universität wien

MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis

Centralized solutions for the collaborative capacitated multi-level lotsizing problem

verfasst von / submitted by

Patrick Födermayr, BSc

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien, 2020 / Vienna 2020

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	UA 066 915
Studienrichtung lt. Studienblatt / degree programme as it appears on the student record sheet:	Masterstudium Betriebswirtschaft
Betreut von / Supervisor:	O. Univ.-Prof. Dipl.-Ing. Dr. Richard F. Hartl
Mitbetreut von / Co-Supervisor:	Univ.-Prof. Mag. Dr. Margaretha Gansterer

Table of Contents

List of Figures

List of Tables

List of Algorithms

1. Introduction

In today's increasingly competitive market, firms often have to collaborate with other actors in their supply chain networks to cut costs and increase profitability. This is usually achieved via division of labor in some form, which requires a coordination effort. However, collaborative planning does not limit itself to just coordinating plans, Stadtler (2009) describe it as an adaptation of individual plans in an effort of joint decision making. Therefore, mutual agreement on those plans needs to occur to ensure that a collaboration will last. This, in turn, may require actors to give up information to each other.

One of the largest cost drivers in supply chains is multi-level lotsizing. According to Lee and Kumara (2007) it is one of the most investigated research areas of supply chain management, largely because of its directly practical applications (e.g. Tempelmeier and Derstroff, 1996; Tempelmeier and Helber, 1994; Dellaert and Jeunet, 2000a; Pitakaso et al., 2007; Buschkühl et al., 2010). In the literature there have been numerous decentralized collaboration mechanisms for distributed multi-level lot-sizing problems (e.g. Lee and Kumara, 2007a; Homberger, 2010; Buer et al., 2013), though only few centralized ones (e.g. Chen, 2015).

Many papers about the lotsizing problem (LSP) with multiple agents try to model the real world by mimicking a certain level of self-interest in the agents. This is typically done by modeling them as rivals (e.g. Buer et. Al, 2015) or as partners with asymmetric information (e.g. Gansterer & Hartl, 2019). In this thesis, we assume "cooperation" to be a situation where every agent has access to all information so that the overall best production plan can be chosen. Alternatively, one could assume that one centralized decision maker has access

to all information of all agents. This could feasibly be accomplished via a software tool, which prevents other agents from viewing sensitive data.

While the addition of capacities is not a novel idea, to our knowledge it has never been attempted on the collaborative multi-level LSP. In this theses we introduce and close this research gap via the following contributions:

- We introduce and mathematically formulate the collaborative, multi-level, capacitated lotsizing problem with setup carryover and transshipments (cf. Gansterer et al. 2020).
- We adapt existing instance data of a similar problem type (the decentralized, multi-level, uncapacitated lotsizing problem (DMLULSP)) to provide a base for future research on this topic.
- We provide solutions for our introduced instances as reference values and examine our results.

The following sections are organized as follows: Section 2 gives an overlook about the Lotsizing Problem as a topic of research. Section 3 covers the problem description. Section 4 focusses on the underlying instance sets. The solution method is detailed in section 5, its results are presented in section 6. Lastly, section 7 presents the conclusion and outlook.

2. Literature Review

In this part, we give a brief overview about the lotsizing problem. Section 2.1 focuses on individual aspects of the problem and why they are relevant. In Section 2.2 solution approaches to different variants of the LSP are examined in closer detail. Section 2.3 concludes this part by going into more detail about collaborations in general.

2.1. Common Extensions of the Lotsizing Problem

The Lotsizing/Lot-Sizing Problem (LSP) is one of the most studied topics in production and logistics. Its aim is to create production plans of at least one product over a finite planning horizon to satisfy customer demand. In its most basic form, it weighs the fixed costs of producing in a given period against the variable costs of storing the produced items until they are needed. There are numerous possible extensions for this problem with the most relevant ones being described briefly here:

Capacity: While it may seem trivial at first, limiting the amount of items that can be produced in any given period can have tremendous impacts on the total costs, especially in larger problems, where a large number of different items need to be produced. Adding capacities also hampers the effectiveness of many heuristics in creating feasible solutions. That being said, the capacitated lotsizing problem (CLSP) is one of the most important extensions in practice, due to its obvious inevitability. The base form of the LSP is therefore often called uncapacitated LSP or ULSP.

Overtime/Additional Capacity: A very similar extension to the one before is adding additional capacity at a premium rate, most commonly in the form of overtime costs though other ways to interpret this exist of course (e.g. using a subcontractor). The general shape of the problem does not change much with this extension, though the number or variables increases significantly due to the additional production variables needed to accurately determine the costs.

Multi-Item: Producing more than one item will be a reality for just about all companies. This means that mathematically, the problem becomes a lot harder to solve to optimality. In most formulations of the model, it introduces the necessity to switch between items and gives the planning decisions to reduce fixed costs more significance.

Multi-Level: Not all products are created as equals. Some of them are intermediary steps to create more elaborate goods. In fact, only a small percentage of products are so-called final products. In multi-level LSPs (MLCLSP or MLULSP for capacitated or uncapacitated cases, respectively), successor and predecessor relations are introduced between items, taken from a bill of materials or similar data.

Lead Times: Sometimes setting up an item i.e. readying a machine for the production of an item may take more time than for other items. Or there are pre/post-processing requirements for an item that cause wait times such as cleaning machinery, letting paint dry, letting materials cool, moving resources in place, etc. Lead times represent a necessary time investment and need to be accounted for in the production plan. Common ways to incorporate them in the model formulation is by either treating lead times as fixed costs but for time resources (such as machine/worker hours) in the capacitated LSP, or in the

uncapacitated LSP declaring the products to be available for further use after a certain number of periods after the production occurred.

Setup Carryover: Setup costs only occur when a different product is being set up. This inherently means that production for an item was set up before, and is now being dismantled. Intuitively, it makes sense that one can simply omit the dismantling stage in order to keep producing in the next period, however, the basic model does not allow for this decision to be made. Adding setup carryover constraints to the model adds realism to the model while potentially reducing costs, at the cost of more complexity.

Multi-Machine: Some production halls may feature multiples of the same machine. This adds a lot of flexibility to the system, though the effects of this only start to show when a few of the other extensions are added as well: multiple machines with setup carryover could allow an item that would incur a disproportionally large amount of setup costs to constantly be assigned to one machine, while a second machine is optimized as a single-machine LSP; multiple machines with lead times makes it so bottlenecks due to these lead times can be mitigated; multiple item levels can have an item A be constantly fed with a production of resource B; and so on.

Multi-Agent: an agent in this context is an independent planning unit, for example different production sites of a company, a subcontractor for hire, or even competitors. It raises the same questions as the multi-machine problem extension but the notable addition of the different parameters of each planning unit. Furthermore, transshipments between agents may have significant effects on the total costs of another agent, as they may be able to omit producing an item that would be very expensive for them otherwise. In other words,

the unique cost benefits of each agent can be utilized to reduce the overall costs of the supply chain.

Multiple Resources: Typically the only resource that needs to be managed in LSPs is time-based. However, one can easily extend this to other factors as well. Secondary resources may include physical raw materials required for the production processes. Or it could manifest as a certain number of workers that are needed to complete a job, possibly even a window of workers for different quality levels of the task. As there is a problem extension for having multiple homogenous machines, the model could also be extended for heterogeneous machines. This, however, leans more into the field of machine scheduling problems.

Backlogging/Lost Sales: In real world scenarios, the idea that all demand can always be fulfilled may not always be feasible or fiscally prudent. However, allowing the production plan to make up for missing production is not as straight-forward as it initially appears. Aside from the problem in trying to specify the costs of lost or belated sales (which is tricky in and of itself as the loss of goodwill needs to be quantified and is not necessarily equal for each customer), it increases the problem complexity significantly.

Uncertainty: Most papers focus on deterministic scenarios, for a number of reasons. Firstly, these problems are easier to solve and model, secondly, they are more reliable in their solution quality by the very nature of deterministic data and are thus more suited for comparison against different solution methods. That being said, the advantages of stochastic solution models should not be overlooked. They often offer a more robust solution framework, can better adapt to sudden changes in the production plan and are of course a

more accurate representation of the real world. Additionally, the stochastic nature of the instance data may apply to many different aspects of the lotsizing problem – stochastic capacity may be interpreted as machine failures, human error, power outages, employees becoming sick, etc; stochastic demand could include cancelled or last-minute orders; stochastic lead times may be caused by a supplier not delivering on time; even production or inventory levels could arguably be stochastic to account for inventory shrinkage or damages.

2.2. Solution Approaches

The oldest solution approaches date back to Wagner & Whitin (1958). It is a dynamic programming algorithm which is now simply known as the Wagner-Whitin algorithm. It constitutes one of the cornerstones of the lotsizing problem as the first exact solution method while being simple to apply, which makes it a useful introductory tool in academic studies to this day. Dynamic programming is primarily used for the single-item LSP due to the problems comparative simplicity, though it often finds usage in solving subproblems of the more complex variants. As such, dynamic programming is the most widely used solution method.

A number of heuristic solution approaches exist for the LSP, though most of them focus on the uncapacitated single-item case. The simplest method is the lot-for-lot procedure, where in every period you produce just enough to match the demand. Silver and Meal (1973) introduced a simple algorithm that determines production based on the average production cost in a given period and adjusting the production horizon until costs increase. The Part Period Balancing algorithm tries to balance production and holding costs for a given periods planning horizon (DeMatteis (1968), Wemmerlöv (1983)).

For more complex problem structures, branch-and-cut methods are often used. Valid inequalities are added to the formulation, to limit the solution space. The specific approach can vary and different authors have provided a multitude of definitions for these inequalities. Barany, Van Roy & Wolsey (1984) are the first to explore this topic, Loparic, Marchand & Wolsey (2003) use knapsack inequalities for the capacitated case.

Other methods include branch-and-bound approaches (e.g. Erenguc and Tufekci (1987), Lotfi and Yoon (1994), Chung, Flynn & Lin (1994), Hardin, Nemhauser & Savelsbergh (2007)), or dual-based approaches (van Hoesel, Wagelmans & Kolen (1991) and Levi, Roundy & Shmoys (2006)).

When trying to solve complex variants of the LSP (such as multi-level LSPs, multi-item LSP with capacity restrictions, LSP with backlogging, etc.) exact algorithms are often not able to solve these problems to optimality, especially so for large instances. Ways to solve these kinds of problems include metaheuristics such as Tabu Search (e.g. Hindi (1995), Hindi (1996)), Genetic Algorithms (Hernandez & Süer (1999), Dellaert, N., & Jeunet, J. (2000b)) or Memetic Algorithms (Berretta & Rodriguez (2004)).

Additionally, methods like Column Generation (e.g. Manne (1958), Cattrysse et al. (1990)), Particle Swarm Optimization (Mishra et al. (2016)) show great potential for solving a multitude of LSP variants.

2.3. Collaboration Frameworks

The Cambridge Dictionary defines a collaboration as "the situation of two or more people working together to create or achieve the same thing.[1]" Different authors provide a very

[1] https://dictionary.cambridge.org/dictionary/english/collaboration

similar definition for collaborations in a supply chain network, e.g. Simatupang and Sridharan (2002) describe it as a joint effort to plan and perform supply chain operations to achieve better results than would be possible independently. Other definitions include Narus & Anderson (1996), with a focus on the sharing of resources to satisfy customer needs, or Lambert et al. (1999) describing it as a relationship to share risks and rewards, leading to a better performance than the partners could achieve individually. As is the case in many other studies, the terms collaboration and cooperation will be used interchangeably throughout this thesis.

A collaboration is driven by an incentive, usually in the form of cost reductions in one way or another; be it by moving the production of goods to a collaborator with better per-unit costs, batching orders to better utilize Economic Order Quantities (EOQ), lowering inventory levels or by better utilization of a players resources, to name a few. Additionally, Morash et al. (1996) name improved reliability in completing orders, delivery speed, more widespread distribution and better pre- & post-sale customer service as some easily overlooked advantages.

Simatupang and Sridharan (2002) differentiate between three types of collaborations in supply chain networks – vertical, horizontal and lateral. When agents on different levels within a supply chain collaborate it is called vertical collaboration. Horizontal collaboration has two or more organizations that are on the same level of a supply chain – often competitors – engage in cooperation. This could for example come in the form of sharing facilities such as warehouses or vehicles and machines that are not fully utilized. The third kind is lateral collaboration, which aims to combine elements of both vertical and horizontal

collaboration, a demonstration of this is shown in Figure 1 (cf. Visser (2007) p.7). The type of collaboration that will be explored in this thesis is a horizontal collaboration between suppliers/shippers.

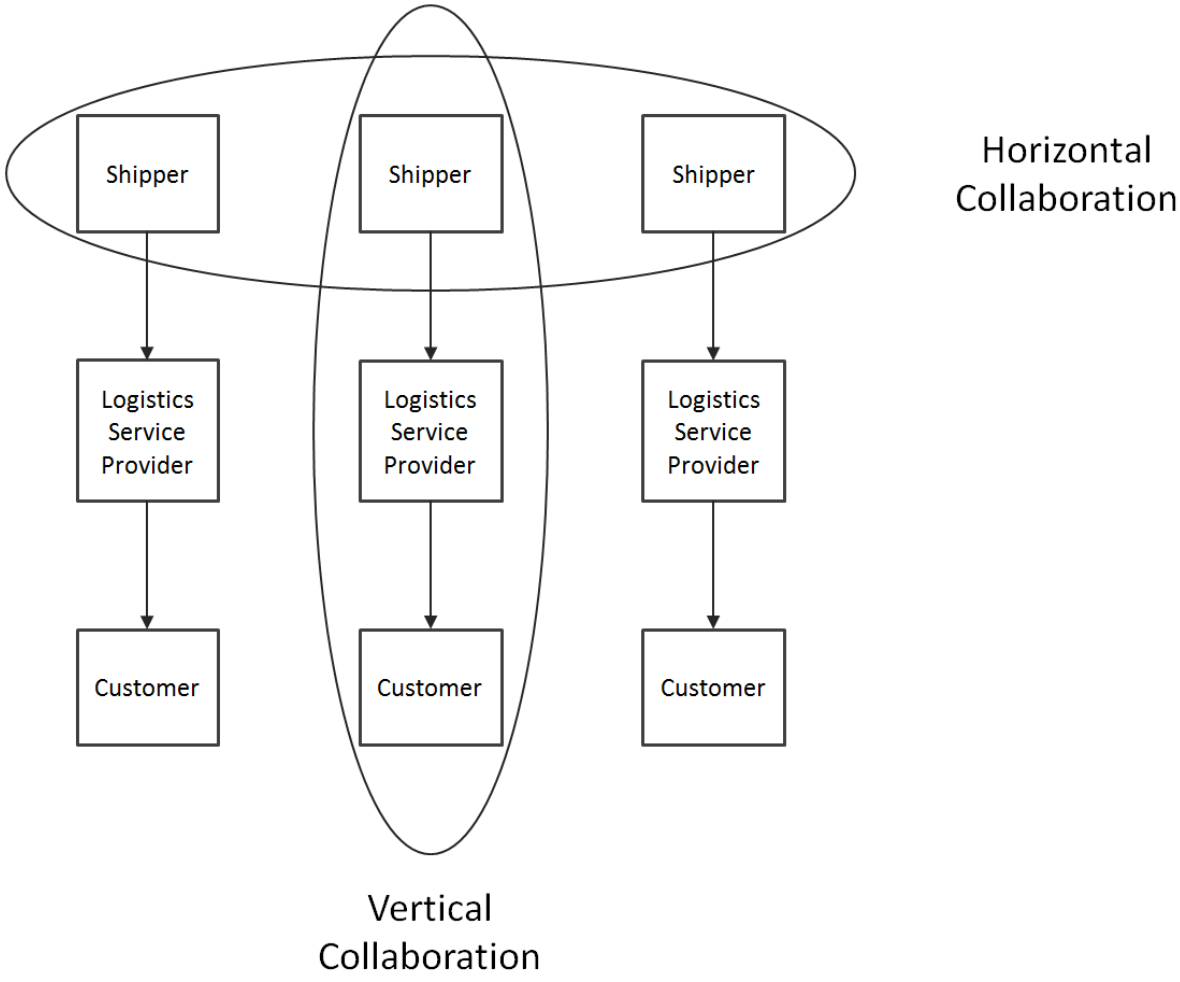

Figure 1: A combination of horizontal and vertical collaboration

Another important distinction lies in how the collaboration conducts affairs. Centralized planning systems have a central entity that decides how the cooperating companies should act. This could be either a third party or a decision making tool or platform. For this it obviously requires information from everyone involved, which, unfortunately, is one of the biggest hindrances of this approach as most private companies would want to share as little information as possible, especially when collaborating with competitors. Decentralized

planning systems on the other hand allow actors to share much less data. This could be handled individually between partners, or, again, via a central authority which only gets part of the information. Another noteworthy method are auction-based decentralized systems. Here, requests may be pooled for collaborators to bid on. This mechanism offers the possibility to share a collaborators' preferences, while not giving away any specific information. Undoubtedly, auctions show a lot more potential than non-auction-based methods, though they come at the cost of being more complex to solve. For a more in-depth look into the game theoretical aspects, we refer to Meca & Timmer (2008).

