A Hybrid VNS/LP Approach to Solve Production Scheduling of Fruit Juice Beverages

Dražen Popović, Nenad Bjelić, and Milorad Vidović
Department of Logistics
Faculty of Transport and Traffic Engineering, University of Belgrade
d.popovic@sf.bg.ac.rs, n.bjelic@sf.bg.ac.rs, m.vidovic@sf.bg.ac.rs

Abstract
A production scheduling problem of fruit juice beverages (also known as two-level soft drink production problem) consists of two integrated segments: scheduling and lot sizing. The solution to this problem is the production plan that defines the quantity of products that will be produced on usually multiple production lines in a given planning horizon. To reduce the possibility of stock-outs, safety stocks are required (products consumption is stochastic in reality). Also, products shelf life can be very important for retailers since they must sell these products to final customers. Therefore, production plan must simultaneously solve scheduling and lot sizing problem so that the stock level of each product does not fall under safety stock level or go over maximum desirable stock regarding shelf life. Additionally, total available warehouse capacity is limited and must be respected by the production plan (excessive stock can be outsourced which incurs additional stock keeping cost). In this paper, we propose a hybrid variable neighbourhood search (VNS) and linear programming (LP) model to solve scheduling and lot sizing problem with several inventories related objectives: minimizing the violation of planned minimum and maximum stock level per each product, minimizing the overflow of warehouse capacity, and finally minimizing total stock and backorder costs in the system. The proposed hybrid approach uses the LP model to solve the lot sizing segment and the VNS heuristics to solve the sequencing segment.

Keywords: production scheduling, variable neighbourhood search, linear programming, inventory management, beverages

JEL classifications: C61

1. Introduction

Production scheduling problem in the beverage industry which simultaneously tries to solve lot sizing and scheduling has recently gained significant attention in academic research. Development of technology and research methodologies has induced many researchers to develop models for solving this complex problem in which manufacturing companies with production lines setup must constantly make production plans. These production plans must allocate a number of products to different production lines in observed planning horizon considering a wide range of constraints such as time available for production, line setup and changeover times, demand for products, available inventories etc.

Clark et al. 2011 defined production planning and scheduling as a problem in which decision maker must efficiently allocate production resources while fulfilling customer requirements and market demand, often by trading-off conflicting objectives. The main objective is to determine the moment of production as well as the production duration (which determines the produced quantity) on multiple production lines. A number of research papers were published on the topic of efficient production planning in the beverage manufacturing industry. Ferreira et al. 2010 developed MILP model for solving integrated lot sizing and scheduling decisions in production planning of small-scale soft drink plant, that was primarily focused on bottling phase (which is often a bottleneck in production). A similar problem was observed by Ferreira et al. 2012 but with a somewhat different approach. They proposed four single-stage formulations to solve the synchronised two-stage lot sizing and scheduling problem, where the first stage's syrup lots in tanks is synchronized with the second stage's soft drink lots on bottling lines. Another research paper on the topic of lot sizing and scheduling was written by Baldo et al. 2014 in the brewery industry, where authors presented MIP-based heuristic which produces relatively good-quality solutions for real-world problem instances. Most recent research paper on the topic of production scheduling was published by Pagliarussi et al. 2017 where authors developed MILP model for solving lot sizing and scheduling of fruit juice beverages production, in multi-period planning horizon and with multiple production lines for relatively small-scale instances (CPLEX solver was not capable of providing good solutions within acceptable computer times for most test instances). Toledo et al. 2014 developed a hybrid approach to solve a lot sizing and scheduling problem in soft drink production motivated by real-world case. Their hybrid model is based on a genetic algorithm used for scheduling segment and mathematical programming used for lot sizing (sequencing) segment. The computational results showed that the hybrid approach outperforms other models available in the literature in terms of objective function value and run times when applied to real-world problem instances.

This paper applies the similar idea of hybrid approach from Toledo et al. 2014 on the production scheduling problem in fruit juice beverages industry with the increased focus on inventory management of finished products. This inventory management is becoming more important due to dynamic markets where shelf life and stock-outs are one of the main drivers for decision makers. Additionally, warehouse capacity for finished products can be scarce (as it was the case for one of the biggest fruit juice production companies located in Serbia) and therefore can have a great influence on decision making. Popović et al. 2018 presented mathematical programming model for inventory management problem (that envelopes shelf life and stock outs as well as warehouse capacity restrictions) in production scheduling of fruit juice beverages to solve only small-scale instances. In this paper, we extend the idea of Popović et al. 2018 by developing the hybrid VNS/LP model to solving real-world large-scale instances. Observed problem is described in detail in Section 2. In Section 3 we present the hybrid VNS/LP approach for production scheduling, while test instance and computational results are given in Section 4. Concluding remarks are presented in Section 5.

2. Problem Description

To test proposed hybrid VNS/LP approach we have used production problem setup from Toledo et al. 2014 and extend it with additional inventory management constraints: each product has planned minimum and maximum stock level, and warehouse has limited capacity. In the following, we will briefly describe the production problem setup. Production lines have raw material tanks from which beverages are bottled on the automated filling lines. The raw material in the tank determines the flavor of beverage, while different bottle types on one filling line can be used for bottling. Therefore, a product is defined by flavor and bottle type. Only one tank is assigned to each filling line. Setup time for preparation of raw material in the tank occurs between two batches (even in the case of the same flavor), also known as flavor changeover. Package changeover is setup time for changing the bottle type at a filling line. The production lines have different processing time for each product. Both changeover times are dependent on the sequence of production (the previous and the following flavor/bottle type). Production capacities of tanks and filling lines cannot be violated.

