On multi-item economic lot-sizing with remanufacturing and uncapacitated production

Jesus O. Cunha*

Departamento de Ciência da Computação, Universidade Federal da Bahia. Avenida Adhemar de Barros, s/n, Salvador, BA 40170-110, Brazil.

Ioannis Konstantaras

Department of Business Administration, School of Business Administration, University of Macedonia, 156 Egnatia Str., Thessaloniki 54636, Greece.

Rafael A. Melo

Departamento de Ciência da Computação, Universidade Federal da Bahia. Avenida Adhemar de Barros, s/n, Salvador, BA 40170-110, Brazil.

Angelo Sifaleras

Department of Applied Informatics, School of Information Sciences, University of Macedonia, 156 Egnatia Str., Thessaloniki 54636, Greece.

Abstract

In this paper we consider the multi-item economic lot-sizing problem with remanufacturing and uncapacitated production. Observing that the problem is composed of several independent single-item problems, we show how very high quality feasible solutions and bounds can be obtained by solving each item separately using an effective approach recently proposed in the literature. Computational experiments show that our approach improves the best known feasible solutions and lower bounds for all the available instances. In addition, 86 instances could be solved to optimality and the remaining open gap was below 0.5% for almost all the unsolved instances.

Keywords: Inventory, Lot-sizing with remanufacturing, Multi-item, Mixed integer programming, Reverse logistics.
2010 MSC: 90B05, 90C11, 65K05

*Corresponding author. Work of this author was supported by the Coordination for the Improvement of Higher Education Personnel (CAPES), Brazil.

Email addresses: `jesus.ossian@ufba.br` (Jesus O. Cunha), `ikonst@uom.gr` (Ioannis Konstantaras), `melo@dcc.ufba.br` (Rafael A. Melo), `sifalera@uom.gr` (Angelo Sifaleras)

1. Introduction

In recent years, several factors (environmental concerns, legislation, voluntary collection for materials recovery, etc) have made it more significant for many businesses to think of reverse material flows when managing their supply chains. For instance, the increase in online sales has generated an important increase in product returns, because customers are unable to physically observe products before purchasing them. Furthermore many companies have started taking back products after customers use them. These have led reverse logistics and closed-loop supply chains to gain substantial interest in business and academia (DeCroix et al., [4]).

In the literature, there has been an increasing interest in the study of lot-sizing problems with remanufacturing due to their large applicability in reverse logistics. Two recent reviews on the modelling of lot-sizing and reverse logistics inventory systems are given by Aloulou et al. [1] and Bazan et al. [2], accordingly. Mainly, the single-item dynamic lot-sizing with remanufacturing, which consists in determining production and remanufacturing plans over a finite time horizon considering that the demands and returns for each period are dynamic and known beforehand, has attracted the attention of several authors during the last years. Recent works include the cutting-edge heuristic approaches presented in Piñeyro and Viera [9], Schulz [12], and Sifaleras et al. [15], as well as the state-of-the-art mixed integer programming (MIP) techniques studied in Cunha and Melo [3] and Retel Helmrich et al. [5].

One of the most important issues that should be dealt with by firms is the determination of the inventory replenishment policy of the different goods (spare parts, raw material, components or finished goods). Product returns increase the complexity of managing an inventory system by introducing an uncertain reverse flow of materials. This is mainly true when only a subset of the components of a product can be recovered for reuse (Decroix et al., [4]). In this paper we consider the multi-item economic lot-sizing with remanufacturing and uncapacitated production. To the best of our knowledge, there are only a few papers in the literature studying the multi-item economic lot-sizing with remanufacturing. Li et al. [6] studied the uncapacitated multi-item economic lot-sizing problem with remanufacturing options and demand substitution and presented an approximate procedure to solve the problem while the same authors [7] studied a variant of their previous model assuming capacity constraints. Sahling [11] considered a multi-item economic lot-sizing with remanufacturing and capacitated production problem in which setup times were also present. The authors proposed a column-generation approach combined with a truncated branch-and-bound method to solve the problem. Sifaleras and Konstantaras [13, 14] introduced the multi-item variation of the economic lot-sizing with uncapacitated production, the one we consider in our work. In [13], the authors proposed a general variable neighborhood search metaheuristic for the problem and a benchmark set with very large instances. In [14], the same authors presented a variable neighborhood descent metaheuristic and another larger benchmark set.

The remainder of this note is organized as follows. In Section 2, we formally define the problem and present a standard MIP formulation. The partial Wagner-Whitin based formulation is described in Section 3. Computational experiments are described in Section 4. Final comments are discussed in Section 5.

2. The multi-item economic lot-sizing with remanufacturing and uncapacitated production

In the multi-item economic lot-sizing with remanufacturing and uncapacitated production [13, 14], there is a set of NI items, each of them with deterministic dynamic demand over a finite discrete time horizon of NT periods. Each item i has a demand d_t^i for each period $t \in \{1 \dots NT\}$. The deterministic amount of returned material of item i arriving at each period is r_t^i. There is no restriction on the amount of new units of an item to be manufactured while the remanufacturing is restricted to the availability of returned material for that item.

Fixed and variable production costs (respectively $f_t^{p,i}$ and $p_t^{p,i}$) as well as fixed and variable remanufacturing costs (respectively $f_t^{r,i}$ and $p_t^{r,i}$) are incurred in case production and/or remanufacturing take place in a given period. There is a per unit cost $h_t^{p,i}$ implied by the storage of finished material as well as a per unit cost $h_t^{r,i}$ implied by the storage of returned material. It is assumed that there are no initial stocks of either finished or returned material and no final stocks of finished material. Besides, all the data are nonnegative and, for each item $i \in \{1, \dots, NI\}$, the cumulated demand in the interval $[k, t]$ is defined as $d_{kt}^i = \sum_{l=k}^{t} d_l^i$ for $1 \le k \le t \le NT$, and the cumulated amount of returned material in the interval $[k, t]$ as $r_{kt}^i = \sum_{l=k}^{t} r_l^i$ for $1 \le k \le t \le NT$.