3. Problem description

The LP model of the lotsizing problem will be described in this section, with each subsection showcasing a different extension of the baseline Capacitated Lotsizing Problem (CLSP). The formulations shown below were taken from different sources but have been normalized to feature the same variable notations.

3.1. The Basic Capacitated Lotsizing Problem with Setup States

Fleischmann & Meyr (1997) define the General Lotsizing Problem as *"determining continuous lotsizes of several products and scheduling them on a single machine subject to capacity constraints. Deterministic, dynamic demands, given over a finite planning horizon, are to be met without backlogging so as to minimize inventory holding costs and […] setup costs."*

From this definition we can determine the attributes of the most basic CLSP: there is a fixed number of items $i \in I$ for which we need to determine the production amounts in each period $t \in T$. This production quantity is called q_{it}, the inventory level at the end of the period is I_{it} and the demand of each product is d_{it}. Each product incurs holding costs hc_i and setup costs sc_i. Setup states y_{it} show whether a product is ready to be produced i.e. whether the machine has been set up to produce a good i. Lastly, the producing machine has an amount of available capacity C_t in every period. The CLSP can be formulated as (cf. Chen (2015), p.27):

$$min\ Z = \sum_i^I \sum_t^T hc_i * I_{it} + \sum_i^I \sum_t^T sc_i * y_{it} \qquad (1a)$$

$$\sum_i^I q_{it} \leq C_t \qquad \forall\ i \in I,\ t \in T \qquad (2a)$$

$$q_{it} \leq y_{it} * M \qquad \forall\, i \in I,\ t \in T \qquad (3a)$$

$$I_{it} = I_{i,t-1} + q_{it} - d_{it} \qquad \forall\, i \in I,\ t \in T \qquad (4a)$$

$$q_{it}, I_{it} \geq 0 \qquad \forall\, i \in I,\ t \in T \qquad (5a)$$

$$y_{it} \in \{0, 1\} \qquad \forall\, i \in I,\ t \in T \qquad (6a)$$

The objective function sums up inventory and setup costs, just as defined by Fleischmann & Meyr. Constraints (2a) ensure that the capacity is not exceeded in any period. It can easily be extended by a factor to simulate different capacity requirements for the different items, though for simplicity it is omitted here. Constraints (3a) make production of i in t impossible if the respective setup variable has not been set to 1, M serves as a very large number to accommodate any amount of production. Constraints (4a) are inventory balance constraints, period t-1 for t = 1 implicitly represents the starting inventory. Lastly, constraints (5a) ensure non-negativity for production and inventory variables. These also prevent the possibility of backordering, which would make the problem a lot harder to solve (Absi et. al (2013)). Constraints (6a) show the binary nature of the setup variables.

Note, that for the most simplistic CLSP setup costs may be omitted, but as it is very rare to find a problem in the real world where setup costs are not present as well as it being the simplest concept to grasp mathematically, nigh all papers factor them in. Without setup states, the model only changes marginally: Constraints (3a) and (6a) are omitted and the objective function (1a) gets reduced down to the first term.

While this LP model does not appear very complex, it is already NP-hard (Karimi et al. (2003), p.367). It is therefore imperative that efficient solution methods are found to make good

results possible within a reasonable timeframe. This only becomes more evident when considering the extensions shown in the following sections.

3.2. Setup-Carryover States

This is a direct extension of the previous section and is also very simple conceptually. If a machine is setup at the end of period 1 to produce a good i and production of that same product continues immediately at the start of period 2 it would be nonsensical to disassemble that setup state in-between periods only to set it up again immediately. By storing the setup state at the end of the period and comparing it to the setup state at the start of the following one we can keep the setup state and in doing so prevent paying setup costs twice, this is embodied in the setup carryover decision variable z_{it}. The model for this extension looks as follows (cf. Chen (2015), p.28):

$$min\ Z = \sum_i^I \sum_t^T hc_i * I_{it} + \sum_i^I \sum_t^T sc_i * (y_{it} - z_{it}) \quad (1b)$$

$$\sum_i^I q_{it} \leq C_t \quad \forall\ t \quad (2b)$$

$$q_{it} \leq y_{it} * M \quad \forall\ i,t \quad (3b)$$

$$I_{it} = I_{i,t-1} + q_{it} - d_{it} \quad \forall\ i,t \quad (4b)$$

$$q_{it}, I_{it} \geq 0 \quad \forall\ i,t \quad (5b)$$

$$y_{it}, z_{it} \in \{0,1\} \quad \forall\ i,t \quad (6b)$$

$$\sum_i^I z_{it} \leq 1 \quad \forall\ t \quad (7b)$$

$$2z_{it} = y_{it} + y_{i,t-1} \quad \forall\ i,t \quad (8b)$$

$$z_{i1} = 0 \quad \forall\ i \quad (9b)$$

$$N_k * (2 - z_{it} - z_{i,t+1}) + 1 \geq \sum_{j \in N_k} y_{jt} \qquad \forall\, i, t \qquad \text{(10b)}$$

Constraints (2b)-(6b) remain unchanged from (2a)-(6a) except for the addition of z_{it} as a binary variable. The objective function (1b) now incurs setup costs only if the setup state at the end of the previous period was 0. Constraints (7b) show that only one product may be carried over to the next period (In certain permutations of this problem, it may be possible to do so with multiple products e.g. when there are multiple identical machines. In such a case, (7b) would hold for one such machine). Constraints (8b) indicate a carryover state can only be 1 if the setup state of both the current and the previous period were 1 as well. Because of this relation, z_{i1} can never be 1, as indicated by (9b). Constraints (10b) deal with the case of multi-period setup carryover, that is if a product was setup at the end of period $t-1$, was active through the entirety of period t and still continues in period $t+1$. N_k denotes the number products that are able to be produced on machine (or with resource) k, therefore the sum of setup states in a period on that machine (resource) may not exceed the number of machines (resources) as that would cause an interruption in the setup state. The term "resource" should not be mistaken with units of raw material but rather means of production i.e. machines, work groups, facilities, etc.

3.3. Multi-Level Product Structure

More often than not, products cannot be created from one production step alone. Succeeding processes take the unfinished goods of a previous step as an input. To model this, a model requires a successor/predecessor matrix. Γ_i^+ is the set of all direct successors of i, whereas Γ_i^- is the set of all direct predecessors of i. This leads to a distinction between items that have one or more successors and items that do not. The second category (called final products) does not have endogenous demand. Additionally, due to how the underlying

instance sets are structured, only final products have exogenous demand. We therefore denote the set of items i where $\Gamma_i^+ = \emptyset$ as the set of final products E. The resulting model changes the basic CLSP as follows (cf. Buer et. al (2015), pp. 327):

$$min\ Z = \sum_i^I \sum_t^T hc_i * I_{it} \qquad (1c)$$

$$\sum_i^I q_{it} \leq C_t \qquad \forall\ t \qquad (2c)$$

$$I_{it} = I_{i,t-1} + q_{it} - d_{it} \qquad \forall\ i \in E, t \qquad (3c)$$

$$I_{it} = I_{i,t-1} + q_{it} - \sum_{j\ \epsilon\ \Gamma_i^+} q_{jt} \qquad \forall\ i \in I \backslash E, t \qquad (4c)$$

$$q_{it}, I_{it}\ \geq 0 \qquad \forall\ i \in I, t \qquad (5c)$$

The objective function (1c) only features holding costs compared to (1a). Constraints (2c) and (5c) remain unchanged from the basic model counterparts (2a) and (5a). Constraints (3c) also remain unchanged for the final products. Inventory balance constraints (4c) are introduced which consider the implicit demand by all successors. For the case of a successor requiring more than one unit of its predecessors, the factor p_{ij} may be added within the sum in (4c), which describes the number of units of good j needed to produce one unit of i.

3.4. Collaborative or Multi-Agent CLSP

Over the last decade, cooperation between manufacturers has become much more of an important topic of research and this trend is only going to increase in the future. There are centralized and decentralized approaches to this topic, typically this changes how much information each individual agent has on other agents. In a decentralized problem, agents

would calculate a feasible plan based on information within their company only, whereas a centralized problem has a separate entity that has all information and decides on the best plan for all agents. Of course, any compromise between these two settings is possible. For our problem, a centralized approach was chosen (Centralized Collaborative CLSP or CCCLSP).

Modelling this requires some substantial changes to the CLSP. Most notably, a new parameter depicting each agent $a \in A$. Additionally, one of the most crucial differences is the importance of production costs. Previously, production costs were considered as unavoidable since all demand needed to be fulfilled anyway. Due to agents having different setups/supply chains/facilities we now need to pay closer attention to them. To benefit from lower costs for any particular product, agents need to cooperate. This comes in the form of transshipments of goods, such that an agent a1 may produce good i1 for agent a2, who needs that good as an input for the production of a good i2. It is assumed that all agents can produce all goods. Due to the centralized nature of this approach, demand is shared across all agents.

The resulting LP model looks as follows (cf. Drechsel & Kimms (2011), p. 2645):

pc_{ia} production costs of product i for agent a

A_{abit} amount of product i shipped from agent a to b in period t

$$min\ Z = \sum_i^I \sum_t^T \sum_a^A (hc_{ia} * I_{ita} + pc_{ia} * q_{ita}) \quad (1d)$$

$$\sum_i^I q_{ita} \leq C_{ta} \qquad \forall\ a, t \quad (2d)$$

$$\sum_a^A I_{ita} = \sum_a^A (I_{1,t-1,a} + q_{ita} - \sum_b^A (A_{abit} + A_{bait})) - d_{it} \qquad \forall\ a, i, t \quad (3d)$$

$$q_{ita}, I_{ita} \geq 0 \qquad \forall\, a, i, t \qquad (4d)$$

$$\begin{cases} A_{abit} \geq 0 & a \neq b \\ A_{abit} = 0 & else \end{cases} \qquad \forall\, a, b \in A, i, t \qquad (5d)$$

As mentioned above the objective function (1d) now includes production costs, constraints (2d) and (4d) remain unchanged except for the additional parameter for the agent. Since demand is shared, constraints (3d) now consist of the sum of all agents' inventory levels. We also need to consider transshipments between agents here and do so with the sum over b. Constraints (5d) are non-negativity constraints for transshipment amounts, transports to oneself are prohibited. Upper bounds for these variables are set implicitly in (3d). Transport costs may also be added to the objective function without needing to change the model in any way.

3.5. The Centralized Collaborative Multi-Level Capacitated LSP (CCMLCLSP)

When combining all of the above mentioned extensions the resulting problem comes relatively close to a full scale real world problem, at the very least when compared to other literature in the field. The model looks as follows:

$$min\ Z = \sum_i^I \sum_t^T \sum_a^A \left(hc_{ia} * I_{ita} + sc_{ia} * (y_{ita} - z_{ita})\right) \qquad (1)$$

$$\sum_i^I q_{ita} \leq C_{ta} \qquad \forall\, a, t \qquad (2)$$

$$q_{ita} \leq y_{ita} * M \qquad \forall\, a, i, t \qquad (3)$$

$$I_{ita} = I_{i,t-1,a} + q_{ita} - \sum_{j \in \Gamma_i} q_{jta} - \sum_b^A (A_{abit} + A_{bait}) \qquad \forall\, a, t, i \in I \backslash E \qquad (4)$$

$$\sum_a^A I_{ita} = \sum_a^A (I_{i,t-1,a} + q_{ita} - \sum_b^A (A_{abit} + A_{bait})) - d_{it} \qquad \forall\, a, t, i \in E \qquad (5)$$

$$\sum_i^I z_{ita} \leq 1 \qquad \forall\, a, t \qquad (6)$$

$$2z_{ita} = y_{ita} + y_{i,t-1,a} \qquad \forall\, a, i, t \quad (7)$$

$$z_{i1a} = 0 \qquad \forall\, a, i \quad (8)$$

$$(N - |R| + |R_a|) * (2 - z_{ita} + z_{i,t+1,a}) + 1 \geq \sum_{j \in N \setminus (R \setminus R_a)} y_{jta} \qquad \forall\, a, i, t \quad (9)$$

$$q_{ita}, I_{ita} \geq 0 \qquad \forall\, a, i, t \quad (10)$$

$$y_{ita}, z_{ita} \in \{0, 1\} \qquad \forall\, a, i, t \quad (11)$$

$$\begin{cases} A_{abit} \geq 0 & a \neq b \\ A_{abit} = 0 & \text{else} \end{cases} \qquad \forall\, a, b \in A, i, t \quad (12)$$

To recapitulate from earlier sections: The objective function (1) sums up holding costs and setup costs and respects setup carryover states. Note, that despite previously mentioning the importance of production costs for this problem, they were omitted to reduce the complexity of the problem. Constraints (2) are capacity constraints for each agent in each period; constraints (3) make production only possible, when the respective setup variable is set to 1. Constraints (4) and (5) are inventory balance constraints and include transshipments between agents, predecessor relations and have a shared demand over all agents for all items $i \in E$, no other items have exogenous demand in our instances. Constraints (6)-(9) set the boundaries for setup carryover variables as described in Section 3.2. R represents the set of all items i which are assigned to a specific agent, with R_a being the set of items assigned to agent a. This is information given by the DULR instances, which will be discussed further in Section 4.1. Therefore, $N \setminus (R \setminus R_a)$ represents all items that are available to an agent a. The term $(N - |R| + |R_a|)$ represents the maximum number of items an agent could theoretically produce in a period. Constraints (10)-(12) define the decision variables as non-negative or binary, respectively.

Lastly, one new set of constraints is introduced alongside a new variable o_{ia} denoting whether a variable is open for production for an agent. Its sole purpose is to forbid production from an agent a if it was fixed to agent a'. Whether an item is assigned to an agent is dictated by the instance data and is not changed at any point. This relation can be seen in (13)-(14):

$$y_{ita} \leq o_{ia} \qquad \forall\, a, i, t \qquad (13)$$

$$\begin{cases} o_{ia} = 0 & \text{if } i \text{ was assigned to an agent } a' \\ o_{ia} = 1 & \text{else} \end{cases} \qquad \forall\, a, i \qquad (14)$$

4. Instance Sets

The instances used for computation were based on the ones provided by Buer et. al (2015)[2], which are for the distributed multi-level uncapacitated lotsizing problem. There are 96 small instances, 40 medium instances and 40 large instances. Each instance can be solved for 2 and 5 agents, leaving a total of 352 instances. The set of small instances is denoted as *s1-s96*, the medium instances as *m1-m40* and the large instances as *l1-l40*. These instances need to be adapted, however, to fit our problem type. The parameters that need to be added are capacities for each agent (for the cases of both 2 and 5 agents). The following parameters were also considered, but ultimately not included to reduce complexity: production, setup and transportation times; production and transportation costs; transport lead times; production lead times.

Additionally, each agent is assigned one "random" product (four products for medium, eight for large instances) that only they may produce denoted as R_a for each agent a.

