	41	1.64	121.64	26.70
Inst20T1.3.G21	20.98	2	622.91	80.68	45	2.65	69.21	13.84
Inst30T1.1.G21	77.52	5	2208.37	103.87	117	2.34	122.69	18.87
Inst30T1.2.G21	2.26	1	533.33	68.60	78	1.39	26.67	6.84
Inst30T1.3.G21	41.42	5	755.74	92.68	132	2.20	37.79	5.73
Inst40T1.1.G21	0.00	0	609.20	99.37	77	1.33	30.46	7.91
Inst40T1.2.G21	5.50	2	1109.36	101.93	112	1.62	46.22	9.91
Inst40T1.3.G21	0.00	0	267.97	79.26	101	1.36	9.92	2.65
Inst50T1.1.G21	0.00	0	395.91	87.61	94	1.27	13.65	4.21
Inst50T1.2.G21	7.56	1	1483.99	104.54	114	1.54	49.47	13.02
Inst50T1.3.G21	0.00	0	629.01	86.39	129	1.30	19.06	4.88
Inst60T1.1.G21	0.00	0	1207.00	98.03	99	1.41	46.42	12.19
Inst60T1.2.G21	38.92	3	977.02	66.35	114	1.48	32.57	8.57
Inst60T1.3.G21	15.82	3	475.85	96.00	121	1.57	16.41	3.93
Inst70T1.1.G21	114.34	14	578.99	59.29	250	2.78	19.30	2.32
Inst70T1.2.G21	0.00	0	933.28	60.12	106	1.28	28.28	8.80
Inst70T1.3.G21	0.00	0	345.41	79.24	140	1.21	8.86	2.47
Inst20T1.1.G27	0.00	0	836.37	88.12	132	1.48	26.98	6.34
Inst20T1.2.G27	15.84	2	1247.35	89.10	178	2.12	38.98	7.01
Inst20T1.3.G27	2.20	1	1211.50	63.88	120	1.43	41.78	10.10
Inst30T1.1.G27	0.00	0	454.94	81.95	159	1.49	11.10	2.86
Inst30T1.2.G27	40.67	9	2461.20	105.90	372	3.61	64.77	6.62
Inst30T1.3.G27	0.00	0	1061.65	95.57	183	1.43	24.13	5.80
Inst40T1.1.G27	0.00	0	938.34	95.74	191	1.35	17.70	4.91
Inst40T1.2.G27	0.00	0	1077.09	95.37	174	1.55	25.65	6.19
Inst40T1.3.G27	34.28	5	1964.49	75.92	289	2.31	41.80	6.80
Inst50T1.1.G27	0.00	0	1668.75	99.55	196	1.63	38.81	8.51
Inst50T1.2.G27	16.80	5	1595.13	103.63	368	2.11	24.54	4.33
Inst50T1.3.G27	0.00	0	646.32	83.72	204	1.32	10.77	3.17
Inst60T1.1.G27	0.00	0	1857.14	99.73	264	1.34	26.16	7.03
Inst60T1.2.G27	298.15	25	1727.42	80.24	901	4.29	22.73	1.92
Inst60T1.3.G27	94.50	9	1535.01	96.90	486	2.69	22.91	3.16
Inst70T1.1.G27	361.68	20	2239.87	97.52	700	3.85	32.00	3.20
Inst70T1.2.G27	251.45	19	2339.09	98.55	728	3.29	28.88	3.21
Inst70T1.3.G27	146.32	15	2362.96	91.28	651	3.48	35.27	3.63
Inst20T1.1.G24	0.00	0	500.37	95.64	148	1.57	14.72	3.38
Inst20T1.2.G24	22.29	4	2256.61	91.80	287	2.66	60.99	7.86
Inst20T1.3.G24	126.32	8	2205.99	96.57	426	4.35	64.88	5.18
Inst30T1.1.G24	0.00	0	816.46	82.71	195	1.40	16.01	4.19
Inst30T1.2.G24	2.89	1	2832.25	101.77	240	1.78	57.80	11.80
Inst30T1.3.G24	30.54	4	2082.22	97.45	271	3.01	54.80	7.68
Inst40T1.1.G24	1.88	1	2364.74	91.16	235	1.55	41.49	10.06
Inst40T1.2.G24	0.00	0	499.41	82.53	226	1.30	7.80	2.21
Inst40T1.3.G24	98.31	10	2008.03	94.77	569	3.62	35.23	3.53
Inst50T1.1.G24	0.00	0	322.24	65.38	204	1.34	5.86	1.58
Inst50T1.2.G24	426.55	20	2606.13	114.00	906	4.60	37.23	2.88
Inst50T1.3.G24	6.05	4	1403.93	96.34	378	2.05	20.06	3.71
Inst60T1.1.G24	0.00	0	797.37	90.49	284	1.31	9.84	2.81
Inst60T1.2.G24	12.10	2	3239.33	103.89	327	1.74	46.95	9.91
Inst60T1.3.G24	0.96	1	2476.19	100.60	335	1.44	30.57	7.39
Inst70T1.1.G24	0.00	0	973.29	88.50	295	1.31	11.87	3.30
Inst70T1.2.G24	99.09	10	2720.66	109.79	746	2.94	28.94	3.65
Inst70T1.3.G24	17.04	12	2342.35	101.51	752	2.71	21.69	3.11

