J Intell Manuf (2017) 28:929–944
DOI 10.1007/s10845-014-1030-4

BFO: a hybrid bees algorithm for the multi-level capacitated lot-sizing problem

Marcos Mansano Furlan · Maristela Oliveira Santos

Received: 11 November 2013 / Accepted: 20 December 2014 / Published online: 3 January 2015

Abstract This paper presents a hybrid heuristic based on the bees algorithm combined with the fix-and-optimize heuristic to solve the multi-level capacitated lot-sizing problem. The bees algorithm can be used as a new method to determine the sequence in which to apply the partition in the fix-and-optimize approach. This new manner of choosing the partition adds diversity to the solution pool and yields different local optima solutions after some iterations. The bees-and-fix-and-optimize (BFO) algorithm attempts to avoid these local optima by performing random search in accordance with the concept of bees algorithm. The BFO has yielded good results for instances from the literature and, in most cases, the results are superior to the best results provided by approaches presented in recent literature. They show that this construction concept is advantageous and illustrate the efficiency of hybrid methods composed of matheuristics and metaheuristics. Furthermore, the BFO approach is a general-purpose heuristic that can be applied to solve other types of production planning problems.

Keywords Lot-sizing problem · Multi-level problem · Bees algorithm · Fix-and-optimize

Introduction

This paper addresses the multi-level capacitated lot-sizing problem (MLCLSP) and presents a new general solution method. The multi-level lot-sizing problem consists in determining a production plan that supplies the demand for a number of products (final items) and/or their components without exceeding the available production capacity. The demand for final items and their components is specified in each period of a finite planning horizon and backlogging is not allowed. The objective is to minimize the setup, holding and overtime costs. The setup time and cost of each item are considered when determining whether to start production in any machine in a given period. This problem has been studied in-depth and many models involving various features and assumptions have been proposed to solve it. Good reviews of the lot sizing problem can be found in Pochet and Wolsey (2006) and Jans and Degraeve (2007).

MLCLSP is a big bucket problem because more than one product may be produced per period. In some problems, such as small bucket problems, the planning horizon is divided into very small periods and there is a limit on the number of items that can be produced in one of these periods. Small bucket problems include the discrete lot sizing problem (DLSP), the continuous setup lot sizing problem (CSLP), the proportional lot sizing and scheduling problem (PLSP) and the general lot sizing and scheduling problem (GLSP). Drexl and Kimms (1997) present and explain the main differences of the formal models for small bucket problems.

Billington et al. (1983) proposed a mathematical formulation that extended the formulation of the single-level capacitated lot-sizing problem. Regarding computational complexity, Bitran and Yanasse (1982) proved that the capacitated lot-sizing problem is NP-hard and Maes et al. (1991) demonstrated that the feasibility problem is NP-complete when setup times are considered. Because of their complexity, most of the solution approaches to solve them are heuristic- or metaheuristic-based. Another reason for the success of such approaches is their flexibility, which enables them to handle large and complex problems (Jans and Degraeve 2007).

M. M. Furlan (✉) · M. O. Santos
Departamento de Matemática Aplicada e Estatística, Instituto de Ciências Matemáticas e de Computação , Universidade de São Paulo, Caixa Postal 668, São Carlos, SP 13560-970, Brazil
e-mail: mafurlan@icmc.usp.br

M. O. Santos
e-mail: mari@icmc.usp.br

Tempelmeier and Derstroff (1996) combined a Lagrangian relaxation and a dynamic programming method to solve problems containing up to 100 items and 16 periods. Stadtler (2003) proposed a rolling horizon procedure based on a relax-and-fix heuristic. This type of method solves the problem iteratively using linear relaxation and decomposition schemes. A reformulation based on the simple plant location problem (SPL) was performed so that a tighter mathematical model could be obtained. Akartunalı and Miller (2009) proposed a framework that uses a relax-and-fix method to solve the same problem and an LP-and-fix method to improve the solution upper bound.

Almeder (2010) proposed a hybrid method that combines ant colony optimization and the commercial LP/MIP solver CPLEX to solve the MLCLSP. The author concluded that his approach is superior to some others for small- and medium-sized instances. For large instances, his hybrid method solution is among the best.

Helber and Sahling (2010) present a heuristic based on mathematical formulation (matheuristic) which yielded high-quality solutions for the MLCLSP while requiring low computational time and outperforming some other algorithms. That matheuristic was initially developed by Pochet and Wolsey (2006) who referred to it as Exchange. Sahling et al. (2009) recalled it fix-and-optimize and proposed decomposition schemes to use it to solve the MLCLSP with setup carryover. In this method, a set of decision variables that represent the binary setup variables is split into a finite number of disjunct subsets. In each iteration of the method, a group of decision variables is fixed to the incumbent solution values, and only the variables of a chosen subset are released to be optimized. Moreover, all real variables are free to be optimized during the entire process. The fix-and-optimize method as proposed by Helber and Sahling (2010) uses a predefined sequence to release the subsets. If a better solution has been found, it becomes the new incumbent solution.

The fix-and-optimize algorithm tends to local optima because it iteratively obtains the optimal (or nearly optimal) solution and analyses only a subset of the decision variables. The method is very similar to a local search based on the mathematical model and can be used embedded in another approach for this purpose, maintaining its advantages. Moreover, it is well known that it is hard to optimality solve medium and large sized instances of MLCLSP. It can be difficult to determine good solutions, in a limited computational time, when positive setup times are considered (Maes et al. 1991).

Wu et al. (2012) proposed a solution method based on linear relaxation. By using the relax-and-fix partial solutions and a rounding-based heuristic solution, their method yields values for the decision variables and decides which variables should be fixed. After these analyses, the fixed sub-problem is solved using a mixed integer solver. These steps are performed iteratively until a stopping criterion has been satisfied. The results were compared with those obtained by Akartunalı and Miller (2009) and those obtained by the solver. In these comparisons, the heuristic yielded better results in most instances considered and improved the average quality of the solution.

Li et al. (2012) proposed a three-step heuristic to solve the single and multi-level lot-sizing problems, both including capacity constraints. The authors considered the problems with and without the use of joint setup costs, which are the costs associated with the decision to produce items in a given period and independent of the quantity and variety of items produced. The proposed method includes an initial step of treating the joint setup costs. Then, an initial solution is determined and subsequently improved. The results were compared with those obtained using CPLEX 11.1 and with the literature instances bounds. In both comparisons, the proposed method yielded better results on average.

Several studies have considered other characteristics associated with the MLCLSP as, for example, demand backlogs (Toledo et al. 2013; Wu et al. 2013; Akartunalı and Miller 2009). These studies consider another set of tests proposed by Akartunalı and Miller (2009), which imposes the necessity of demand backlogs concerning. Wu et al. (2013) included the setup carry-over feature, which was also considered by Sahling et al. (2009). Goren et al. (2010) conducted a review of genetic algorithms applied to solve lot-sizing problems by grouping the studies according to the lot-sizing characteristics considered and the genetic algorithm operators used.

This paper proposes a hybrid method to address the MLCLSP by combining the fix-and-optimize heuristic with the metaheuristic bees algorithm proposed by Pham et al. (2005). According to Puchinger and Raidl (2005), the proposed heuristic is categorized as an integrative method because the fix-and-optimize heuristic is incorporated into the bees algorithm. A survey of some important lines of hybridization is presented in Blum et al. (2011). The authors classified the methods according to the classes of the component methods. Based on their classification, the BFO fits the "hybridizing metaheuristics with (meta)heuristics" category.

A similar hybrid approach was developed by James and Almada-Lobo (2011) to solve the single and parallel machine capacitated lot-sizing and scheduling problems. Their method is composed of two steps. In the first, the relax-and-fix algorithm is used to construct an initial solution. In the next step, the fix-and-optimize algorithm is used to improve the initial solution by using period and product decompositions. The choice of partitions is probabilistic and the initial chances are equal. A partition has a lower chance of being selected again when it is applied more often than the current partition. To escape from local optima, the method applies a tool inspired by variable neighbourhood search (VNS),

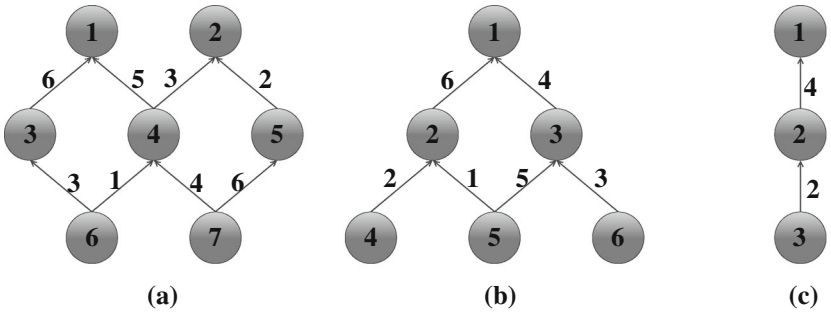

Fig. 1 Product structures. **a** General structure. **b** Assembly structure. **c** Serial structure

which combines partitions with more periods. Once a better solution has been found, the search returns to the initial partition sizes. The results were compared with literature methods and the results of CPLEX 12.1. The method has shown superior performance for single machine instances. There were some convergence difficulties for parallel machine instances and the authors argue that they were caused by the increased number of local minima and the inclusion of the machine assignment problem.