The production plan must define the quantity of products that will be produced on each production line in a given planning horizon. For each product start time of production and the production duration must be determined. Production should satisfy products' expected demands and this is the main driver of production. In the case of product shortage, backorder is required which incurs additional costs. A products demand is stochastic value in reality and therefore safety stocks are required to reduce the possibility of shortages. On the other hand, products shelf life can be very important for retailers since they must sell these products to final customers. In that regard, high stock levels can reduce the shelf life of products which can lead to obsolete products (these products cannot be delivered to retailers and therefore must be destroyed which incurs additional costs). Therefore, planned minimum and maximum stock level, that are directly dependent on demand (d_{jt}), per each product should not be violated (Figure 1). For example, if a product's shelf life is 200 days with expected daily consumption of 2 pallets and retailer wants a minimum of 80% product's shelf life on delivery, then $f_{max}(d_{jt})$ is equal to 40 days of 2 pallets consumption (80 pallets of maximum desirable stock level of product j). The same logic applies to safety stock determination.

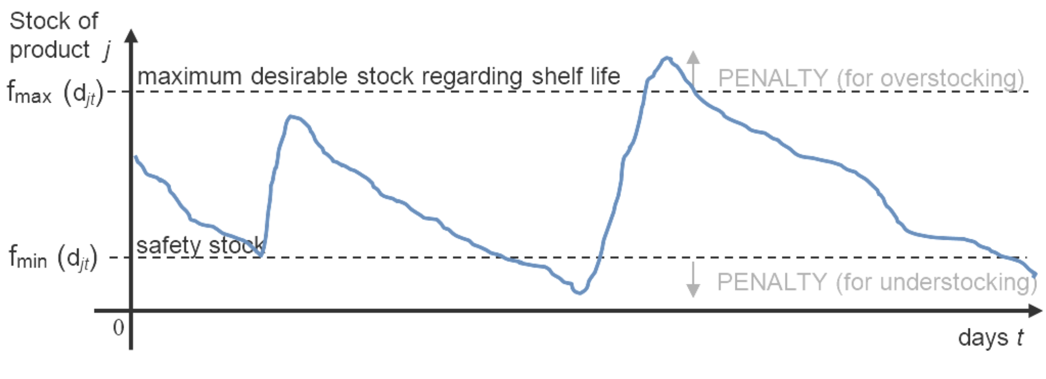


Figure 1: Maximum and minimum desirable stock level of a product, based on safety stock and required shelf life (Popović et al., 2018)

Considering available stock of products at the start of the planning horizon, as well as expected demands, production plan must solve lot sizing and scheduling problem so that the stock level of each product incurs minimal violation of safety stock level and desirable stock regarding shelf life. Additionally, the production facility has limited warehouse capacity for product storage. In the case of warehouse capacity overflow, excessive quantities can be stored at an outsourced warehouse which incurs additional costs (more expensive than the storage in the production facility warehouse).

3. The Hybrid VNS/LP Approach for Production Scheduling

Mixed integer programming (MIP) model for the integrated scheduling and lot sizing in beverage production cannot solve real-world instances in reasonable computational time because of the complexity of the problem. For instance, Toledo et al. 2007 developed MIP model for the similar problem where CPLEX could solve only small-to-moderate scale size instances. The MIP model consists of binary variables (for the scheduling segment) as well as continuous variables (for the lot sizing segment), and combined with real-world large scale dimensions it has unacceptable computational time for obtaining the solution. Therefore, Toledo et al. 2014 divided this integrated approach and formulated LP model for lot sizing segment which includes only continuous variables, and the genetic algorithm (GA) heuristic for sequencing segment. This LP model can obtain the solution to lot sizing for real-world instances practically immediately (CPU time is less than 1 second). To be able to solve lot sizing segment, LP model requires input data from scheduling segment. More precisely, for each period of the planning horizon, the sequence of products to be produced on each line is required. The sequence is obtained by the heuristic approach and LP model is used to solve lot sizing segment of the observed production problem. Illustration of cooperation between heuristic and LP model is given in Figure 2 (2 production lines and 3 days) where heuristic model must define the scheduling of production for each line per days of planning horizon (what products and in which order should be produced). This schedule is imported to LP model as a parameter for obtaining the lot sizing of production runs for products (the width of each field in the final solution illustrates the length of production runs).

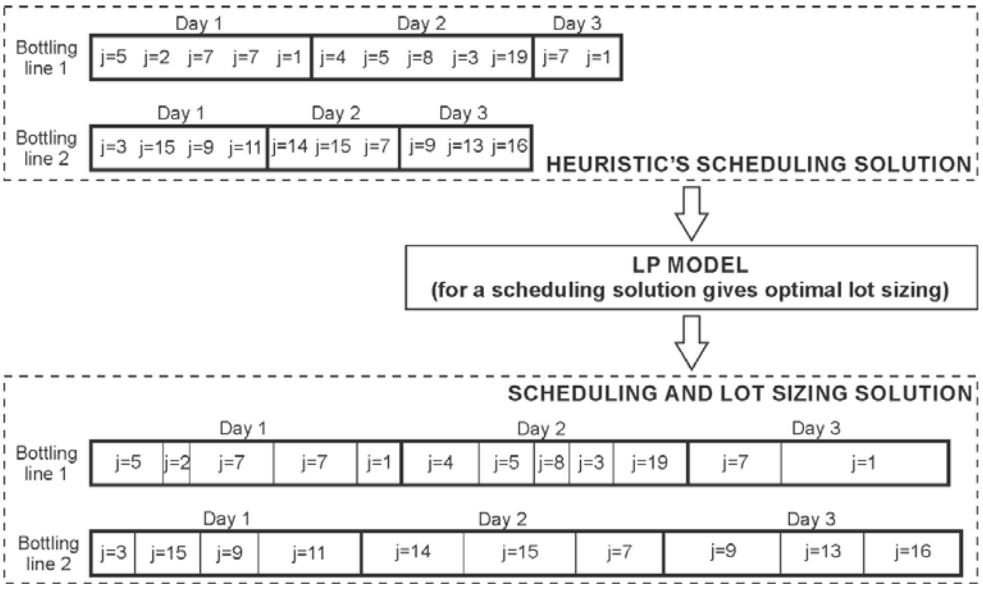


Figure 2: Illustration of heuristic and LP model tasks (j is product type)

