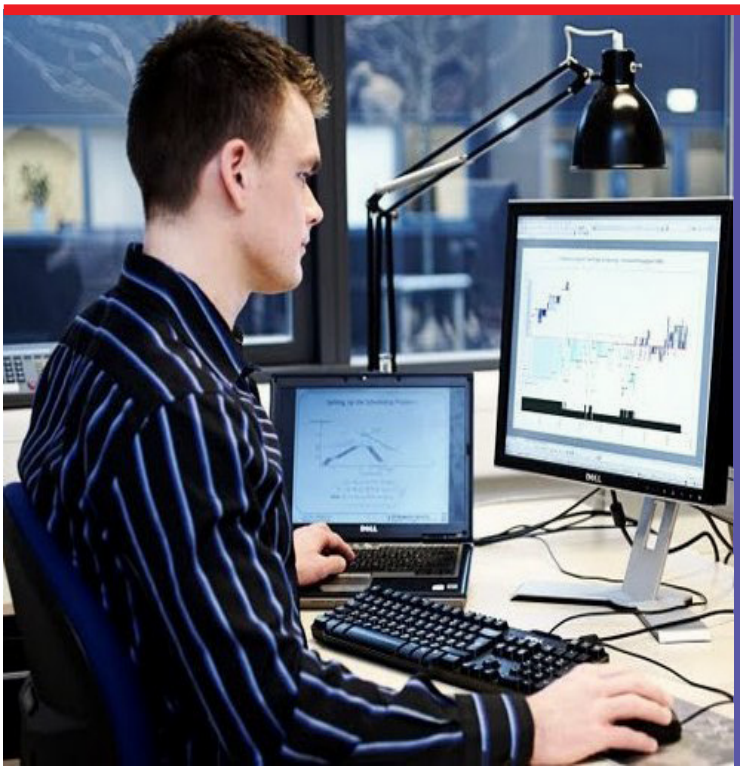
A hybrid adaptive large neighborhood search algorithm applied to a lot-sizing problem

Muller, Laurent Flindt; Spoorendonk, Simon

Publication date:
2010

Document Version
Publisher's PDF, also known as Version of record

A hybrid adaptive large neighbor-hood search algorithm applied to a lot-sizing problem



Report 1.2010

DTU Management Engineering

Laurent Flindt Møller
Simon Spoorendonk
January 2010

DTU Management Engineering
Department of Management Engineering

A hybrid adaptive large neighborhood search algorithm applied to a lot-sizing problem

Laurent Flindt Muller Simon Spoorendonk

Department of Management Engineering,
Technical University of Denmark
Produktionstorvet, Building 426, DK-2800 Kgs. Lyngby, Denmark
lafm@man.dtu.dk, spoo@man.dtu.dk

Abstract

This paper presents a hybrid of a general heuristic framework that has been successfully applied to vehicle routing problems and a general purpose MIP solver. The framework uses local search and an adaptive procedure which choses between a set of large neighborhoods to be searched. A mixed integer programming solver and its built-in feasibility heuristics is used to search a neighborhood for improving solutions. The general reoptimization approach used for repairing solutions is specifically suited for combinatorial problems where it may be hard to otherwise design operations to define a neighborhood of a solution and to investigate the feasibility of elements in such a neighborhood. The hybrid heuristic framework is applied to the multi-item capacitated lot sizing problem with dynamic lot sizes, where experiments have been conducted on a series of instances from the literature. On average the heuristic solutions are within 0.2% of the previously best known solutions and we found new improved upper bounds for 3 out of 12 instances.

Keywords: Adaptive large neighborhood search, capacitated lot sizing problem with setup times

1 Introduction

The adaptive large neighborhood search (ALNS) heuristic is a concept introduced by Røpke and Pisinger [15]. The ALNS heuristic is an improvement heuristic that operates on top of a construction heuristic. The improvement is done using a local search method, e.g., simulated annealing or tabu search, choosing between different neighborhoods. In each iteration of the search a *destroy neighborhood* is chosen to destroy the current solution, and a *repair neighborhood* is chosen to repair the solution. The neighborhoods are weighted according to their success and weights are adjusted as the ALNS heuristic progresses.