[2] Available at http://www.dmlulsp.com/ (last accessed 17.12.2019)

4.1. Description of DULR Instances

Small instances[3]: These instances all have 5 items and 12 periods. Every group of 4 instances (e.g. instances s1-s4 or s17-s20) shares the exact same demand structure, and there are 6 different demand structures which repeat, as shown in Figure 2. The total demand is the same for all 96 instances. Within each group of four, the same four product structures are used, i.e. *s1, s5, s9*, etc. share the first product structure, *s2, s6*, etc. share the second product structure and so on. Additionally, the groups which share demand structures also split holding and setup costs among them: s1-4 and s25-28 share setup costs, s1-4 and s49-52 share holding costs, s25-28 and s73-76 share holding costs, s49-52 and s73-76 setup costs. Essentially, there are two baseline holding and setup cost sets for each group[4].

Medium instances[5]: These instances can be split into ones with 50 or 40 items and ones with 12 or 24 periods, and there are 10 instances for each permutation of these parameters, as depicted in Figure 3. As before, only a very limited number of product structures are used throughout: every odd numbered instance has the same build, and every even numbered instance does as well for all instances with 50 items (m1-20). The ones with 40 items

Instance	1-4	5-8	9-12	13-16	17-20	21-24	25-28	29-32	33-36	37-40	41-44	45-48	49-52	53-56	57-60	61-64	65-68	69-72	73-76	77-80	81-84	85-88	89-92	93-96
1	80	125	50	80	10	230	80	125	50	80	10	230	80	125	50	80	10	230	80	125	50	80	10	230
2	100	50	80	100	10	180	100	50	80	100	10	180	100	50	80	100	10	180	100	50	80	100	10	180
3	125	100	180	80	15	10	125	100	180	80	15	10	125	100	180	80	15	10	125	100	180	80	15	10
4	100	50	80	180	20	20	100	50	80	180	20	20	100	50	80	180	20	20	100	50	80	180	20	20
5	50	100	0	95	70	70	50	100	0	95	70	70	50	100	0	95	70	70	50	100	0	95	70	70
6	50	100	0	10	180	40	50	100	0	10	180	40	50	100	0	10	180	40	50	100	0	10	180	40
7	100	125	180	150	250	0	100	125	180	150	250	0	100	125	180	150	250	0	100	125	180	150	250	0
8	125	125	150	50	270	15	125	125	150	50	270	15	125	125	150	50	270	15	125	125	150	50	270	15
9	125	80	10	0	230	10	125	80	10	0	230	10	125	80	10	0	230	10	125	80	10	0	230	10
10	100	100	100	180	40	250	100	100	100	180	40	250	100	100	100	180	40	250	100	100	100	180	40	250
11	50	50	180	0	0	10	50	50	180	0	0	10	50	50	180	0	0	10	50	50	180	0	0	10
12	100	100	95	180	10	270	100	100	95	180	10	270	100	100	95	180	10	270	100	100	95	180	10	270

Figure 2: Demand Structure of the Small Instances

[3] Originally constructed by Coleman and McKnew (1991)

[4] Note, that this applies only to the first agent of each instance; the other four agents deviate slightly around that baseline, assumedly randomly.

[5] Originally constructed by Dellaert and Jeunet (2000a)

(m21-m40) work differently, though, as they allow for intermediate products to be used for multiple successor products. Additionally, every even numbered instance with 40 products has three end products instead of just one. Again, each odd numbered instance shares the same product structure, as well as every even numbered one. Unlike the small instances before, they do not share demand structures, holding or setup costs, making every instance unique in that regard.

Instances	Items	Periods	multiple successors possible?	Enditems
m1-10	50	12	no	1
m11-20	50	24	no	1
m21-30	40	12	yes	1 or 3, alternating
m31-40	40	24	yes	1 or 3, alternating
l1-5	500	36	no	1
l6-20	500	36	yes	5
l21-25	500	52	no	1
l26-40	500	52	yes	5

Figure 3: Analysis of Instances; Classes m and l

Large instances[5]: All instances in this category have 500 items and either 36 periods for the first half or 52 periods for the latter. Instances l1-5 and l21-25 feature one end product, all others have five. All instances with multiple end products contain multiple successor relations. There are 20 different product structures in total and as many demand structures, i.e. l1 and l21 are identical in this regard; only l21 has an additional 16 periods.

4.2. Extension of DULR Instance Sets

As documentation of how Buer et. al constructed their instances is no longer available under the provided link, as well as the different instance classes being constructed by different

authors, we will now describe the underlying assumptions for our extension of these instances.

There are four different types of structures used in our extension denoted as Predictive, Frontloaded, Seasonal and Constant. The type was decided at random for each instance using a uniform distribution; probabilities for each type are listed in Figure 4. The first step for all of these scenarios consists of calculating the total required capacity in each period and in total. From that, we apply a uniformly randomly selected factor (Total Capacity Factor or TCF) between 150% and 350% for the small instances (between 120% and 250% for medium and large instances) to this amount to get the total available capacity for all agents, rounded up to increments of 100. Capacity is split among agents in each period using uniformly distributed random numbers between 35% and 65% for the case of two agents. For the case of five agents, normally distributed numbers are generated with mean 20 and a variance of 5 for the first three agents, while the last two agents share the remainder with a split that is, again, uniformly distributed between 40% and 60%. These values are rounded to increments of 10 for both cases. Should rounding differences occur for the total, randomly selected elements will be altered in increments of 10, to get nice round numbers.

Predictive: This scenario assumes that a company already has an idea of how demand will be distributed over the planning horizon. The TCF is applied to each period's total demand, which ensures that enough capacity is available for a lot-for-lot solution. The only compromise that was made for this scenario was for the large instances; here most early periods have zero or extremely little production comparatively (sometimes less than 0.0001% - though not zero - of the total production for the first 13 periods in some

Type	Target Distribution	Actual distribution		
		small	medium	large
Predictive	50%	50.0%	45.0%	47.5%
Frontloaded	20%	18.8%	25.0%	20.0%
Seasonal	15%	17.7%	15.0%	17.5%
Constant	15%	13.5%	15.0%	15.0%

Figure 4: Distribution of the Scenarios that were applied to the Instances

instances). Therefore the production was manually set to at least be able to pool some periods' demand, to enable some meaningful choices.

Frontloaded: Here, most of the available capacity is assigned in the first 8 periods of a 12 period instance, for planning horizons of 24 and 36 periods it was interpreted as a yearly cycle, where each year received an equal amount of capacity. The final four periods of a year receive between 0 and 25% of the total yearly capacity, between 30 and 55% are allocated to the first three periods and the rest goes to the remaining 5 periods, all normally distributed. Within each of these blocks, demand is split among each period in a similar fashion as the split between agents, as explained above.

Seasonal: The aim of this scenario was to represent industries where seasonal stopping of production is necessary, fiscally sensible or regulated by law. Similar to the previous variant, instances with 24 or 36 periods or considered as planning horizons of multiple years, with each year having an equal amount of available capacity. These years are split into four quarters, one of which will have a capacity of zero. This can apply to either Q2, Q3 or Q4 (50%, 40% and 10%, respectively). Q1 is exempt from this as it would lead to infeasibilities. One quarter receives between 10-25% of the yearly capacity, another quarter 45-70% and the last quarter gets the remainder. The allocation is random, though there is a bias to having the highest capacity before the quarter with zero resources. Within each quarter, two

periods get between 25-40% of the quarterly capacity, while the third period receives the remainder.

Constant: This scenario is characterized by equal production capacities in every period and for each agent. The total capacity is split evenly among all periods (though rounding differences may occur). Capacity is split among agents only once, instead of in each period, and then applied to every period to reflect this very rigid system.

5. Solution Approach

Our proposed solution method is based on Chen (2015) and is a Fix-and-Optimize approach with random elements, introduced by Helber & Sahling (2010) and is typically used on mixed integer problems (MIP). As the name implies, Fix-and-Optimize sets a set of (binary/integer) variables to a certain value and solves the problem for the remaining variables using standard optimization. The selection criterion for this subset originally consisted of product oriented, resource oriented or process oriented selection rules. Chen (2015) combined these rules to form a new „hybrid“ rule which incorporates some randomness in the algorithm, dubbed as interrelatedness oriented.

5.1. Interrelatedness in the CCMLCLSP

The definition of the subproblem to be examined is based on the binary variables for setup and setup carryover (y_{ita} and z_{ita}). A variable is considered related to another variable y_{ita} if they interact in one of the main problems constraints, which is the case in constraints (6), (7) and (9). Were y_{ita} to change, it would be possible that $y_{i,t-1,a}$ and $y_{i,t+1,a}$ may change in turn due to (7). Additionally, these variables are related between all agents $y_{i,t-1,a'}$, $y_{ita'}$and $y_{i,t+1,a'}$ as it may be prudent for a different agent to produce a given product, this link can be derived via the link of (3) and (4) or (3) and (5), respectively. Lastly, a change in y_{ita} may affect the predecessors and successors of i in $y_{i'ta}$ where $i' \epsilon \Gamma_i^-$ and$i' \epsilon \Gamma_i^+$ through the inventory balance constraints (4) and (5), implicitly representing the bill of materials. Via the same reasoning as before, $y_{i'ta'}$ are related to it. It is related to carryover variables z_{ita}, $z_{i,t-1,a}$ and $z_{i,t+1,a}$ as well as $z_{ita'}$, $z_{i,t-1,a'}$ and $z_{i,t+1,a'}$ via constraints (7) and to $z_{i',t-1,a}$, $z_{i'ta}$ and $z_{i',t+1,a}$ as well as $z_{i',t-1,a'}$, $z_{i'ta'}$and $z_{i',t+1,a'}$ through constraints (9). However,

this would apply to every single i, which would turn the hybrid selection rule into a process oriented one. We thus arbitraily limit the relatedness to direct successors and predecessors to preserve this characteristic and to limit the number of variables in the subproblem.

For the setup carryover variable z_{ita}, it is connected to $z_{i,t-1,a}$ and $z_{i,t+1,a}$ via constraints (9) as well as y_{jta} for all j ∈ $N \setminus (R \setminus R_a)$. It is connected to $z_{i'ta}$ where $i' \in \Gamma_i^-$ and$i' \in \Gamma_i^+$ through the link from (4) to (3) to (7) and to $z_{i'ta'}$ to allow for switching the production to another agent.

The underlying instances also have items that are assigned to a specific agent, denoted as R_a. To take this into account, all items that are assigned to the randomly selected agent are declared related as well, to give the algorithm a bit more flexibility. This adds y_{jta}, z_{jta}, $z_{j,t-1,a}$ and $z_{j,t+1,a}$ where $j \in R_a$ to the related variables. For these items it is not necessary to consider any other agent as no agent other than a may produce these items, as given from the instances.

All of the above described connections form the set of related variables IR(y_{ita}) to a variable y_{ita}. This can be referred to as relatedness level 1, or 1-interrelated. The definition of IR(y_{ita}) is given by:

$$\text{IR}(y_{ita}) = \{y_{i,t-1,a'}, y_{ita'}, y_{i,t+1,a'} | a' \in A\} \cup \{y_{i'ta'} | i' \in \Gamma_i^- \cup \Gamma_i^+, a' \in A\} \cup \{y_{i'ta} | i' \in R_a\} \cup \{z_{i,t-1,a'}, z_{ita'}, z_{i,t+1,a'} | a' \in A\} \cup \{z_{i',t-1,a'}, z_{i'ta'}, z_{i',t+1,a'} | i' \in \Gamma_i^- \cup \Gamma_i^+, a' \in A\} \cup \{z_{i',t-1,a}, z_{i'ta}, z_{i',t+1,a} | i' \in R_a\}$$

$$IR(z_{ita}) = \{y_{i,t-1,a'}, y_{ita'}, y_{i,t+1,a'}\} \cup \{z_{i,t-1,a}, z_{ita}, z_{i,t+1,a}\} \cup \{z_{i,t-1,a}, z_{ita}, z_{i,t+1,a} | i' \in \Gamma_i^- \cup \Gamma_i^+ \cup R_a\}$$

By default, a variable is 1-interrelated with itself. For any integer $l \geq 1$, we say two setup variables y_{ita} and $y_{i't'a'}$ are l-interrelated if and only if there is a series of setup variables $y_{i_j t_j a_j}$ (j = 0, 1,…, l) such that

1) $(i_0, t_0, a_0) = (i, t, a)$ and $(i_l, t_l, a_l) = (i', t', a')$
2) $y_{i_j t_j a_j}$ and $y_{i_{j+1}, t_{j+1}, a_{j+1}}$ are 1-interrelated for $0 \leq j \leq l - 1$

We can therefore say that two variables that are l-interrelated are also interrelated for $l' > l$. Consequently, relatedness is symmetric, reflexive and transitive. Any variable that is

As an example, let us look at the following example: Suppose we have an instance with 2 agents, 5 items and 12 periods (this is identical to the S2 instance set). The BOM structure is shown in Figure 5, product 4 is assigned to agent 2 and product 5 is assigned to agent 1.

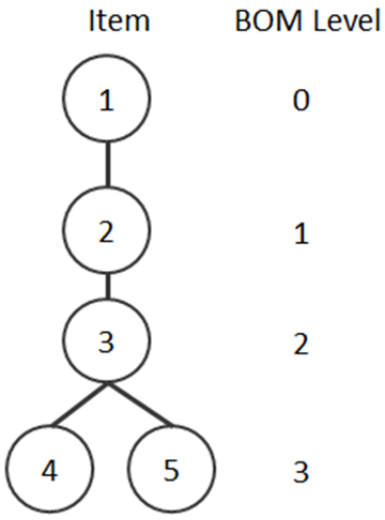

Figure 5: Relatedness Example - BOM Structure

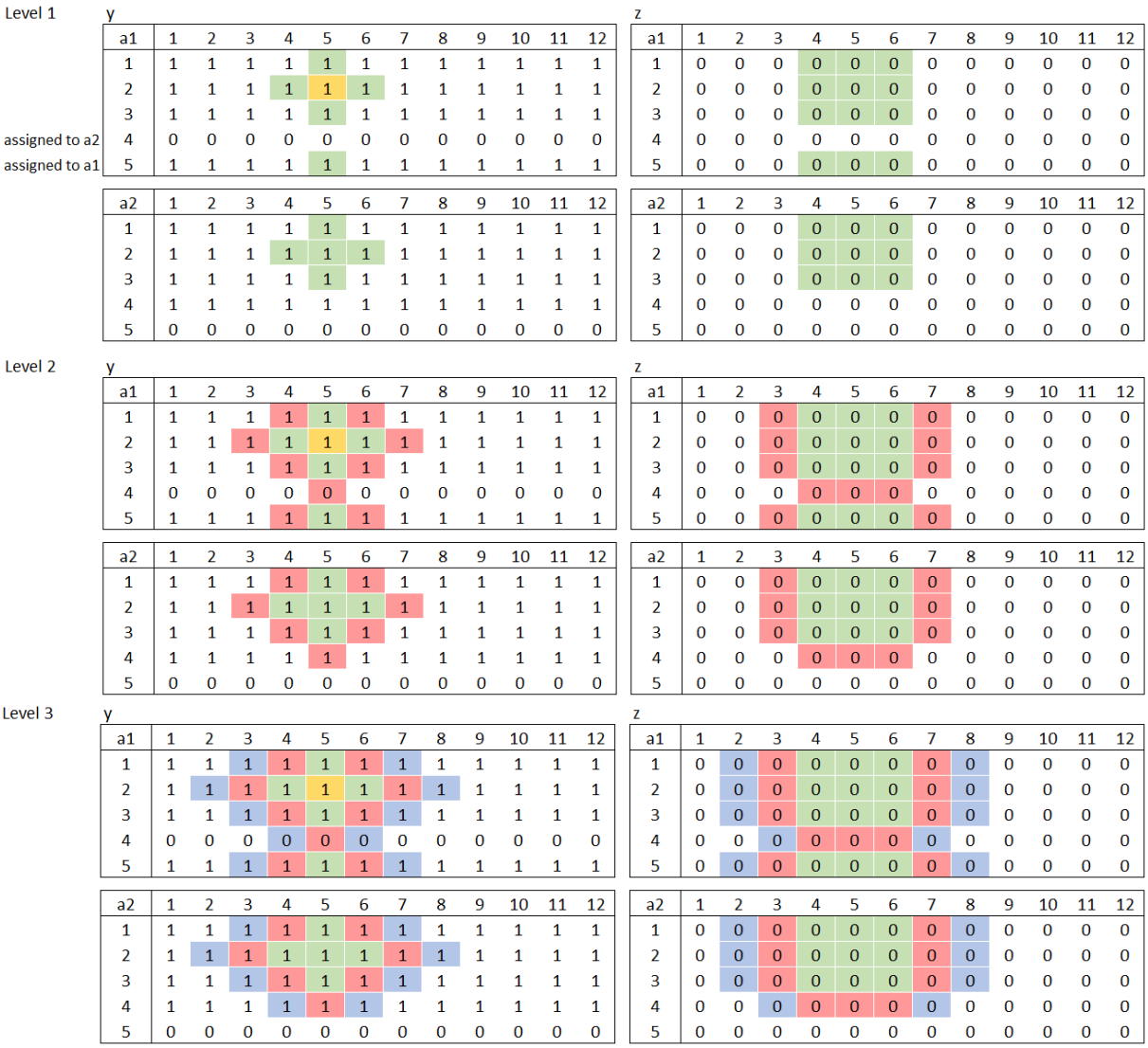

Figure 6: Demonstration of Interrelated Variables

The randomly selected variable is y_{251}. Figure 6 shows all related variables to it with relatedness level 1 in green (y_{251} is yellow for visibility, but is also 1-interrelated), relatedness level 2 is shown in red, and level 3 in blue. Again, any variable that is l-interrelated with the selected variable is automatically interrelated on any higher level.