The overview of hybrid VNS/LP approach for the integrated scheduling and lot sizing in beverage production is presented in Figure 3. The detailed description of the LP model and hybrid VNS approach is given in the rest of this section.

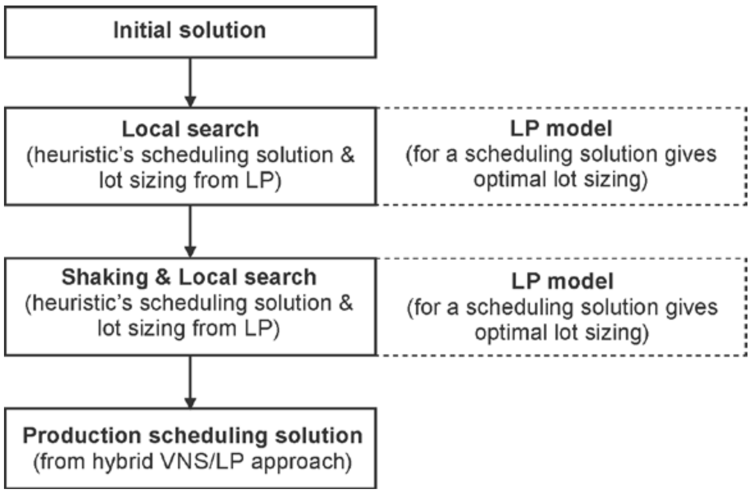

Initial solution
Local search
(heuristic's scheduling solution & lot sizing from LP)
LP model
(for a scheduling solution gives optimal lot sizing)
Shaking & Local search
(heuristic's scheduling solution & lot sizing from LP)
LP model
(for a scheduling solution gives optimal lot sizing)
Production scheduling solution
(from hybrid VNS/LP approach)

Figure 3: The overview of hybrid VNS/LP approach

3.1 LP Model for the Lot Sizing

LP model formulation for the lot sizing segment is taken from Toledo et al. 2014 and it is extended by additional inventory related constraints. In the following we present notations, objective function and constraints of the proposed LP model for lot sizing segment.

Indices:

j	- number of products (j =1, …, J)
t	- number of days (t = 1, …, T)
m	- number of production lines (m = 1, …, M)
l	- raw material used for production

Input parameters defined by the heuristic scheduling solution:

ns_{mjt}	- number of production runs (lots) of product j on line m in day t
w_{mt}	- total production setup time for line m in day t (lost time)

Parameters:

h_j	- daily inventory cost of product j (per unit)
gj	- daily backorder cost of product j (per unit)
Aj	- daily cost of safety stock violation for product j (per unit)
B_j	- daily cost of shelf-life stock violation for product j (per unit)
C	- daily cost of total warehouse capacity overflow (per unit)
d_{jt}	- demand of product j in day t
r_{jl}	- quantity of raw material l required to produce one unit of product j
a^{II}_{mj}	- production time of one unit of product j in production line m
K^{I}_{m}	- total capacity of tank m (designated to production line m) for storing raw materials

K^{II}_{mt} - total time capacity available for production on line *m* in day *t*

q^{I}_{lm} - minimum quantity of raw material *l* necessary to fill tank at production line *m*

α_m - set of products that can be produced on line *m*

β_m - set of raw materials that can be used for production on line *m* (that can be stored in designated tank), which directly depends on the products that can be produced on line *m*.

SS_j - minimum desirable safety stock level for product *j*

SL_j - maximum desirable stock based on shelf-life for product *j*

WH - total warehouse capacity

Decision variables:

u^{II}_{mjt} - main decision variable which represents lot size of product *j* in line *m* in day *t*. Other decision variables are derived from u^{II}_{mjt} and used in objective function to determine values of five inventory relates sub-objectives.

I^{+}_{jt} - total stock cost in the system for product *j* in day *t*

I^{-}_{jt} - backorder cost in the system for product *j* in day *t*

V^{SS}_{jt} - violation of planned minimum stock level (safety stock) for product *j* in day *t*

V^{SL}_{jt} - violation of planned maximum stock level (shelf-life) for product *j* in day *t*

V^{WH}_{t} - violation of warehouse capacity in day *t*

Objective function (1) has five inventories related sub-functions with minimization of: total stock and backorder costs in the system which are same as in Toledo et al. 2014, while we extended the objective function with cost of violation of planned minimum and maximum stock level (more violation means greater risk of backorder or lost sales due to expiration of expected shelf life, where both cases incurs additional costs), and cost of warehouse capacity overflow (additional cost incurred by storing the products at 3PL warehouse).