The key idea of the BFO method is to use the fix-and-optimize heuristic as the local search method for the bees algorithm and therefore obtain the advantages of high solution quality, low computational time, and ability to avoid local minima or reach higher quality local minima. This approach differs from the method proposed by Helber and Sahling (2010) in the manner the partitions are applied and the number of stored incumbent solutions. In Helber and Sahling (2010) only one solution is stored and the chosen sequence of partitions is predicted. In contrast, the BFO has a pool of solutions (bees) and the partitions implemented for each bee are randomly selected from a set of possibilities. Period and product decompositions was employed and three different variants was developed based on these decompositions. The first uses only product decomposition, the second uses only period decomposition and the third uses both decompositions.

The paper is organized as follows: "Problem formulation" section introduces the formulation of the MLCLSP; "Hybrid method" section presents the proposed hybrid algorithm and presents a more detailed explanation of the fix-and-optimize heuristic, the bees algorithm, the construction of initial solutions and the decomposition schemes employed; "Computational experiment" section describes the computational experiments that demonstrate the high quality of solutions; Finally, In section "Conclusion" draws some conclusions and suggests some future work.

Problem formulation

The lot sizing problem consists in finding a production plan that minimizes the sum of setup costs, inventory holding costs and overtime costs and meets demands of items in time. To formulate the multi-level problem, a product structure represented by an acyclic graph whose nodes correspond to the components and edges correspond to precedence relations is considered. Figure 1 shows three types of product structures. In Fig. 1a, there are two end items (1 and 2) and the other items (3–7) are components. For example, the production of one unit of item 1 requires six units of component 3 and five units of component 4.

To better understand the MLCLSP mathematical model presented below, consider the following sets, indices, parameters and variables.

Sets and indices

N	Number of products ($i \in \{1, \ldots, N\}$);
M	Number of resources ($m \in \{1, \ldots, M\}$);
T	Planning horizon ($t \in \{1, \ldots, T\}$);
$S(i)$	Set of successors of product i;
$A(i)$	Set of predecessors of product i;
K_m	Set of products that require resource m.

Parameters

cs_i	Setup cost of product i;
h_i	Holding cost of product i per period;
oc_m	Overtime cost per unit of overtime for resource m;
r_{ij}	Units of product i required for the production of one unit of product j;
d_{it}	External demand for product i in period t;
a_i	Processing time per unit of product i;
ts_i	Setup time for product i;
C_{mt}	Capacity of resource m in period t.

Decision variables

O_{mt}	Overtime quantity for resource m in period t;
X_{it}	Amount of item i produced in period t;
I_{it}	Inventory level of product i at the end of period t;

Y_{it} Setup decision variable for product i in period t;
($Y_{it} = 1$, if $X_{it} > 0$
$Y_{it} = 0$, otherwise.)

Consider B_{it}, an upper bound in the production amount of item i in period t (X_{it}), which is defined as the cumulative echelon demand. The echelon demand can be calculated as $D_{it} = d_{it} + \sum_{j \in S(i)} D_{jt}$ where d_{it} is the external demand and $\sum_{j \in S(i)} D_{jt}$ is the component demand. The cumulative demand is calculated as follows: $B_{it} = \sum_{\pi=t}^{T} D_{i\pi}$.

$$\text{minimize} \quad Z = \sum_{i=1}^{N}\sum_{t=1}^{T}(cs_i \cdot Y_{it} + h_i \cdot I_{it}) + \sum_{m=1}^{M}\sum_{t=1}^{T} oc_m \cdot O_{mt} \tag{1}$$

$$\text{subject to:} \quad I_{i,t-1} + X_{it} - \sum_{j \in S(i)} r_{ij} \cdot X_{jt} - I_{it} = d_{it} \qquad \forall i, t \tag{2}$$

$$\sum_{i \in K_m} (a_i \cdot X_{it} + ts_i \cdot Y_{it}) \leq C_{mt} + O_{mt} \qquad \forall m, t \tag{3}$$

$$X_{it} \leq B_{it} \cdot Y_{it} \qquad \forall i, t \tag{4}$$

$$X_{it}, I_{it} \geq 0 \qquad \forall i, t \tag{5}$$

$$Y_{it} \in \{0, 1\} \qquad \forall i, t \tag{6}$$

The objective function (1) minimizes the sum of setup, holding and overtime costs. Constraints (2) represent the inventory balance constraints. Constraints (3) ensure that the production and setup times do not exceed the capacity increased by overtime. Because the overtime cost is high, the solution tends to use the minimum necessary overtime. A product can be produced at a given time t only when the machine is prepared to produce it in the same period t, which is guaranteed by constraints (4). Constraints (5) ensure that the quantities produced and the inventories are positive. The domain of the binary decision variables is treated in constraints (6).

It is well known that this formulation yields a poor LP relaxation and is not adequate for the branch-and-bound phase of a standard MIP solver. According to Stadtler (2003), several reformulations proposed in the literature can yield tighter relaxations. Here, the SPL reformulation (Stadtler 2003) is used to improve the solution quality of the solver.

Hybrid method

This section presents a hybrid method that attempts to avoid local minima, based on BA fashion, and maintain the fix-and-optimize benefits. The fix-and-optimize approach can be considered a local search procedure in which the application of different partitions to an incumbent solution produces different neighbourhoods. The BFO uses the fix-and-optimize heuristic to perform the BA local search. This type of composition enables the BFO to solve the sub-problems quickly and avoid local minima by using the random search from the BA framework. In this paper, the performance of this type of hybridization applied to the problem is investigated, because the literature lacks an approach by which partitions are chosen in a non-deterministic manner. In the following subsections, the initial solution, the fix-and-optimize heuristic, the decomposition schemes, the bees algorithm and the proposed method are presented.

Initial solution

To obtain an initial solution three different procedures are considered. The first procedure consists of the lot-for-lot (LFL) approach, i.e. each product i is produced in each period t to supply the demand. The initial population is homogeneous because the same solution is replicated and the random choice of partitions adds the characteristics necessary to obtain different final solutions.

The second method uses the populate function available in the CPLEX framework. This function solves one problem until a predefined number of integer solutions has been found. To avoid long run times to find the initial solutions, a run time limit equal to 10 % of the BFO time limit is used. If necessary, the pool of solutions is fulfilled with LFL solutions.

The third procedure is a greedy random constructive heuristic, which is a modified version of the LPH1 procedure proposed by Maes et al. (1991). At each iteration of the LPH1 procedure, the solution of the current LP relaxation is used to select the setup variable to be fixed. If the value of the selected variable is an integer, the algorithm fixes it and moves to the next iteration without resolving the LP relaxation. Otherwise, this variable is fixed to one and the LP relaxation is resolved. Algorithm 1 provides the pseudo-code for this constructive procedure, which is referred to as a Greedy Randomly Round Constructive Heuristic (GRRCH).

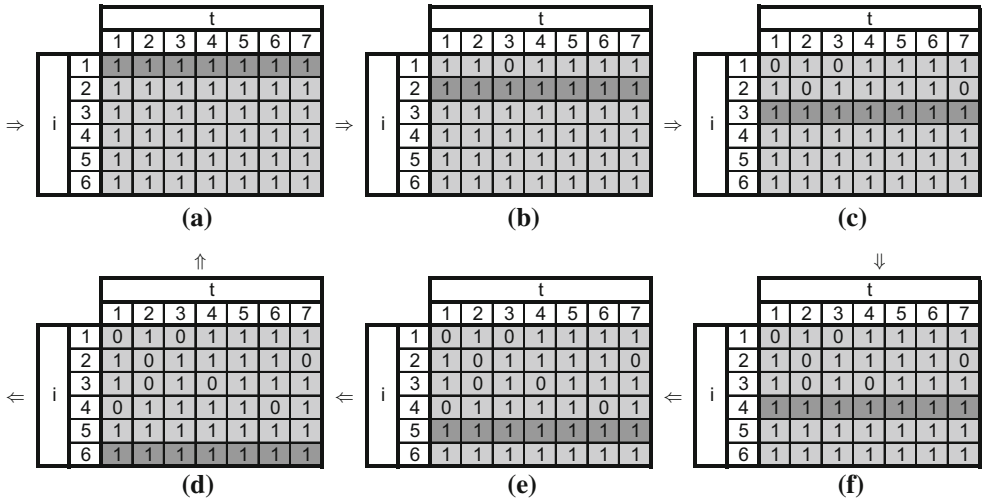

Fig. 2 Fix-and-optimize steps with product decomposition. **a** Release product 1. **b** Release product 2. **c** Release product 3. **d** Release product 6. **e** Release product 5. **f** Release product 4.

Algorithm 1: The GRRCH pseudo-code

```
Given k (the number of variables to be fixed in each iteration);
Given α (the setup variables selection interval for the round method);
Solve the LP relaxation;
while Any relaxed variable remains do
    Find the maximum non-integer setup value (Y_max);
    Select all setup variables that differ by at most α from Y_max,
    i.e., C = {Y_it | Y_it + α >= Y_max ∀i, t};
    Randomly select k setup variables from C;
    Switch their values to one and fix them all;
    Resolve the LP relaxation using the fixed variables;
    if the problem is infeasible then
        Release the variables fixed in this iteration;
    end
    if Any non-fixed variable has an integer value then
        Fix this variable;
    end
    if The current solution requires more overtime than the initial
    solution determined by the LFL approach then
        Release all variables fixed to zero;
    end
end
```

Fix-and-optimize heuristic

This section describes the fix-and-optimize heuristic proposed by Sahling et al. (2009) and illustrates it with a small example. The basic idea of the heuristic is to iteratively solve a series of MIP sub-problems derived from the overall MIP problem. In each iteration, a set of integer variables is fixed and the sub-problem is solved. This strategy reduces the number of integer variables in the remaining MIP in each iteration. Thus, the execution time of the remaining sub-problem is expected to be shorter because fewer binary variables are released.

