

Document downloaded from:

<http://hdl.handle.net/10251/184789>

This paper must be cited as:

Lujak, M.; Fernandez Gil, A.; Onaindia De La Rivaherrera, E. (2021). Spillover Algorithm: A decentralised coordination approach for multi-robot production planning in open shared factories. *Robotics and Computer-Integrated Manufacturing*. 70:1-12.  
<https://doi.org/10.1016/j.rcim.2020.102110>



The final publication is available at

<https://doi.org/10.1016/j.rcim.2020.102110>

Copyright Elsevier

Additional Information

## Highlights

### **Spillover Algorithm: A Decentralised Coordination Approach for Multi-Robot Production Planning in Open Shared Factories**

Marin Lujak, Alberto Fernández, Eva Onaindia

- Formulation of the decentralised MCLSP-BOSI problem for open and shared factories
- Spillover Algorithm: multi-agent heuristics for MCLSP-BOSI that protects private information
- Spillover Algorithm: high computational efficiency and good solution quality

# Spillover Algorithm: A Decentralised Coordination Approach for Multi-Robot Production Planning in Open Shared Factories

Marin Lujak<sup>a</sup>, Alberto Fernández<sup>b</sup> and Eva Onaindia<sup>c</sup>

<sup>a</sup>IMT Lille Douai, Université de Lille, CERI Numerique, 59000 Lille, France

<sup>b</sup>University Rey Juan Carlos, 28933 Móstoles, Madrid, Spain

<sup>c</sup>Valencian Research Institute for AI, Universitat Politècnica de València, 941, 46022 Valencia, Spain

---

## ARTICLE INFO

### Keywords:

Capacitated production planning  
multi-robot systems  
multi-agent coordination  
decentralised algorithm  
shared factories

## Abstract

Open and shared manufacturing factories typically dispose of a limited number of industrial robots and/or other production resources that should be properly allocated to tasks in time for an effective and efficient system performance. In particular, we deal with the dynamic capacitated production planning problem with sequence independent setup costs where quantities of products to manufacture need to be determined at consecutive periods within a given time horizon and products can be anticipated or back-ordered related to the demand period. We consider a decentralised multi-agent variant of this problem in an open factory setting with multiple owners of robots as well as different owners of the items to be produced, both considered self-interested and individually rational. Existing solution approaches to the classic constrained lot-sizing problem are centralised exact methods that require sharing of global knowledge of all the participants' private and sensitive information and are not applicable in the described multi-agent context. Therefore, we propose a computationally efficient decentralised approach based on the spillover effect that solves this NP-hard problem by distributing decisions in an intrinsically decentralised multi-agent system environment while protecting private and sensitive information. To the best of our knowledge, this is the first decentralised algorithm for the solution of the studied problem in intrinsically decentralised environments where production resources and/or products are owned by multiple stakeholders with possibly conflicting objectives. To show its efficiency, the performance of the Spillover Algorithm is benchmarked against state-of-the-art commercial solver CPLEX 12.8.


---

## 1. Introduction

The smart factory concept (*e.g.*, Shrouf, Ordieres and Miragliotta (2014); Hozdić (2015)) in Industry 4.0 revolution is opening up new ways of addressing the needs of sustainability and efficiency in the manufacturing industry of today's global economy. In particular, we are seeing the emergence of shared factories (*e.g.*, Jiang and Li (2019)) in which the owners of production resources (robots) do not necessarily produce their own products. Indeed, they may offer their facilities and available production resources therein to various other manufacturers that manufacture their products in the same shared facility. Even more, multiple factories could be linked in a more flexible global virtual facility (*e.g.*, Hao, Helo and Shamsuzzoha (2018)).

In this paper, we consider an (intrinsically decentralised) shared factory scenario that requires decentralised methods for efficient allocation of robots (production resources) of the same shared manufacturing plant to individually rational and self-concerned firms (users) that use them to manufacture their products in a given time horizon. We consider open firms that may collaborate at times on common projects based on individual interest such that the factory's production capacity may vary from one period to another based on available shared resources. The requisite for shared factories is that they follow the concept of an open firm, *i.e.*, a common standard for the production of components and their interfaces is shared by distinct firms and each firm is open to all industry participants (*e.g.*, Farrell, Monroe and Saloner (1998); Arora and Bokhari (2007)). An important aspect to consider in this context is that the different firms participating in such manufacturing ecosystems are not willing to share information or business strategy. Therefore, solutions in which a central coordinator receives all the information regarding resource availabilities and production demands are not applicable.