Consider variables $x_t^{p,i}$ ($x_t^{r,i}$) to be the amount of item i produced (remanufactured) in period t. Also, let $s_t^{p,i}$ ($s_t^{r,i}$) denote the amount of finished (returned) item i in stock at the end of period t. Furthermore, $y_t^{p,i}$ ($y_t^{r,i}$) is equal to 1 if production (remanufacturing) happens in period t and 0 otherwise. The problem can thus be formulated as

$$z = \min \sum_{i=1}^{NI} \sum_{t=1}^{NT} (h_t^{p,i} s_t^{p,i} + p_t^{p,i} x_t^{p,i} + f_t^{p,i} y_t^{p,i}) + \sum_{i=1}^{NI} \sum_{t=1}^{NT} (h_t^{r,i} s_t^{r,i} + p_t^{r,i} x_t^{r,i} + f_t^{r,i} y_t^{r,i}) \quad (1)$$

$$s_{t-1}^{p,i} + x_t^{p,i} + x_t^{r,i} = d_t^i + s_t^{p,i}, \quad \text{for } 1 \le i \le NI,\ 1 \le t \le NT, \quad (2)$$

$$s_{t-1}^{r,i} + r_t^i = x_t^{r,i} + s_t^{r,i}, \quad \text{for } 1 \le i \le NI,\ 1 \le t \le NT, \quad (3)$$

$$x_t^{p,i} \le d_{t,NT}^i y_t^{p,i}, \quad \text{for } 1 \le i \le NI,\ 1 \le t \le NT, \quad (4)$$

$$x_t^{r,i} \le \min\{r_{1t}^i, d_{t,NT}^i\} y_t^r, \quad \text{for } 1 \le i \le NI,\ 1 \le t \le NT, \quad (5)$$

$$x^p, x^r, s^p, s^r \in \mathbb{R}_+^{NI \times NT}, \quad (6)$$

$$y^p, y^r \in \{0, 1\}^{NI \times NT}. \quad (7)$$

The objective function (1) minimizes the total cost. Constraints (2) are balance constraints for the final material. Constraints (3) are balance constraints related to the returned material. Constraints (4) and (5) force the setup variables to one if production/remanufacturing are incurred. Constraints (6) and (7) are, respectively, nonnegativity and integrality constraints on the variables.

Observation 1. *The problem is composed of NI independent economic lot-sizing with remanufacturing problems. In addition, let z_i be the optimal solution for the problem related to item i and $\underline{z}_i$ a lower bound on z_i, then $z = \sum_{i=1}^{NI} z_i$ and $\sum_{i=1}^{NI} \underline{z}_i \leq z$.*

The observation follows from the fact that there are no constraints linking the economic lot-sizing with remanufacturing problems for different items.

3. The partial Wagner-Whitin based formulation

In this section we present how to use strong formulations for the single-item economic lot-sizing with remanufacturing related to each item in order to obtain an improved formulation for the multi-item problem. We use the best performing approach presented in Cunha and Melo [3] for each of the single-item economic lot-sizing with remanufacturing problems, namely a partial Wagner-Whitin based formulation with size determined automatically in a heuristic way. Let K_t^p and K_t^r, for $1 \leq t \leq NT$, be integer values in the interval $[0, NT-1]$. The partial Wagner-Whitin based formulation is

$$z = \min \sum_{i=1}^{NI} \sum_{t=1}^{NT} (h_t^{p,i} s_t^{p,i} + p_t^{p,i} x_t^{p,i} + f_t^{p,i} y_t^{p,i}) + \sum_{i=1}^{NI} \sum_{t=1}^{NT} (h_t^{r,i} s_t^{r,i} + p_t^{r,i} x_t^{r,i} + f_t^{r,i} y_t^{r,i})$$

$$(2) - (7),$$

$$s_l^{r,i} + \sum_{k=t}^{l} r_{tk}^i y_k^{r,i} \geq r_{tl}^i, \text{for } 1 \leq i \leq NI, 1 \leq t \leq l \leq NT,\ l \leq t + K_t^{r,i}, \tag{8}$$

$$s_{t-1}^{p,i} + \sum_{k=t}^{l} d_{kl}^i y_k^{p,i} + \sum_{k=t}^{l} \min\{r_{1k}^i, d_{kl}^i\} y_k^{r,i} \geq d_{tl}^i, \text{for } 1 \leq i \leq NI, 1 \leq t \leq l \leq NT,\ l \leq t + K_t^{r,i}, \tag{9}$$

$$\sum_{k=1}^{t-1} x_k^{p,i} + \sum_{k=t}^{l} \underline{d}_{kl}^i y_k^{p,i} \geq \underline{d}_{1l}^i, \text{for } 1 \leq i \leq NI, 1 \leq t \leq l \leq NT,\ l \leq t + K_t^{p,i}. \tag{10}$$

Inequalities (8) are variations of the Wagner-Whitin $(l,S)-$inequalities related to the returned material and were used in Retel Helmrich et al. [5]. Inequalities (9) and (10) were presented in Melo and Cunha [3]. Inequalities (9) are extensions of the $(l,S)-$inequalities associated to the demands. Defining $\underline{d}^p \in \mathbb{R}_+^{NT}$ to be the vector of minimum demands that must be satisfied by production of new items as in [3], inequalities (10) are $(l,S)-$inequalities for $\underline{d}^p$, with $\underline{d}_{kl}^p = \sum_{j=k}^{l} \underline{d}_j^p$.

The key idea of this partial formulation is to heuristically determine values for the parameters K_t^p and K_t^r based on the problem's cost structure. Assuming time invariant costs, estimations of the intervals in which production and remanufacturing setups are likely to occur are determined as in [3], i.e., $K^{p'} = \arg\min_{k \in \{1,\ldots,NT\}} \left(d_{avg}^i \times k \times (p^{p,i} + h^{p,i}) \geq f^{p,i}\right)$ and $K^{r'} = \arg\min_{k \in \{1,\ldots,NT\}} \left(r_{avg}^i \times k \times (p^{r,i} + h^{p,i}) \geq f^{r,i}\right)$, considering $d_{avg}^i = \dfrac{d_{1,NT}^i}{NT}$ and

$r^i_{avg} = \dfrac{r^i_{1,NT}}{NT}$. The values K^p and K^r are thus calculated as $K^p = \max\{5, \lceil 0.5 \times K^{p'} \rceil\}$ and $K^r = \max\{5, \lceil 0.5 \times K^{r'} \rceil\}$.