The fix-and-optimize heuristic can be considered a local search method that, given an incumbent solution, attempts to iteratively find better solutions. Consider that integer variables (Q) have been decomposed into R subsets ($Q^r\ \forall r = 1, \ldots, R$), where $Q = Q^1 \cup \ldots \cup Q^R$ and $Q^1 \cap \ldots \cap Q^R = \emptyset$. In each iteration, one subset is released and the remaining variables are fixed to the incumbent values. For example, in the first step, Q^1 is released and $Q^2 \cup, \ldots, \cup Q^R$ is fixed; in the second step, Q^2 is released and $Q^1 \cup Q^3 \cup, \ldots, \cup Q^R$ is fixed, and so forth. Once a better solution has been determined, the incumbent solution is updated and all partitions can be used again. If all partitions have been tried and there has been no improvement, then the heuristic ends.

Figure 2 shows an example of a fix-and-optimize algorithm applied to the problem addressed in Section "Problem formulation". Each cell represents a decision variable (Y_{it}), each table represents a solution and each color represents a state of variable Y_{it}. If the cell's color is light gray, then the integer variable is fixed. Conversely, if the cell's color is dark gray, then the variable is "free" to be optimized. In this example, the product decomposition is considered and the subsets are used in an incremental sequence. However, another decomposition can be introduced and different sequences can be used. Each table represents one iteration of the algorithm and in each iteration, only the integer variables are fixed and the other variables are optimized. Figure 2a shows the situation in which the setup variables for product 1 ($Y_{1t}\ \forall t$) are released to be optimized and the remaining binary variables are fixed to their incumbent values.

The procedure is repeated until a local minimum has been reached or another stop criterion has been met. Because this approach is deterministic, it may lead to a poor local minimum for difficult instances. James and Almada-Lobo (2011) developed a stochastic MIP-based local search improvement

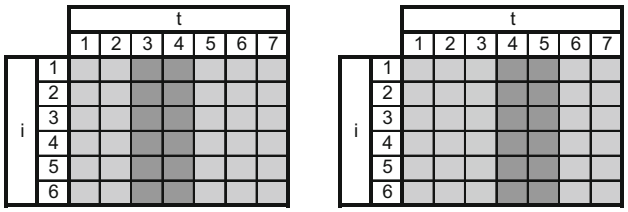

Fig. 3 Example of a period decomposition

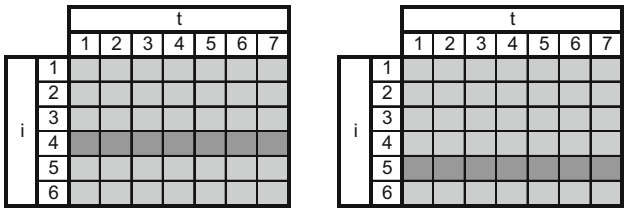

Fig. 4 Example of a product decomposition

heuristic that attempts to improve any given feasible solution. The key idea is to solve a series of sub-problems randomly selected using an iterative fix-and-optimize approach. In the approach proposed by the authors, local minima are avoided. Here, a similar process that attempts to avoid the same drawback is used.

For a detailed description of this heuristic the reader is referred to Helber and Sahling (2010).

Decomposition schemes

This paper proposes a period decomposition that considers setup decisions of two successive periods to be released at each partition. This implies that overlapping between two consecutive partitions is considered. A product decomposition proposed by Helber and Sahling (2010) is also used. In this case, the setup decisions for only one product are released for each partition throughout the planning horizon, i.e., the number of partitions is equal to the number of products, therefore there is no overlapping in this decomposition. Figures 3 and 4 illustrate two successive partitions for both decompositions.

As in Helber and Sahling (2010), the decomposition types are combined in the following three variants of the hybrid heuristic: the BFO-P1 variant uses only the product decomposition, the BFO-P2 variant uses the period decomposition and the BFO-P3 uses product and period decompositions randomly selected.

Bees algorithm

Many solution approaches based on the food foraging behaviour of honey bees have been proposed to address problems that include constrained optimization, supplier selection and dynamic optimization problems. Examples of optimization problems tackled by these methods are: dynamic optimization problem (Castellani et al. 2012), job scheduling problems (Pham et al. 2007), constrained optimization problems (Brajevic and Tuba 2013), process planning problem (Wen et al. 2014), supplier selection-order allocation (Jain et al. 2013), production control systems (Ajorlou and Shams 2013), among others. The bees algorithm (BA) proposed by Pham et al. (2005) was chosen.

The idea of the BA is based on the food foraging behaviour of honey bees. The swarm's objective is to maximize the food obtained by considering the distance from the food sites (sources or patches) and the facility of food purchase. In the first phase, scout bees are randomly sent to find food sites. When these bees return to the hive, they deposit the nectar and go to the "dance floor" to perform a dance called the "waggle dance". This dance indicates the direction in which the flower source has been found, the distance from the hive to the food source and the quantity of food. Using this information the hive sends bees to the sites such that more bees are sent to the more promising food sources.

In optimization terms, each bee represents a possible solution to a given problem and the bee's search is performed in two phases. In the first phase, a random search is performed and the sites are defined. In the second phase, a number of sites is selected according to the method parameters and local searches are made at each site to improve their solution. Conversely, a random search is performed at the other sites to maintain diversity and reach better ones. The fix-and-optimize decomposition scheme was chosen to perform the local searches due to its efficiency.

The BA was initially constructed for continuous problems; however some papers have proposed its use to solve combinatorial problems. Özbakir et al. (2010) used the BA to solve the generalized assignment problem and Dereli and Das (2011) proposed using it to solve the container loading problem. Algorithm 2 shows a basic bees algorithm pseudocode for the algorithm proposed by Pham et al. (2005).

The bees algorithm has the following parameters: nb is the number of scout bees, mb is the number of scout bees chosen for the local search, eb is the number of best scout bees chosen for a more intense local search, nep is the number of flower bees sent to follow the "eb" best scout bees, and nsp is the number of flower bees sent to follow the "$mb-eb$" other selected scout bees.

Main procedure

The main procedure defines the proposed method. The method starts by populating the pool of solutions (bees) using the preselected method. The number of bees, the bees selected to attend the local search and other parameters are

Algorithm 2: Basic bees algorithm pseudo-code

```
1 Initiate with random solutions;
2 Evaluate and classify these solutions;
3 while stopping criteria are not met do
4     Randomly select "mb" scout bees (sites);
5     Recruit bees to perform local search in the selected sites
      (amount of bees is defined according to their promise) and
      evaluate their fitness;
6     Select the best bee from each site;
7     Assign the remaining bees to perform the random search and
      evaluate their fitness;
8     Classify the new population;
9 end
```

also predetermined and must be adjusted according to the problem addressed. Given the pool of solutions, the bee algorithm starts by applying the fix-and-optimize partitions as its local search method.

In each local search iteration, the method apply at most "nep" (best sites) or "nsp" (others) partitions randomly chosen to the incumbent solution and iteratively solve one sub-problem for each partition. Once a superior solution has been found, the local search procedure stops, the new incumbent solution returns and all the partitions used are available for this site again, i.e., all partitions are considered for the new incumbent. Otherwise, the procedure stops without determining a new solution. The local search can be used until the incumbent solution does not have any available partitions to apply. Figure 5 illustrates one iteration of the local search procedure for a given solution.

The random search procedure is applied only when a site does not have any partitions to be used (a local minimum has been attained). The aim of this procedure is to avoid the local minima obtained by the decomposition scheme in the local search approach. It releases the $RP\%$ setup variables randomly selected to be optimized, given that RP is the release percentage used by the random search. If a superior solution is found after the random search procedure, the local search procedure restarts with all partitions for this site.

The BFO algorithm continues until all the bees have made all possible local searches, i.e. after the random search procedure has been applied. The BFO also limit total time (T_{max}), number of iterations (l_{max}) and time to solve each sub-problem. The time limit (in seconds) to solve the sub-problem is proportional to the number of "free" setup variables and is set using a multiplier β ($TL = |free\ variables| * \beta$).

Algorithm 3 introduces the BFO pseudo-code, which shows the main steps of this approach. First, the initial population of scout bees is determined using one out of the three policies mentioned above (line 1). In the LFL approach, the deterministic solution is copied to fill the pool of initial solutions. At each iteration, the BFO chose at most mb scout bees

Algorithm 3: BFO pseudo-code

```
1  Create the initial population of scout bees (nb solutions) using
   the LFL policy, the CPLEX populate method or the GRRCH
   procedure;
2  Classify the scout bees according to the solution quality;
3  while l < l_max and there are bees with unused partitions and
   time ≤ T_max do
4      Randomly choose "mb" scout bees to perform the local
       search into among the bees with partitions available;
5      for Each chosen scout bee do
6          if this solution is one of the eb best solutions then
7              Randomly choose nep available partitions to be
               applied (recruit bees);
8              Do a local search with the chosen partitions;
9              Remove the partition from the unused set of this scout
               bee;
10         else
11             Randomly choose nsp available partitions to be
               applied (recruit bees);
12             Do a local search with the chosen partitions;
13             Remove the partition from the unused set of this scout
               bee;
14         end
15         if a better solution has been found then
16             Store the new solution as the scout bee;
17             Reset the partition set including each applied partition;
18         end
19     end
20     for each random bee (local minima) do
21         Randomly choose RP % of the setup variables to be
           released;
22         Do a random search using the chosen released setup
           variables;
23         if a better solution has been found then
24             Reset the partition set including each applied partition;
25         end
26     end
27     Classify the new population according to the solution quality;
28 end
29 Return the best solution found;
```

to perform the local search (line 4). Only non-random bees, i.e. bees with unused partitions can be subjected to the local search. After this step, nep or nsp partitions are chosen to perform the local search, in which the number of partitions depends on the solution quality: nep partitions are chosen for the eb best solutions and nsp partitions are chosen for the $mb - eb$ other solutions (lines 6–14). If a site does not have nep or nsp partitions to be applied, the method uses all the available partitions. The best solutions found by the local search procedure are stored as the new population of scout bees (each best solution overwrites its own scout bee) (lines 15–18). Finally, random searches are performed for each random bee with no available partitions and if a better solution is found, this random bee returns to the scout bees group (lines 20–26). The BFO returns the best solution found in the last iteration because the solutions do not become worse (line 9).