---

 marin.lujak@imt-lille-douai.fr (M. Lujak); alberto.fernandez@urjc.es (A. Fernández); onaindia@dsic.upv.es (E. Onaindia)

ORCID(s): 0000-0001-8565-9194 (M. Lujak); 0000-0002-8962-6856 (A. Fernández); 0000-0001-6931-8293 (E. Onaindia)

Production planning considers the best use of resources to satisfy the demand in a given planning time horizon. A production plan must meet conflicting objectives of guaranteeing service quality and minimizing production and inventory costs. To this end, the Constrained Lot-Sizing Problem (CLSP) answers the question of when and how much of each product (item) must be produced so that the overall costs are minimised (e.g., Buschkühl, Sahling, Helber and Tempelmeier (2010); Karimi, Ghomi and Wilson (2003)).

In this paper, we focus on deterministic, dynamic, single-level, multi-item CLSP with back orders and sequence independent setup costs, which we will refer to as MCLSP-BOSI (e.g., Pochet and Wolsey (2006)). The concept of dynamic in MCLSP-BOSI means that the production demands vary through time; single-level indicates that the end product or item is directly manufactured in one step from raw materials with no intermediate sub-assemblies and no item can be a predecessor of another item; multi-item refers to the existence of multiple product types and capacitated denotes a scarcity between the available limited manufacturing resources and the production demand. Moreover, the presence of back orders implies that it is possible to satisfy the demand of the current period in future periods by paying shortage costs (back-ordering). We consider a lost-sales inventory model where a unit of a back-ordered product that cannot be produced at the period of its demand is requested nonetheless for when resources become available again in some later period of the planning time horizon, while a backlogged unit is not produced at all and the demand is lost resulting in lost sales (e.g., Bijvank and Vis (2011)). The sequence independent setup costs include the setup of robots and other production resources for producing an item at some period independently of its previous production dynamics (e.g., Pochet and Wolsey (2006)). Contrary to most of the related work, in our studied MCLSP-BOSI model, all costs are time-dependent.

The studied MCLSP-BOSI is a variant of the CLSP where production can be anticipated or delayed in relation to the product demands, resulting in time-dependent holding and back order costs, respectively. The inclusion of back orders is crucial in the contexts of high demands, since, otherwise, no feasible production plan would exist. Item back-ordering results in not producing (lost sales) if the overall demand is higher than the production capacity over a given time horizon. The main challenge in back-ordering is to select lost sales (if any), the items to back-order, the ones to produce at the time of demand release, and the ones beforehand (Drexl and Kimms (1997); Jans and Degraeve (2008)). For an overview of the CLSP and its variations, we refer the reader to excellent descriptions in Bitran and Yanasse (1982); Pochet and Wolsey (2006); Quadt and Kuhn (2008).

The MCLSP-BOSI is an NP-hard problem (Chen and Thizy (1990); Bitran and Yanasse (1982); Dixon and Silver (1981)) and has been mostly addressed by heuristic approximations that do not guarantee an optimal solution, but find a reasonably good solution in a moderate amount of computation time. In its classical mathematical form, MCLSP-BOSI is intrinsically centralised, i.e., a single decision-maker has the information of all the production resources and all the items at disposal. This formulation and related state-of-the-art solution approaches are not applicable in intrinsically decentralised contexts, such as open and shared factories.

With the aim of finding a production plan in intrinsically decentralised open and shared factories, in this paper, we mathematically formulate a decentralised variant of the MCLSP-BOSI problem. As a solution approach to this problem, we propose the *Spillover Algorithm*, a decentralised and heuristic production planning method whose approximation scheme is based on the *spillover effect*. The spillover effect is defined as a situation that starts in one place and then begins to expand or has an effect elsewhere. In ecology, this term refers to the case where the available resources in one habitat cannot support an entire insect population, producing an “overflow”, flooding adjacent habitats and exploiting their resources (Rand, Tylianakis and Tschamntke (2006)). In economy, it is related to the interconnection of economies of different countries where a slight change in one country’s economy leads to the change in the economy of others.

We leverage the spillover effect in a multi-agent system considering both the self-concerned owners of production resources and the self-concerned owners of the items’ demands. For a better explanation of the proposed multi-agent system, the spillover effect is modelled in a metaphor of a liquid flow network with buffers, where the capacity of each buffer represents available production capacity at a specific period. Liquid agents assume the role of demand owners, with liquid volume proportional to the production demand. Buffer agents, on the other hand, act in favour of resource owners by allocating their production capacities to liquid agents. Liquid demand that cannot be allocated where requested, spills over through a tube towards other neighbouring buffers based on the rules that dictate how the spillover proceeds. In the proposed Spillover algorithm, we assume that all agents are truthful (non-strategic), i.e. the information they provide is obtained from their true internal values and is not strategically modified to obtain better resource allocation.