Objective function:

$$\min \rightarrow \sum_{j=1}^{J}\sum_{t=1}^{T}\left(h_j \cdot I^{+}_{jt} + g_j \cdot I^{-}_{jt}\right) + \sum_{j=1}^{J}\sum_{t=1}^{T}\left(A_j \cdot V^{SS}_{jt} + B_j \cdot V^{SL}_{jt}\right) + \sum_{t=1}^{T} C \cdot V^{WH}_{t} \tag{1}$$

Subject to:

$$I^{+}_{j(t-1)} + \sum_{m\in\lambda_j} u^{II}_{mjt} + I^{-}_{jt} = I^{+}_{jt} + I^{-}_{j(t-1)} + d_{jt} \quad j=1,...,J, t=1,...,T \tag{2}$$

$$\sum_{j\in\alpha_m} a^{II}_{mj} \cdot u^{II}_{mjt} + w_{mt} \le K^{II}_{mt} \quad m=1,...,M, t=1,...,T \tag{3}$$

$$r_{jl} \cdot u^{II}_{mjt} \le K^{I}_{m} \cdot ns_{mjt} \quad m=1,...,M, l\in\beta_m, j=1,...,J, t=1,...,T \tag{4}$$

$$r_{jl} \cdot u^{II}_{mjt} \ge q^{I}_{lm} \cdot ns_{mjt} \quad m=1,...,M, l\in\beta_m, j=1,...,J, t=1,...,T \tag{5}$$

$$V^{SS}_{jt} \ge I^{-}_{jt} - I^{+}_{jt} + SS_j \quad j=1,...,J, t=1,...,T \tag{6}$$

$$V^{SL}_{jt} \ge I^{+}_{jt} - SL_j \quad j=1,...,J, t=1,...,T \tag{7}$$

$$V_t^{WH} \geq \sum_{j=1}^{J} I_{jt}^{+} - WH \quad t = 1,...,T \tag{8}$$

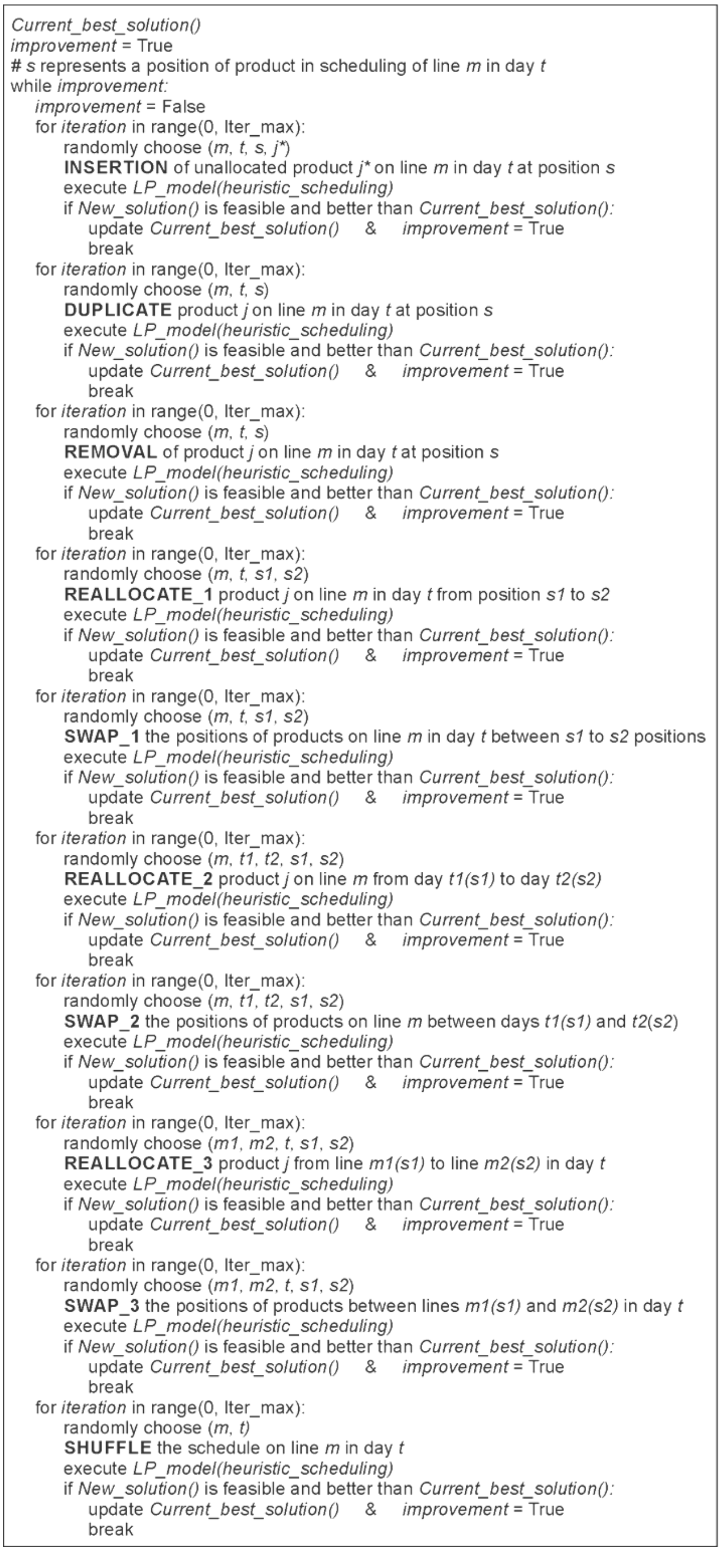
$$I_{jt}^{+}, I_{jt}^{-}, u_{mjt}^{II}, V_{jt}^{SS}, V_{jt}^{SL}, V_t^{WH} \geq 0 \quad m = 1,...,M, j = 1,...,J, t = 1,...,T \tag{9}$$

Constraints (2-5) are taken from Toledo et. al (2014) lot sizing formulation, while we extended this formulation by constraints (6-8). Constraints (2) defines the lot sizing of production considering product demand, stock and understock in the system. Usage of available total production time per each line for production and setup is defined by constraints (3). Constraints (4) assures that one lot size cannot exceed available tank capacity on the production line. This means that production run of quantities larger than tank capacity must be realized in two separate runs (one example is given in Figure 2, for line 1 in day 1 where product 7 has two production runs). Additionally, constraints (5) defines the minimal production lot size considering the minimum quantity of raw material that must be filled in the tank. Amount of violation of planned safety stock and shelf life stock is respectively defined by constraints (6) and (7). The overflow of warehouse capacity is defined by constraints (8). Constraints (9) defines the nature of the variables, where all variables can take only non-negative continuous values.