The main motivation for extending the ALNS heuristic to a hybrid version is, that not all problem types are equally well suited for defining neighborhoods. Especially the construction and the exploration of repair neighborhoods can be a challenge, both w.r.t. finding a meaningful repair operation and testing feasibility of such an operation. To address this problem, we propose to use a mixed integer programming (MIP) solver in the repair phase of

the ALNS heuristic. The idea is to solve a restricted subproblem that is based on a partial solution where variables are fixed (or partly fixed). The process of constructing a subproblem, and the following reoptimization of the subproblem with the use of a MIP solver, can in the context of an ALNS heuristic be seen as the application of a destroy and a repair neighborhood. As such the hybrid ALNS can be viewed as a specialization of the ALNS framework, which removes the task of defining repair neighborhoods.

The ALNS framework by Røpke and Pisinger [15] is extended from a large neighborhood search framework by Shaw [16]. The heuristic has several similarities with variable neighborhood search, see e.g., Mladenović and Hansen [10], and hyper-heuristics, see e.g., Burke et al. [2]. However, there are no adaptiveness built into the basic idea of the variable neighborhood search. This approach mainly relies on the diversity of the neighborhoods being used. The hyper-heuristic approach operates on simpler low-level heuristics where the ALNS heuristic operates on neighborhoods. Furthermore, an evaluation function is used to calculate a score for each low-level heuristic and chooses the best scoring heuristic for the next iteration. The ALNS heuristic uses a random selection between all weighted neighborhoods to choose the neighborhood for next iteration. In both cases this allows for bad performing or seldom used neighborhoods to be chosen which can provide diversity in the solution process. An ALNS heuristic has been implemented for vehicle routing problems with great success, see Pisinger and Røpke [12], Røpke and Pisinger [15]. Examples of an application of the framework outside a routing problem context are few, see Cordeau et al. [4] that schedules technicians and tasks in a telecommunications company and Muller [11] presented an ALNS heuristic for the resource constrained project scheduling problem. For a recent survey on large neighborhood search and the ALNS framework we refer to Pisinger and Røpke [13].

The multi-item capacitated dynamic lot sizing problem with setup times (CLSP) can be defined as: Schedule the production of a set of items over a given number of time periods such that all demands of items are met, and such that the capacity of the resource is not exceeded. The production of each unit of an item and the setup of the production consume capacity on the resource. The CLSP is $\mathcal{NP}$-hard, see e.g., Pochet and Wolsey [14]. Few papers concern meta-heuristic approaches for the CLSP, see e.g., the surveys of Jans and Degraeve [8] and Buschkühl et al. [3]. One of the few is the variable neighborhood search heuristic by Hindi et al. [7]. Exact approaches for the CLSP includes branch-and-cut algorithms by Belvaux and Wolsey [1], Wolsey [18], Miller et al. [9] and a branch-and-price algorithm by Degraeve and Jans [6]. The good performance of the branch-and-cut algorithms suggest that using a MIP solver to solve restricted subproblems of the CLSP can be done in reasonable time.

The contribution of this paper is an ALNS heuristic which combines the strength of modern MIP solvers with that of the ALNS heuristic, creating a "hybrid" approach. The repair neighborhoods employ the MIP solver in a generic fashion and neighborhoods are thus applicable to a large variety of problems. An evaluation of the hybrid ALNS heuristic is applied on the CLSP, and its usefulness is indicated by the results where the heuristic is on average within 0.2 % of the best known upper bounds and is able to find improved upper bounds on 3 out of 12 instances.

The paper is organized as follows: Section 2 gives an outline of the ALNS framework, Section 3 describes the hybrid variant proposed in this paper, Section 4 presents an application of the hybrid ALNS heuristic to the CLSP, and Section 5 contains the experimental results performed on the instances of Trigeiro et al. [17] and an extension of these described in Miller

et al. [9]. Section 6 concludes the paper and suggest new directions for future research.

2 ALNS framework

In this section, an outline of the the ALNS framework is presented as described by Pisinger and Røpke [12]. The framework is divided into three parts, i) a master local search framework, ii) a set of large neighborhoods that either destroyor repaira possible solution, and iii) a procedure for choosing neighborhoods which adapts based on historical information.

At the top level (also denoted *master level*) any local search heuristic can be applied, e.g., simulated annealing, tabu search, guided local search, or GRASP (greedy randomized adaptive search procedure). A neighborhood of a solution is a set of solutions obtained by performing some operation on the original solution. In large neighborhoods these operations involve changing several settings in the solution at once leading to a neighborhood of potentially exponential size. The procedure for choosing neighborhoods can be pictured as a roulette wheel spinning and randomly choosing a neighborhood. The weights of all neighborhoods are updated based on historical success. Hence, successful neighborhoods have a higher probability to be used as time passes.