5.2. Fix-and-Optimize Algorithm for the CCMLCLSP

Our presented algorithm solves a series of smaller subproblems, as was the case in Helber & Sahling (2010) and Chen (2015) among many others. We attain a feasible initial solution by setting all binary decision variables y and z to 1 for all a, i and t. From there, a subproblem

with a randomly selected starting point, a combination of uniformly distributed a, i and t values, is chosen and solved. Additionally, between 0 and 5 (or fewer if the number of potential available items is too small) other items are included in this subproblem, this number is denoted as *randomNumberItems* in Algorithm 2. How many and which specific items are included is decided on at the start of every single iteration. The relatedness level l directly correlates to the size of the subproblem, it should be lowered if the computation time is too high, though in our implementation a time limit of 15 seconds per subproblem was implemented. This may lead to less than optimal solutions, but testing has shown that it improves the computation time drastically, as the number of real variables is still quite high ($N * T * A * 2 + N * T * A^2$).

A solution is accepted if it was found to be feasible, which means worse solutions are considered. A worse solution is accepted a certain number of times before returning the best known solution, called *consecWorseSolutions* in Algorithm 1 This helps the program reach different neighborhoods, though if no new best solution is found after a few iterations the algorithm will back up to the best known solution. If no new best solution was found after a certain number of iterations n the algorithm will terminate.

Algorithm 1: Fix-and-Optimize for the CCMLCLSP

```
    Input: instance data, l, n
1.  Iter, consecWorseSolutions, consecInfeasSolutions ← 0
2.  Find an initial feasible solution by setting y_ita and z_ita = 1 for all a, i, t
3.  bestSolution, secondbestSolution, incumbentSolution ← initialSolution
4.  while Iter < n do
5.      Select a combination of (i, t, a) from I * T * A randomly
6.      Define related variables based on selection
7.      Define additional related variables randomly          } see Algorithm 2
8.      for each non-related variable fix its current value
9.      Set time limit for subproblem = 15 seconds
10.     currentSolution ← Solve(subproblem)
11.     if currentSolution feasible then
12.         if currentSolution < bestSolution then
13.             secondbestSolution ← bestSolution
14.             bestSolution ← currentSolution
15.             Iter, consecInfeasSolutions, consecWorseSolutions ← 0
16.         else
17.             Iter ← Iter +1
18.             consecInfeasSolutions ← 0
19.             consecWorseSolutions ← consecWorseSolutions+1
20.                 if consecWorseSolutions ≥ 5 then
21.                     currentSolution ← bestSolution
22.                     consecWorseSolutions ← 0
23.                 end if
24.         end if
25.     else
26.         currentSolution ← secondbestSolution
27.         consecInfeasSolutions ← consecInfeasSolutions+1
28.         if consecInfeasSolutions ≥ 10 then
29.             currentSolution ← bestSolution
30.             consecInfeasSolutions ← 0
31.         end if
32.     end if
33. end while
34. bestSolution ← repair(bestSolution)
35. return bestSolution
```

Infeasible solutions do not count against the iteration limit, which helps solution quality as the algorithm is not prematurely ended and the solution is not accepted either way. The parameter *consecInfeasSolutions* describes the limit of how many infeasible solutions in a row are tolearated before the algorithm resets to the best known solution. Usually this precaution performs well enough that the program does not drive itself into a corner,

however, from testing we noticed that the algorithm may sometimes become ”stuck“ in a certain solution. This happens when a relatively optimal solution has been found that lies within a neighborhood with a particularly small number of feasible solutions. In such a case, the algorithm will still terminate eventually, but it bloats the computation time decently and the iteration count significantly without providing any improvements. Therefore, runs with such cases were not taken out of our results, as it is a behaviour of our implementation that should be improved upon in future research. Generally, this is much more likely to happen with larger problems i.e. medium instances with 5 agents. This aspect will be touched upon again in Section 6. It is unclear to us, why infeasible solutions occur in the first place seeing as every iteration starts from a known feasible solution and theoretically the algorithm should return to that solution on its own. We suspect that it is a quirk of the CPLEX MIP solver, though we cannot prove it, unfortunately.

Algorithm 2: Defining Related Variables

Input: instance data, l, i, t, a
1. $related\_y_{ita}$, $related\_z_{ita}$, *numberRandomItems* ← 0
2. **For each** related variable to y_{ita} and z_{ita} set $related\_y_{ita}$, $related\_z_{ita}$= 1
3. **For each** item assigned to a set $related\_y_{ita}$, $related\_z_{ita}$ = 1
4. *numberRandomItems* ← Random(0, 5)
5. Randomly select *numberRandomItems* additional items from pool of unselected and unassigned items
6. **For each** selected item set $related\_y_{ita}$, $related\_z_{ita}$ = 1
7. **For each** l **do**
8. **For each** $related\_y_{i't'a'}$ == 1 **do**
9. Set related variables to $y_{i't'a'}$ to $related\_y_{i't'a'}$, $related\_z_{i't'a'}$= 1
10. **For each** $related\_y_{ita}$== 1 fix the value of y_{ita}
11. **For each** $related\_z_{ita}$== 1 fix the value of z_{ita}
12. **return**

Algorithm 3: Repair Function

Input: *bestSolution*
1. **For each** i, t, a **do**
2. Fix values of real variables I_{ita}, q_{ita}
2. **For each** i, t, a **do**
3. **if** y_{ita} == 1 **then**
3. Lift upper and lower bound restrictions y_{ita}, z_{ita}
4. **else** fix y_{ita} = 0
5. **end if**
5. *bestSolution* ← Solve(subproblem)
6. **return** *bestSolution*

The repair function described in Algorithm 3 is very basic and only serves to mitigate bad solutions caused by not reaching certain parts of the problem space. This may happen due to the random nature of the algorithm, as it is theoretically possible that the exact same random selection of i, t and a is chosen n times in a row. With an increasing problem size, it becomes more and more likely that some variables are never reached. To counteract this, the repair function identifies decision variables y_{ita} which have been set to 1 and lifts the upper and lower bound restrictions on them and their corresponding z_{ita} (i.e. ”unfixing“ them). Variables y_{ita} that have been set to 0 are fixed, such that the final solution may be trimmed without transforming the production plan in any way.

5.3. Heuristic Solution

The initial solution used for the algorithm is not a qualitatively good solution. This way, there is no preselection of neighborhoods and the algorithm can take its course freely. As this is a new problem definition, we want to provide an alternative solution to compare our values against.

Our approach is a myopic upstream approach based on the BOM level of each item. On each level, a single-level CLSP is solved based on the result on the previous level. Each item is

assigned randomly, such that every item is assigned to an agent. The problem formulation does not change, except for constraints (4) which are changed to apply to $\forall\, a \in A, t \in T, i \in \{I \backslash E \mid m_i \leq M\}$ where m_i represents the BOM level of item i and M is the current BOM level. In each iteration M is raised by 1 until the final level has been reached.

A basic feasibility check is performed when assigning the items. Should an agent not have the capacity to produce an equal share of items in any of the periods, an item on the lowest BOM level is pushed to another agent that has excess capacity. If no items on the lowest level are in their assignment, the second lowest level is considered, etc. This way, every agent gets a fair share of items to produce, when considering their capacities.

Capacitated multi-level problems are much more susceptible to shortages due to greedy behavior when used in this way, therefore we need to correct retroactively infeasible assignments by pushing capacity into later periods. Specifically, for an agent that does not have enough capacity to add another BOM level M+1 to their production plan, an item highest production amount is selected. In the case of a tie, an item with a lower BOM level is preferred, to prevent further infeasibilites. The shifted amount is equal to the indirect demand of the product or, if this indirect demand is already satisfied, the maximum possible amount while maintaining feasibility.

This approach ultimately leads toward a lot-for-lot solution. While this leads to feasible solutions much more consistently, it does not tend to their solution quality. It is important to note that it is still possible for the heuristic to not find a feasible solution at all. This can happen for different reasons e.g. the overall capacity is sufficient, but without cooperation

no feasible production plan exists, or at least one agent does not have enough capacity to provide the other agents with their required predecessor items. The computational results of this heuristic will be discussed in the next section.

6. Computational Results

As all instances were extended by us to fit the new problem, no reference values exist for them yet. The problem definition is sufficiently different from previous works, that it makes no sense to compare them directly.

All algorithms were coded using C++ in the Visual Studio 2007 environment and solved using the MIP solver of CPLEX. All computations were performed on a Intel Core i5-2400 CPU with 3.10 GHz and 8 GB RAM.

The parameter l depicts the relatedness level, as explained in Section 5.1, and n represents the number of iterations without improvement before the algorithm terminates. This factor is larger for the small instances, as the increase in computation time overall is negligible, as there are only a total of 5*12*2 ($N * T * A$) total variables and computes very quickly. Table 1 shows this for the calculations in Table 3, Table 4 and Table 5. Note, that the listed best times correspond to the iteration of the algorithm that gave the best value for each instance. Our implementation was capable of computing results for the large instances, however, calculation times were too high to be practical given the immense increase in the number of variables compared to the medium instances.

For the case of Table 4, average values and best values are extremely close if not identical, with notable exceptions in s24, s34 and s69. This happens because these instances are trivial in their definition. Since there are 5 items in every instance of class S and there are also 5 agents, with the every agent having *at least* one item assigned to them, the only possible outcome is that every agent has *exactly* one item assigned to them, with no other items in

Table 1: Comparison of Instance Categories
l = 2

Instance	Value (avg)	Value (best)	Time (avg)	Time (best)	Avg Iteration length (sec)	Avg Improvements found
s_2Agents	651.1	634.1	91.2	96.6	0.37	20.7
s_5Agents	268.8	267.6	20.2	21.8	0.14	5.4
m_2Agents	278,148.7	272,723.1	835.5	1,099.5	1.54	131.9
m_5Agents	251,763.0	245,641.9	4,287.7	5,536.5	5.29	123.6

$N \setminus (R \setminus R_a)$ remaining. Therefore, the optimal solution is for every agent to set up their respective item in period 1 and producing lot-for-lot as this incurs no holding costs. This also explains why the computation time is lower than for the S2 instances. However, there are cases where the algorithm is not able to find this "only sensible" solution, though during testing in 98.3% of the cases it did.

The surprising stability of the algorithm is shown in Table 1 with the average solution quality being 97.4% of the best found solution overall for the small instances with 2 Agents (S2), 99.6% for 5 Agents (S5), 98.1% for the medium instances with 2 Agents (M2) and 97.6% for 5 Agents (M5). All instances were run 10 times, except for class M5 which were run 5 times each due to the much larger computation time.

Table 2 shows the results differentiated between the different types within the medium class. As shown in Figure 3, described in Section 4.1, there is a lot more variation within the class. From this we can see how much the problem size affects computation times, though not because of longer individual iteration computation times, but rather there is much more room for improvement. These two characteristics are shown by the average number of improvements in an instance run as well as the average iteration length.

Table 2: Comparison of Different Instances within Medium Class

$l = 2$

Instance 2 Agents	Value (avg)	Value (best)	Time (avg)	Time (best)	Avg Iteration length (sec)	Avg Improvements found
m1-10	185,372.1	179,405.8	1,122.2	1,052.4	2.27	105.9
m11-20	350,996.1	341,641.1	996.7	1,844.8	1.44	198.9
m21-30 (odd)	201,286.7	199,092.7	262.9	269	0.94	66.4
m21-30 (even)	177,065.6	176,199.4	447.1	601	1.40	92.6
m31-40 (odd)	442,369.8	436,412.8	1,097.1	1,377.6	1.49	137.1
m31-40 (even)	331,731.5	327,985.8	639.1	753.6	1.08	149.8

Instance 5 Agents	Value (avg)	Value (best)	Time (avg)	Time (best)	Avg Iteration length (sec)	Avg Improvements found
m1-10	166,104.2	161,413.2	3,245.8	3,488.6	4.70	104.7
m11-20	318,989.0	306,966.1	6,001.3	9,685.9	5.67	183.2
m21-30 (odd)	184,346.9	182,233.8	2,763.0	2,539	4.45	72.0
m21-30 (even)	160,255.3	159,081.1	3,449.5	2,946	6.36	87.0
m31-40 (odd)	407,866.3	398,027.5	4,427.1	4,506.4	5.61	116.7
m31-40 (even)	291,449.4	289,034.5	5,167.8	7,951.4	5.19	137.6

Having more than one end product seems to affect computation time in an interesting manner, as shown in Table 2. When comparing instances m21-30 (odd) with m21-m30 (even), the even numbered instances (i.e. those with three end procuts) require a longer average computation time, as one would expect. However, for instances m31-40, the even numbered instances were quicker to compute on average. This is largely due to m33, where the average computation time is very distorted due to one run where the algorithm got stuck in a very confined neighborhood and had a disproportionately high number of total iterations before it terminated. A similar case can be made for m1-10, during testing multiple runs occurred where the program spent a long time trying to leave an unfavorable neighborhood. In fact, most of these cases occurred within these first 10 instances.