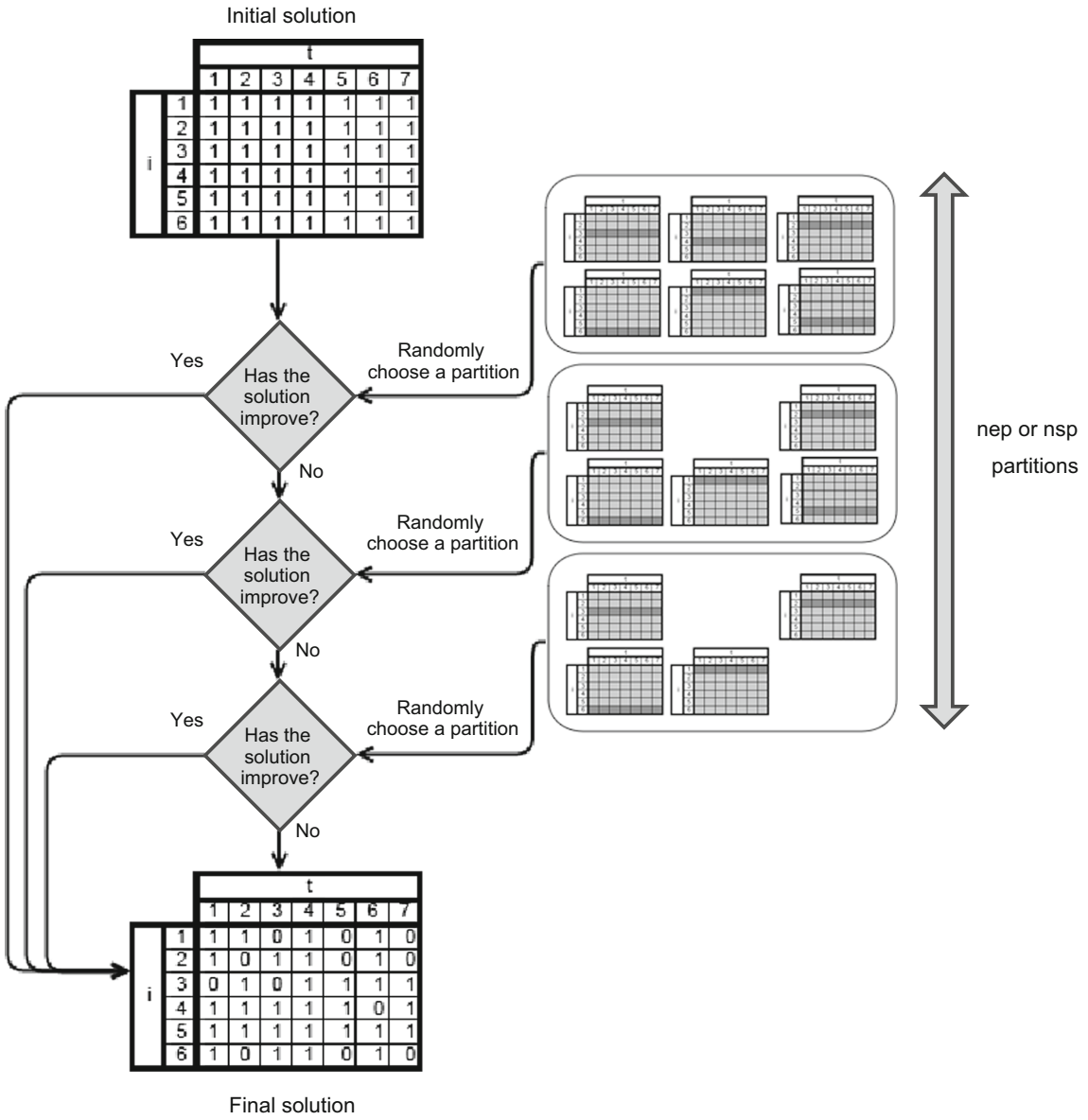

Fig. 5 Local search scheme proposed

Computational experiments

This section reports on some computational experiments performed in the test sets of Tempelmeier and Derstroff (1996) and Stadtler and Sürie (2000). It discusses the benefits and disadvantages of the BFO in comparison with three other heuristics (from Helber and Sahling (2010); Tempelmeier and Derstroff (1996) and Stadtler (2003)) and the standard LP/MIP solver CPLEX 12.2.

The proposed algorithm is coded in C++ and runs on 2.8GHz Intel Pentium i5 with 4GB of memory. The CPLEX use is limited to only one thread, since the other methods use only one. Each variant of the proposed heuristic was run 5 times for each test problem.

The parameter values used in the computational experiments were set empirically as follows: $T_{max} = 600$ seconds (total time limit), $l_{max} = 10,000$ (maximum number of iterations), $\alpha = 0.5$ (interval length of rounding for the constructive heuristic), $\beta = 0.1$ second/released variable (multiplier for the time limit for solving the sub-problem), $nb = 4$ (number of scout bees), $mb = 2$ (number of scout bees chosen for the local search), $eb = 1$ (number of best scout bees chosen for a more in-depth local search), $nep = 5$ (number of bees recruited for the "eb" best sites), $nsp = 2$ (number of bees recruited for the "mb-eb" other sites) and $RP = 20$ (release percentage used for the random search).

Table 1 shows some characteristics of the test sets, which include assembly and general product structures containing

Table 1 Parameters of the Tempelmeier and Derstroff (1996)* and Stadtler and Sürie (2000)** test sets

Test set	No. products	No. periods	No. resources	Setup times profiles	No. test instances
A+**	10	24	3	No	120
B+**	10	24	3	Yes	312
C*	40	16	6	No	180
D*	40	16	6	Yes	80
C+**	40	48	6	No	10
D+**	40	48	6	Yes	10
E+**	100	48	10	No	10

10–40 products and a planning horizon between 16 and 24 periods. Test sets C+, D+ and E+ (Stadtler and Sürie 2000) were also considered. Although not commonly used in the literature, these data were used to verify the asymptotic performance of the BFO in comparison with CPLEX.

Results

This section provides the computational results of the test sets described in Table 1. For the four first test set and for each solution method, the tables below show the average perceptual deviation (Gap) between the upper bound reported by the authors and the average best solution obtained by each solution method ($Gap = 100 * \frac{feasible\ solution - upper\ bound}{upper\ bound}$), the average improvement from the initial solution to the final solution obtained by the proposed algorithm (Improv.), the standard deviation calculated using all instances from the test set (SD), the percentage of the feasible solutions (Feas), the average solution time in seconds (Time(s)) and the solution time limit in seconds (Limit(s)).

These computational tests aim to verify the performance, compare results with those of approaches from the literature and ascertain the reliability using the standard deviations and performance charts.

For the four first test sets, the BFO results is compared with the best fix-and-optimize version by Helber and Sahling (2010) (HS-FO4), which uses a product-oriented decomposition first, then a resource-oriented decomposition and finally a process-oriented decomposition (variant 4). The BFO solutions is also compared with the results reported by Helber and Sahling (2010) for the lagrangian decomposition method developed by Tempelmeier and Derstroff (1996) (TDH) and the rolling horizon method proposed by Stadtler (2003) (Sta).

The three heuristics were run by Helber and Sahling (2010) on a 2.13 GHz Intel Pentium Core 2 of 4GB memory, whereas the BFO was run on a different computer, as mentioned above. The tables also provide the results of the proposed method using each of the three initial solution populations. The variants of the hybrid method are BFO-P1, which uses only the product decomposition, BFO-P2, which uses the period decomposition, and BFO-P3, which uses product and period decompositions randomly selected.

For test sets C+, D+ and E+, the performance of the BFO variants is briefly compared with that of CPLEX. The computational time limit was set to 3,600 seconds, since the planning horizon of such instances are larger and harder to solve. The computational performance of the test sets is addressed in Section "Additional computational tests".

Table 2 provides the results of the three methods for test set A+. In this set, only 108 out of 120 instances have at least one known solution that does not require overtime. The numbers without brackets were calculated considering only the feasible solutions that do not require overtime and the numbers with brackets were calculated considering all solutions.

Regarding the quality of the solutions of the upper bound (gaps), Table 2 shows that the HS-FO4 method obtains solutions that are, on average, superior to those obtained by the BFO method with product decomposition in all cases. However, for other variants of the hybrid method, the solutions are better. When the Sta and TDH heuristics is considered, all versions of the hybrid method performed better.

In contrast, the BFO requires more computational time than HS-FO4 and TDH to solve the problem. In relative terms, the BFO-P2 and BFO-P3 methods require significantly more time than HS-FO4, however, in absolute terms, the computational time required for this test set is short. The standard deviation (SD) indicates that the BFO results have a small dispersion, i.e. the differences in the solution quality among different instances and runs are relatively small.