To the best of our knowledge, this is the first decentralised heuristic approach that tackles the high computa-

tional complexity of the studied MCLSP-BOSI problem with time-dependent costs and is applicable in intrinsically decentralised shared and open factories.

This paper is organized as follows. We give an overview of the related work in Section 2. In Section 3, we present the background of the studied classic and centralised MCLSP-BOSI problem. In Section 4, we present the motivation and mathematical formulation of the decentralised MCLSP-BOSI problem and discuss our problem decomposition and decentralisation approach. Our multi-agent architecture together with the Spillover Algorithm is presented in Section 5. Section 6 presents the evaluation results of simulated experiments whose setup was taken from related work, while Section 7 concludes the paper and outlines future work.

## 2. Related work

Due to its complexity, the literature on the classic (centralised) MCLSP-BOSI problem is scarce and can be classified into: (a) *period-by-period* heuristics, which are special-purpose methods that work in the time horizon from its first to the last period in a single-pass construction algorithm; (b) *item-by-item* heuristics, which iteratively schedule a set of non-scheduled items; and (c) *improvement heuristics* that are mathematical-programming-based heuristics that start with an initial (often infeasible) solution for the complete planning horizon usually found by uncapacitated lot sizing techniques, and then try to enforce feasibility conditions, by shifting lots from period to period at minimal extra cost. The aim of improvement heuristics is to maximise cost savings as long as no new infeasibilities are incurred (e.g., Maes and Wassenhove (1998); Buschkühl et al. (2010)). However, these general heuristic approaches in many cases do not guarantee finding a feasible solution of the MCLSP-BOSI problem even if one exists ( Millar and Yang (1994)).

Pochet and Wolsey (1988) approach this problem by a shortest-path reformulation solved by integer programming and by a plant location reformulation for which they propose a cutting plane algorithm. Both approaches produce near optimal solutions to large problems with a quite significant computational effort. Millar and Yang (1994) present two Lagrangian-based algorithms to solve the MCLSP-BOSI based on Lagrangian relaxation and Lagrangian decomposition. However, their MCLSP-BOSI model is limited as it considers setup, holding, and back order costs while considering production costs fixed and constant throughout the planning horizon. They find a feasible primal solution at each Lagrangian iteration and guarantee finding a feasible solution if one exists with valid lower bounds, thus, guaranteeing the quality of the primal solutions.

Karimi, Ghomi and Wilson (2006) present a tabu search heuristic method consisting of four parts that provide an initial feasible solution: (1) a demand shifting rule, (2) lot size determination rules, (3) checking feasibility conditions and (4) setup carry over determination. The found feasible solution is improved by adopting the setup and setup carry over schedule and re-optimising it by solving a minimum-cost network flow problem. The found solution is used as a starting input to a tabu search algorithm. An overview of metaheuristic approaches to dynamic lot sizing can be found in, e.g., Jans and Degraeve (2007). Furthermore, there exist heuristic-based algorithms based on local optima (e.g., Cheng, Madan, Gupta and So (2001)), approaches that examine the Lagrangian relaxation and design heuristics to generate upper bounds within a subgradient optimisation procedure (e.g., Süral, Denizel and Wassenhove (2009)), or a genetic algorithm for the multi-level version of this problem that uses fix-and-optimise heuristic and mathematical programming techniques (e.g., Toledo, de Oliveira and França (2013)).

Zangwill (1966) presents a CLSP model with backlogging where the demand must be satisfied within a given maximum delay while Pochet and Wolsey (1988) study uncapacitated lot sizing problem with backlogging and present the shortest path formulation and the cutting plane algorithm for this problem. Furthermore, Quadt and Kuhn (2008) study a MCLSP-BO with setup times and propose a solution procedure based on a novel aggregate model, which uses integer instead of binary variables. The model is embedded in a period-by-period heuristic and is solved to optimality or near-optimality in each iteration using standard procedures (CPLEX). A subsequent scheduling routine loads and sequences the products on the parallel machines. Six versions of the heuristic were presented and tested in an extensive computational study showing that the heuristics outperforms the direct problem implementation as well as the lot-for-lot method.