4. Computational experiments

In this section we report on the performed computational experiments. We compare the results obtained using three approaches: (a) solving a problem for each item separately (*mi_lsr*$_i$), (b) solving all the items in a single formulation (*mi_lsr*), and (c) best results reported in Sifaleras and Konstantaras [14] (*vnd*) in whose work the running time of each execution was limited to 90 s. All executions were performed on a machine running under Xubuntu, with an Intel(R) Core(TM) i7-4770S CPU @ 3.10GHz processor and 8Gb of RAM memory, using FICO Xpress 7.9. The codes were written in C++ and compiled with g++ 4.8.4. The solver's default settings were used, with exception of the optimality tolerance which was set to 10^{-6}, and the running time of each execution was limited to 3600 s.

The benchmark instances used in our experiments, which have $NI = 300$ items and $NT = 52$ periods each, were presented in [13, 14], where a detailed description of how they were generated can be encountered. The instances are divided into three groups: *group 1*, *group 2* and *group 3*. Instances in *group 1* are the ones from [14] with demand and return values generated using a normal distribution. Instances in *group 2* are those from [14] with demand and return values generated using an uniform distribution. Instances in *group 3* are those presented in [13].

The approach for solving each item separately was implemented using two cycles. In *cycle 1*, the total available running time is initially distributed uniformly amongst all items. After every ten treated items, the available running time for each untreated item is calculated by dividing the remaining available running time uniformly among the items which were not treated yet. After all items are treated, and in case there is still available time, in *cycle 2* the unsolved items are revisited until there is available time (i.e., larger than $3600/NI$ s) or until they are solved to optimality.

The computational results are summarized in Tables 1, 2, 3, and 4. The columns in *mi_lsr*$_i$ are related to the approach considering a formulation for each item separately, columns in *mi_lsr* refer to the formulation consisting of all items, and columns in *vnd* are associated to the variable neighborhood descent heuristic proposed in [14]. The first column in each table identifies the instances. Next, there are four columns associated with each of the cycles (*cycle k*, $k \in \{1, 2\}$) in *mi_lsr*$_i$. Column *bi* gives the best integer solution found; $gap = 100 \times \frac{bi - bb}{bi}$ shows the remaning open gap, with *bb* being the best achieved bound using any of the tested approaches, which was obtained using *mi_lsr*$_i$ for all instances; *time* indicates the time in seconds; and # displays the number of items solved to optimality in each instance. The next four columns give, for *mi_lsr* and *vnd*, the best integer solution found (*bi*) and the remaining open gap (*gap*) at the end of the execution. The open gap for each instance related to *vnd* is calculated using the best bound achieved using our approaches, as they were always better than the ones obtained in [14]. The last column presents the improvement over the previously best known feasible solution (*%imp*), calculated as $100 \times \frac{bi_{vnd} - bi_{mi\_lsr_i}}{bi_{vnd}}$, in which $bi_{mi\_lsr_i}$ is the value bi in mi_lsr_i and bi_{vnd} is the value bi in *vnd*.