The average computational times are not significantly different for the solutions with and without infeasible instances (those that use overtime). In Table 2, the gaps and solution times are very similar for each initial population. These results show that, in this test set, the different methods for constructing the initial population do not disturb the proposed hybrid search method, suggesting that the method is stable. The improvement values obtained for test set A+ show the differences in the quality of the solutions yielded by each procedure used to produce initial solutions. LFL procedure generates solutions improved in almost 50 %, CPLEX returns superior solutions and GRRCH returns solutions with quality between those returned by the other methods. These results show that the BFO improvement over the initial population is significant (over 20 %).

The results of all variants of the hybrid method for the B+ test set show the same tendency observed in Table 2. In Table 3, some negative average percentage deviations can be found, indicating improvements over the literature upper bounds. For example, the BFO-P3 method with initial solutions obtained using the GRRCH approach yields, on average, an improvement of 0.42 %. For the B+ test set, small

Table 2 Results for the A+ test set: a summary of the solutions

Init. Sol.	Method	Gap (%)	Improv. (%)	SD (%)	Feas (%)	Time (s)	Limit (s)
	CPLEX	0.93 (0.81)	#	1.68 (1.64)	100.00 (90.00)	600.00 (600)	600
	TDH	13.19	#	#	100.00	0.08	#
	Sta	4.14	#	#	99.07	13.49	14
LFL	HS-FO4	0.78	#	#	100.00	6.38	#
LFL	BFO-P1	1.58 (1.47)	48.40 (46.39)	1.62 (1.59)	100.00 (90.00)	12.07 (12.19)	600
	BFO-P2	0.39 (0.33)	49.04 (47.01)	1.35 (1.31)	100.00 (90.00)	25.15 (25.12)	600
	BFO-P3	0.17 (0.12)	49.15 (47.11)	1.35 (1.30)	100.00 (90.00)	25.06 (25.03)	600
CPLEX	BFO-P1	1.71 (1.59)	23.21 (22.08)	1.70 (1.68)	100.00 (90.00)	11.49 (11.48)	600
	BFO-P2	0.30 (0.27)	24.31 (23.12)	1.32 (1.27)	100.00 (90.00)	24.48 (24.37)	600
	BFO-P3	0.18 (0.13)	24.40 (23.22)	1.32 (1.27)	100.00 (90.00)	24.94 (24.94)	600
GRRCH	BFO-P1	1.67 (1.55)	36.68 (34.77)	1.77 (1.73)	100.00 (90.00)	11.25 (11.19)	600
	BFO-P2	0.33 (0.27)	37.56 (35.62)	1.28 (1.24)	100.00 (90.00)	24.14 (24.13)	600
	BFO-P3	0.20 (0.15)	37.64 (35.70)	1.28 (1.24)	100.00 (90.00)	24.71 (24.82)	600

Table 3 Results for the B+ test set: a summary of the solutions

Init. Sol.	Method	Gap (%)	Improv. (%)	SD (%)	Feas (%)	Time (s)	Limit (s)
	CPLEX	0.62	#	1.95	98.08	600.00	600
	TDH	12.52	#	#	100.00	0.08	#
	Sta	4.11	#	#	84.62	13.50	22
LFL	HS-FO4	0.41	#	#	100.00	8.40	#
LFL	BFO-P1	1.14	49.28	1.85	100.00	13.11	600
	BFO-P2	−0.19	49.98	1.68	100.00	27.20	600
	BFO-P3	−0.40	50.08	1.65	100.00	27.09	600
CPLEX	BFO-P1	1.10	10.64	1.68	100.00	17.54	600
	BFO-P2	−0.36	11.93	1.68	100.00	31.61	600
	BFO-P3	−0.48	12.03	1.67	100.00	34.01	600
GRRCH	BFO-P1	1.25	36.50	1.85	100.00	12.23	600
	BFO-P2	−0.19	37.45	1.67	100.00	24.80	600
	BFO-P3	−0.42	37.59	1.64	100.00	26.71	600

gaps and standard deviations show the good performance and stability of the BFO.

Table 4 shows that all variants of the hybrid heuristic outperformed all the other heuristics and the solution time required by all heuristics is higher. A possible explanation is that the number of products increased in comparison to the numbers for test sets A+ and B+, increasing both problem size and number of possible solutions. The negative gaps indicate improvements and they are similar in Tables 2 and 3.

Finally, Table 5 shows that the solution quality for the hybrid heuristic applied to test set D and the improvement values are similar to those obtained for test set C (Table 4). However, for a few instances with unknown feasible solutions, the BFO-P1 method yields poor-quality solutions when the GRRCH procedure is used to obtain the initial solutions, which explains the difference between the gap values calculated including and excluding such instances (0.90 and 6.60 %). According to the literature upper bound, 90 % of the instances from test set D have a known solution that does not require overtime. However in many cases, the BFO feasible percentage values are higher than 90 %. High values for the standard deviation may indicate that the BFO method has yielded both poor and good solutions.

Figures 6, 7, 8 and 9 show the convergence curves for the BFO-P1 method, which uses the LFL, CPLEX and GRRCH initial solutions and two runs for one hard instance of test set D. This instance has a general acyclic structure with a profile of 90% resource utilization (for more details, see instance code DG012132 on page 4 of Stadtler and Sürie (2000)). These curves show that the BFO variants converge very rapidly (Fig. 6 for the first run and Fig. 7 for the second run). All the BFO variants continue to improve the solutions

Table 4 Results for the C test set: a summary of the solutions

Init. Sol.	Method	Gap (%)	Improv. (%)	SD (%)	Feas (%)	Time (s)	Limit (s)
	CPLEX	1.69 (1.30)	#	1.99 (1.86)	100.00 (73.33)	600.00 (600.00)	600
	TDH	6.76	#	#	100.00	0.43	#
	Sta	2.14	#	#	99.24	279.21	263
LFL	HS-FO4	1.61	#	#	100.00	40.90	#
LFL	BFO-P1	0.02 (−0.05)	50.09 (41.67)	1.63 (1.47)	100.00 (73.33)	163.55 (168.88)	600
	BFO-P2	−0.94 (−0.85)	50.59 (42.10)	1.35 (1.25)	100.00 (73.33)	563.75 (563.92)	600
	BFO-P3	−1.65 (−1.40)	50.93 (42.37)	1.34 (1.34)	100.00 (73.33)	525.84 (532.10)	600
CPLEX	BFO-P1	0.10 (0.00)	27.53 (22.22)	1.75 (1.59)	100.00 (73.33)	157.69 (161.68)	600
	BFO-P2	−1.40 (−1.21)	28.63 (23.12)	1.32 (1.30)	100.00 (73.33)	541.91 (542.28)	600
	BFO-P3	−1.72 (−1.47)	28.86 (23.31)	1.37 (1.38)	100.00 (73.33)	520.65 (524.87)	600
GRRCH	BFO-P1	−0.05 (−0.11)	32.44 (25.93)	1.66 (1.49)	100.00 (73.33)	158.58 (161.37)	600
	BFO-P2	−1.37 (−1.19)	33.34 (26.69)	1.32 (1.29)	100.00 (73.33)	543.65 (544.07)	600
	BFO-P3	−1.77 (−1.51)	33.61 (26.91)	1.39 (1.40)	100.00 (73.33)	510.94 (516.23)	600

Table 5 Results for the D test set: a summary of the solutions

Init. Sol.	Method	Gap (%)	Improv. (%)	SD (%)	Feas (%)	Time (s)	Limit (s)
	CPLEX	2.25 (0.96)	#	10.85 (11.81)	97.22 (88.75)	600.00 (600.00)	600
	TDH	5.68	#	#	100.00	0.28	#
	Sta	6.99	#	#	88.89	76.73	81
LFL	HS-FO4	2.17	#	#	100.00	34.40	#
LFL	BFO-P1	0.96 (1.08)	55.90 (59.42)	2.29 (14.65)	100.00 (91.00)	99.32 (115.80)	600
	BFO-P2	−1.14 (−3.91)	56.86 (60.53)	1.82 (10.93)	100.00 (92.50)	481.07 (487.67)	600
	BFO-P3	−1.31 (−3.94)	56.93 (60.59)	1.97 (10.55)	100.00 (92.25)	429.30 (441.44)	600
CPLEX	BFO-P1	0.90 (2.51)	22.74 (27.50)	2.22 (20.92)	100.00 (91.25)	134.09 (157.74)	600
	BFO-P2	−1.20 (−1.01)	24.92 (29.95)	1.82 (18.32)	100.00 (91.75)	439.81 (455.82)	600
	BFO-P3	−1.33 (−1.36)	24.67 (29.79)	1.93 (16.59)	100.00 (92.25)	418.60 (436.74)	600
GRRCH	BFO-P1	0.90 (6.60)	36.72 (40.32)	2.26 (34.39)	100.00 (90.00)	96.53 (119.80)	600
	BFO-P2	−1.20 (−1.92)	38.07 (42.55)	1.87 (18.28)	100.00 (92.00)	462.90 (472.78)	600
	BFO-P3	−1.38 (−2.59)	38.18 (42.71)	1.91 (13.10)	100.00 (91.50)	411.19 (427.13)	600

in steps, as shown in Figs. 8 and 9. A step which appears in a curve after a long time without improvement demonstrates the overlap of a local optima. The BFO-P2 and BFO-P3 variants exhibited a similar behaviour with better improvement values, as shown in Tables 2 , 3 , 4 and 5.