3.2 The VNS Model

The concept of VNS metaheuristic was developed by Mladenovic and Hansen 1997 which is based on a systematic change of a neighborhood within a search algorithm. VNS has many variants that are usually based on three general steps: construction of the initial solution, the local search, and the shaking procedure. Both local search and shaking have neighborhoods in which different search procedures are applied.

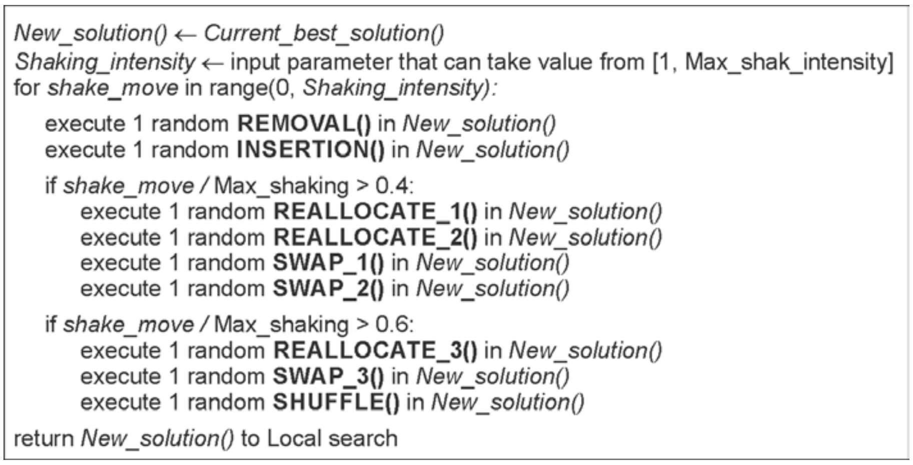
Due to the complexity of production scheduling problem and large search space we developed a variant of Reduced VNS (RVNS) with the first improvement approach. By Hansen et al. 2017 the RVNS is the simplest VNS approach where classical local search improvement is discarded and where neighborhoods are searched by shaking moves (in a stochastic manner). In our RVNS model, we have included the local search procedure but with randomized moves (instead of searching all possible moves within each neighbourhood) as shown in Figure 4, and therefore we kept the reduced approach. By parameter *Iter_max* we define the number of possible local search moves within each neighbourhood (the intensity of local search) before continuing to the next neighbourhood. In total, we applied 10 neighbourhoods in the reduced local search procedure which are presented and explained in Figure 4. The first neighbourhood of local search is based on insertion heuristic which can be used to construct the initial solution. Therefore, in our approach the initial solution is an empty set (no production at all) where local search (mainly reduced insertion) adds and moves products on a production schedule. Our intention was to obtain a different initial solution of good quality for different solution runs of RVNS.



```
Current_best_solution()
improvement = True
# s represents a position of product in scheduling of line m in day t
while improvement:
   improvement = False
   for iteration in range(0, Iter_max):
      randomly choose (m, t, s, j*)
      INSERTION of unallocated product j* on line m in day t at position s
      execute LP_model(heuristic_scheduling)
      if New_solution() is feasible and better than Current_best_solution():
         update Current_best_solution()    &    improvement = True
         break
   for iteration in range(0, Iter_max):
      randomly choose (m, t, s)
      DUPLICATE product j on line m in day t at position s
      execute LP_model(heuristic_scheduling)
      if New_solution() is feasible and better than Current_best_solution():
         update Current_best_solution()    &    improvement = True
         break
   for iteration in range(0, Iter_max):
      randomly choose (m, t, s)
      REMOVAL of product j on line m in day t at position s
      execute LP_model(heuristic_scheduling)
      if New_solution() is feasible and better than Current_best_solution():
         update Current_best_solution()    &    improvement = True
         break
   for iteration in range(0, Iter_max):
      randomly choose (m, t, s1, s2)
      REALLOCATE_1 product j on line m in day t from position s1 to s2
      execute LP_model(heuristic_scheduling)
      if New_solution() is feasible and better than Current_best_solution():
         update Current_best_solution()    &    improvement = True
         break
   for iteration in range(0, Iter_max):
      randomly choose (m, t, s1, s2)
      SWAP_1 the positions of products on line m in day t between s1 to s2 positions
      execute LP_model(heuristic_scheduling)
      if New_solution() is feasible and better than Current_best_solution():
         update Current_best_solution()    &    improvement = True
         break
   for iteration in range(0, Iter_max):
      randomly choose (m, t1, t2, s1, s2)
      REALLOCATE_2 product j on line m from day t1(s1) to day t2(s2)
      execute LP_model(heuristic_scheduling)
      if New_solution() is feasible and better than Current_best_solution():
         update Current_best_solution()    &    improvement = True
         break
   for iteration in range(0, Iter_max):
      randomly choose (m, t1, t2, s1, s2)
      SWAP_2 the positions of products on line m between days t1(s1) and t2(s2)
      execute LP_model(heuristic_scheduling)
      if New_solution() is feasible and better than Current_best_solution():
         update Current_best_solution()    &    improvement = True
         break
   for iteration in range(0, Iter_max):
      randomly choose (m1, m2, t, s1, s2)
      REALLOCATE_3 product j from line m1(s1) to line m2(s2) in day t
      execute LP_model(heuristic_scheduling)
      if New_solution() is feasible and better than Current_best_solution():
         update Current_best_solution()    &    improvement = True
         break
   for iteration in range(0, Iter_max):
      randomly choose (m1, m2, t, s1, s2)
      SWAP_3 the positions of products between lines m1(s1) and m2(s2) in day t
      execute LP_model(heuristic_scheduling)
      if New_solution() is feasible and better than Current_best_solution():
         update Current_best_solution()    &    improvement = True
         break
   for iteration in range(0, Iter_max):
      randomly choose (m, t)
      SHUFFLE the schedule on line m in day t
      execute LP_model(heuristic_scheduling)
      if New_solution() is feasible and better than Current_best_solution():
         update Current_best_solution()    &    improvement = True
         break
```