The ALNS framework can be described as follows: Given a starting solution the heuristic iteratively tries to improve it. The set of neighborhoods is divided in two – destroy neighborhoods N^- and repair neighborhoods N^+. Given a current solution x a destroy neighborhood $n^- \in N^-$ performs an operation on x, stores the removed elements in an *item bank* B and leaves a partial solution $\overline{x}$. A repair neighborhood inserts elements from the item bank into $\overline{x}$ creating a new solution x'. In the case of the hybrid ALNS heuristic presented in this paper, the repair neighborhoods make use of a MIP solver. A roulette wheel for each of the sets N^- and N^+ is used in each iteration to choose which destroy and which repair neighborhood should be used. This is based on a weight π for each neighborhood that is initialized at the beginning according to the quality of the neighborhood (this is a user defined consideration to be made a priori). During execution the weights are updated according to the quality of the solutions produced with the given neighborhoods. The motivation behind this is, that not all neighborhoods perform equally good on all problem instances – hence the weights of the neighborhoods adapt to the instance during the execution of the algorithm and hopefully produce better solutions overall.

As mentioned above, the weights of the neighborhoods are updated according to how successful a neighborhood is in obtaining better and new solutions. In the paper by Pisinger and Røpke [12] the weights are updated after certain given time segments and the weight of a neighborhood is based on the *observed* weight $\overline{\pi}_{i,t}$ of neighborhood i at time segment t which in each iteration is incremented with values dependent on the quality of the new solution x'.

At the end of a time segment the *smoothened* weight $\pi_{i,t}$ is calculated for use in the roulette wheel. This is based on $\overline{\pi}_{i,t}$ and a *reaction factor* r which indicates how much the roulette weight depends on previous success. Previous success is calculated as the observed weight $\overline{\pi}_{i,t}$ divided by the number of times $a_{i,t}$ neighborhood i has been chosen in time segment t. The updated smoothened weight is: $\pi_{i,t+1} = r\frac{\overline{\pi}_{i,t}}{a_{i,t}} + (1-r)\pi_{i,t}$ A low reaction factor keeps the weight at about the same level during the algorithm. A neighborhood i has the probability: $p_t(i) = \frac{\pi_{i,t}}{\sum_{j \in N} \pi_{j,t}}$ of being chosen in time segment t.

In Figure 1 the pseudo-code is given for the ALNS framework. The choices made for the hybrid ALNS heuristic will be described in the following sections.

```
ALNS
 1  x is an initial solution; set x* := x
 2  repeat
 3          Choose n⁻ ∈ N⁻ and a n⁺ ∈ N⁺ based on π
 4          Generate solution x′ based on x, n⁻, and n⁺
 5          if x′ is accepted
 6             then x := x′
 7          if x′ < x*
 8             then x* := x′
 9          Update π for N⁻ and N⁺
10     until stop criteria is meet
11  return x*
```

Figure 1: The criteria for accepting a new solution in line 5 depends on the choice of local search framework and the score update on line 9 can be performed using different strategies.

3 A hybrid ALNS algorithm

In this section, we describe the elements of the proposed hybrid ALNS algorithm, i) the master level local search procedure, ii) the genereic MIP repair neighborhoods and iii) an adjusted weight calculation method used in the adaptive procedure.

3.1 Local search

In this paper steepest descent has been chosen as the local search procedure. Because the the destroy neighborhoods merely removes parts of that solution, and because the MIP repair neighborhoods use the current solution as an initial upper bound, it is always possible for the MIP solver to find that solution again. Therefor, the MIP solver never returns a solution that is worse than the current one. Hence, a selection process for choosing worse solutions would never apply. It is possible that, when considering all valid solutions explored in a given iteration of a repair neighborhood worse solutions are found. These are stored and used as restart points of the local search.

To diversify the search, the search is restarted at different solutions when no improvements have occurred in a given number of iterations. The number of iterations is chosen to be equal to the segment size for updating the neighborhood weights. For the initial restart the second best solution is chosen (the current solution is the best solution). In subsequent restarts, the local search either switches back to the best solution if it is not the current one or switches to next best solution that has not previously been used for a restart. The reason for returning to try and improve on the best solution is that the neighborhoods may have obtained different scores in the meantime yielding a diversified exploration of the neighborhood of the solution.