The charts of Figs. 10, 11, 12 and 13 show the performance curves of the proposed heuristic in comparison to the performance of CPLEX. This metric of comparison was proposed by Dolan and Moré (2002). The charts were constructed using the normalization of the solutions obtained by each approach and the best solution found in each instance. Given a set of solution approaches (S), a set of instances (P) and a metric of evaluation (t), it is possible to define the quality of a particular solution approach s for a given instance p as $r_{ps} = \frac{t_{ps}}{min\{t_{ps}:\, s \in S\}}$. With this value, the probability that each solution approach will perform better or equal to τ can be calculated as $p_s(\tau) = \frac{|p \in P:\, r_{ps} \leq \tau|}{NP}$, where NP is the number of instances considered. In the charts below the abscissa gives the τ values and the ordinate gives the probabilities of obtaining a solution whose quality is better than or equal to τ. For example, in Fig. 10 the MIP solver line crosses the abscissa value of 1.02 above 70 %, indicating that this tool has a probability equal to 70 % of obtaining a solution with maximum deviation of 2 % in comparison with the best solution of the approaches analyzed.

Figure 10 shows that CPLEX (MIP) has a greater number of superior solutions in class A+ (performance values with $\tau = 1$), but its performance curve is quickly dominated by those of the proposed method variants. Such result shows that the BFO provides high-quality solutions more often. In addition, the probability that the BFO will obtain a solution whose deviation is lower than 2 % from the best

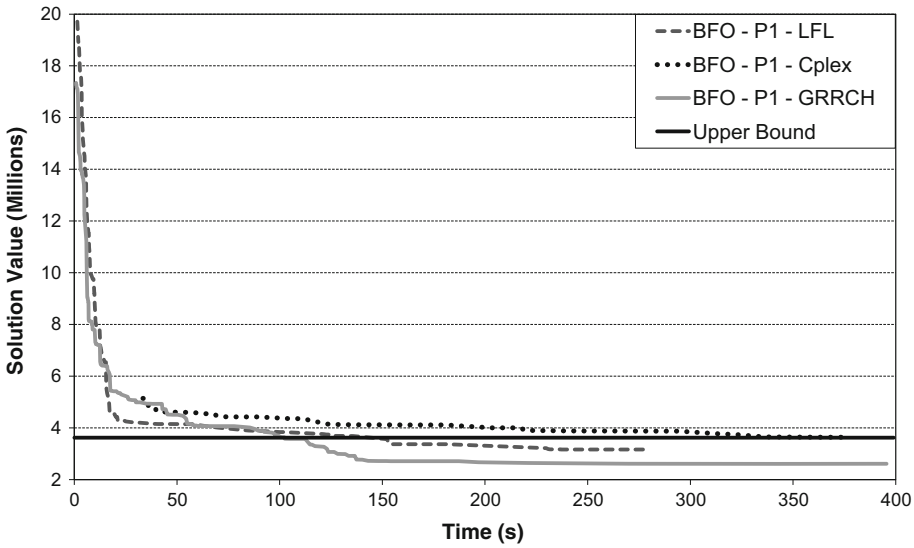

Fig. 6 Convergence curves for the BFO-P1 heuristic using the LFL, CPLEX and GRRCH initial solutions for the complete first run

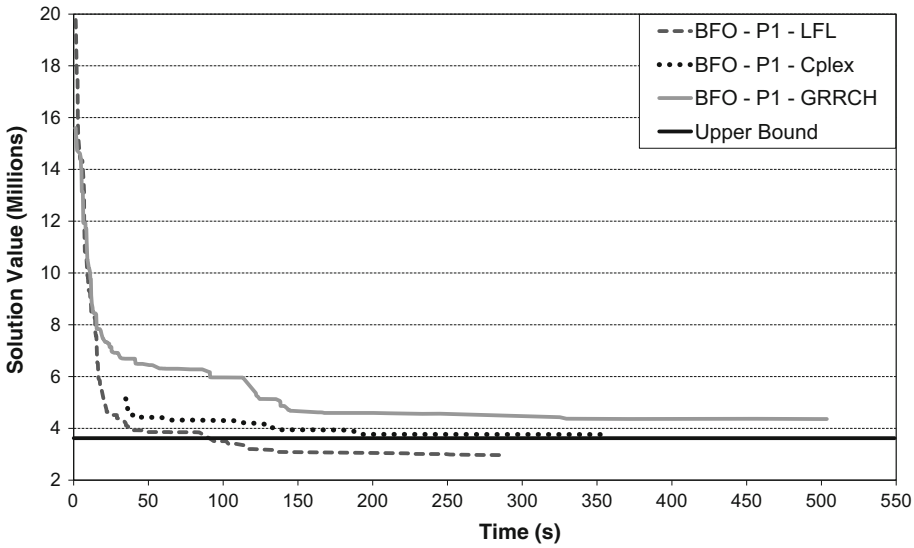

Fig. 7 Convergence curves for the BFO-P1 heuristic using LFL, CPLEX and GRRCH initial solutions for the complete second run

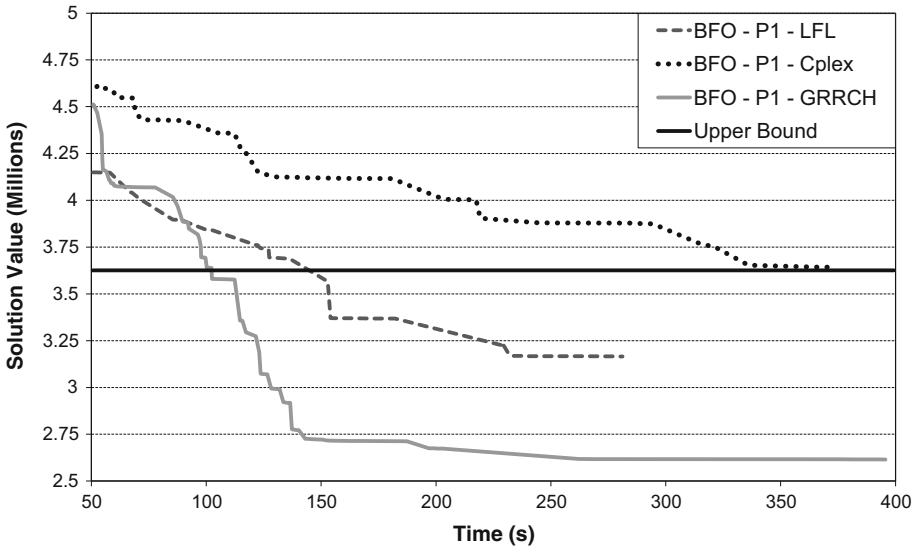

Fig. 8 Convergence curves for the BFO-P1 heuristic using the LFL, CPLEX and GRRCH initial solutions for the first run after 50s

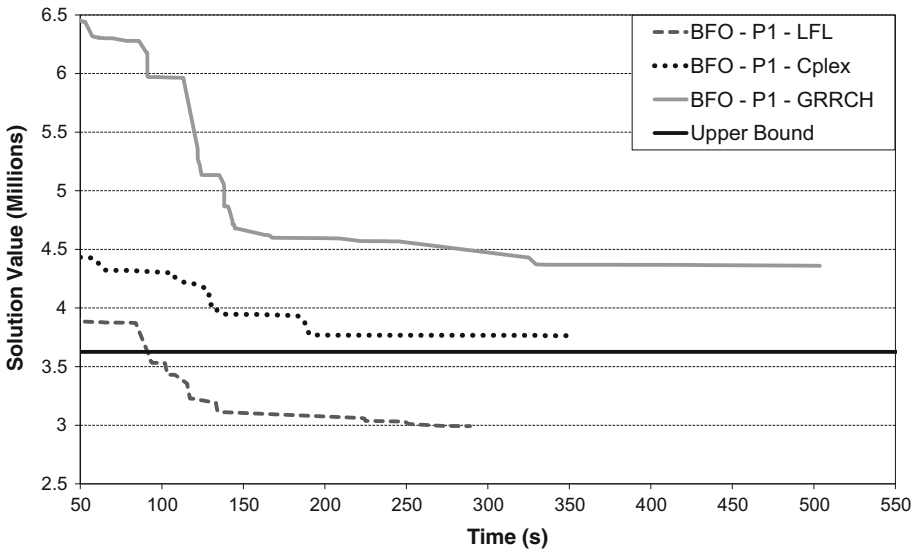

Fig. 9 Convergence curves for the BFO-P1 heuristic using the LFL, CPLEX and GRRCH initial solutions for the second run after 50s