Figure 4: Pseudo code of local search procedure

The shaking procedure is used to obtain a new starting point for a local search and therefore to allow the good quality global search of solution space (to overcome the local optima "trap"). We used identical neighborhoods as in the local search procedure to randomly shake the best-known solution in such a manner that small changes are made at the beginning and the size of those changes increases toward the end of the shaking procedure. The pseudo code of the shaking procedure is presented in Figure 5.



```
New_solution() ← Current_best_solution()
Shaking_intensity ← input parameter that can take value from [1, Max_shak_intensity]
for shake_move in range(0, Shaking_intensity):
    execute 1 random REMOVAL() in New_solution()
    execute 1 random INSERTION() in New_solution()
    if shake_move / Max_shaking > 0.4:
        execute 1 random REALLOCATE_1() in New_solution()
        execute 1 random REALLOCATE_2() in New_solution()
        execute 1 random SWAP_1() in New_solution()
        execute 1 random SWAP_2() in New_solution()
    if shake_move / Max_shaking > 0.6:
        execute 1 random REALLOCATE_3() in New_solution()
        execute 1 random SWAP_3() in New_solution()
        execute 1 random SHUFFLE() in New_solution()
return New_solution() to Local search
```

Figure 5: Pseudo code of shaking procedure

In the local search and shaking procedures, infeasible solutions can occur due to the randomness of changes where these infeasibilities are related to the maximum production time of a line in a single time period (too many products assigned to a production line in one day). Only feasible moves are accepted in the proposed approach. The algorithm of the proposed hybrid VNS/LP model is presented in Figure 6. Two parameters define the use of shaking procedure: *shak_intensity* define the "destruction" intensity of by which current best solution is moved to some point in the neighbourhood (larger intensity represents a move with further distance from the current best solution); *shak_pass* is a number of shaking repetitions within the same shaking intensity value.

4. Test Instance and Computational Results

In order to test proposed hybrid VNS/LP approach we will use real-world instance I1 from papers Ferreira et al. 2009 and Toledo et al. 2014 in modified form due to the fact that some of the input data required to solve the original problem are not publicly available (or not clearly stated) in these two papers or in any other (to the best of our knowledge). Secondly, we introduced additional inventory related constraints and therefore we defined some new values. We observed 23 products, 18 raw materials, 2 production lines (1 tank for raw material storage is dedicated to each line) and planning horizon of 3 weeks (15 working days in total). Maximum capacity of both tanks for raw materials is set to 24000 units while the minimum quantity of raw material necessary to fill the tank is set to 0.2*24000 units. Maximum warehouse capacity is 400000 units. The safety stock of products is equal to 5 daily demands and maximum desirable stock regarding shelf life is equal to 20 daily demands. Daily product demand is evenly distributed on 5 working days from weekly demand. The rest of the input parameters (h_j, g_j, r_{jl}, a^{II}_{mj}, α_m, β_m, starting stock, flavour and product changeover times and costs, products that can be produced on lines, raw materials used to produce products, production line capacities and products production unit time etc.) are the same as in the Ferreira et al. 2009 and Toledo et al. 2014. Additionally, the daily cost of safety stock violation A_j and the daily cost of shelf-life stock violation B_j are equal to g_j*0.5, while daily cost of total warehouse capacity overflow C is equal to average(h_j)*10.