Gören and Tunali (2018) integrate several problem-specific heuristics with fix-and-optimise (FOPT) heuristic. FOPT heuristic is a MIP-based heuristic in which a sequence of MIP models is solved over all real-valued decision variables and a subset of binary variables. Time and product decomposition schemes are used to decompose the problem. Eight different heuristic approaches are obtained and their performances are shown to be promising in simulations.

From the decision distribution perspective, the above mentioned solution approaches solve the MCLSP-BOSI problem by a single decision-maker having total control over and information of the production process and its elements. Since the MCLSP-BOSI problem is highly computationally expensive, decomposition techniques may be used for simplifying difficult constraints. They were applied by Millar and Yang (1994) who used Lagrangian relaxation of setup constraints, while Thizy and Van Wassenhove (1985) studied the multi-item CLSP problem without back orders and used Lagrangian relaxation of capacity constraints to decompose the problem into single item uncapacitated lot sizing sub-problems solvable by the Wagner-Whitin algorithm (see, Wagner and Whitin (1958)) and its improvements (e.g., Aggarwal and Park (1993); Brahimi, Absi, Dauzère-Pérès and Nordli (2017)). Moreover, Lozano, Larraneta and Onieva (1991), Diaby, Bahl, Karwan and Zionts (1992), and Hindi (1995) are some other works using Lagrangian relaxation. A broad survey of single-item lot-sizing problems can be found in Brahimi et al. (2017). All these Lagrangian relaxation and decomposition methods focus on solving the CLSP problem centrally by one decision maker on a single processor. State-of-the-art centralised heuristic solution approaches and related surveys can be found in, e.g., Karimi et al. (2003), Karimi et al. (2006), Quadt and Kuhn (2009), Gören and Tunali (2018), and Jans and Degraeve (2008).

While the MCLSP-BOSI state-of-the-art solution approaches improve solution quality in respect to previous work, they are all centralised approaches that can only be applied for a single decision maker. Giordani, Lujak and Martinelli (2009) and Giordani, Lujak and Martinelli (2013) deal with distributed multi-agent coordination in the constrained lot-sizing context. Both works assume the existence of a single robot (production resource) owner agent responsible for achieving globally efficient robot allocation by interacting with product (item) agents through an auction. The problem decomposition is done to gain computational efficiency since item agents can compute their bids in parallel. The allocation of limited production resources is done through the interaction between competing item agents and a robot owner (a single autonomous agent) having available all the global information. However, these approaches are not adapted to intrinsically decentralised scenarios as is the case of competing shared and open factories since the resource allocation decisions are still made by a single decision maker (robot owner) based on the bids of the item agents that are computed synchronously.

In shared and open factories, both the interests of rational self-concerned owners of the production resources as well as the interests of self-interested rational owners of the items' demands must be considered with minimal exposure of their sensitive private information. Distributed constraint programming approaches are computationally expensive (e.g., Faltings (2006)) and are, therefore, not applicable in such large systems producing multitudes of items.

Cloud manufacturing is an interesting paradigm that integrates distributed resources and capabilities, encapsulating them into services that are managed in a centralized way Xu (2012). Important issues with cloud manufacturing are how to schedule multiple manufacturing tasks to achieve optimal system performance or how to manage the resources precisely and timely. Workload-based multi-task scheduling methods Liu, Xu, Zhang, Wang and Zhong (2017) or approaches of resource socialization Qian, Zhang, Sun, Rong and Zhang (2020) have been proposed to address the limitations inherent of the centralized structure of cloud platforms. Unlike cloud manufacturing approaches, we advocate for a decentralized approach to the manufacturing planning problem that achieves a good quality solution while protecting sensitive private information of the stakeholders.

In this paper we present a first solution approach for the decentralised variant of the MCLSP-BOSI problem; a sketchy outline of this approach can be seen in Lujak, Fernández and Onaindia (2020).

### 3. Overview of the studied MCLSP-BOSI problem

The objective of the MCLSP-BOSI is to find a production schedule for a set  $I$  of different types of items that minimizes the total back order, holding inventory, production, and setup costs over a given finite time horizon  $T$  subject to demand and capacity constraints. This problem belongs to the class of deterministic dynamic lot-sizing problems well known in the inventory management literature (e.g., Pochet and Wolsey (2006); Buschkühl et al. (2010); Jans and Degraeve (2008); Karimi et al. (2003); Millar and Yang (1994); Quadt and Kuhn (2008)). In the following, we give the formulation of the baseline problem.