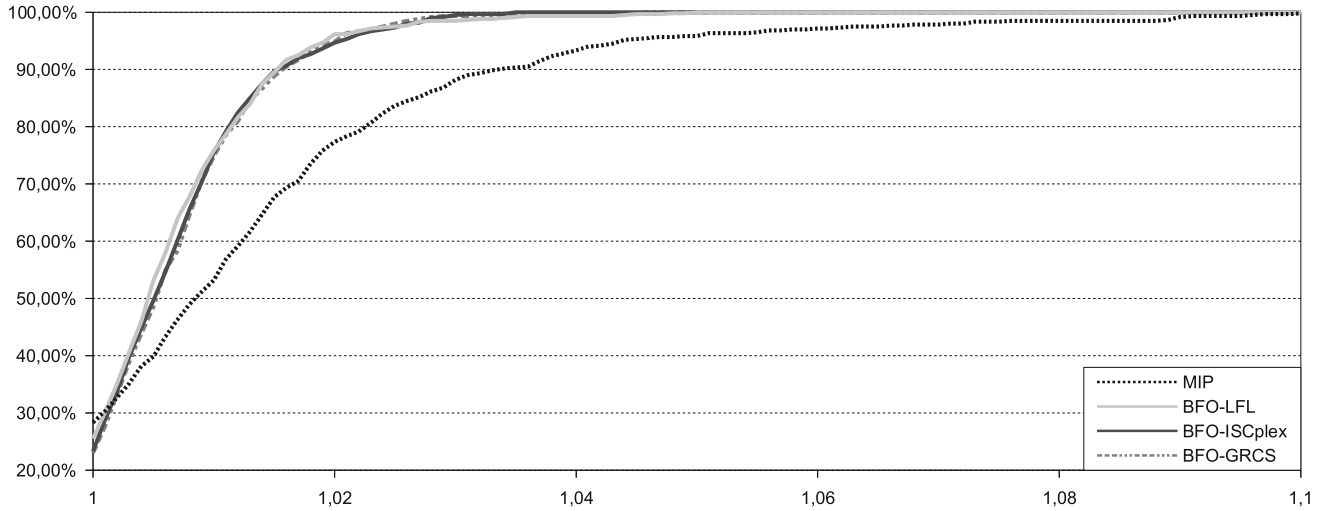

Fig. 10 Evaluation of the performance of BFO variants and CPLEX for test set A+

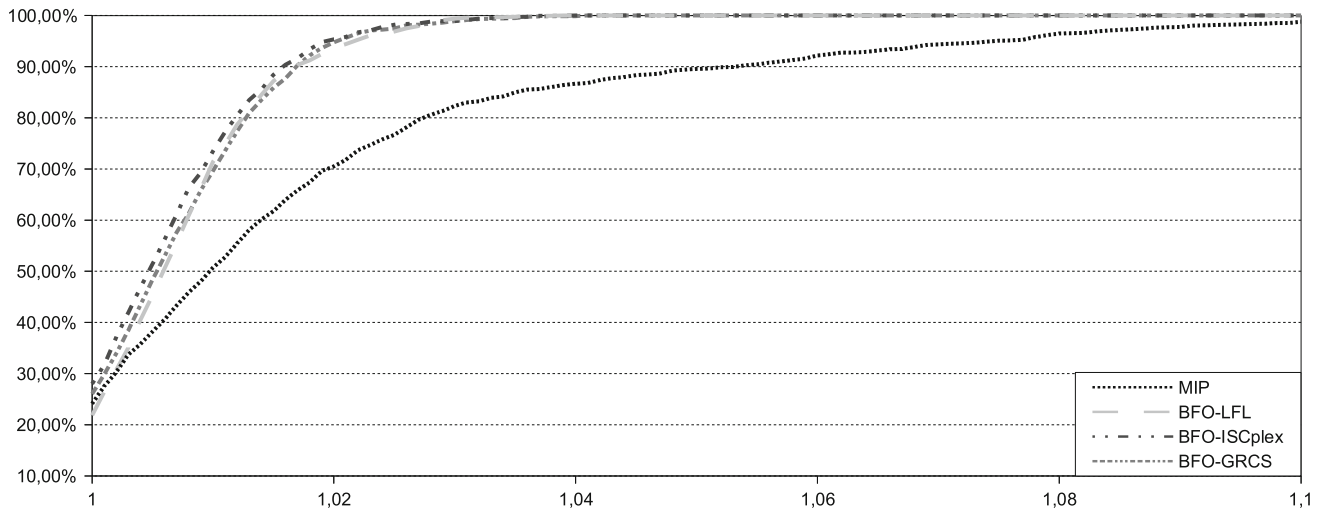

Fig. 11 Evaluation of the performance of BFO variants and CPLEX to test set B+

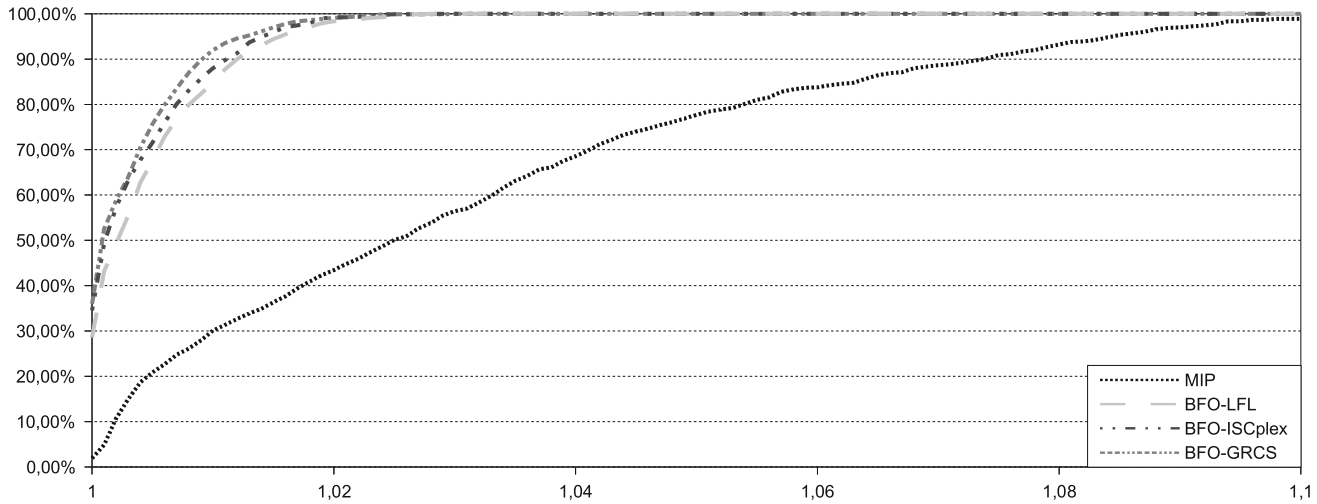

Fig. 12 Evaluation of the performance of BFO variants and CPLEX for test set C

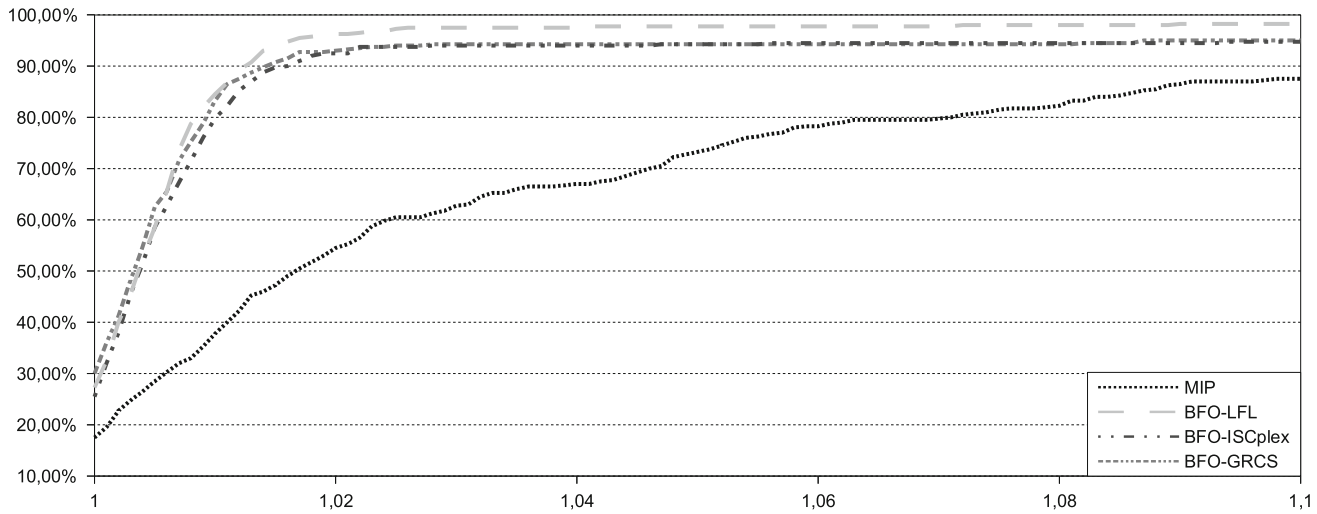

Fig. 13 Evaluation of the performance of BFO variants and CPLEX for test set D

solution yielded by all the approaches is higher than 90 %, demonstrating the method robustness. The similarity of the performance curves among all BFO variants shows that it is not strongly depend on the quality of the initial solution.

Figure 11 shows the results of test set B+. The performance of BFO is even better in comparison with CPLEX because the quantities of the best solutions are relatively similar and the curves of the three variants are always above those of the MIP solver. The difference between test sets A+ and B+ is the inclusion of positive setup times in the latter. This characteristic seems to reduce the amount of best solutions obtained by the MIP solver, but is does not affect significantly the performance of the proposed method. Furthermore, the MIP solver shows a lower performance curve for test set B+ than for A+.

Test set C shows the largest performance gap between CPLEX and the BFO variants (Fig. 12). The chart shows that all solutions produced by the BFO are at most 3 % worse than the best solution found, i.e. the point at which the curve reaches 100 % of probability has $\tau \leq 1.03$.

The results of the chart in Fig. 13 are similar to those of the previous chart. Here, in more than 90 % of the cases the gap in the BFO solution is lower than or equal to 2 %. The performance of the MIP solver is better in comparison with its performance for test set C, however the BFO remains superior.

The performance curves have demonstrated the BFO robustness, given the reduced percentage of infeasible solutions and the high-quality of the them. Furthermore, the BFO proper functioning does not appear to be strongly dependent

on the initial solution quality, as observed by the similarities between the performance of the three variants.