Table 1: Computational results for instance group 1

	mi_lsr_i												
	cycle 1				*cycle 2*				*mi_lsr*		*vnd*		
inst	*bi*	*gap*	*time*	#	*bi*	*gap*	*time*	#	*bi*	*gap*	*bi*	*gap*	*%imp*
1	2837199.8	0.327	2724	128	2837181.2	0.285	-	167	3266811.6	14.9	3270282.6	13.5	13.2
2	3121601.0	0.393	1673	200	3121482.5	0.337	-	220	3606109.5	15.5	3495448.5	11.0	10.7
3	3382696.2	0.625	1514	200	3382653.2	0.610	-	202	4165302.4	21.1	3748498.8	10.3	9.8
4	3635224.2	0.003	2229	296	3635224.2	0.000	2380	300	4112513.2	12.7	4027173.0	9.7	9.7
5	4189026.5	0.172	3586	173	4189026.5	0.172	-	174	4427388.5	6.8	4479928.0	6.7	6.5
6	4604167.0	0.821	2679	114	4604085.4	0.793	-	149	4878990.8	7.6	4875797.6	6.3	5.6
7	5248904.8	0.000	860	300	5248904.8	0.000	860	300	5300283.8	1.6	5796958.8	9.5	9.5
8	6634672.5	0.000	1146	300	6634672.5	0.000	1146	300	7010674.0	6.2	6984412.0	5.0	5.0
9	7574772.2	0.000	1080	300	7574772.2	0.000	1080	300	7982017.8	5.7	8081856.8	6.3	6.3
10	4297670.4	0.743	2493	200	4297568.4	0.731	-	202	5617896.6	27.4	4858495.8	12.2	11.5
11	4575374.0	0.075	1840	262	4575035.0	0.000	3009	300	7124586.0	38.4	5089078.5	10.1	10.1
12	4841805.8	0.001	1226	298	4841805.8	0.000	1254	300	7083701.8	33.4	5278825.4	8.3	8.3
13	5145537.2	0.202	3013	204	5145533.0	0.181	-	227	6349446.8	21.4	5892350.8	12.8	12.7
14	5693519.0	0.649	3466	96	5693519.0	0.647	-	103	7218031.0	24.6	6321038.5	10.5	9.9
15	6087632.0	0.919	2416	191	6087549.2	0.904	-	201	8484889.6	31.4	6704859.6	10.0	9.2
16	6946349.4	0.000	950	300	6946349.4	0.000	950	300	7449673.6	7.3	7414346.8	6.3	6.3
17	8220551.5	0.000	1197	300	8220551.5	0.000	1197	300	8775444.0	7.4	8742185.0	6.0	6.0
18	9142101.8	0.000	1117	300	9142101.8	0.000	1117	300	9659954.4	6.1	9880458.4	7.5	7.5
19	8319817.8	0.078	2060	247	8318825.8	0.000	2732	300	11993376.4	32.8	8930651.8	6.9	6.9
20	8849610.0	0.197	3069	216	8849409.0	0.166	-	243	13086520.5	34.4	9380007.0	5.8	5.7
21	9339915.4	0.546	-	133	9339915.4	0.546	-	133	15418572.2	41.5	9814944.8	5.4	4.8
22	9154784.8	0.282	2949	178	9154554.0	0.234	-	219	12736666.0	30.9	10142751.4	10.0	9.7
23	9824402.5	1.252	-	8	9824402.5	1.252	-	8	14323726.0	35.4	10621326.0	8.7	7.5
24	10438585.2	2.346	-	0	10438585.2	2.346	-	0	15614710.2	37.2	11077982.2	8.0	5.8
25	11416615.2	0.000	1900	300	11416615.2	0.000	1900	300	12966823.4	13.6	12404914.8	8.0	8.0
26	12514570.0	0.013	2703	293	12514403.0	0.000	2921	300	14843864.0	18.0	13681913.0	8.5	8.5
27	13367702.6	0.000	1967	300	13367702.6	0.000	1967	300	16938032.8	23.5	14802946.0	9.7	9.7
28	2844512.2	0.516	3162	114	2844488.8	0.506	-	131	3238023.8	14.2	3194939.6	11.4	11.0
29	3127829.5	0.348	2218	170	3127730.5	0.257	-	222	3604858.5	15.6	3470471.5	10.1	9.9
30	3368773.0	0.453	1692	198	3368615.4	0.394	-	218	4076169.0	19.7	3675663.0	8.7	8.4
31	3634686.0	0.005	2390	295	3634686.0	0.000	2589	300	4176219.4	14.2	3989579.0	8.9	8.9
32	4177896.5	0.154	3245	212	4177889.0	0.142	-	232	4430459.5	7.3	4492250.0	7.1	7.0
33	4577700.4	0.461	2601	168	4577493.2	0.409	-	212	4880633.8	8.1	4880649.8	6.6	6.2
34	5246282.8	0.000	845	300	5246282.8	0.000	845	300	5300435.4	1.6	5828545.2	10.0	10.0
35	6611761.0	0.000	1133	300	6611761.0	0.000	1133	300	6964529.5	5.9	7115807.5	7.1	7.1
36	7543658.8	0.000	1067	300	7543658.8	0.000	1067	300	7887536.6	5.1	8110102.4	7.0	7.0
37	4303264.2	0.676	3032	188	4303271.0	0.653	-	201	5623638.8	27.4	4837334.8	11.6	11.0
38	4604686.0	0.265	2406	222	4604385.5	0.193	-	260	6976425.0	37.0	5086428.0	9.7	9.5
39	4870024.0	0.041	1736	280	4869822.8	0.000	2387	300	7052187.2	33.2	5295171.4	8.0	8.0
40	5144432.8	0.209	3470	181	5144432.8	0.208	-	186	6389111.8	21.9	5824516.8	11.9	11.7
41	5690303.5	0.772	-	69	5690303.5	0.772	-	69	7237013.0	25.0	6339034.0	10.9	10.2
42	6084812.2	0.665	2911	177	6084605.4	0.624	-	204	8435319.2	31.2	6710371.4	9.9	9.3
43	6945897.8	0.000	932	300	6945897.8	0.000	932	300	7387604.8	6.6	7442889.6	6.7	6.7
44	8199921.5	0.000	1112	300	8199921.5	0.000	1112	300	8635130.0	6.1	8718339.0	5.9	5.9
45	9114334.2	0.000	1093	300	9114334.2	0.000	1093	300	9606982.6	6.0	9785732.0	6.9	6.9
46	8326307.8	0.116	2310	235	8324648.2	0.000	3186	300	12358125.0	34.8	8952738.6	7.0	7.0
47	8846827.5	0.281	3303	185	8846827.5	0.272	-	201	13200236.0	35.1	9389338.5	6.0	5.8
48	9302945.8	0.306	3515	175	9302960.6	0.305	-	179	15626739.4	42.4	9841728.0	5.8	5.5
49	9163390.2	0.351	3271	142	9163390.2	0.337	-	165	12710152.4	30.8	10120022.2	9.8	9.5
50	9817251.0	1.264	-	17	9817251.0	1.264	-	17	14234903.5	35.1	10617420.5	8.7	7.5
51	10379591.9	1.866	-	5	10379591.9	1.866	-	5	15469234.6	36.7	11055554.0	7.9	6.1
52	11404705.8	0.000	1597	300	11404705.8	0.000	1597	300	12983464.0	13.7	12410460.8	8.1	8.1
53	12497549.0	0.010	2597	295	12497512.5	0.000	2813	300	14930557.0	18.6	13644827.5	8.4	8.4
54	13368330.8	0.003	2289	297	13368330.8	0.000	2328	300	16642804.4	22.3	14678743.2	8.9	8.9

Table 2: Computational results for instance group 1 (continued)