Before the beginning of the time horizon  $T$ , the customer demand arrives for each item  $i \in I$  and each period  $t \in T$  of the time horizon and is received as incoming order. Items  $i \in I$ , with given resource requirements  $r_i$ , are produced at the beginning of each period by robots (manufacturing resources) of a given limited and time-varying production capacity  $R_t$  known in advance for the whole given time horizon  $T = \{1, \dots, |T|\}$ . Open and shared factories may collaborate at times based on individual interest such that the factory's production capacity may vary through time

**Table 1**

List of symbols used in the MCLSP-BOSI problem modelling

Sets and indices	
$T$	planning time horizon, set of consecutive periods $t \in T$ of equal length
$I$	set of items $i \in I$
Parameters	
$d_{it}$	demand for item $i$ at period $t$
$r_i$	resource requirement for production of a unit of item $i$
$c_{it}$	unitary production cost for item $i$ at period $t$
$h_{it}$	unitary holding cost for item $i$ at period $t$
$b_{it}$	unitary back order cost for item $i$ at period $t$
$s_{it}$	fixed setup cost for item $i$ at period $t$
$R_t$	production capacity at period $t$
Decision variables	
$u_{it}$	number of items $i$ produced at period $t$
$x_{it}^+, x_{it}^-$	storage and back order inventory level of item $i$ at the end of period $t$
$y_{it}$	equals 1 if item $i$ is produced at period $t$ , 0 otherwise.
Objective function	
$z(\mathbf{u})$	generalized cost objective function of solution $\mathbf{u}_{it}$ , $i \in I, t \in T$

based on available shared resources. We assume that a time-varying and sequence independent setup cost  $s_{it}$  that includes the setup of the production resources for the production of item  $i$  occurs if item  $i$  is produced at period  $t$  and a linear production cost  $c_{it}$  is added for each unit produced of item  $i$  at the same period. Setup costs usually include the preparation of the robots, *e.g.*, changing or cleaning a tool, expediting, quality acceptance, labour costs, and opportunity costs since setting up the robot consumes time. Moreover, demand  $d_i$  for each item  $i \in I$  can be anticipated or delayed in respect to the period in which it is requested. If anticipated, it is at the expense of a linear holding cost  $h_{it}$  for each unit of item  $i$  held in inventory per unit period and if delayed, a linear back order cost  $b_{it}$  is accrued for every unit back ordered per unit period. The inventory level and back orders are measured at the end of each period.

Each item demand  $\mathbf{d}_i = \{d_{i1}, \dots, d_{i|T|}\}$  is associated with inventory level  $x_{it}$  at the end of period  $t \in T$  that can be positive (if a stock of completed items  $i$  is present in the buffer at the end of period  $t$ ), zero (no stock and no back order), or negative (if a back order of demands for item  $i$  is in the queue at the end of period  $t$ ). The inventory level is increased by production quantity  $u_{it}$  and reduced by demand  $d_{it}$  at each period  $t$ .

Using a standard notation, we denote  $x_{it}^+ := \max\{x_{it}, 0\}$  as the stock level, and  $x_{it}^- := \max\{-x_{it}, 0\}$  as the back order level at the end of period  $t$ . Stock  $x_{it}^+$  available at the end of period  $t$  for product  $i$  corresponds to the amount of product  $i$  that is physically stored on stock and is available for the demand fulfilment. Notice that, for each period  $t$  and item  $i$ ,  $x_{it}^+ \cdot x_{it}^- = 0$ , *i.e.*, positive back order value and storage value cannot coexist at the same time since if there is a back order of demand, it will be first replenished by the items we have in the storage and then, we will produce the rest of the demand.

The problem is to decide when and how much to produce in each period of time horizon  $T$  so that all demand is satisfied at minimum cost. The mathematical program ( $P$ ) of the studied MCLSP-BOSI problem is given in the following; related sets, indices, parameters, and decision variables are summarised in Table 1.