Table 3: Results for Small Instances with 2 Agents

n = 100, l = 2, 10 repetitions

Instance	Value (avg)	Value (best)	Time (avg)	Time (best)	Avg Iteration length	Avg Improvements found	Instance	Value (avg)	Value (best)	Time (avg)	Time (best)	Avg Iteration length	Avg Improvements found
s01	215.0	202.3	80.6	99	0.42	17.2	s49	542.1	529.8	97.0	108	0.42	20.3
s02	426.2	415.5	86.3	97	0.45	22.2	s50	651.0	642.4	105.3	78	0.45	21.5
s03	579.1	562.8	81.5	114	0.28	18.0	s51	711.1	692.2	81.6	78	0.38	24.5
s04	704.7	690.3	92.9	75	0.45	22.1	s52	926.0	906.4	92.0	151	0.26	20.2
s05	300.5	277.5	88.1	125	0.33	21.0	s53	538.9	528.5	90.0	129	0.45	21.4
s06	394.6	386.4	96.0	83	0.45	20.2	s54	655.9	644.6	87.6	98	0.48	20.2
s07	648.8	638.5	153.6	120	0.51	26.1	s55	701.8	681.1	75.9	73	0.46	18.8
s08	776.0	755.7	76.1	98	0.28	21.4	s56	909.9	893.3	105.1	102	0.53	20.8
s09	222.8	199.0	84.4	78	0.30	21.7	s57	591.2	566.9	108.8	148	0.43	20.6
s10	447.4	432.1	95.1	204	0.38	23.3	s58	640.1	620.1	85.9	111	0.37	20.0
s11	519.5	506.1	86.3	61	0.36	20.0	s59	752.1	710.8	83.5	114	0.31	20.1
s12	705.5	673.4	91.6	78	0.21	23.4	s60	1,020.4	1,013.1	76.4	82	0.26	16.8
s13	214.1	188.6	72.5	58	0.28	18.5	s61	523.1	510.5	58.8	67	0.38	17.7
s14	449.4	435.5	96.0	83	0.42	25.3	s62	753.6	732.3	90.2	110	0.49	19.1
s15	536.0	510.8	109.7	199	0.30	21.6	s63	720.6	708.8	91.0	84	0.43	20.5
s16	765.1	747.7	100.3	50	0.32	22.2	s64	860.0	844.7	90.8	91	0.38	21.8
s17	438.9	437.7	82.7	79	0.19	22.0	s65	494.1	475.1	83.3	101	0.34	22.0
s18	351.0	333.2	86.1	85	0.36	23.8	s66	733.3	710.5	96.7	111	0.41	20.1
s19	556.2	538.6	83.6	50	0.36	20.2	s67	869.5	843.4	83.2	125	0.42	20.3
s20	683.2	671.4	102.3	77	0.37	22.4	s68	787.5	766.8	87.5	95	0.35	23.7
s21	239.5	221.9	91.0	117	0.27	21.6	s69	582.8	535.5	44.3	46	0.18	16.1
s22	361.9	338.1	78.8	102	0.34	16.3	s70	550.6	526.0	65.7	67	0.38	17.0
s23	562.6	541.3	105.1	199	0.37	24.5	s71	645.5	628.7	103.1	81	0.41	22.9
s24	613.8	598.1	75.0	80	0.35	20.2	s72	895.8	881.7	94.8	151	0.28	27.0
s25	306.6	279.3	92.0	151	0.40	21.6	s73	704.4	703.8	110.3	96	0.52	20.9
s26	688.9	669.8	73.1	85	0.27	19.0	s74	994.2	954.6	84.2	78	0.36	19.2
s27	636.1	616.4	94.5	90	0.49	19.3	s75	823.2	814.7	96.4	57	0.57	18.0
s28	765.8	759.0	99.4	67	0.46	22.4	s76	784.9	765.3	113.2	111	0.47	21.0
s29	301.4	295.9	79.9	90	0.39	19.4	s77	724.9	720.7	90.7	69	0.52	16.6
s30	463.6	453.8	95.0	108	0.44	17.6	s78	1,125.1	1,102.7	77.8	61	0.34	17.1
s31	623.8	617.5	131.0	118	0.49	26.1	s79	841.6	823.0	95.5	84	0.49	19.4
s32	744.6	737.5	101.8	107	0.44	21.4	s80	1,030.2	1,002.2	75.9	48	0.26	15.9
s33	265.7	237.9	99.4	132	0.35	24.2	s81	665.5	652.2	115.1	147	0.45	22.4
s34	910.8	898.3	69.5	45	0.27	18.8	s82	770.1	755.6	104.1	84	0.47	20.8
s35	591.8	580.8	88.9	83	0.32	19.7	s83	887.4	865.3	87.4	91	0.33	24.2
s36	725.9	719.4	91.6	147	0.34	25.1	s84	810.4	793.3	74.0	109	0.38	18.0
s37	293.9	281.4	104.2	68	0.32	22.6	s85	796.3	785.3	120.1	80	0.46	22.8
s38	858.9	848.8	94.7	133	0.27	22.3	s86	772.0	761.9	120.3	124	0.51	20.2
s39	615.8	602.8	90.2	92	0.35	21.6	s87	783.6	763.1	86.5	106	0.50	17.8
s40	812.3	791.3	83.6	86	0.31	20.6	s88	785.3	754.0	79.0	93	0.35	16.7
s41	281.9	270.3	96.1	148	0.36	24.6	s89	806.2	803.4	105.1	107	0.34	22.1
s42	399.4	390.4	87.4	59	0.38	21.6	s90	885.1	866.6	110.1	108	0.34	21.5
s43	556.6	529.6	89.8	92	0.35	20.2	s91	721.1	709.6	93.0	68	0.42	22.1
s44	756.2	736.4	77.9	48	0.32	20.3	s92	880.4	864.3	85.1	109	0.37	18.9
s45	688.3	665.6	106.2	120	0.17	21.2	s93	769.1	756.4	103.5	54	0.28	16.9
s46	787.6	770.3	88.6	77	0.16	18.2	s94	659.0	640.1	87.0	74	0.36	21.2
s47	558.2	545.0	93.7	138	0.44	20.2	s95	884.5	862.7	101.0	102	0.38	19.9
s48	578.6	554.9	65.9	47	0.30	18.7	s96	619.4	606.1	72.4	75	0.29	20.0

For m21-40 with 5 agents we see a much more gradual increase in computation time as the problem size increases, which is what we expected. Again, the larger the problem size, the more improvements are found on average.

We tested the algorithm for relatedness levels 1, 2 and 3 and compared them to the solutions from our heuristic described in Section 5.3, the results can be seen in Table 7. For

Table 4: Results for Small Instances with 5 Agents

n = 100, l = 2, 10 repetitions

Instance	Value (avg)	Value (best)	Time (avg)	Time (best)	Avg Iteration length	Avg Improvements found	Instance	Value (avg)	Value (best)	Time (avg)	Time (best)	Avg Iteration length	Avg Improvements found
s01	92.1	92.1	16.9	19	0.14	4.0	s49	168.7	168.7	17.8	18	0.14	4.4
s02	143.2	143.2	18.8	20	0.13	5.4	s50	215.6	215.6	15.6	20	0.12	4.4
s03	241.9	240.9	22.5	27	0.13	5.6	s51	216.4	216.4	22.2	18	0.14	6.8
s04	214.7	214.7	14.8	15	0.13	3.5	s52	370.5	370.5	20.7	30	0.15	5.2
s05	127.8	125.9	16.5	14	0.14	4.1	s53	184.3	184.3	17.2	15	0.15	4.3
s06	144.7	144.7	20.0	25	0.13	6.5	s54	186.7	186.7	23.1	16	0.13	6.9
s07	189.3	189.3	13.9	15	0.13	3.8	s55	205.8	205.8	16.2	22	0.13	4.4
s08	403.8	403.8	23.5	15	0.13	5.7	s56	256.1	256.1	22.4	31	0.13	7.0
s09	78.0	78.0	18.1	23	0.14	4.3	s57	166.0	166.0	16.2	22	0.15	4.1
s10	211.8	211.8	16.7	14	0.14	4.3	s58	211.1	208.7	15.8	14	0.13	4.1
s11	187.6	187.6	24.9	30	0.14	7.1	s59	319.6	319.6	26.6	16	0.13	8.1
s12	256.5	256.5	17.5	18	0.14	4.3	s60	453.4	453.4	26.7	25	0.11	7.0
s13	81.5	81.5	17.0	19	0.15	3.9	s61	201.8	201.8	24.7	27	0.14	6.5
s14	176.8	176.8	22.5	28	0.13	6.3	s62	232.1	232.1	21.6	14	0.13	6.5
s15	173.2	173.2	18.1	24	0.14	3.9	s63	215.3	215.3	14.8	16	0.15	3.6
s16	203.5	203.5	21.0	20	0.15	6.0	s64	336.7	336.7	24.9	22	0.13	7.2
s17	317.6	316.5	15.8	17	0.14	3.7	s65	170.2	170.2	13.7	23	0.11	4.0
s18	125.6	125.6	23.4	24	0.14	6.3	s66	192.1	192.1	24.9	19	0.13	7.3
s19	181.8	181.8	21.4	18	0.12	6.6	s67	217.5	217.5	19.6	19	0.14	4.8
s20	224.5	224.5	15.7	16	0.13	4.1	s68	242.4	238.0	16.6	20	0.13	4.3
s21	156.5	156.5	18.1	13	0.14	4.1	s69	312.5	295.8	16.5	15	0.14	4.3
s22	147.4	147.4	22.4	26	0.13	7.5	s70	207.3	207.3	15.0	14	0.13	3.9
s23	287.7	287.7	21.7	22	0.14	5.7	s71	210.6	210.6	14.7	16	0.13	3.9
s24	281.6	258.9	20.6	18	0.13	5.0	s72	515.9	515.9	22.0	20	0.13	6.5
s25	86.4	86.4	17.2	28	0.15	4.2	s73	278.3	278.3	24.9	28	0.15	5.5
s26	374.3	374.3	27.5	36	0.15	6.2	s74	501.5	498.2	22.0	23	0.14	5.3
s27	171.7	171.7	22.2	28	0.15	6.3	s75	217.7	217.7	16.9	18	0.14	4.6
s28	210.9	210.9	23.2	23	0.15	6.8	s76	231.5	231.5	14.8	17	0.13	3.7
s29	127.8	127.8	18.9	17	0.15	4.8	s77	173.4	173.4	21.1	23	0.15	5.9
s30	126.3	126.3	17.2	16	0.15	3.9	s78	663.2	659.2	24.4	30	0.15	6.9
s31	185.2	185.2	17.0	15	0.15	3.8	s79	226.1	226.1	24.5	27	0.15	6.4
s32	211.3	211.3	21.0	21	0.15	6.1	s80	477.6	477.6	23.6	28	0.15	7.0
s33	86.0	86.0	18.9	19	0.15	4.7	s81	177.5	177.5	26.9	22	0.15	6.3
s34	664.4	626.0	19.2	22	0.14	4.5	s82	265.9	265.9	20.2	20	0.15	5.2
s35	178.2	178.2	22.2	26	0.15	6.1	s83	547.1	542.1	22.8	41	0.15	6.0
s36	237.4	237.4	28.8	28	0.15	7.5	s84	251.7	251.7	21.2	20	0.12	6.5
s37	78.1	78.1	17.4	19	0.16	3.5	s85	429.2	429.2	18.2	19	0.15	5.4
s38	690.9	685.2	19.1	39	0.13	4.5	s86	210.2	210.2	22.8	14	0.15	5.8
s39	170.3	170.3	24.0	30	0.15	7.2	s87	217.4	217.4	25.9	28	0.14	7.4
s40	368.5	368.5	22.3	32	0.13	6.5	s88	259.9	259.9	23.1	26	0.15	6.8
s41	78.5	78.5	17.0	16	0.15	4.0	s89	982.1	982.1	21.1	28	0.15	5.8
s42	127.6	127.6	26.4	23	0.16	6.7	s90	213.7	213.7	15.3	16	0.13	4.1
s43	189.7	189.7	16.3	19	0.15	3.7	s91	218.6	218.6	15.9	17	0.13	3.9
s44	211.8	211.8	22.3	21	0.14	6.4	s92	234.1	234.1	25.1	47	0.15	6.9
s45	881.1	881.1	21.6	22	0.15	5.1	s93	638.7	638.7	19.9	19	0.15	4.6
s46	800.7	800.7	19.4	19	0.15	4.3	s94	214.7	214.7	20.9	19	0.15	6.8
s47	198.4	198.4	23.0	26	0.14	6.5	s95	542.6	535.5	18.8	22	0.15	4.8
s48	229.8	229.8	21.5	21	0.13	6.3	s96	284.0	284.0	15.6	20	0.13	3.9

level 1, the starting heuristic outperforms the algorithm for instance sets S2 and S5, demonstrating that it is not able to take enough of the production plan into account to find better solutions. As to why the lot-shifting heuristic outperforms our algorithm we emphasize that assignments of items can differ from the given assignments in the instances. Therefore, a better assignment may have been predetermined. We would like to remind, however, that the solutions are trivial for S5, as there is no possibility for an agent to produce more than 1 product, leading to lot-for-lot solutions throughout. Level 2

significantly outperforms level 1 as well as the heuristic for all instance sets except S5, but level 3 only slightly improves on them. Surprisingly, level 3 provides worse solutions for M5 than level 2 on average. It should be noted that the heuristic was unable to find feasible solutions for some instances[6] in M2 and M5 (the affected instances are marked with an asterisk (*) in their respective tables). This demonstrates that even heuristic solutions may be difficult to find in these larger problems. Detailed results for each instance can be seen in Table 9-Table 11.

Table 6 compares the average performance (time in seconds) of the algorithm at relatedness levels 1, 2 and 3 as well as the uncapacitated problem at relatedness level 2 as reference values. For the small instances with 2 agents, results are as expected: increasing the number of variables in each subproblem lowers costs while increasing the amount of time until the algorithm terminates. This decrease is 13.5% from level 1 to level 2, while the additional effort in computation time is marginal at a 2.5% increase. Though it should be noted, that the results for each individual instance may paint a very different picture: 45 of the s2 instances had both a better solution value while simultaneously having lower calculation times. This is most likely due to the limited size of the subproblems at level 1.

[6] Specifically m27, 30, 32 and 33 for M2 and
m10, 13, 16, 17, 18, 20, 22, 26, 27, 29, 30, 32, 33, 35, 37 and 38 for M5

Table 5: Results for Medium Instances with 2 Agents (left) and 5 Agents (right)

n = 50, l = 2, 5 repetitions (5 Agents)

Instance	Value (avg)	Value (best)	Time (avg)	Time (best)	Avg Iteration length (sec)	Avg Improvements found
m01	287,086.3	275,006.1	1,066.4	5,318	0.52	85.2
m02	163,447.7	162,278.4	392.7	371	0.89	120.3
m03	154,883.5	147,955.6	245.0	332	0.89	85.5
m04	178,816.1	156,074.9	680.9	1,216	0.24	65.4
m05	185,972.4	184,000.9	361.9	574	0.90	117.3
m06	191,648.7	189,897.2	728.8	597	1.55	128.3
m07	173,231.2	172,102.8	354.3	369	1.03	112.2
m08	140,013.4	135,021.3	491.6	700	1.48	121.9
m09	162,281.1	160,607.3	703.1	538	1.54	142.1
m10	216,396.1	211,114.0	756.4	509	0.38	79.5
m11	307,816.8	306,140.0	1,170.8	1,626	1.73	247.0
m12	346,244.2	337,579.1	981.1	1,119	1.68	213.5
m13	279,796.3	269,394.1	848.5	977	1.17	191.0
m14	475,697.4	448,514.0	422.8	651	0.93	101.5
m15	332,211.2	320,184.4	1,376.6	2,181	1.13	184.1
m16	337,709.0	328,630.9	733.5	834	1.16	168.1
m17	296,694.4	292,941.9	908.2	1,191	1.73	221.4
m18	391,933.3	379,563.2	1,183.1	7,143	1.52	162.3
m19	358,455.2	354,730.5	1,159.2	1,175	1.67	252.7
m20	383,402.8	378,733.2	1,182.7	1,551	1.66	247.4
m21	127,860.0	127,343.4	192.0	252	0.72	67.3
m22	165,637.2	165,041.6	311.5	384	0.94	87.6
m23	350,695.4	349,753.5	446.6	347	1.69	83.9
m24	228,456.0	226,952.6	652.2	646	1.53	108.6
m25	144,802.2	142,020.0	222.7	251	0.59	62.6
m26	168,600.6	167,905.2	265.5	442	0.95	83.2
m27	165,328.2	164,527.7	243.5	258	1.01	77.5
m28	131,143.4	130,412.2	369.2	669	1.40	84.9
m29	217,747.9	211,819.1	209.5	237	0.68	40.6
m30	191,491.0	190,685.3	637.3	865	2.19	98.8
m31	351,429.8	349,880.1	995.7	1,279	2.10	157.0
m32	250,690.7	249,530.4	714.0	916	1.14	191.6
m33	339,343.9	335,518.6	2,316.1	841	1.14	139.3
m34	295,270.9	293,700.3	632.6	765	1.05	160.1
m35	317,168.0	315,687.9	523.3	749	1.44	141.9
m36	279,842.1	278,090.8	606.5	804	1.26	163.1
m37	837,590.7	815,638.0	1,095.2	3,240	1.43	88.2
m38	328,079.2	323,811.7	499.5	818	1.39	142.0
m39	366,316.5	365,339.1	555.2	779	1.35	159.0
m40	504,774.6	494,795.6	742.9	465	0.55	92.1