	mi_lsr_i												
	cycle 1				*cycle 2*				*mi_lsr*		*vnd*		
inst	*bi*	*gap*	*time*	#	*bi*	*gap*	*time*	#	*bi*	*gap*	*bi*	*gap*	*%imp*
55	2826263.0	0.347	2967	183	2826263.0	0.325	-	208	3181368.6	13.3	3276712.2	14.0	13.7
56	3117468.5	0.158	2075	211	3117380.0	0.064	-	276	3578028.5	15.2	3521771.5	11.5	11.5
57	3365281.4	0.176	1616	221	3365034.6	0.074	-	279	4038201.2	19.0	3734584.0	10.0	9.9
58	3613126.0	0.002	2022	298	3613084.0	0.000	2098	300	3928229.6	9.3	4030910.6	10.4	10.4
59	4175611.5	0.251	3568	182	4175611.5	0.250	-	183	4383123.5	6.3	4505687.5	7.6	7.3
60	4578955.2	0.374	2453	199	4578625.4	0.309	-	228	4909098.4	8.4	4890417.0	6.7	6.4
61	5214572.8	0.000	773	300	5214572.8	0.000	773	300	5255045.2	1.4	5719050.4	8.8	8.8
62	6600259.0	0.000	1032	300	6600259.0	0.000	1032	300	6929837.0	5.5	7002678.5	5.7	5.7
63	7555365.4	0.000	1032	300	7555365.4	0.000	1032	300	7908192.8	5.2	8089596.8	6.6	6.6
64	4263633.6	0.463	2968	184	4263647.8	0.438	-	204	5515908.8	26.6	4858676.4	12.6	12.2
65	4581185.0	0.343	2314	203	4581103.0	0.256	-	250	6615350.0	34.0	5085849.5	10.2	9.9
66	4850430.2	0.108	1999	245	4850014.0	0.000	3526	300	7193514.6	34.8	5275865.6	8.1	8.1
67	5108814.4	0.081	3046	245	5108814.4	0.064	-	268	5976552.2	17.1	5854389.6	12.8	12.7
68	5665087.0	0.552	3576	130	5665087.0	0.551	-	131	6954775.0	22.2	6327424.5	11.0	10.5
69	6062483.6	0.606	2537	183	6062308.2	0.556	-	212	7838717.6	26.0	6704104.2	10.1	9.6
70	6907298.4	0.000	844	300	6907298.4	0.000	844	300	7417128.6	7.5	7414142.4	6.8	6.8
71	8183353.5	0.000	1067	300	8183353.5	0.000	1067	300	8570157.0	5.6	8724241.5	6.2	6.2
72	9116241.6	0.000	1022	300	9116241.6	0.000	1022	300	9548423.2	5.3	9846869.8	7.4	7.4
73	8269287.2	0.013	1887	292	8269163.8	0.000	1990	300	11046402.8	27.5	8917010.2	7.3	7.3
74	8796339.0	0.064	2641	267	8795614.5	0.000	3284	300	12361527.0	31.2	9398291.5	6.4	6.4
75	9259589.9	0.061	3212	260	9259583.6	0.054	-	274	12579604.0	28.8	9838522.0	5.9	5.9
76	9108642.8	0.165	2955	221	9108288.0	0.120	-	258	12346541.4	29.2	10116626.2	10.1	10.0
77	9761036.0	0.821	-	69	9761036.0	0.821	-	69	13914509.0	33.9	10619852.0	8.8	8.1
78	10351400.6	1.636	-	35	10351400.6	1.636	-	35	15011114.6	35.1	11089695.8	8.2	6.7
79	11348873.8	0.000	1270	300	11348873.8	0.000	1270	300	12818360.0	13.1	12375946.8	8.3	8.3
80	12465397.5	0.003	2393	297	12465397.5	0.000	2453	300	14612898.5	17.1	13634882.5	8.6	8.6
81	13343954.6	0.000	2084	300	13343954.6	0.000	2084	300	16630194.6	22.4	14777131.2	9.7	9.7
82	2824505.6	0.330	2993	196	2824476.4	0.303	-	220	3215036.6	14.3	3214135.8	12.4	12.1
83	3114287.5	0.185	2203	200	3114187.0	0.095	-	267	3572959.5	15.3	3477940.0	10.5	10.5
84	3351589.4	0.141	1697	226	3351318.8	0.031	-	290	4054006.4	19.6	3675077.6	8.8	8.8
85	3610405.6	0.006	2143	294	3610398.2	0.000	2429	300	3905477.6	8.9	3982416.4	9.3	9.3
86	4160359.0	0.123	3168	227	4160358.0	0.111	-	245	4378143.5	6.6	4505007.5	7.8	7.7
87	4555634.6	0.243	2360	211	4555521.4	0.176	-	252	4841262.8	7.6	4893901.6	7.1	6.9
88	5213627.0	0.000	784	300	5213627.0	0.000	784	300	5269270.6	1.7	5721915.8	8.9	8.9
89	6581138.5	0.000	1035	300	6581138.5	0.000	1035	300	6922297.5	5.8	7067795.0	6.9	6.9
90	7523706.8	0.000	1055	300	7523706.8	0.000	1055	300	7849215.2	4.9	8095371.8	7.1	7.1
91	4263606.4	0.404	3252	176	4263606.4	0.394	-	191	5465322.6	25.9	4832667.0	12.1	11.8
92	4593008.5	0.341	2681	191	4592910.5	0.282	-	236	6597773.0	33.8	5084756.5	9.9	9.7
93	4861960.2	0.156	2166	230	4861585.6	0.053	-	288	7017309.2	33.2	5300186.0	8.3	8.3
94	5105480.0	0.074	3135	254	5105465.6	0.059	-	273	5985456.2	17.3	5815482.8	12.3	12.2
95	5656235.5	0.551	-	137	5656235.5	0.551	-	137	6844448.5	21.1	6319621.5	11.0	10.5
96	6058478.6	0.499	3050	181	6058411.6	0.460	-	204	7666380.2	24.5	6690277.4	9.9	9.4
97	6906395.0	0.000	845	300	6906395.0	0.000	845	300	7436369.2	7.8	7413442.2	6.8	6.8
98	8163563.5	0.000	1052	300	8163563.5	0.000	1052	300	8530315.0	5.4	8697765.0	6.1	6.1
99	9086654.8	0.000	1051	300	9086654.8	0.000	1051	300	9429917.0	4.6	9768606.2	7.0	7.0
100	8268147.8	0.018	1978	289	8267947.4	0.000	2116	300	11088531.8	27.8	8930673.6	7.4	7.4
101	8784617.0	0.052	2575	268	8784033.0	0.000	3029	300	12298362.0	30.9	9383498.0	6.4	6.4
102	9234426.6	0.051	2888	269	9234233.2	0.017	-	292	12569350.8	28.9	9828182.6	6.1	6.0
103	9106166.8	0.171	3126	219	9106147.0	0.143	-	247	12374200.4	29.4	10121253.4	10.2	10.0
104	9749583.0	0.769	-	89	9749583.0	0.769	-	89	13847893.0	33.7	10602131.0	8.7	8.0
105	10301753.1	1.248	-	51	10301753.1	1.248	-	51	14843606.0	34.4	11067832.0	8.1	6.9
106	11342483.8	0.000	1256	300	11342483.8	0.000	1256	300	12851712.2	13.4	12396198.4	8.5	8.5
107	12449613.0	0.006	2303	297	12449613.0	0.000	2419	300	14646731.0	17.4	13606139.0	8.5	8.5
108	13330411.8	0.003	2260	298	13330411.8	0.000	2301	300	16410810.8	21.5	14660716.8	9.1	9.1