3.2 Generic MIP repair neighborhoods

As the name indicates the generic MIP repair neighborhoods make use of a MIP solver in order to repair or reoptimize a solution. Consider the minimization problem $\min\{cx|Ax = b, Bx \leq d, x \geq 0\}$, where A is an $n \times m$ matrix, B is an $n \times l$ matrix, and c, b, d and x are

n-dimensional vectors. Given the partial solution $\bar{x}$ returned from a destroy neighborhood, a subproblem based on the MIP formulation above is constructed. This is done by using one of the two rules described below for fixing the lower and upper bounds bound of the variables x given the values of $\overline{x}$.

Bound by solution value. Bound non-removed variables x to a fraction of their current value, i.e., if x_t is a variable which has not been removed by a destroy neighborhood, fix $x_t \geq \delta \overline{x}_t$ for $\delta \in [0, 1]$. In this paper $\delta = 0.5$ was chosen.

Bound by equality constraints. If for a given equality constraint with a right-hand-side of b there is only one non-zero variable, say x_t, then fix $x_t \geq b$.

One may argue, that these bounding rules are in fact further relaxations of the partial solution, and should therefor be thought of as second level destroy neighborhoods. However, since the MIP solver is used for finding feasible solutions, we stay with the concept of repair neighborhoods.

To speed up the subproblem solution process, we suggest to limit the number of explored branch nodes or end the search after a given time limit. MIP heuristics, such as relaxation induced neighborhood search by Danna et al. [5], are applied in the MIP solver to obtain feasible solutions rapidly. If no feasible solution is found, the invalid solution is discarded (i.e. not accepted) and the ALNS heuristic continues to the next iteration.

3.3 Adaptive weight adjustments

In this paper a slightly different approach is used for updating the observed weights: Let cx be the value of the current solution and let cx' be the value of the new solution. The observed weight $\overline{\pi}_{i,t}$ of neighborhood i at time segment t is updated as follows

$$\bar{\pi}_{i,t} = \bar{\pi}_{i,t} + b^{(cx - cx')/cx}$$

where b is some constant. The observed weights are updated differently because of the choice of local search procedure. For the selected procedure, a solution is only accepted when it is better than the current solution, which may result in long periods without any accepted solutions. If a fixed scoring scheme was used all neighborhoods would score equally bad, while for the employed scheme it is possible to differentiate the neighborhoods who produce solutions which are almost as good as the current one and the ones that produce solutions which are far worse.

4 An application to the CLSP

In this section we present an application of the hybrid ALNS heuristic to the CLSP. First, a description of the mathematical model is given, followed by a description of the neighborhoods employed, and finally the parameter values are presented.

4.1 Problem description

This section briefly describes the mathematical formulation of the CLSP. Let $I = \{1, \ldots, n\}$ be the set of items and $T = \{1, \ldots, m\}$ be the set of time periods. The data set is given by:

- h_t^i is the unit inventory cost of item i at time t.
- p_t^i is the unit production cost of item i at time t.
- f_t^i is the fixed cost for producing item i at time t.
- d_t^i is the demand of item i at time t.
- α_t^i is the capacity used for producing item i at time t.
- β_t^i is the capacity used for setting up the production of item i at time t.
- C_t is the capacity of the resource at time t.
- M is a large constant.

and the variables are

- s_0^i is the number of units of item i in the initial inventory.
- s_t^i is the number of units of item i in stock after time t.
- x_t^i is the number of units of production of item i at time t.
- y_t^i indicates if a setup for production of item i at time t has been done.

All variables except the y-variables are positive continuous variables. The y-variables are binary and implicitly force the other variables to obtain integer values (if all constants are also integer).

The mathematical model for the CLSP:

$$\min \sum_{i\in I}\left(h_0^i s_0^i + \sum_{t\in T}(h_t^i s_t^i + p_t^i x_t^i + f_t^i y_t^i)\right) \tag{1}$$

$$\text{s.t. } s_{t-1}^i + x_t^i = d_t^i + s_t^i \qquad t\in T,\ i\in I \tag{2}$$

$$x_t^i \le M y_t^i \qquad t\in T,\ i\in I \tag{3}$$

$$\sum_{i\in I}\left(\alpha_t^i x_t^i + \beta_t^i y_t^i\right) \le C_t \qquad t\in T \tag{4}$$

$$s_t^i, s_t^0, x_t^i \ge 0,\ y_t^i \in \mathbb{B} \qquad t\in T,\ i\in I \tag{5}$$

The objective (1) is to minimize holding, production, and setup cost. Constraints (2) ensure flow conservation of the item. That is, items in stock plus the items produced in a time period must equal the number of items demanded in this time period plus the number of items in stock after this time period. Constraints (3) ensure that production of an item can only occur if the resource is set up to produce that item. Constraints (4) impose the restriction on the resource capacity such that the combined production and setup cannot exceed. The variable domains are specified by constraints (5).