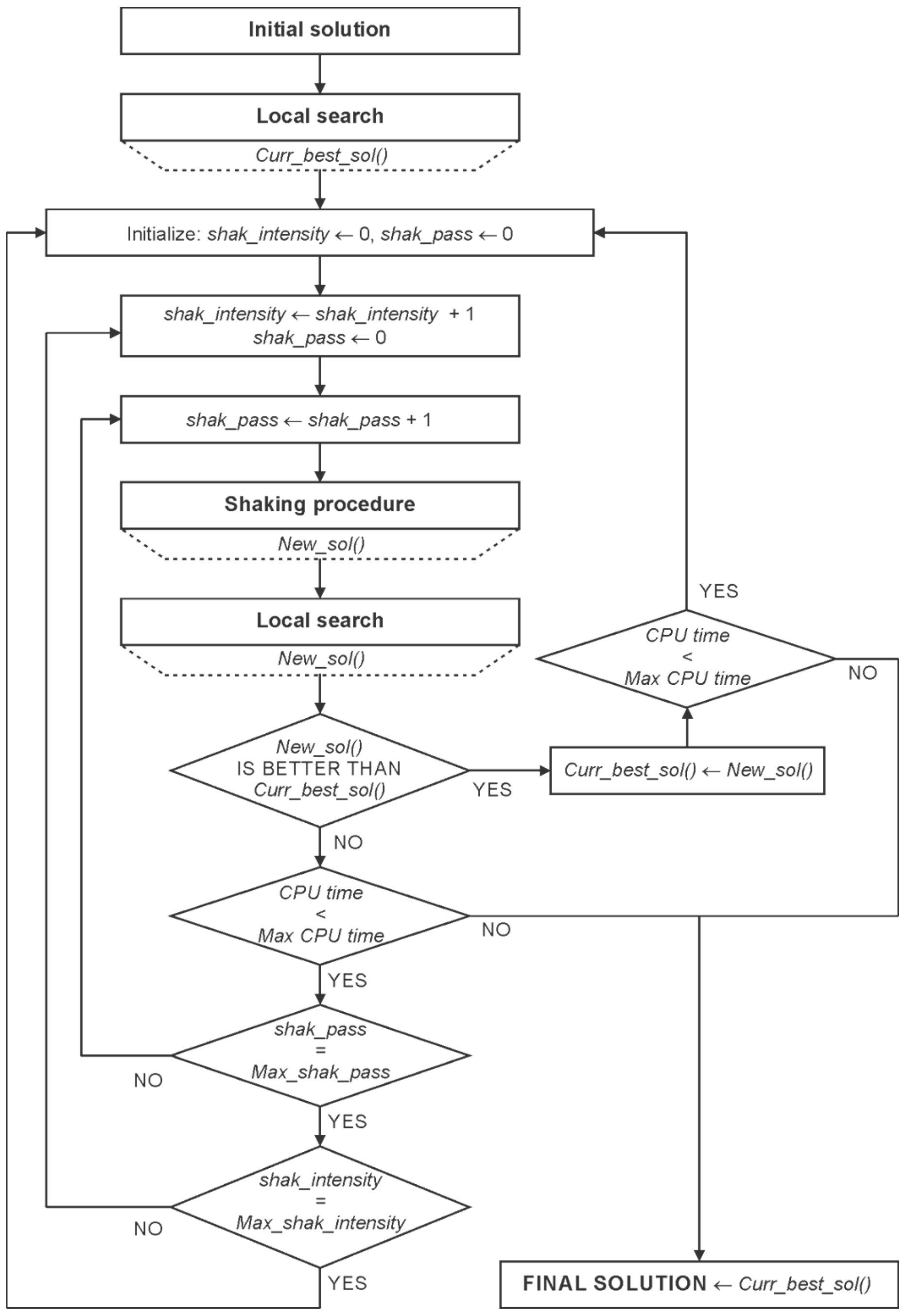

Initial solution
Local search
Curr_best_sol()
Initialize: shak_intensity ← 0, shak_pass ← 0
shak_intensity ← shak_intensity + 1
shak_pass ← 0
shak_pass ← shak_pass + 1
Shaking procedure
New_sol()
Local search
New_sol()
New_sol() IS BETTER THAN Curr_best_sol()
YES
Curr_best_sol() ← New_sol()
CPU time < Max CPU time
YES
NO
NO
CPU time < Max CPU time
NO
YES
shak_pass = Max_shak_pass
NO
YES
shak_intensity = Max_shak_intensity
NO
YES
FINAL SOLUTION ← Curr_best_sol()

Figure 6: Algorithm of hybrid VNS/LP approach

The hybrid VNS/LP model has three tune-up parameters: *Iter_max* – maximum number of iterations for each local search neighborhoods, *Max_shak_intensity*, and *Max_shak_pass*. Max CPU time for obtaining the solution is used as termination criteria and we tested our model for two values, 2 and 4 hours. Because of the stochastic nature of VNS metaheuristic search, we solved modified instance I1 for 100 times in each test run (each test run has unique values of three tune-up parameters and CPU time) to analyse the impact on the quality of solutions. The solution results for max CPU time of 2 h and different values of three tune-up parameters is given in Table 1, and for max CPU time of 4 h in Table 2. In total, we made 1100 test runs for max CPU time of 2 h and 600 test runs for max CPU time of 4 h. The best heuristic solution out of those 1700 solutions is used to evaluate the quality of the model with different test parameters. Total costs are obtained as the sum of the LP model objective function plus changeover costs derived from the heuristic. To evaluate performance of neighborhoods used in the local search procedure we measured total number of solution improvements and average values for different max CPU times are presented in Table 3. The model was implemented by Python 2.7 and API CPLEX 12.6 on PC Intel® Core™ i5-3470 CPU @ 3.20 GHz with 8 GB RAM.

5. Conclusions

We have presented the hybrid VNS/LP approach to solve integrated scheduling and lot sizing in fruit juice production. The problem is complex and cannot be solved to optimality in reasonable computational time and therefore we have developed a tailormade heuristic approach for one specific real-world case where we use LP model to solve lot sizing of a predetermined sequence of production (obtained by VNS heuristic).

The results from Tables 1 and 2 show good convergence of the solutions especially regarding the stochastic nature of initial solution construction. Also, the model returns stable results expressed by small values of standard deviation in 100 repetitions in each test run. As expected, solutions have better convergence and stability for the case of 4 hours maximal CPU time: 2.06 % average error from the best obtained solution and 1.09 % standard deviation in 100 repetitions of the test run with max CPU time of 4 h and *Iter_max*=16, *Max_shak_intensity*=16, *Max_shak_pass*=16.

As for the performance of local search neighborhoods, results from Table 3 shows that the Insertion and Removal are most effective, while Duplicate and Reallocate_1 share second best performance. However, two facts must be taken into consideration regarding the evaluation of neighborhoods performance: the initial solution is mostly constructed by the Insertion; and the order of neighborhoods in local search procedure (neighborhoods located at the beginning of the local search should find more improvements than the ones at the end). Therefore, further elaborate testing of local search neighborhoods effectiveness is required in the future.