Table 3: Computational results for instance group 2

	mi_lsr_i												
	cycle 1				*cycle 2*				*mi_lsr*		*vnd*		
inst	*bi*	*gap*	*time*	#	*bi*	*gap*	*time*	#	*bi*	*gap*	*bi*	*gap*	*%imp*
109	2768179.4	0.228	2860	212	2768164.2	0.194	-	240	3084193.4	12.4	3288156.2	16.0	15.8
110	3085888.5	0.163	2123	235	3085813.5	0.066	-	285	3708014.0	19.2	3579426.0	13.8	13.8
111	3337251.6	0.059	1582	261	3337131.2	0.000	2584	300	4015498.4	19.1	3757871.6	11.2	11.2
112	3594372.0	0.005	1943	297	3594372.0	0.000	2085	300	3855673.6	8.2	4021857.8	10.6	10.6
113	4147315.0	0.046	2499	273	4147214.0	0.000	3530	300	4325688.5	5.7	4612075.5	10.1	10.1
114	4555839.2	0.061	1908	267	4555638.0	0.000	2990	300	4879350.4	8.2	5030035.0	9.4	9.4
115	5260013.2	0.000	710	300	5260013.2	0.000	710	300	5305694.0	1.5	5772314.2	8.9	8.9
116	6680277.0	0.000	942	300	6680277.0	0.000	942	300	7067194.0	6.3	7183598.5	7.0	7.0
117	7660408.8	0.000	958	300	7660408.8	0.000	958	300	8022926.8	5.3	8229016.0	6.9	6.9
118	4212971.4	0.170	3003	225	4212964.4	0.151	-	247	5053720.6	20.4	4762549.0	11.7	11.5
119	4577655.5	0.281	2705	208	4577583.0	0.219	-	246	6427191.0	32.3	5007422.5	8.8	8.6
120	4871114.6	0.225	2315	224	4870821.8	0.122	-	271	6589181.4	29.0	5259170.0	7.5	7.4
121	5106351.6	0.104	3422	251	5106351.6	0.102	-	257	5724268.6	13.4	5743878.0	11.2	11.1
122	5656662.5	0.351	3553	184	5656662.5	0.351	-	185	6732667.5	19.4	6321740.5	10.8	10.5
123	6070019.8	0.251	3040	219	6069978.8	0.226	3587	241	7311129.0	20.4	6749591.0	10.3	10.1
124	6997204.4	0.000	746	300	6997204.4	0.000	746	300	7073002.6	1.9	7511577.2	6.8	6.8
125	8297090.5	0.000	953	300	8297090.5	0.000	953	300	8592491.5	4.6	8880716.5	6.6	6.6
126	9245314.6	0.000	1123	300	9245314.6	0.000	1123	300	9502674.4	3.7	9955046.2	7.1	7.1
127	8201131.6	0.064	1804	274	8200515.2	0.000	2135	300	10099748.0	21.3	8917528.0	8.0	8.0
128	8737821.0	0.048	2023	274	8737261.0	0.000	2391	300	11231832.5	24.9	9382673.5	6.9	6.9
129	9201993.8	0.022	2035	290	9201652.2	0.000	2281	300	11253018.0	20.8	9845523.6	6.5	6.5
130	9094420.8	0.284	2883	224	9093508.8	0.194	-	267	12073343.4	27.8	10168450.4	10.7	10.6
131	9760574.2	0.734	3580	115	9760574.2	0.734	-	115	13556816.5	32.4	10662331.0	9.1	8.5
132	10318131.6	0.992	-	104	10318131.6	0.992	-	104	14522452.2	32.9	11139763.8	8.3	7.4
133	11422544.8	0.000	1010	300	11422544.8	0.000	1010	300	12925612.4	13.4	12569296.4	9.1	9.1
134	12567794.0	0.003	2020	297	12567794.0	0.000	2107	300	14720317.0	17.2	13906463.0	9.6	9.6
135	13486006.4	0.005	2262	297	13486006.4	0.000	2335	300	16524439.6	21.2	14939059.4	9.7	9.7
136	2561054.0	0.130	2917	219	2561050.0	0.103	-	254	2794553.2	11.3	2942610.2	13.1	13.0