Instance	Value (avg)	Value (best)	Time (avg)	Time (best)	Avg Iteration length (sec)	Avg Improvements found
m01	255,274.5	249,106.6	3,505.8	3,754	0.88	103.4
m02	147,169.1	145,408.9	1,646.6	1,747	3.77	107.4
m03	131,658.3	129,948.5	4,261.8	1,560	3.04	100.6
m04	153,029.0	142,722.3	4,344.6	3,085	0.35	85.0
m05	167,028.0	165,761.4	2,591.8	3,370	7.29	122.2
m06	172,752.3	171,567.4	2,529.6	4,014	5.06	97.4
m07	151,193.2	149,957.6	3,457.6	5,004	7.76	117.0
m08	127,203.9	125,629.8	2,799.0	2,863	6.41	108.2
m09	146,592.0	145,464.0	4,967.8	6,265	10.40	125.8
m10	209,141.7	188,565.2	2,353.2	3,224	2.09	79.6
m11	286,154.6	279,137.1	5,208.0	7,571	9.02	192.2
m12	307,440.9	304,989.2	6,058.2	8,621	7.03	216.8
m13	248,467.6	243,976.7	5,230.4	6,620	5.12	189.8
m14	439,745.1	388,184.4	3,653.6	8,825	3.51	123.6
m15	306,356.3	289,999.6	8,710.2	10,340	2.99	204.4
m16	301,220.7	296,429.2	6,205.2	7,080	2.14	161.8
m17	265,078.6	263,732.1	4,773.2	4,566	8.24	207.6
m18	357,952.4	343,883.2	9,157.2	26,634	2.78	168.0
m19	328,596.0	324,847.9	6,121.4	8,102	9.89	191.2
m20	348,878.1	334,482.0	4,895.8	8,500	5.98	176.4
m21	112,367.7	111,602.5	2,089.4	2,314	7.97	64.0
m22	144,939.1	144,270.9	2,401.4	3,039	2.68	88.6
m23	332,971.3	332,399.2	4,228.0	4,224	5.55	85.8
m24	211,522.3	209,476.0	3,860.4	3,271	8.43	85.6
m25	127,182.3	125,904.6	1,657.4	2,135	3.54	72.6
m26	148,978.7	147,545.8	2,405.2	1,961	3.61	72.4
m27	145,533.3	145,076.8	2,388.4	1,716	4.13	76.2
m28	118,530.4	117,679.0	2,937.0	3,338	6.37	83.6
m29	203,679.7	196,185.8	3,451.6	2,306	1.07	61.2
m30	177,305.8	176,433.7	5,643.6	3,121	10.72	105.0
m31	334,097.1	316,929.7	3,718.4	6,739	9.58	108.4
m32	222,158.5	220,266.5	4,289.4	6,087	7.46	133.0
m33	299,401.7	297,200.4	5,039.2	3,180	5.21	129.2
m34	257,742.1	255,591.0	5,107.8	6,484	7.10	147.8
m35	280,902.1	278,789.7	3,893.2	4,057	5.77	124.0
m36	249,992.5	244,014.6	3,033.6	4,102	4.60	117.8
m37	802,854.1	779,733.4	5,988.0	5,157	2.61	103.4
m38	278,595.2	277,658.2	5,674.8	6,631	5.96	152.6
m39	322,076.7	317,484.5	3,496.6	3,399	4.86	118.4
m40	448,758.7	447,642.2	7,733.6	16,453	0.85	136.8

Since the algorithm can only select a smaller number of variables in each subproblem, it takes more steps for the program to find the best solution, and in some cases it fails to do so entirely as the random nature of the selected base variable may exclude a portion of them if "edge" variables are simply never selected. Higher relatedness levels are able to overcome this issue. From level 2 to 3, the reduction is only a mere 2% on average but with a rather significant increase in computation time. Again, 13 instances had better computation times than their level 1 counterparts, 5 of which even outperformed the level 2 calculations. Nonetheless, level 3 consistently found better solutions than level 2 on average.

Table 7: Comparing Solution Quality to Lot-shifting Heuristic

Instance set	L = 1	L = 2	L = 3
	Relative cost reduction (%)	Relative cost reduction (%)	Relative cost reduction (%)
s2	-2.19%	11.87%	13.90%
s5	-4.95%	-1.07%	-0.76%
m2	1.37%	10.64%	12.81%
m5	12.07%	22.52%	21.76%

The small instances with 5 agents are special cases, as was discussed earlier. As was the case for the s2 instances, level 1 offers worse results when compared to level 2 and 3 both in solution quality and computation time, for the same reasons as outlined in the last paragraph.

Table 6: Comparison of Solution Quality at different Relatedness Levels

Instance	L = 1		L = 2		L = 3		uncapacitated, L = 2	
	Cost	Time	Relative cost reduction	Relative time increase (%)	Relative cost reduction	Relative time increase (%)	Relative cost reduction	Relative time increase (%)
s2	750.6	90.1	13.51	2.49	15.54	52.02	26.14	8.22
s5	278.9	27.4	3.06	-23.90	3.35	-23.89	20.76	-27.30
m2	303,808.6	605.3	8.19	25.58	10.05	570.26	18.43	35.09
m5	282,174.1	1,676.5	9.28	240.18	7.81	207.99	20.57	259.65

Although the differences for the m2 instances in Table 6 do show a decent improvement for level 2 compared to level 1 on average, this average does not give a good representation of the performance overall. The improved solution quality ranges between 1.72% in m36 and 28.57% in m10 and the additional time it took for the algorithm to terminate is just as varried; for some instances the algorithm finished more quickly at level 2 while for some others it took a significantly longer time, m33 stands out among them. Taking a closer look at this particular instance shows an interesting aspect about our implementation: As described earlier, some runs have the solution temporarily be stuck in a very restrictive

neighborhood, unable to escape easily as there is no mechanism dedicated specifically to that situation. When this happens a comparatively extremely large number of iterations is needed for the algorithm to randomly find its way out of it; m33 at relatedness level 2 has two such runs. The average computation time of the 10 runs is 2,316 seconds with 14,407 iterations, but without these two runs it would be only 741 seconds with 841 iterations, while the average cost would go from 339,343.92 to 339,969.51. Thus, although the computation time is very bloated, the algorithm does eventually find better than average solutions for these runs. A dedicated heuristic for escaping these local optima would very likely help in keeping the solution quality while drastically reducing the computation times of these cases, making our algorithm a lot more competitive.

On the other end of the spectrum there is m14 for example, where multiple local optimum situations happened in the level 1 runs, while only one such run happened at level 2. And while the "stuck" runs of level 1 had better results than the "non-stuck"ones, level 2 outperformed them by roughly 13.5% on average.

At level 3 the marginal improvement to level 2 is only 1.86%, with computation times nearly six times longer than at level 1. This is due to the longer computation times for each iteration due to the much larger subproblems, and not a higher density of stuck runs. In fact, they appeared less frequently than at level 2 or 1, showing a better capability of escaping local optima or not falling into them in the first place. There is not a single instance in which the solution quality of level 2 outperforms level 3 (this is true for all instances in s2 and s5 as well) and there are 12 instances[7] for which at level 3 the algorithm performed better than the uncapacitated reference cases. Unfortunately, for half of the m2 instances the marginal

[7] m02, m03, m07, m21, m22, m25, m26, m32, m34, m35, m36 & m38

cost reduction is less than 1% (see Table 10), which is not enough to justify the additional time spent on solving these problems. On the other hand, this suggests a good cost-benefit-ratio for relatedness level 2 on these medium scale instances.

The m5 instances show this even clearer: here, the average improvement of level 2 over level 1 is 9.28% with a 240% increase in computation time, on top of the already much longer average computation times of the m2 instances. This is, of course, due to the instances being significantly larger, but it is a factor that is easily overlooked when just comparing percentages. Additionally, level 3 significantly underperformed in these instances, with solution qualities being worse in 31 of the 40 instances when compared to level 2 and even worse than level 1 in 6 instances. Our algorithm seems to struggle with the large subproblems of m5; the average per iteration computation time is much higher in most instances, much fewer improvements are found compared to level 2 despite a similar number of total iterations. This is also the reason for the poor performance of level 3: it is not able to find improvements effectively and therefore stops prematurely before it is able to comb through a sufficient part of the solution space. Curiously, if level 3 finds a solution that is on par or even better than the level 2 solutions, it does so in fewer iterations; in other words the average increase in solution quality per improvement found is significantly better at level 3. Therefore finding a method to help the algorithm better find improvements could help in increasing solution quality in the future.

7. Conclusion

In this thesis, a new problem variant of the CLSP was introduced. Existing instances from Buer et. al (2015) were extended to fit the criteria of this new problem, the CCMLCLSP, and a fix-and-optimize approach to solve them was created. It is an iterative approach that selects a subproblem in each iteration, where most binary decision variables are fixed while the remaining ones are reoptimized. The definition of the subproblems was derived from Chen (2015), where the selection is based on the relatedness of binary variables in the MIP model, while also including some random elements to avoid getting stuck in a local optimum. Baseline results were computed for the extended instances using the described method for 272 out of the 352 created instances – the small and medium sized instances. Our algorithm was not able to find solutions in an acceptable timeframe for the large instances, due to the vast number of decision variables they possess. Therefore, an algorithm that can solve these instances in particular more effectively needs to be created. A branch-and-cut approach or a way to optimize certain subproblems of the instances could prove as good starting points for this.

Further improvements can be made in the stability of each indivial iteration to find a feasible result, this is currently the biggest time loss overall. This happens when a solution is found that lies in a narrow neighborhood e.g. when capacity on a certain agent is used for items that are not mandatory to produce for him but when other production is shifted, it cannot be freed unless the specific variables are selected for reallocation again. It is much more likely to occur in lower relatedness levels, as they have smaller neighborhoods to work with. We would like to give suggestions to overcome this problem: infeasible solutions may simply

be allowed as intermediary solutions to get out of restrictive neighborhoods - this would require the algorithm to have a way to analyze where bottlenecks of a given solution lie and to focus on neighborhoods that alleviate this bottleneck, most likely in the form of a repair heuristic; allowing the algorithm to switch between relatedness levels for the purposes of escaping these restrictive neighborhoods; storing more intermediary solutions to call back on, should such a restrictive neighborhood be reached. Alternatively, it may have been our implementation that caused this likelyhood of infeasibilities in the first place. Using a different solver could potentially solve this issue as well.

Different solution approaches may yield better results for the underlying problem; branch-and-cut, column generation, ant-colony optimisation come to mind. As for branch-and-cut: tightening the solution space with valid constraints for this problem type may yield improved results in terms of computation time. More efficient implementations may also be capable of computing results for the large instances in a more reasonable timeframe.

The computational study showed consistent behavior of our algorithm; increasing the size of the subproblems provided better solution quality coupled with an increased computation time, the extent varying depending on the size of the problem. The medium sized instances with 5 agents break this trend, as relatedness level 3 on average performed worse than level 2 for these instances, showing a weakness of our approach. As we did not calculate results for the large instances, we can only assume that they would behave similary to the medium instances. Results on the small instances with 5 agents need to be evaluated separately due to the way these instances are built, as was explained in this thesis.

Appendix

Table 9: Comparison of different Relatedness Levels, Class S, 2 Agents

	L = 1		L = 2		L = 3		uncapacitated, L = 2	
Instance	Cost	Time	Relative cost reduction (%)	Relative time increase (%)	Relative cost reduction (%)	Relative time increase (%)	Relative cost reduction (%)	Relative time increase (%)
s01	277.7	89.4	22.56	-9.84	27.14	32.89	25.14	12.98
s02	481.3	96.7	11.44	-10.72	14.23	59.62	16.27	17.10
s03	688.7	89.5	15.92	-8.94	17.47	-4.13	18.50	14.30
s04	781.3	85.5	9.81	8.65	11.16	125.85	12.95	47.49
s05	367.8	89.0	18.31	-1.01	20.80	33.48	38.22	3.26
s06	480.9	91.8	17.95	4.58	19.65	99.56	19.54	6.21
s07	714.2	85.8	9.14	79.02	10.05	120.51	25.43	7.20
s08	953.3	82.1	18.60	-7.31	20.05	45.92	30.66	24.14
s09	307.9	84.7	27.65	-0.35	31.77	-4.01	33.23	-7.28
s10	508.2	97.4	11.96	-2.36	14.97	67.76	26.60	-0.68
s11	603.9	87.8	13.96	-1.71	15.61	33.49	19.28	-1.03
s12	859.5	86.8	17.91	5.53	21.48	0.12	23.05	9.68
s13	289.8	103.3	26.13	-29.82	34.77	-7.16	31.17	-8.78
s14	488.6	97.9	8.02	-1.94	10.29	70.07	29.86	-15.39
s15	634.5	100.6	15.52	9.05	17.60	-10.93	18.81	-2.65
s16	880.7	98.3	13.14	2.03	15.27	26.25	22.67	1.25
s17	530.8	108.1	17.32	-23.50	17.38	-14.62	62.52	-15.17
s18	470.6	83.8	25.40	2.74	28.02	71.12	32.24	1.13
s19	684.3	79.6	18.72	5.03	20.97	40.45	26.99	14.28
s20	818.5	66.4	16.53	54.07	18.07	71.08	24.13	36.95
s21	299.2	88.7	19.95	2.59	25.82	-9.92	35.28	-1.09
s22	445.8	71.5	18.82	10.21	25.18	38.88	21.04	16.64
s23	640.3	94.4	12.15	11.33	14.51	26.59	33.34	-9.89
s24	743.5	84.2	17.44	-10.93	19.52	12.47	21.23	2.65
s25	342.4	127.5	10.48	-27.84	18.44	11.14	14.50	-19.61
s26	793.3	101.2	13.16	-27.77	16.80	-5.43	39.46	14.36
s27	683.9	83.2	6.99	13.58	8.62	94.95	8.60	25.80
s28	813.9	98.1	5.92	1.33	6.83	64.02	10.23	8.80
s29	357.4	99.0	15.68	-19.29	17.22	33.54	16.97	10.40
s30	511.0	99.1	9.28	-4.14	11.13	30.07	10.93	17.86
s31	691.1	84.4	9.74	55.21	10.19	83.65	9.69	47.51
s32	786.6	89.4	5.34	13.87	6.00	32.21	6.37	13.12
s33	338.9	114.6	21.60	-13.26	25.79	-3.40	21.63	-16.40
s34	1,027.0	86.6	11.31	-19.75	11.92	1.39	58.64	13.39
s35	732.3	101.2	19.19	-12.15	20.60	3.56	22.07	-2.96
s36	825.4	93.4	12.05	-1.93	12.70	16.92	16.65	3.46
s37	344.8	98.8	14.76	5.47	17.43	-7.29	19.46	-9.35
s38	932.2	76.9	7.87	23.15	8.93	43.56	52.05	22.37
s39	710.1	104.1	13.28	-13.35	14.31	29.30	17.18	0.29
s40	886.9	79.5	8.42	5.16	10.31	73.96	19.39	14.68
s41	340.0	91.8	17.10	4.68	20.12	37.91	22.49	33.22
s42	470.9	97.3	15.20	-10.17	17.03	33.30	16.35	5.89
s43	630.0	92.5	11.66	-2.92	15.58	13.62	18.45	14.20
s44	907.8	66.2	16.70	17.67	18.90	84.59	29.14	45.62
s45	710.2	100.6	3.08	5.57	6.28	20.08	63.30	-15.90
s46	876.0	89.0	10.09	-0.45	12.05	73.15	56.75	-2.25
s47	639.5	89.3	12.72	4.93	13.06	26.32	20.97	-0.67
s48	766.1	61.8	24.47	6.63	26.11	16.18	25.83	25.35
s49	628.7	88.2	13.77	9.98	14.61	86.05	29.20	-14.66
s50	734.7	89.2	11.39	18.05	11.90	85.65	21.25	-0.49
s51	794.1	99.5	10.45	-17.99	12.03	68.14	21.03	-6.37
s52	1,107.6	80.7	16.39	14.00	18.02	49.69	34.62	24.25
s53	614.5	76.8	12.30	17.19	13.38	187.76	29.34	24.70
s54	743.9	89.8	11.83	-2.45	13.30	104.34	21.08	15.37
s55	732.0	106.1	4.13	-28.46	5.88	75.21	6.78	-3.83
s56	969.0	93.6	6.10	12.29	7.06	167.31	18.58	11.25
s57	696.2	91.3	15.09	19.17	18.48	59.69	38.85	-4.97
s58	745.6	91.8	14.14	-6.43	15.77	31.81	23.57	-5.05
s59	924.0	94.8	18.60	-11.92	22.47	5.59	31.75	-9.49
s60	1,501.1	87.6	32.02	-12.79	32.30	21.35	50.93	4.22
s61	596.5	85.8	12.31	-31.47	13.12	56.88	25.56	8.86
s62	843.6	86.8	10.67	3.92	13.14	147.24	30.87	10.14
s63	778.5	96.7	7.44	-5.89	9.49	49.22	14.50	-1.45
s64	932.0	94.6	7.72	-4.02	8.55	102.75	22.39	8.39
s65	613.9	66.0	19.50	26.21	22.36	70.15	37.10	18.08
s66	873.8	84.2	16.08	14.85	17.89	158.43	37.15	6.37
s67	979.0	80.4	11.19	3.48	12.85	115.67	39.65	2.78
s68	880.0	88.6	10.51	-1.24	12.60	43.12	23.20	11.02
s69	772.9	80.5	24.60	-44.97	30.58	-14.41	52.39	-5.51
s70	624.9	99.9	11.88	-34.23	14.44	20.12	19.05	-10.04
s71	714.6	107.9	9.67	-4.45	9.96	-1.30	22.62	-24.81
s72	1,061.6	68.2	15.62	39.00	16.89	75.95	40.74	9.82
s73	791.6	99.8	11.01	10.52	11.09	52.40	11.61	20.47
s74	1,192.3	83.4	16.62	0.96	19.52	22.30	36.27	55.76
s75	880.2	101.1	6.48	-4.65	6.99	78.93	14.06	9.86
s76	886.6	82.5	11.47	37.21	12.30	72.36	21.45	18.91
s77	751.5	81.1	3.54	11.84	3.98	170.28	4.02	35.02
s78	1,364.3	87.5	17.53	-11.09	19.06	12.57	41.93	29.14
s79	938.6	84.8	10.34	12.62	13.09	74.88	19.55	26.06
s80	1,191.8	97.0	13.56	-21.75	15.05	19.07	33.51	-2.61
s81	750.8	78.6	11.37	46.44	12.41	122.01	13.88	24.98
s82	820.8	93.1	6.19	11.82	7.92	95.81	11.62	0.04
s83	1,084.1	84.3	18.15	3.68	19.76	48.75	37.66	10.87
s84	946.1	91.8	14.35	-19.39	15.41	14.05	23.41	6.54
s85	893.3	105.6	10.85	13.73	11.76	82.10	24.22	-11.52
s86	833.5	78.7	7.38	52.86	8.46	131.00	10.75	33.50
s87	885.5	97.3	11.51	-11.10	13.27	45.73	17.21	2.33
s88	904.3	118.4	13.16	-33.28	14.49	-0.25	18.25	-18.69
s89	866.3	95.2	6.93	10.40	7.21	35.92	33.33	9.98
s90	944.5	94.1	6.29	17.00	7.62	114.13	29.33	6.34
s91	789.9	85.8	8.72	8.39	10.73	55.36	17.96	7.30
s92	1,016.2	86.8	13.37	-1.96	14.91	101.15	32.73	2.80
s93	852.0	79.4	9.73	30.35	11.53	75.31	37.55	18.22
s94	759.2	81.0	13.20	7.41	14.96	37.04	18.44	28.52
s95	1,032.0	90.6	14.29	11.48	16.18	43.05	41.52	3.13
s96	749.3	77.5	17.33	-6.58	18.59	-8.65	21.25	0.95