4.2 Destroy neighborhoods

Except for the random removal neighborhood each neighborhood focuses on specific structural problems in a solution to the CLSP, e.g., too many items in stock, or often changing production of items. The destroy neighborhoods are divided in two steps. First, a candidate set of production to be removed are constructed and secondly the candidates is removed randomly.

Random. Removes production (positive x-variables in the current solution) at random throughout the production plan. The neighborhood is good at creating different solutions, and is useful if the search is stuck in a local minimum.

Production causing stock. This neighborhood removes production that causes items to be placed in stock. Hopefully, the production can be inserted at a later time in the production plan, hence saving inventory expenses.

Capacity critical. To obtain diversity in a solution where the resource is fully loaded are removed. This allows for a reshuffling of production. When the production of an item is removed, the production of that item in the previous and the next time periods are also removed to open the possibility of shifting the production between these time periods.

Stocked items. This neighborhood removes production of items that have the largest amount of units in stock throughout the production plan. The idea is to attempt a reshuffle of stocked items between those types of items that are favorable to put in stock, e.g., due to low holding cost. A minimum of two item types are removed in an iteration.

Time periods with high stock density. This neighborhood removes production from time periods where there are many items in stock. The idea is to reshuffle the production (and thereby the stocked items) into the previous or succeeding time periods. The time periods are sorted according to the number of stocked items. When the production in a time period is cleared, the time period before and after are also cleared. The remaining time periods are then resorted before the next period is selected.

Production higher than demand. When there are productions that are very high compared to the demand of the corresponding item in the time period, a lot of items are put in stock. Often a solution can be shifted, such that the majority of the production is moved to the following time period.

4.3 Repair neighborhoods

The generic MIP repair neighborhoods employed for the MCLS are:

Bound by solution value. In CLSP, the production variables x imply the setup variables y. Hence, whenever $\overline{x}_t^i > 0$ we bound the variable $y_t^t = 1$. The x variables are bounded as described earlier. Stock is generally to be avoided, therefor we fix all lower bounds of the stock variables s to 0 so that we do not accidentally force unnecessary stock.

Bound by equality constraints. We only consider the production variables x in the context of constraints (2) when $x_t^i = d_i^t$. Then we bound $x_t^i = \overline{x}_t^i$ where we have $\overline{x}_t^i = d_i^t$. In solutions for the CLSP, this scenario happens frequently in consecutive time periods for the production of an item. In order to allow some diversification in the production we only fix production when there is no stock in the preceding nor the succeeding

item periods, i.e., we do not fix the production in the first and the last time period in consecutive time periods, where production equals demand for a certain item.

4.4 ALNS parameters

A time limit of 60 seconds is chosen as the termination criteria of the steepest decent local search. The reaction factor r is set fairly high at 0.8 since we expect few iterations, and therefor would like the neighborhood weights to converge fast. All neighborhoods are initialized with a weight of 1. During the first iteration, an estimate of the overall number of iterations is calculated based on the running time of that iteration. The segment size for updating the weights is set to one hundredth of the estimated number of overall iterations, or at least 10 and at most 50 iterations. Q is a parameter controlling how large a part of the current solution is destroyed which is measured in a percentage of combined production for the CLSP. Muller [11] suggests an exponential decrease of Q from given start value to some end value. In this paper we propose a linear decrease beginning at $Q = 0.4$ and decreasing toward $Q = 0.1$. The linear decrease provides room for large neighborhoods in more iterations than the exponential decrease which is crucial when few iterations are expected.

5 Experimental results

The experiments are divided in two: i) a neighborhood performance analysis, and ii) a comparison of solution quality of the ALNS heuristic with the best known solutions from the literature and the MIP solver ILOG CPLEX. The experiments are performed on a 2.66 GHz Intel(R) Xeon(R) X5355 machine with 8 GB memory using ILOG CPLEX version 10.2. The time limit for the ALNS heuristic and the tests with ILOG CPLEX is 60 seconds. For the ALNS heuristic the average is calculated based on 100 runs. The ALNS heuristic is initialized with a heuristic solution found by ILOG CPLEX after solving the root node.