Table 4: Computational results for instance group 3

	mi_lsr_i												
	cycle 1				*cycle 2*				*mi_lsr*		*vnd*		
inst	*bi*	*gap*	*time*	#	*bi*	*gap*	*time*	#	*bi*	*gap*	*bi*	*gap*	*%imp*
137	2845802.5	0.235	3168	193	2845802.5	0.217	-	218	3305414.5	17.4	3187454.0	10.9	10.7
138	3061621.6	0.166	2769	230	3061590.8	0.121	-	264	3613781.6	18.7	3375163.4	9.4	9.3
139	3198456.2	0.000	1078	300	3198456.2	0.000	1078	300	3470794.8	9.1	3543668.4	9.7	9.7
140	3691161.5	0.001	1545	299	3691159.5	0.000	1561	300	3835314.5	5.3	4145741.0	11.0	11.0
141	4064161.6	0.012	1863	291	4064125.2	0.000	2031	300	4461194.6	10.7	4543054.2	10.5	10.5
142	4457465.6	0.000	677	300	4457465.6	0.000	677	300	4471142.8	0.8	4804402.4	7.2	7.2
143	5678441.5	0.000	993	300	5678441.5	0.000	993	300	5870010.0	3.9	5988207.0	5.2	5.2
144	6534347.6	0.000	1013	300	6534347.6	0.000	1013	300	6783870.4	4.4	6857867.2	4.7	4.7
145	3761483.2	0.058	2296	268	3761268.0	0.000	3017	300	4776611.4	25.4	4337867.6	13.3	13.3
146	4082135.0	0.330	3219	178	4082123.5	0.313	-	201	5248075.0	27.2	4571979.0	11.0	10.7
147	4329257.8	0.385	3310	177	4329253.2	0.374	-	192	5794539.0	29.5	4768544.0	9.6	9.2
148	4474457.0	0.000	1257	300	4474457.0	0.000	1257	300	4971584.0	12.4	5041837.0	11.3	11.3
149	4953918.0	0.054	2427	270	4953680.5	0.000	3022	300	5740727.0	17.2	5623766.0	11.9	11.9
150	5330086.8	0.131	2867	238	5330052.8	0.074	-	279	6437671.0	21.0	6036566.0	11.8	11.7
151	5931974.6	0.000	739	300	5931974.6	0.000	739	300	6162480.4	4.3	6445542.8	8.0	8.0
152	7050148.5	0.000	856	300	7050148.5	0.000	856	300	7250282.0	3.5	7508623.0	6.1	6.1
153	7870790.8	0.000	943	300	7870790.8	0.000	943	300	8019008.0	2.6	8386224.0	6.1	6.1
154	7104151.0	0.000	1303	300	7104151.0	0.000	1303	300	8404164.0	18.5	7780606.6	8.7	8.7
155	7546268.0	0.000	1577	300	7546268.0	0.000	1577	300	9114981.0	21.0	8138222.5	7.3	7.3
156	7919664.6	0.006	1610	295	7919644.2	0.000	1677	300	8837244.2	13.7	8495880.0	6.8	6.8
157	7835826.8	0.000	1754	300	7835826.8	0.000	1754	300	9409652.2	20.6	8747360.2	10.4	10.4
158	8395537.0	0.069	2567	271	8394881.0	0.000	3121	300	9454965.0	16.7	9256106.0	9.3	9.3
159	8863227.6	0.187	2976	241	8862335.4	0.109	-	277	10078872.4	17.0	9682844.0	8.6	8.5
160	9748595.8	0.000	956	300	9748595.8	0.000	956	300	10803327.6	11.2	10635183.6	8.3	8.3
161	10709300.5	0.000	1584	300	10709300.5	0.000	1584	300	12190881.5	14.5	11702127.5	8.5	8.5
162	11485385.2	0.000	1741	300	11485385.2	0.000	1741	300	13496694.2	17.7	12562994.2	8.6	8.6

The computational results show that much better gaps can be obtained using mi_lsr_i

when compared to the ones achieved by *mi_lsr* and *vnd*, as we can observe in Figures 1, 2, and 3. Observe that neither *mi_lsr* nor *vnd* could find optimal solutions, while mi_lsr_i solved 51 instances to optimality in *cycle 1* and this number increased to 86 at the end of *cycle 2*.

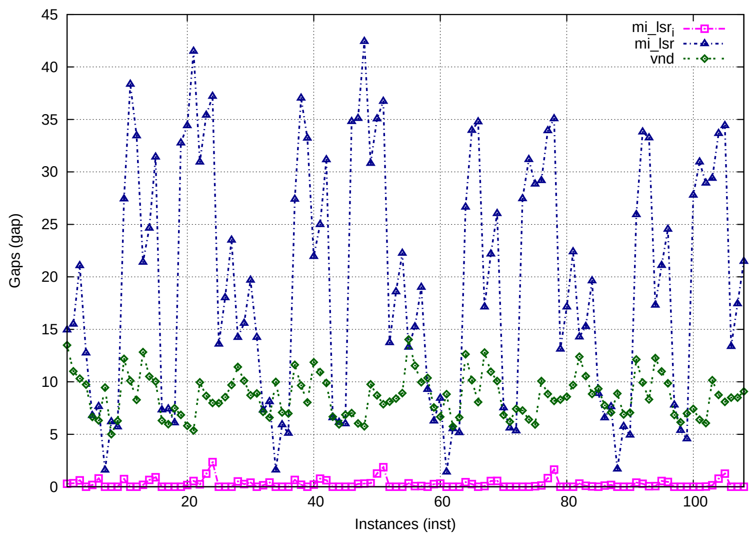

Figure 1: Obtained gaps for instance group 1.

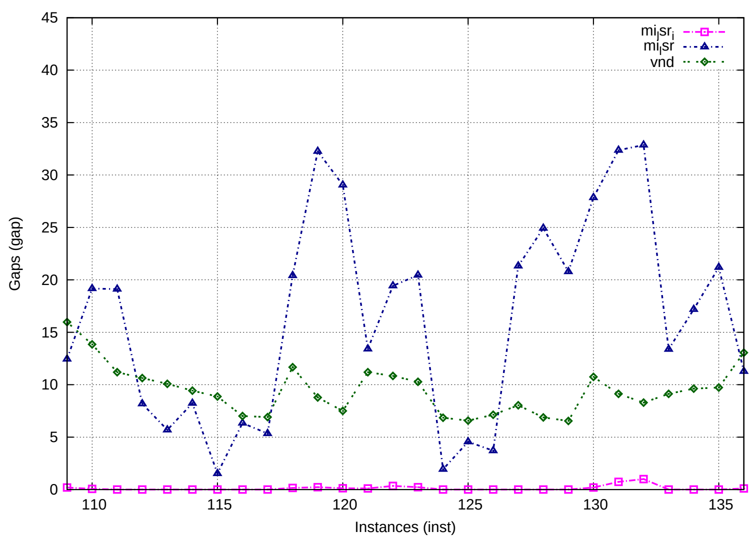

Figure 2: Obtained gaps for instance group 2.