Table 8: Comparison of different Relatedness Levels, Class S, 5 Agents

	L = 1		L = 2		L = 3		uncapacitated, L = 2	
Instance	Cost	Time	Relative cost reduction (%)	Relative time increase (%)	Relative cost reduction (%)	Relative time increase (%)	Relative cost reduction (%)	Relative time increase (%)
s01	92.1	14.5	0.00	16.55	0.00	21.38	0.00	13.56
s02	143.2	25.5	0.00	-26.27	0.00	-4.71	0.00	-21.57
s03	292.5	20.0	17.32	12.50	17.64	24.00	38.76	10.16
s04	214.7	21.5	0.00	-31.16	0.00	-31.16	0.00	-30.53
s05	126.5	23.5	-1.06	-29.79	0.42	-25.53	34.67	-33.29
s06	144.7	33.5	0.00	-40.30	0.00	-38.51	0.00	-36.72
s07	189.3	25.5	0.00	-45.49	0.00	-41.18	0.71	-38.30
s08	415.5	25.5	2.80	-7.84	2.80	-40.39	43.77	-30.85
s09	104.7	28.0	25.50	-35.36	25.50	-31.79	25.50	-43.57
s10	211.8	19.0	0.00	-12.11	0.00	-11.58	32.09	-11.75
s11	240.2	35.5	21.88	-29.86	21.88	-23.38	22.42	-37.56
s12	283.8	25.0	9.60	-30.00	9.60	-31.20	18.24	-31.07
s13	115.0	26.0	29.14	-34.62	29.14	-29.62	29.14	-35.13
s14	176.8	28.0	0.00	-19.64	0.00	-25.71	24.53	-18.45
s15	182.4	22.5	5.04	-19.56	5.04	-6.67	5.04	-27.70
s16	228.5	27.0	10.94	-22.22	10.94	-14.81	11.99	-8.15
s17	320.5	25.0	0.90	-36.80	1.23	-22.00	75.61	-39.60
s18	125.6	34.0	0.00	-31.18	0.00	-29.12	0.00	-32.84
s19	181.8	31.0	0.00	-30.97	0.00	-43.87	0.00	-34.09
s20	224.5	22.5	0.00	-30.22	0.00	-34.67	0.00	-28.44
s21	156.5	20.0	0.00	-9.50	0.00	2.00	44.22	-22.33
s22	147.4	25.5	0.00	-12.16	0.00	-9.80	5.60	-6.54
s23	287.7	30.0	0.00	-27.67	0.00	-23.33	35.78	-38.67
s24	258.9	23.5	-8.76	-12.34	0.00	-19.15	12.36	-14.04
s25	86.4	26.0	0.00	-33.85	0.00	-36.15	0.58	-36.92
s26	423.6	29.0	11.63	-5.17	11.63	-40.69	69.47	-26.09
s27	171.7	30.0	0.00	-26.00	0.00	-13.67	0.00	-24.78
s28	210.9	36.0	0.00	-35.56	0.00	-23.06	0.00	-38.80
s29	127.8	32.5	0.00	-41.85	0.00	-44.31	35.99	-51.69
s30	126.3	23.5	0.00	-26.81	0.00	-31.91	0.00	-31.91
s31	185.2	26.0	0.00	-34.62	0.00	-38.46	0.00	-34.87
s32	211.3	37.5	0.00	-44.00	0.00	-39.20	0.00	-39.91
s33	86.0	25.5	0.00	-25.88	0.00	-29.80	2.33	-34.12
s34	776.3	19.0	14.42	1.05	19.36	-10.53	83.52	-10.70
s35	178.2	26.5	0.00	-16.23	0.00	-0.75	0.00	-2.64
s36	247.3	22.0	3.98	30.91	3.98	11.82	3.98	10.45
s37	78.7	27.5	0.69	-36.73	0.69	-32.00	0.69	-39.39
s38	685.2	31.0	-0.83	-38.39	0.00	-47.42	80.13	-51.94
s39	170.3	36.5	0.00	-34.25	0.00	-34.25	0.00	-32.69
s40	368.5	31.0	0.00	-28.06	0.00	-17.42	41.81	-31.18
s41	78.5	28.5	0.00	-40.35	0.00	-5.96	0.00	-45.38
s42	127.6	40.5	0.00	-34.81	0.00	-35.31	0.00	-39.75
s43	190.1	30.5	0.21	-46.56	0.21	-43.61	0.21	-42.95
s44	234.4	34.5	9.62	-35.36	9.62	-28.70	9.62	-37.29
s45	881.1	22.5	0.00	-4.00	0.00	4.00	91.31	6.67
s46	803.7	23.0	0.37	-15.65	0.37	-26.09	83.98	-25.65
s47	198.4	39.5	0.00	-41.77	0.00	-38.48	2.72	-43.97
s48	254.8	26.0	9.81	-17.31	9.81	-9.23	9.81	-23.08
s49	168.7	19.0	0.00	-6.32	0.00	-14.74	0.00	-20.35
s50	215.6	21.0	0.00	-25.71	0.00	-32.86	0.00	-16.67
s51	216.4	29.0	0.00	-23.45	0.00	-20.00	0.00	-22.07
s52	395.5	28.0	6.32	-26.07	6.32	-24.29	40.12	-34.88
s53	184.3	24.5	0.00	-29.80	0.00	-26.12	0.00	-32.65
s54	186.7	45.0	0.00	-48.67	0.00	-49.11	0.00	-46.44
s55	205.8	24.0	0.00	-32.50	0.00	-40.00	0.00	-33.75
s56	256.1	30.0	0.00	-25.33	0.00	-34.67	1.41	-38.33
s57	166.0	17.5	0.00	-7.43	0.00	33.14	5.81	-5.71
s58	231.8	22.0	8.96	-28.18	9.96	-15.45	9.96	-30.15
s59	345.1	25.5	7.37	4.31	7.37	-7.84	35.57	-10.20
s60	682.5	21.5	33.57	24.19	33.57	23.26	65.14	-4.96
s61	201.8	23.0	0.00	7.39	0.00	3.48	18.88	1.45
s62	232.1	23.0	0.00	-6.09	0.00	-8.26	24.24	-0.87
s63	215.3	43.0	0.00	-65.58	0.00	-62.56	0.00	-58.99
s64	336.7	28.5	0.00	-12.63	0.00	-30.53	31.54	-18.13
s65	198.0	18.5	14.02	-25.95	14.02	-36.22	14.02	-25.41
s66	192.1	24.0	0.00	3.75	0.00	-10.42	0.21	-7.08
s67	217.9	38.0	0.18	-48.42	0.18	-47.89	2.57	-43.95
s68	238.0	27.0	-1.83	-38.52	0.00	-47.04	0.00	-38.89
s69	325.2	31.0	3.90	-46.77	9.03	-32.26	48.75	-45.48
s70	207.3	18.0	0.00	-16.67	0.00	-25.00	9.59	-6.67
s71	210.6	21.0	0.00	-30.00	0.00	-35.24	1.41	-19.52
s72	515.9	26.5	0.00	-16.98	0.00	-18.49	52.99	-33.21
s73	278.3	37.0	0.00	-32.70	0.00	-50.27	41.87	-54.05
s74	509.2	29.5	1.50	-25.42	2.15	-36.27	61.75	-45.76
s75	217.7	22.0	0.00	-23.18	0.00	-31.36	0.00	-23.18
s76	231.5	25.5	0.00	-41.96	0.00	-43.53	0.00	-40.00
s77	173.4	33.0	0.00	-36.06	0.00	-24.55	0.00	-38.79
s78	659.2	32.0	-0.61	-23.75	0.00	-36.88	66.29	-26.88
s79	298.1	29.0	24.14	-15.52	24.14	-20.34	24.14	-25.52
s80	477.6	28.0	0.00	-15.71	0.00	0.00	47.48	-22.86
s81	177.5	28.0	0.00	-3.93	0.00	-34.64	0.68	-33.21
s82	265.9	29.0	0.00	-30.34	0.00	-18.28	25.27	-28.62
s83	595.8	25.0	8.17	-8.80	9.01	-34.80	59.75	-16.40
s84	251.7	21.0	0.00	0.95	0.00	-15.71	0.00	5.71
s85	429.2	28.5	0.00	-36.14	0.00	-5.26	58.71	-22.46
s86	210.2	42.0	0.00	-45.71	0.00	-49.76	0.00	-46.19
s87	217.4	38.5	0.00	-32.73	0.00	-39.22	0.69	-41.82
s88	287.3	24.0	9.52	-3.75	9.52	-5.83	9.59	-2.08
s89	982.1	29.5	0.00	-28.47	0.00	-21.69	82.31	-32.88
s90	213.7	26.5	0.00	-42.26	0.00	-40.00	9.03	-36.60
s91	218.6	23.5	0.00	-32.34	0.00	-32.34	0.00	-27.66
s92	234.6	25.0	0.21	0.40	0.21	-5.20	0.21	-14.00
s93	638.7	23.0	0.00	-13.48	0.00	-6.09	75.53	-29.13
s94	236.9	34.5	9.35	-39.42	9.35	-24.35	17.79	-44.64
s95	573.4	27.5	5.38	-31.64	6.60	-18.18	63.53	-31.64
s96	284.0	23.5	0.00	-33.62	0.00	-37.87	9.40	-23.40

Table 10: Comparison of different Relatedness Levels, Class M, 2 Agents

	L = 1		L = 2		L = 3		uncapacitated, L = 2	
Instance	Cost	Time	Relative cost reduction (%)	Relative time increase (%)	Relative cost reduction (%)	Relative time increase (%)	Relative cost reduction (%)	Relative time increase (%)
m01	328,368.1	580.3	12.57	83.78	16.52	450.93	45.69	-6.25
m02	180,238.6	389.0	9.32	0.95	10.24	721.80	9.83	49.36
m03	183,695.5	240.4	15.68	1.91	19.77	1229.78	19.27	69.55
m04	235,951.8	485.6	24.22	40.22	33.82	558.32	40.47	53.21
m05	199,199.1	406.2	6.64	-10.91	7.73	687.00	7.90	35.75
m06	214,271.4	504.0	10.56	44.60	11.37	534.29	21.98	9.13
m07	176,770.1	504.0	2.00	-29.70	2.85	534.29	2.84	50.60
m08	149,849.4	411.0	6.56	19.61	9.80	677.81	18.26	74.06
m09	191,678.2	395.4	15.34	77.82	16.16	708.50	23.44	62.57
m10	302,967.0	377.4	28.57	100.42	31.75	747.06	43.95	39.80
m11	316,619.0	1,191.6	2.78	-1.75	3.08	168.28	3.18	55.81
m12	362,793.8	952.2	4.56	3.04	7.27	235.73	14.24	45.22
m13	329,517.4	655.6	15.09	29.42	17.79	387.61	19.97	140.67
m14	549,697.7	1,168.0	13.46	-63.80	22.05	173.70	37.22	-9.78
m15	368,107.5	598.2	9.75	130.12	12.36	434.40	12.54	162.89
m16	381,645.9	573.2	11.51	27.97	14.77	457.71	17.25	164.58
m17	306,409.7	1,040.6	3.17	-12.72	4.30	207.21	5.49	80.70
m18	439,248.7	574.8	10.77	105.83	14.33	456.16	14.71	199.06
m19	369,461.0	1,355.0	2.98	-14.45	4.00	135.93	4.78	15.78
m20	399,111.8	1,062.9	3.94	11.28	4.84	200.77	7.35	34.84
m21	132,534.4	323.0	3.53	-40.56	3.98	889.72	3.55	-30.03
m22	169,937.3	376.2	2.53	-17.20	2.95	749.76	2.63	-8.35
m23	356,820.6	313.0	1.72	42.68	1.86	921.34	51.47	-4.66
m24	245,033.0	364.2	6.77	79.08	7.14	777.76	32.56	-50.80
m25	181,744.3	166.4	20.33	33.83	22.33	1821.15	21.87	54.57
m26	173,607.4	403.6	2.88	-34.22	3.33	692.07	2.75	-19.33
m27	170,179.8	341.6	2.85	-28.72	3.28	835.83	3.58	-12.88
m28	134,391.1	353.8	2.42	4.35	3.14	803.56	12.15	28.72
m29	272,334.7	221.0	20.04	-5.20	23.62	1346.52	48.90	25.25
m30	199,443.5	362.8	3.99	75.66	4.33	781.15	25.21	0.72
m31	359,072.9	777.6	2.13	28.05	2.59	311.11	15.99	-19.57
m32	259,853.8	832.6	3.53	-14.24	4.09	283.95	3.16	-17.61
m33	378,263.6	449.8	10.29	414.92	11.50	610.72	15.36	103.60
m34	301,201.2	747.9	1.97	-15.41	2.63	327.46	2.06	-33.22
m35	323,131.6	831.2	1.85	-37.04	2.36	284.62	1.83	-12.00
m36	284,734.4	826.8	1.72	-26.64	2.26	286.65	1.83	-11.42
m37	922,964.1	532.4	9.25	105.71	12.11	500.45	60.40	66.53
m38	344,806.7	664.0	4.85	-24.77	5.93	381.45	4.25	26.05
m39	374,740.6	821.0	2.25	-32.38	2.32	289.38	2.63	-5.85
m40	581,945.7	1,036.8	13.26	-28.35	15.36	208.33	54.51	-3.53