5.1 Instances

The first set of instances is a small subset of the randomly generated instances from Trigeiro et al. [17]. These 6 instances are assumed to be among the hardest, see Belvaux and Wolsey [1]. They have 6, 12, and 24 different types of items respectively, and the planning horizon is either 15 or 30 time periods. The instances are named as `trI-T` with `I` being the number of items and `T` being the number of time periods. The second set of instances is an extension of the first set by Miller et al. [9] and is an attempt to produce even harder instances. These instances are denoted `xtrI-T`, and have the same number of items and time periods as the instances for which they were derived, but with no setup costs and a constant of 10 larger setup times for all items. Note, that the instance `xtr12-15` appears to be further altered since Miller et al. [9] report solutions with no initial inventory although the lower bounds found by ILOG CPLEX implies that no solution exists without initial inventory or Miller et al. [9] have an error in their solution.

5.2 Neighborhood performance

Table 1 specifies how often a neighborhood was chosen (in percentage of the total number of iterations). The destroy neighborhoods are the random (Random), production causing stock

Instance	Destroy nhs.						Repair nhs.	
	Random	Prod. caus. stock	Crit. cap.	Stock. items	High stock dens.	High prod.	Bound value	Bound equal.
`tr6-15`	16.6	16.8	16.5	16.6	16.7	16.8	49.8	50.2
`tr6-30`	18.2	16.9	17.1	12.7	17.6	17.5	50.8	49.2
`tr12-15`	16.5	16.5	17.0	16.7	16.9	16.4	50.0	50.0
`tr12-30`	16.2	16.9	15.9	17.8	17.0	16.2	48.9	51.1
`tr24-15`	17.1	16.6	16.6	17.1	16.0	16.6	50.1	49.9
`tr24-30`	17.6	16.9	15.2	17.2	16.6	16.5	49.0	51.0
`xtr6-15`	16.7	16.6	16.6	16.6	16.7	16.8	50.1	49.9
`xtr6-30`	17.0	16.6	16.9	15.6	17.5	16.4	49.6	50.4
`xtr12-15`	17.5	14.9	17.8	18.0	17.1	14.7	56.2	43.8
`xtr12-30`	19.1	14.9	17.5	15.3	17.4	15.9	50.1	49.9
`xtr24-15`	16.4	16.8	17.0	16.4	18.0	15.5	51.4	48.6
`xtr24-30`	18.4	15.6	18.2	15.0	17.3	15.4	49.5	50.5
Average	17.3	16.3	16.9	16.3	17.1	16.2	50.5	49.5

Table 1: Destroy and repair neighborhood. Number of times a neighborhood is chosen in percentage.

(Prod. caus. stock), capacity critical (Crit. cap), stocked items (Stock. items), time periods with high stock density (High stock dens.), and production higher than demand (High prod.). The repair neighborhoods are the bound by solution value (Bound value) and the bound by equality constraints (Bound equal). Note that the number of times a neighborhood is chosen also reflects its quality since good neighborhoods scores better than bad neighborhoods. The difference in usage of less than 1.1 and 1.0 percentage points for the destroy and repair neighborhoods, respectively, indicates that the neighborhoods all perform equally well.

5.3 Upper bounds

Table 2 shows a comparison of the upper bounds obtained in different papers. The first column is the instance name (Instance) and the second column reports the best known upper bounds found in the literature (Prev. best). The next two columns are results from the ALNS heuristic, best upper bound (ALNS best) and average upper bound of the 100 runs (ALNS avg.). This is followed by two columns reporting results using ILOG CPLEX with a 60 second time limit given the default settings (`CPLEX` default), and given the MIP emphasis parameter of ILOG CPLEX set to find hidden feasible solutions (`CPLEX` hidden). Further listed are the relax-and-fix heuristic (BW) by Belvaux and Wolsey [1] based on their branch-and-cut algorithm. Thier algorithm is running (on a 200 Mhz Intel(R) Pentium(R) Pro computer) for about 40 seconds on the smaller instances to a little less than 5 minutes on the largest instance. Miller et al. [9] considers a branch-and-cut and a cut-and-branch algorithm. In the latter approach cutting planes are only applied in the root node. In the column (MNS) the best integer solution for either of these approaches is given. Last in column (DJ) the results of the branch-cut-and-price-algorithm by Degraeve and Jans [6] is presented. The integer solutions are found with a limit of 2000 branch nodes on an Intel(R) Pentium(R) III 750 Mhz. A boldface entry indicates the best upper bound for an instance or the best total, and a '-' entry indicates that no result is available.