It is well known that an effective way to produce strong formulations for several multi-item production planning and inventory control problems (such as the present one) is to strengthen the relaxations for the different single-item relaxations [10]. Therefore, it is possible to see that our approach, which solves each of the single-item subproblems using

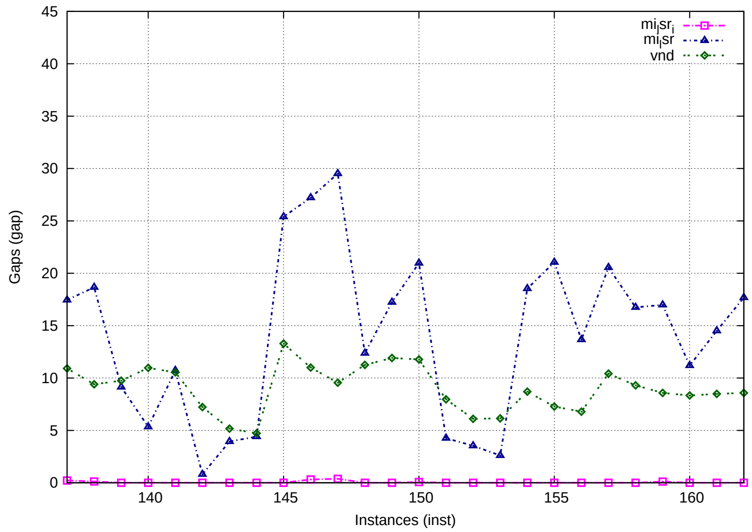

Figure 3: Obtained gaps for instance group 3.

an effective reformulation based on valid inequalities is a nice contribution in the direction of finding good solutions. Such reformulations play an important role in solving or finding high quality solutions for the multi-item economic lot-sizing problem with remanufacturing. Based on our experimental results, we can conclude that solving the subproblem for each item separately using a strengthened formulation for each of them allowed us to achieve optimal or much better almost-optimal solutions.

5. Final comments

We studied the multi-item economic lot-sizing with remanufacturing and uncapacitated production. We proposed an approach which solves each of the items separately using an effective reformulation based on valid inequalities [3]. The approach is composed of two cycles, considering that in the first cycle, the total available running time is initially uniformly distributed amongst all items, and the allowed running time is repeatedly updated after a certain number of items is treated. In case there are still unsolved items and the time limit is not yet exceeded, the second cycle revisits these unsolved items until there is available time or until they are solved to optimality.

The computational results showed that our approach can generate very strong bounds and high quality feasible solutions, allowing to improve over all the best known results available in the literature. Besides, several instances could be solved to optimality. It is important to remark that the remaining open gap was below 0.5% for almost all of the instances which were not optimally solved.

We remark that it would be interesting to treat more challenging problems in which the items are not independent from each other. A possibility would be to consider the multi-item economic lot-sizing problem in which different items can share the capacity for the same type of returned items. Other remanufacturing problems which could be further studied are capacitated variants of the multi-item economic lot-sizing with remanufacturing [11] and multi-item remanufacturing lot-sizing problems with stochastic demands [8].

References

[1] M.A. Aloulou, A. Dolgui, M.Y. Kovalyov, A bibliography of non-deterministic lot-sizing models, International Journal of Production Research 52 (2014) 2293–2310.

[2] E. Bazan, M.Y. Jaber, S. Zanoni, A review of mathematical inventory models for reverse logistics and the future of its modeling: An environmental perspective, Applied Mathematical Modelling 40 (2016) 4151–4178.

[3] J.O. Cunha, R.A. Melo, A computational comparison of formulations for the economic lot-sizing with remanufacturing, Computers & Industrial Engineering 92 (2016) 72–81.

[4] G.A. DeCroix, J.S. Song, P.H. Zipkin, Managing an assemble-to-order system with returns, Manufacturing & Service Operations Management 11 (2009) 144–159.

[5] M.J.R. Helmrich, R. Jans, W. van den Heuvel, A.P.M. Wagelmans, Economic lot-sizing with remanufacturing: complexity and efficient formulations, IIE Transactions 46 (2014) 67–86.

[6] Y. Li, J. Chen, X. Cai, Uncapacitated production planning with multiple product types, returned product remanufacturing, and demand substitution, OR Spectrum 28 (2006) 101–125.

[7] Y. Li, J. Chen, X. Cai, Heuristic genetic algorithm for capacitated production planning problems with batch processing and remanufacturing, International Journal of Production Economics 105 (2007) 301–317.

[8] P. Macedo, A. Douglas, S. Maristela, L. Muris, A. Moreno, Hybrid manufacturing and remanufacturing lot-sizing problem with stochastic demand, return, and setup costs, The International Journal of Advanced Manufacturing Technology 82 (2016) 1241–1257.

[9] P. Piñeyro, O. Viera, The economic lot-sizing problem with remanufacturing: analysis and an improved algorithm, Journal of Remanufacturing 5 (2015) 1–13.

[10] Y. Pochet, Mathematical programming models and formulations for deterministic production planning problems, in: M. Jünger, D. Naddef (Eds.), Computational Combinatorial Optimization: Optimal or Provably Near-Optimal Solutions, Springer Berlin Heidelberg, Berlin, Heidelberg, 2001, pp. 57–111.

[11] F. Sahling, A column-generation approach for a short-term production planning problem in closed-loop supply chains, Business Research 6 (2013) 55–75.

[12] T. Schulz, A new silver–meal based heuristic for the single–item dynamic lot sizing problem with returns and remanufacturing, International Journal of Production Research 49 (2011) 2519–2533.

[13] A. Sifaleras, I. Konstantaras, General variable neighborhood search for the multi-product dynamic lot sizing problem in closed-loop supply chain, Electronic Notes in Discrete Mathematics 47 (2015) 69–76.

[14] A. Sifaleras, I. Konstantaras, Variable neighborhood descent heuristic for solving reverse logistics multi-item dynamic lot-sizing problems, to appear in Computers & Operations Research (2017).

[15] A. Sifaleras, I. Konstantaras, N. Mladenović, Variable neighborhood search for the economic lot sizing problem with product returns and recovery, International Journal of Production Economics 160 (2015) 133–143.