Table 11: Comparison of different Relatedness Levels, Class M, 5 Agents

	L = 1		L = 2		L = 3		uncapacitated, L = 2	
Instance	Cost	Time	Relative cost reduction (%)	Relative time increase (%)	Relative cost reduction (%)	Relative time increase (%)	Relative cost reduction (%)	Relative time increase (%)
m01	336,291.1	731.8	24.09	379.07	24.47	349.52	53.31	408.91
m02	152,684.4	826.8	3.61	99.15	-3.25	297.87	5.96	265.60
m03	151,971.0	463.6	13.37	819.28	14.00	609.58	14.57	544.18
m04	292,494.9	1,589.6	47.68	173.31	51.19	106.95	57.74	119.41
m05	173,527.2	958.6	3.75	170.37	2.94	243.17	4.98	289.19
m06	180,388.8	897.0	4.23	182.01	3.59	266.73	18.68	227.11
m07	184,429.2	677.4	18.02	410.42	16.83	385.62	19.10	668.55
m08	137,013.4	781.8	7.16	258.02	3.18	320.77	18.27	272.58
m09	157,021.4	776.6	6.64	539.69	0.00	323.59	16.69	420.99
m10	285,496.8	2,685.2	26.74	-12.36	32.27	22.51	48.32	49.17
m11	293,476.3	2,486.0	2.49	109.49	0.88	32.33	3.46	327.34
m12	318,664.1	2,511.8	3.52	141.19	-2.02	30.97	14.52	228.24
m13	309,846.4	2,515.2	19.81	107.95	19.43	30.79	24.77	288.77
m14	597,711.7	2,782.6	26.43	31.30	34.09	18.22	40.52	57.23
m15	341,562.4	2,942.7	10.31	196.00	10.95	11.79	14.66	185.33
m16	353,899.2	2,569.0	14.89	141.54	12.58	28.05	20.14	164.87
m17	283,693.0	2,231.8	6.56	113.87	3.21	47.40	7.36	215.06
m18	372,034.2	2,252.2	3.79	306.59	3.26	46.06	9.74	225.62
m19	347,850.7	2,288.4	5.54	167.50	0.35	43.75	8.76	189.61
m20	356,467.8	2,782.4	2.13	75.96	-4.77	18.23	11.30	158.14
m21	115,181.2	523.4	2.44	299.20	0.99	528.51	3.38	264.08
m22	149,528.0	631.8	3.07	280.09	2.31	420.67	3.75	300.06
m23	339,321.3	796.4	1.87	430.89	-0.01	313.06	54.50	260.55
m24	215,127.1	868.0	1.68	344.75	-1.03	278.99	32.79	158.59
m25	143,589.4	496.2	11.43	234.02	11.03	562.96	12.01	432.16
m26	154,097.5	463.6	3.32	418.81	0.90	609.58	3.87	393.87
m27	150,022.5	598.6	2.99	299.00	1.14	449.55	5.97	244.57
m28	125,144.3	587.0	5.29	400.34	2.81	460.41	17.87	331.04
m29	260,832.3	1,015.8	21.91	239.79	24.66	223.84	53.18	143.73
m30	188,287.5	715.8	5.83	688.43	1.73	359.57	32.51	273.15
m31	333,892.7	1,737.2	-0.06	114.05	0.19	89.36	19.93	180.88
m32	231,029.6	1,321.4	3.84	224.61	0.69	148.95	6.38	282.73
m33	337,326.1	1,589.2	11.24	217.09	6.23	107.00	18.96	160.46
m34	264,581.6	1,439.2	2.59	254.91	0.46	128.57	3.38	321.15
m35	287,075.8	1,629.8	2.15	138.88	0.21	101.84	3.51	260.56
m36	253,666.4	1,523.2	1.45	99.16	1.31	115.97	4.86	273.54
m37	940,792.3	2,250.0	14.66	166.13	16.11	46.20	65.36	125.81
m38	287,120.8	1,608.2	2.97	252.87	-0.35	104.55	4.42	444.88
m39	331,948.6	1,620.0	2.97	115.84	2.27	103.06	5.03	277.48
m40	551,876.9	9,893.4	18.68	-21.83	17.61	-66.75	58.44	-49.30

Abstract

Collaborative operations planning is a key element of modern supply chains, where firms are forced to cut inefficiencies and improve profitability in order to stay in business. This thesis introduces the collaborative, centralized, multi-level, capacitated lotsizing problem with setup carryover (CCMLCLSP). Collaborative, in this context, means that agents try to minimize total costs rather than individual costs and share all information with all other agents and transshipments of intermediary products may take place. Companies are faced with this problem whenever they produce more than one product and can reduce their fixed costs through planning decisions. Although firms are rarely voluntarily giving up their private information, it is vital to know what the potential benefit of doing so would be. The purpose of this approach is to provide reference values for future research of this problem. A Fix-and-Optimize approach (FO) was chosen, where each subproblem is selected using a combination of deterministic and random criteria. As there were no instance sets available for this problem type, known instances for the decentralized, multi-level, uncapacitated lotsizing problem (DMLULSP) were modified to fit this new problem variant. Our approach shows consistent results both in terms of solution quality and computation time, though this trait becomes less apparent as the problem size increases. Out of the 352 instances, 272 were solved – the small and medium sized instances – as our approach was not suited to compute results in a reasonable timespan for the large instances.

Zusammenfassung

In modernen Versorgungsketten/Supply Chains müssen Firmen vermehrt mit ihren Partnern vernetzt arbeiten, um Kosten zu mindern und Profitabilität und Effizienz zu steigern. Diese Arbeit stellt das Kollaborative, Zentralisierte, Multi-Level, Kapazitierte Losgrößenproblem mit Rüstkostenübertragung (CMLCLSP) vor. Kollaborativ bedeutet hier, dass die Agenten versuchen die Kosten des gesamten Systems zu minimieren, anstatt nur der eigenen Kosten. Dabei teilen sie alle Informationen mit allen anderen Agenten und Zulieferung von Zwischenprodukten untereinander darf stattfinden. Firmen stehen diesem Problem dann gegenüber, wenn sie mehr als ein Produkt produzieren und ihre Fixkosten durch kurz- oder mittelfristige Entscheidungen senken können. Obwohl Unternehmen selten freiwillig Informationen preisgeben, ist es dennoch essenziel zu wissen, wie hoch der potenzielle Vorteil daraus sein könnte. Ein Fix-and-Optimize Verfahren (FO) wurde ausgewählt, in dem jedes Subproblem anhand von einer Kombination aus deterministischen und zufälligen Kriterien bestimmt wird. Da es für diesen Problemtyp keine Testinstanzen gab, wurden bekannte Instanzen des Dezentralisierten, Multi-Level, Unkapazitierten Lösgrößenproblems abgewandelt, um unserer neuen Variante zu entsprechen. Unsere Methode zeigt beständige Ergebnisse sowohl in den Berechnungszeiten als auch der Lösungsqualität, jedoch werden sie weniger stetig bei den größeren Beispielen. Von den 352 Instanzen konnten 272 gelöst werden – die kleinen und mittleren Instanzen – da sich zeigte, dass unsere Methode nicht geeignet ist, die großen Instanzen in einem annehmbaren Zeitrahmen zu lösen.

Sources

Absi, N., Detienne, B., & Dauzère-Pérès, S. (2013). Heuristics for the multi-item capacitated lot-sizing problem with lost sales. *Computers & Operations Research*, 40(1), 264-272.

Arkin, E., Joneja, D., Roundy, R., (1989). Computational complexity of uncapacitated multi-echelon production planning problems. *Operations Research Letters 8* (2), 61 - 66.

Barany, I., Van Roy, T., & Wolsey, L. (1984). Uncapacitated lot-sizing: The convex hull of solutions. In B. Korte, & K. Ritter (Eds.), Mathematical programming at Oberwolfach II . In *Mathematical Programming Studies: 22* (pp. 32–43). Berlin, Heidelberg: Springer.

Berretta, R., Rodriguez, L.F., (2004). A memetic algorithm for a multistage capacitated lot sizing problem, *International Journal of Production Economics* 87, 67-81.

Billington, P.J., McClain, J.O., and Thomas, L.J., (1983). Mathematical programming approaches to capacity-constrained MRP systems: review. *Formulation and Problem Reduction, Management Science*, 29, 1126–1141.

Brahimi, N., Absi, N., Dauzère-Pérès, S., & Nordli, A. (2017). Single-item dynamic lot-sizing problems: An updated survey. *European Journal of Operational Research*, 263(3), 838-863.

Buer, T., Homberger, J., Gehring, H., (2013). A collaborative ant colony metaheuristic for distributed multi-level uncapacitated lot-sizing. *International Journal of Production Research* 51 (17), 5253-5270.

Buer, T., Ziebuhr, M., & Kopfer, H. (2015). A coordination mechanism for a collaborative lot-sizing problem with rivaling agents. In *Logistics Management* (pp. 325-337). Springer, Cham.

Buschkühl, L., Sahling, F., Helber, S., Tempelmeier, H., (2010). Dynamic capacitated lot-sizing problems: a classification and review of solution approaches. *OR Spectrum* 32 (2), 231-261.

Cattrysse, D., Maes, J., Van Wassenhove, L.N., (1990). Set partitioning and column generation heuristics for capacitated dynamic lot sizing. *European Journal of Operational Research* 46(1):38–47.

Chen, H. (2015). Fix-and-optimize and variable neighborhood search approaches for multi-level capacitated lot sizing problems. *Omega*, *56*, 25-36.

Chung, C.-S., Flynn, J., & Lin, C.-H. M. (1994). An effective algorithm for the ca- pacitated single item lot size problem. *European Journal of Operational Research*, 75 (2), 427–440.

Coleman, B. J., & McKnew, M. A. (1991). An improved heuristic for multilevel lot sizing in material requirements planning. *Decision Sciences*, 22(1), 136-156.

Dellaert, N., & Jeunet, J. (2000a). Solving large unconstrained multilevel lot-sizing problems using a hybrid genetic algorithm. *International Journal of Production Research*, 38(5), 1083-1099.

Dellaert, N., & Jeunet, J. (2000b). A genetic algorithm to solve the general multi-level lot sizing problem with time varying costs. *International journal of production Economics* 68, 241-257.

DeMatteis, J. (1968). An economic lot-sizing technique I: The part-period algorithm. *IBM Systems Journal*, 7 (1), 30–38.

Drechsel, J., & Kimms, A. (2011). Cooperative lot sizing with transshipments and scarce capacities: solutions and fair cost allocations. *International Journal of Production Research*, 49(9), 2643-2668.

Erenguc, S. S., & Tufekci, S. (1987). A branch and bound algorithm for a single-item multi-source dynamic lot sizing problem with capacity constraints. *IIE Transactions*, 19 (1), 73-80.

Fleischmann, B., & Meyr, H. (1997). The general lotsizing and scheduling problem. *Operations-Research-Spektrum*, 19(1), 11-21.

Gansterer, M., & Hartl, R. F. (2019). The collaborative multi-level lot-sizing problem with cost synergies. *International Journal of Production Research*, 1-18.

Gansterer, M., Födermayr, P., Hartl, R.F. (2020), The collaborative capacitated multi-level lot sizing problem, *Technical Report*, University of Klagenfurt.

Hardin, J. R., Nemhauser, G. L., & Savelsbergh, M. W. P. (2007). Analysis of bounds for a capacitated single-item lot-sizing problem. *Computers & Operations Research*, 34 (6), 1721–1743.

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.

Hernández, W. and G. Süer, (1999). Genetic Algorithms in Lot Sizing Decisions, *Proceedings of theCongress on Evolutionary Computing (CEC99)*, July 6-9, Washington DC.

Hindi, KS. (1995) Solving the single-item, capacitated dynamic lot-sizing problem with startup and reservation costs by tabu search. *Computers and Industrial Engineering* 28(4): 701-707.

Hindi KS. (1996) Solving the CLSP by a tabu search heuristic. *Journal of the Operational Research Society* 47(1):151–161.

Homberger, J., (2010). Decentralized multi-level uncapacitated lot-sizing by automated negotiation. *4OR* 8 (2), 155-180.

Karimi, B., Ghomi, S. F., & Wilson, J. M. (2003). The capacitated lot sizing problem: a review of models and algorithms. *Omega*, 31(5), 365-378.

Lambert, D. M., Emmelhainz, M. A., Gardner, J. T., (1999), Building Successful Partnerships, *Journal of Business Logistics*, Vol. 20, No. 1, pp. 165-181.

Lee, S., & Kumara, S., (2007). Decentralized supply chain coordination through auction markets: dynamic lot-sizing in distribution networks. *International Journal of Production Research* 45 (20), 4715-4733.

Levi, R., Roundy, R. O., & Shmoys, D.B., (2006). Primal-dual algorithms for determin- istic inventory problems. *Mathematics of Operations Research*, 31 (2), 267–284.

Loparic, M., Marchand, H., & Wolsey, L. A. (2003). Dynamic knapsack sets and capacitated lot-sizing. *Mathematical Programming*, 95 (1), 53–69.

Lotfi, V. , & Yoon, Y.-S. (1994). Algorithm for the single-item capacitated lot-sizing problem with concave production and holding costs. *Journal of the Operational Research Society*, 45 (8), 934–941.

Manne, A. S. (1958). Programming of economic lot sizes. *Management Science* 4(2):115-135

Meca, A., & Timmer, J. (2008). Supply chain collaboration. In *Supply Chain*. IntechOpen.

Mishra, S. K., Sahithi, V. V. D., & Rao, C. S. P. (2016). A hybrid binary particle swarm optimization for large capacitated multi item multi level lot sizing (CMIMLLS) problem. *MS&E*, 149(1), 012040.

Morash, E. A., Droge, C. L., & Vickery, S. K. (1996). Strategic logistics capabilities for competitive advantage and firm success. *Journal of business Logistics*, 17(1), pp. 1-21

Narus, J. A., Anderson, J. C. (1996), Rethinking Distribution: Adaptive Channels, *Harvard Business Review*, Vol. 74, No. 4, pp. 112-120.

Pitakaso, R., Almeder, C., Dörner, K. F., Hartl, R. F., (2007). A max-min ant system for unconstrained multi-level lot-sizing problems. *Computers & Operations Research* 34 (9), 2533-2552.

Simatupang, T. M. and Sridharan, R. (2002). The collaborative supply chain. In: *The International Journal of Logistics Management* 13.1, pp. 15-30.

Stadtler, H. J. (2009). A framework for collaborative planning and state-of-the-art. *OR Spectrum* 31 (1), 5-30.

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.

Tempelmeier, H., Helber, S., (1994). A heuristic for dynamic multi-item multi-level capacitated lotsizing for general product structures. *European Journal of Operational Research* 75 (2), 296-311.

van Hoesel, S. , Wagelmans, A. , & Kolen, A. (1991). A dual algorithm for the economic lot-sizing problem. *European Journal of Operational Research*, 52 (3), 315–325 .

Visser, L. (2007). Logistics Collaboration between Shippers and Logistics Service Providers. Observations in the chemical industry. Tech. rep. Fontys University of Applied Sciences.

Wagner, H. M., & Whitin, T. M. (1958). Dynamic version of the economic lot size model. *Management science*, 5(1), 89-96.

Wemmerlöv, U. (1983). The part-period balancing algorithm and its look ahead- look back feature: A theoretical and experimental analysis of a single stage lot- sizing procedure. *Journal of Operations Management*, 4 (1), 23–39.