In 11 out of the 12 cases (not `tr24-30`) the ALNS heuristic finds a solution that is at least as good as the previously best known upper bound, and in three of these cases (`xtr12-30`, `xtr24-15` and `xtr24-30`) the heuristic finds better solutions. On average, the ALNS heuristic

Instance	Prev. best	ALNS best	ALNS avg.	CPLEX default	CPLEX hidden	BW	MNS	DJ
tr6-15	**37721**	**37721**	37776.4	**37721**	**37721**	**37721**	**37721**	38162
tr6-30	**61746**	**61746**	61792.2	61782	61765	**61746**	**61746**	62162
tr12-15	**74634**	**74634**	74720.8	**74634**	**74634**	74752	**74634**	75035
tr12-30	**130596**	**130596**	130675.0	130705	130599	130599	**130596**	131234
tr24-15	**136509**	**136509**	136510.0	136568	**136509**	**136509**	**136509**	136860
tr24-30	**287929**	287942	287966.0	287981	287937	287950	**287929**	288383
xtr6-15	**5170**	**5170**	5185.1	5203	**5170**	-	**5170**	-
xtr6-30	**7880**	**7880**	7902.4	7985	**7880**	-	**7880**	-
xtr12-15	-*	**817451**	824002.0	817647	**817451**	-	7889*	-
xtr12-30	14799	**14386**	14800.6	14925	14510	-	14799	-
xtr24-15	14120	**13978**	14147.0	14165	**13978**	-	14120	-
xtr24-30	19289	**18318**	18711.9	18687	18641	-	19289	-
Total tr	729135	729148	729440.4	729391	729165	729277	**729135**	731836
Total xtr$^\diamondsuit$	61258	**59732**	60747.0	60965	60179	-	61258	-
Total xtr	-	**877183**	884749.0	878612	877630	-	-	-
Total$^\diamondsuit$	790393	**788880**	790187.4	790356	789344	-	790393	-
Total	-	**1606331**	1614189.4	1608003	1606795	-	-	-

Table 2: Comparison of upper bounds with other approaches. The value marked with * is less than the lower bound reported by ILOG CPLEX. This indicates that modification additional to the ones described was made to this instance by Miller et al. [9]. A total marked with $^\diamondsuit$ is minus the xtr12-15 instance.

is within 0.2% of the best solutions, and it outperforms ILOG CPLEX solver with defaults settings.

6 Conclusion

We have presented a hybrid heuristic solution approach based on the ALNS framework and the generic use of a MIP solver in the repair phase. This results in a general hybrid ALNS algorithm where i) the "difficult" part of creating repair neighborhoods has been eliminated and ii) the strength of modern MIP solvers can be exploited.

The proposed hybrid algorithm has been applied to the CLSP and the computational results indicates that the heuristic is competitive w.r.t to solution quality, within a 60 second time limit when compared with the commercial MIP optimizer ILOG CPLEX. The proposed neighborhoods all benefited the solution process, both the problem specific destroy neighborhoods and the more general repair neighborhood. Furthermore, the ALNS heuristic is, to the best of our knowledge, able to find new best known upper bounds for three instances. Finally, the solution values provided by the ALNS are on average within less than 0.2% of the best known solutions including upper bounds found in this paper. This indicates the use of general MIP based repair neighborhoods in combination with problem specific destroy neighborhoods is effective.

A suggestion for a future improvement for the CLSP and other kinds of problems, is to apply the hybrid ALNS heuristic within the reoptimization process of the subproblems. In order to solve larger problems the MIP solver may become too slow to use in the repair neighborhoods, and it may be beneficial to apply a heuristic approach to reoptimize the

subproblem. This approach can be applied recursively until the subproblems are small enough for the MIP solver to handle efficiently.

Acknowledgements

Thanks to Stefan Røpke for valuable heuristic suggestions and advice on the workings of the ALNS framework. This research is partly supported by the Danish Research Council for Independent Research — Technology and Production Sciences (project 274-08-0353).

References

[1] G. Belvaux and L.A. Wolsey. bc–prod: A specialized branch-and-cut system for lot-sizing. *Management Science*, 46(5):724–738, 2000.