All the instances tested were solved in less than 0.03 seconds. In some instances, the total tardiness is large. This may occur, because the products due dates are generated randomly, and there may be a huge order with a due date that can not be fulfilled even if all the resources were assigned to it.

There is a positive correlation between the size of the instance (measured in terms of the number of components and the number of machines) and the total deviation, particularly for the instances with higher average machine utilization, although this does not hold for all the set of instances tested (see for example Inst30T1.1.G21, one of the smaller instances with a total deviation of approximately 2208 hours and an average machine utilization of approximately 104%, and Inst60.T1.1.G24, one of the greater instances with a total deviation of approximately 797 hours and an average machine utilization of 90.5%). The results show that there is not a direct relationship between the size of the instance and the total tardiness. Almost certainly, the total tardiness is more dependent of both the machines loads and the required due dates. The number of lots increases with the size of the instance. The average number of lots per component tends to increase with the size of the instance too, although there are exceptions (see for example Inst20T1.3.G21 and Inst70T1.1.G21). On the other hand, the average deviation by lot tends to decrease as the instance size increases. In the set of 18 instances of gauge 21, there are six instances with an average deviation by lot greater or equal to 10 hours, in the set of 18 instances that belong to gauge 27, only one instance has an average deviation by lot greater or equal to 10 hours, and, finally, in the set of 18 instances that belong to gauge 24, there are three instances with an average deviation by lot higher or equal to 10 hours.

There is a positive correlation between the average machine utilization and total tardiness. If we consider only the instances, of the set of 54 instances, that have an average machine utilization greater or equal to 98% (18 instances), only four of the eighteen instances do not have tardiness (Inst40T1.1.G21, Inst60T1.1.G21, Inst50T1.1.G27 and Inst60T1.1.G27).

The instances with higher number of lots have higher total tardiness. This can be in part explained because the LS algorithm splits the components into more lots when they are late.

There seems to exist a direct relationship between the total tardiness and the total deviation of a product, even though there are exceptions. The total tardiness and the total deviation have a positive correlation with: the average machine utilization, the number of lots, and the average number of lots per component.

5. CONCLUSIONS

The main motivation for this study arose from the interest of a Textile company to increase the efficiency of their knitting scheduling plans. As in such type of problems the solution times are a major concern, we developed a constructive heuristic for the lot splitting and scheduling problem existent in the knitting section of the factory. The heuristic is extremely fast, solving instances greater than the real ones in less than one second. The total tardiness and the total

deviations are for some instances high, denoting a potential field of improvement. In the context of the real problem, the minimization of the total tardiness is very important to assure a high level customer service. Nonetheless, the total deviations are of major interest since the production process after the components knitting, is joining the several components that belong to the same product and that process can only occur after completing all the components production. The developed heuristic, takes both objectives into consideration. As far as we are aware, none scheduling published work consider this kind of objective.

REFERENCES

Cheng, T. and C. Sin (1990). A state-of-the-art review of parallel-machine scheduling research. *European Journal of Operational Research,* **Vol.** 47, pp. 271-292.

Graham, R. el al. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics,* **Vol.** 5, pp. 287-326.

Hurink, J. and S. Knust (2001). List scheduling in a parallel machine environment with precedence constraints and setup times. *Operations Research Letters,* **Vol.** 29, pp. 231-239.

Mokotoff, E. (2001). Parallel machine scheduling problems: a survey. *Asia-Pacific Journal of Operational Research,* **Vol.** 18, pp. 193-242.

Sheen, G. and L. Liao (2007). Scheduling machine-dependent jobs to minimize lateness on machines with identical speed under availability constraints. *Computers and Operations Research,* **Vol.** 34, pp. 2266-2278.

Tahar, D. et al. (2006). A linear programming approach for identical parallel machine scheduling with job spliting and sequence-dependent setup times. *International Journal of Production Economics,* **Vol.** 99, pp. 63-73.

Xing, W. and J. Zhang (2000). Parallel machine scheduling with splitting jobs. *Discrete Applied Mathematics,* **Vol.** 103, pp. 259-269.

Yalaoui, F. and C. Chu (2003). An efficient heuristic approach for parallel machine scheduling with job splitting and sequence-dependent setup times. *IIE Transactions,* **Vol.** 35, pp. 183-190.