Additional computational tests

For the additional tests (test sets C+, D+ and E+), the gap was calculated as the relative difference between BFO and CPLEX results: $Gap = 100 * \frac{heuristic\ solution - CPLEX\ solution}{CPLEX\ solution}$. In this case, values lower than zero show that the BFO variant considered provide better results than CPLEX. The variants with partition 3 associated with all initial solution options (LFL, CPLEX and GRRCH) were tested. For test set C+, the best variant (BFO-ISCPLEX) produced solutions with an average gap equal to −20.66 % and the worst variant (BFO-LFL) generated solutions with a gap equal to −20.26 %, on average. The results are quite similar for test set D+, in which BFO-ISCPLEX delivered solutions with average gap equal to −21.47 % and BFO-LFL created production plans with average gap of −20.97 %. For the largest test class (E+), CPLEX solved only one instance, whereas BFO solved all. For the instance solved by CPLEX, the average gap found by BFO ranges between −28.98 % (BFO-ISCPLEX) and −23.53 % (BFO-LFL).

Conclusions

This paper has proposed a hybrid heuristic (BFO) to solve the multi-level capacitated lot-sizing problem. This heuristic is a combination of the bees algorithm and the fix-and-optimize heuristic. The key idea is to use the bees algorithm features to overcome the local minima produced by the decomposition scheme and maintain the fix-and-optimize heuristic's ability to quickly return high-quality solutions. The BFO method has outperformed the standard CPLEX 12.2 solver and other heuristics reported in the literature in terms of quality.

This paper has also reported on the development of a heuristic that improves the fix-and-optimize solutions, demonstrating it is possible to use this approach as an efficient sub-problem solver. The computational results show that an initial good-quality solution is not necessary for the method and, in some cases, the results could be worse even starting with better initial solutions.

For test sets C+, D+ and E+, the BFO outperformed CPLEX with average gaps higher than 20 %. The proposed method provided better results for all instances even considering the worst run. The percentage of feasibility is also higher in test set E+, where CPLEX solved only one instance and the BFO solved all. These results show the BFO higher performance and stability for medium and large instances.

Future research could involve the use of other types of decompositions, the extension to multi-objective problems (Gen and Lin 2014; Alvarado-Iniesta et al. 2014) and consider less expensive methods to control the partition use. Considering other ways to solve the sub-problems is also important because of the high number of sub-problems solved by this type of heuristic. The use of methods more efficient than a standard LP/MIP solver may reduce the solution time and yield higher-quality solutions enabling the use of larger populations. Other metaheuristics, such as GRASP and Tabu Search could be used combined with the fix-and-optimize method to construct hybrid heuristics. The proposed method can be easily modified or extended to other problems by changing the decomposition schemes and setting the parameters according to the problem addressed.

Acknowledgements This work was supported by the Conselho Nacional de Desenvolvimento Científico (CNPq) and by Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP).

References

Ajorlou, S., & Shams, I. (2013). Artificial bee colony algorithm for CONWIP production control system in a multi-product multi-machine manufacturing environment. *Journal of Intelligent Manufacturing*, *24*(6), 1145–1156.

Akartunalı, K., & Miller, A. J. (2009). A heuristic approach for big bucket multi-level production planning problems. *European Journal of Operational Research*, *193*(2), 396–411.

Almeder, C. (2010). A hybrid optimization approach for multi-level capacitated lot-sizing problems. *European Journal of Operational Research*, *200*(2), 599–606.

Alvarado-Iniesta, A., García-Alcaraz, J. L., Piña-Monarrez, M., & Pérez-Domínguez, L. (2014). Multiobjective optimization of torch brazing process by a hybrid of fuzzy logic and multiobjective artificial bee colony algorithm. *Journal of Intelligent Manufacturing*. doi:10.1007/s10845-014-0899-2.

Billington, P. J., McClain, J. O., & Thomas, L. J. (1983). Mathematical programming approaches to capacity-constrained MRP systems: Review formulation and problem reduction. *Management Science*, *29*(10), 1126–1141.

Bitran, G. R., & Yanasse, H. H. (1982). Computational complexity of the capacitated lot size problem. *Management Science*, *28*(10), 1174–1186.

Blum, C., Puchinger, J., Raidl, G. R., & Roli, A. (2011). Hybrid metaheuristics in combinatorial optimization: A survey. *Applied Soft Computing*, *11*(6), 4135–4151.

Brajevic, I., & Tuba, M. (2013). An upgraded artificial bee colony (ABC) algorithm for constrained optimization problems. *Journal of Intelligent Manufacturing*, *24*(4), 729–740.

Castellani, M., Pham, Q. T., & Pham, D. T. (2012). Dynamic optimisation by a modified bees algorithm. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, *226*(7), 956–971.

Dereli, T., & Das, G. S. (2011). A hybrid 'bee(s) algorithm' for solving container loading problems. *Applied Soft Computing*, *11*(2), 2854–2862.

Dolan, E. D., & Moré, J. J. (2002). Benchmarking optimization software with performance profiles. *Mathematical Programming*, *91*, 201–213.

Drexl, A., & Kimms, A. (1997). Lot sizing and scheduling—Survey and extensions. *European Journal of Operational Research*, *99*(2), 221–235.

Gen, M., & Lin, L. (2014). Multiobjective evolutionary algorithm for manufacturing scheduling problems: State-of-the-art survey. *Journal of Intelligent Manufacturing*, *25*(5), 849–866.

Goren, H. G., Tunali, S., & Jans, R. (2010). A review of applications of genetic algorithms in lot sizing. *Journal of Intelligent Manufacturing*, *21*(4), 575–590.

Helber, S., & Sahling, F. (2010). A fix-and-optimize approach for the multi-level capacitated lot sizing problem. *International Journal of Production Economics*, *123*(2), 247–256.

Jain, V., Kundu, A., Chan, F. T. S., & Patel, M. (2013). A Chaotic Bee Colony approach for supplier selection-order allocation with different discounting policies in a coopetitive multi-echelon supply chain. *Journal of Intelligent Manufacturing*. doi:10.1007/s10845-013-0845-8.

James, R. J. W., & Almada-Lobo, B. (2011). Single and parallel machine capacitated lotsizing and scheduling: New iterative MIP-based neighborhood search heuristics. *Computers & Operations Research*, *38*(12), 1816–1825.

Jans, R., & Degraeve, Z. (2007). Meta-heuristics for dynamic lot sizing: A review and comparison of solution approaches. *European Journal of Operational Research*, *177*(3), 1855–1875.

Li, Y., Tao, Yi, & Wang, F. (2012). An effective approach to multi-item capacitated dynamic lot-sizing problems. *International Journal of Production Research*, *50*(19), 5348–5362.

Maes, J., McClain, J. O., & Wassenhove, L. N. V. (1991). Multilevel capacitated lotsizing complexity and LP-based heuristics. *European Journal of Operational Research*, *53*(2), 131–148.

Özbakir, L., Baykasoglu, A., & Tapkan, P. I. (2010). Bees algorithm for generalized assignment problem. *Applied Mathematics and Computation*, *215*(11), 3782–3795.

Pham, D. T., Ghanbarzadeh, A., Koc, E., Otri, S., Rahim, S., & Zaidi, M. (2005). *The Bees Algorithm*. Technical report, Cardiff University, UK.

Pham, D. T., Koc, E., Lee, J. Y., & Phrueksanant, J. (2007). Using the Bees Algorithm to schedule jobs for a machine. *In Proceedings of Eighth International Conference on Laser Metrology* (pp. 430–439). Uk: Euspen.

Pochet, Y., & Wolsey, L. A. (2006). *Production planning by mixed integer programming*. New York: Springer.

Puchinger, J., & Raidl, G. R. (2005). Combining metaheuristics and exact algorithms in combinatorial optimization: A survey and classification. *In artificial intelligence and knowledge engineering applications: A bioinspired approach* (vol. 3562, pp. 113–124). Berlin: Springer.

Sahling, F., Buschkühl, L., Tempelmeier, H., & Helber, S. (2009). Solving a multi-level capacitated lot sizing problem with multi-period setup carry-over via a fix-and-optimize heuristic. *Compututers & Operational Research*, *36*(9), 2546–2553.

Stadtler, H. (2003). Multilevel lot sizing with setup times and multiple constrained resources: Internally rolling schedules with lot-sizing windows. *Operations Research*, *51*(3), 487–502.

Stadtler, H., & Sürie, C. (2000). *Description of MLCLSP test instances*. Technicalreport, Darmstadt: Technische Universität Darmstadt.

Tempelmeier, H., & Derstroff, M. (1996). A Lagrangean-based heuristic for dynamic multi-level multi-item constrained lotsizing with setup times. *Management Science*, *42*(5), 738–757.

Toledo, C. F. M., de Oliveira, R. R. R., & França, P. M. (2013). A hybrid multi-population genetic algorithm applied to solve the multi-level capacitated lot sizing problem with backlogging. *Computers & Operations Research*, *40*(4), 910–919.

Wen, X. Y., Li, X. Y., Gao, L., & Sang, H. Y. (2014). Honey bees mating optimization algorithm for process planning problem. *Journal of Intelligent Manufacturing*, *25*(3), 459–472.

Wu, T., Akartunalı, K., Song, J., & Shi, L. (2013). Mixed integer programming in production planning with backlogging and setup carryover: Modeling and algorithms. *Discrete Event Dynamic Systems*, *23*(2), 211–239.

Wu, T., Shi, L., & Song, J. (2012). An MIP-based interval heuristic for the capacitated multi-level lot-sizing problem with setup times. *Annals of Operations Research*, *196*, 635–650.