[2] E.K. Burke, G. Kendall, J. Newall, E. Hart, P. Ross, and S. Schulenburg. Hyper-heuristics: An emerging direction in modern search technology. In Fred Glover and Gary Kochenberger, editors, *Handbook of Meta-heuristics*, chapter 16, pages 457–474. Kluwer, 2003.

[3] L. Buschkühl, F. Sahling, S. Helber, and H. Tempelmeier. Dynamic capacitated lotsizing problems – a classification and review of solution approaches. *OR Spectrum*, In press. doi: 10.1007/s00291-008-0150-7.

[4] J.-F. Cordeau, G. Laporte, F. Pasin, and S. Røpke. Scheduling technicians and tasks in a telecommunications company. *Journal of Scheduling*, 2009. Forthcoming.

[5] E. Danna, E. Rothberg, and C. Le Pape. Exploring relaxation induced neighborhoods to improve mip solutions. *Mathematical Programming*, A(102):71–90, 2005.

[6] Z. Degraeve and R. Jans. A new dantzig-wolfe reformulation and branch-and-price algorithm for the capacitated lot-sizing problem with setup times. *Operations Research*, 55 (5):909–920, 2007.

[7] K.S. Hindi, K. Fleszar, and C. Charalambous. An effective heuristic for the clsp with set-up times. *Journal of the Operational Research Society*, 54:490 – 498, 2003.

[8] R. Jans and Z. Degraeve. Meta-heuristics for dynamic lot-sizing: A review and comparison of solution approaches. *European Journal of Operation Research*, 177(3):1855–1875, 2007. doi: 10.1016/j.ejor.2005.12.008.

[9] A.J. Miller, G.L. Nemhauser, and M.W.P. Savelsbergh. Solving multi-item capacitated lot-sizing problems with setup times by branch-and-cut. Technical Report 39, Center for Operations Research and Econometrics, Universite Catholique de Louvain, Belgium, 2000.

[10] N. Mladenović and P. Hansen. Variable neighborhood search. *Computers and Operations Research*, 24:1097–1100, 1997.

[11] L. F. Muller. An adaptive large neighborhood search algorithm for the resource-constrained project scheduling problem. In *Proceedings of the VIII Metaheuristics International Conference (MIC) 2009*, Hamburg, Germany, 13-16 July 2009.

[12] D. Pisinger and S. Røpke. A general heuristic for vehicle routing problems. *Computers and Operations Research*, 34(8):2403–2435, 2007. doi: 10.1016/j.cor.2005.09.012.

[13] D. Pisinger and S. Røpke. Large neighborhood search. In M. Gendreau and J.-Y. Potvin, editors, *Handbook of Metaheuristics*. Springer, 2nd edition, 2009. Forthcoming.

[14] Y. Pochet and L.A. Wolsey. *Production Planning in Mixed Integer Programming*. Springer, 2006.

[15] S. Røpke and D. Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40(4):455–472, 2006. doi: 10.1287/trsc.1050.0135.

[16] P. Shaw. Using constraint programming and local search methods to solve vehicle routing problems. *CP-98 (Fourth International Conference on Principles and Practice of Constraint Programming)*, 1520:417–431, 1998.

[17] W.W. Trigeiro, L.J. Thomas, and J.O. McClain. Capacitated lot sizing with setup times. *Management Science*, 35(3):353–366, 1989.

[18] L.A. Wolsey. Solving multi-item lot-sizing problems with an MIP solver using classification and reformulation. *Management Science*, 48(12):1587–1602, 2002.

This paper presents a hybrid of a general heuristic framework that has been successfully applied to vehicle routing problems and a general purpose MIP solver. The framework uses local search and an adaptive procedure which choses between a set of large neighborhoods to be searched. A mixed integer programming solver and its built-in feasibility heuristics is used to search a neighborhood for improving solutions. The general reoptimization approach used for repairing solutions is specifically suited for combinatorial problems where it may be hard to otherwise design operations to define a neighborhood of a solution and to investigate the feasibility of elements in such a neighborhood. The hybrid heuristic framework is applied to the multi-item capacitated lot sizing problem with dynamic lot sizes, where experiments have been conducted on a series of instances from the literature. On average the heuristic solutions are within 0.2% of the previously best known solutions and we found new improved upper bounds for 3 out of 12 instances.

ISBN 978-87-90855-64-2

DTU Management Engineering
Department of Management Engineering
Technical University of Denmark

Produktionstorvet
Building 424
DK-2800 Kongens Lyngby
Denmark
Tel. +45 45 25 48 00
Fax +45 45 93 34 35

www.man.dtu.dk