The solutions quality is dependent on the values of tune-up parameters and more intensive testing of the proposed approach should be performed regarding a larger set of instances and values of tune-up parameters. The hybrid VNS/LP approach can be modified to solve other variants of production scheduling problems, and this is also one of the possible future research directions.

Table 1: The results for 100 solution runs of modified instance I1 for max CPU time of 2 h

	Tune-up parameters *(Iter_max=Max_shak_intensity=Max_shak_pass)*										
	6	7	8	9	10	11	12	13	14	15	16
min	96381.2	95869.3	96068.1	95189.9	95322.3	95293.5	95143.0	95357.3	95375.6	95064.4	94984.7
average	**101994.3**	**100770.2**	**100382.3**	**99129.2**	**99119.7**	**98435.3**	**98249.1**	**98109.3**	**98321.4**	**97997.2**	**98316.6**
max	121938.0	145222.9	123169.8	111180.9	113827.5	109578.4	105346.7	106774.3	125829.2	106704.4	118171.7
stdev	4548.9	5435.2	4312.9	2475.2	2824.2	2354.3	1912.5	1991.6	3584.8	2257.6	2839.6
The best	**94577.2**	**94577.2**	**94577.2**	**94577.2**	**94577.2**	**94577.2**	**94577.2**	**94577.2**	**94577.2**	**94577.2**	**94577.2**
Error [%]	**7.84%**	**6.55%**	**6.14%**	**4.81%**	**4.80%**	**4.08%**	**3.88%**	**3.73%**	**3.96%**	**3.62%**	**3.95%**

Table 2: The results for 100 solution runs of modified instance I1 for max CPU time of 4 h

	Tune-up parameters *(Iter_max=Max_shak_intensity=Max_shak_pass)*					
	8	10	12	14	16	18
min	96041.6	95400.9	95304.7	94619.1	94577.2	94649.6
average	**98778.9**	**97487.2**	**96976.8**	**96824.1**	**96522.0**	**96581.6**
max	107126.1	104912.4	104588.9	101456.9	100409.2	101384.7
stdev	2179.6	1525.1	1455.0	1234.9	1050.2	1333.6
The best	**94577.2**	**94577.2**	**94577.2**	**94577.2**	**94577.2**	**94577.2**
Error [%]	**4.44%**	**3.08%**	**2.54%**	**2.38%**	**2.06%**	**2.12%**

Table 3: The average number of solution improvements for 10 local search neighborhoods (1100 and 600 solution runs for max CPU time 2 and 4 h respectively)

Max CPU time	Insertion	Duplicate	Removal	Reallocate 1	Swap 1	Reallocate 2	Swap 2	Reallocate 3	Swap 3	Shuffle
2 h	336.1	86.3	352.7	76.1	45.4	26.2	31.7	10.6	6.6	14.5
4 h	530.0	129.4	578.6	125.5	69.2	37.4	46.0	15.8	9.8	20.1

Acknowledgements

This work was supported by the Ministry of Education, Science and Technical development of the Government of the Republic of Serbia through the project TR36006, for the period 2011-2018.

References

Baldo T.A., Santos M.O., Almada-Lobo B., Morabito R. (2014). “An optimization approach for the lot sizing and scheduling problem in the brewery industry”, Computers & Industrial Engineering, 72, pp. 58-71.

Clark A., Almada-Lobo B., Almeder C. (2011). “Lot sizing and scheduling: industrial extensions and research opportunities. International Journal of Production Research, 49(9), pp. 2457-2461.

Ferreira D., Clark A.R., Almada-Lobo B., Morabito R. (2012). “Single-stage formulations for synchronised two-stage lot sizing and scheduling in soft drink production”, International Journal of Production Economics, 136(2), pp. 255-265.

Ferreira D., Morabito R., Rangel S. (2009), “Solution approaches for the soft drink integrated production lot sizing and scheduling problem”, European Journal of Operations Research, 196, pp. 697-706.

Ferreira D., Morabito R., Rangel S. (2010). “Relax and fix heuristics to solve one-stage one-machine lot-scheduling models for small-scale soft drink plants”, Computers & Operations Research, 37(4), pp. 684-691.

Hansen P., Mladenović N.,Todosijević R., Hanafi S., (2017), "Variable neighborhood search: basics and variants”, EURO Journal on Computational Optimization, 5(3), pp. 423-454.

Mladenovic N., Hansen P. (1997), “Variable neighborhood search”, Computers & Operations Research, 24, pp. 1097-1100.

Pagliarussi M.S., Morabito R., Santos M.O. (2017). “Optimizing the production scheduling of fruit juice beverages using mixed integer programming models”, Gestão & Produção, 24(1), pp. 64-77.

Popović D., Bjelić N., Vidović M., Radivojević G., Ratković B. (2018), “The MIQP model for inventory management problem in production scheduling of fruit juice beverages”, Proceedings of the International Scientific Conference QUANTITATIVE METHODS IN ECONOMICS Multiple Criteria Decision Making XIX, Trenčianske Teplice, Slovakia, pp. 291-297, ISBN 978-80-89962-08-2.

Toledo C.F.M., França P.M., Morabito R., Kimms A. (2007), “Um modelo de otimização para o problema integrado de dimensionamento de lotes e programação da produção em fábricas de refrigerantes”, Pesquisa Operacional, 27(1), pp.155-186.

Toledo C.F.M., Oliveira L., Pereira R.F., França P.M., Morabito R. (2014), “A genetic algorithm/mathematical programming approach to solve a two-level soft drink production problem”, Computers & Operations Research, 48, pp. 40-52.