( $P$ ):

$$z(\mathbf{u}) = \min \sum_{i \in I, t \in T} (h_{it}x_{it}^+ + b_{it}x_{it}^- + c_{it}u_{it} + s_{it}y_{it}) + \sum_{i \in I} (b_{i,|T|+1}x_{i,|T|+1}^- + h_{i,|T|+1}x_{i,|T|+1}^+) \quad (1)$$

subject to:

$$x_{it}^+ - x_{it}^- = x_{i,t-1}^+ - x_{i,t-1}^- + u_{it} - d_{it}, \quad \forall i \in I, \forall t \in T \quad (2)$$

$$x_{i,|T|+1}^- - x_{i,|T|+1}^+ = x_{i,|T|}^- - x_{i,|T|}^+, \quad \forall i \in I \quad (3)$$

$$\sum_{i \in I} u_{it} r_i \leq R_t, \quad \forall t \in T \quad (4)$$

$$u_{it} \leq y_{it} \sum_{k \in T} d_{ik}, \quad \forall i \in I, \forall t \in T \quad (5)$$

$$u_{it} \in \mathbb{Z}_{\geq 0}, \quad \forall i \in I, \forall t \in T \quad (6)$$

$$y_{it} \in \{0, 1\}, \quad \forall i \in I, \forall t \in T \quad (7)$$

$$x_{it}^+, x_{it}^- \in \mathbb{Z}_{\geq 0}, \quad \forall i \in I, \forall t \in \{1, \dots, |T| + 1\} \quad (8)$$

The values of  $x_{i0}^+$  and  $x_{i0}^-$  represent the initial conditions, *i.e.*, the stock and back order level of item  $i$  at the beginning of the planning time horizon.

For this model to accommodate not only back orders but also lost sales (backlog) and surplus (unsold) stock after the end of the planning time horizon  $T$ , in addition to the sum of the sustained costs in a given time horizon, in (1), there is also an additional backlog (lost sales) cost *i.e.*,  $\sum_{i \in I} b_{i,|T|+1} x_{i,|T|+1}^-$  and surplus (unsold) stock cost *i.e.*,  $\sum_{i \in I} h_{i,|T|+1} x_{i,|T|+1}^+$ . Generally, the decisions on surplus stock  $x_{i,|T|+1}^+$  and backlog  $x_{i,|T|+1}^-$  (demand that is not produced in the time horizon) will depend on the value of surplus stock cost  $h_{i,|T|+1}$  and backlog cost  $b_{i,|T|+1}$  and their relation to production  $c_{it}$ , holding  $h_{it}$ , and back order  $b_{it}$  costs, as well as the relation between overall demand and the overall production capacity throughout time horizon  $T$ .

Constraints (2) are flow-balance constraints among product demand  $d_{it}$ , stock level  $x_{it}^+$ , back order level  $x_{it}^-$ , and production level  $u_{it}$ , for each period  $t \in T$ . Constraints (3) are the flow-balance constraints for the end of the time horizon  $|T| + 1$ , which is the reason why the domain of index  $t$  of  $x_{it}^-$  and  $x_{it}^+$  in (8) runs in interval  $\{1, \dots, |T| + 1\}$ . Without loss of generality and assuming that for each period  $t$  and item  $i$ , the values of  $h_{it}$  and  $b_{it}$  are non-negative (zero or positive), the constraint  $x_{it}^+ \cdot x_{it}^- = 0$  is implicitly satisfied by the optimal solution, and, hence, omitted in the formulation.

Constraints (4) limit the overall resource usage for the production of all the items not to be greater than the production capacity at period  $t$ . Note that production capacity  $R_t$  of each period  $t \in T$  may vary from one period to another. Contrary to the classic MCLSP-BOSI model, our studied model does not have a strong assumption that the overall demand in the time horizon is lower than the overall production capacity. Here, the following constraints on resource requirements for production of a unit of each item  $i \in I$  hold:

$$r_i \leq \max_{t \in T} \{R_t\}, \quad \forall i \in I, \quad (9)$$

$$R_t \geq \min_{i \in I} \{r_i\}, \quad \forall t \in T. \quad (10)$$



Items  $i$  that violate assumption (9) and periods  $t$  that violate assumption (10) can be removed. By constraints (5), we model independent setups and the fact that if the production is launched for item  $i$  in period  $t$ , the quantity produced should not be larger than the overall demand for item  $i$  in a given time horizon. Constraints (6) and (8) are nonnegativity and integrality constraints on production, back order and storage decision variables, while constraints (7) limit the setup decision variables to binary values, *i.e.*, the fixed setup cost is accrued if there is (any) production at a period.

The solution to problem  $P$  is a production plan composed of a number of items  $u_{it}$  to produce for each item type  $i \in I$  in response to demand  $d_{it}$  in each period  $t \in T$  of a given time horizon.

*Complexity of the MCLSP-BOSI problem.* The single-item capacitated lot-sizing problem is NP-hard even for many special cases (*e.g.*, Florian, Lenstra and Rinnooy Kan (1980); Bitran and Yanasse (1982). Chen and Thizy (1990)) proved that multi-item capacitated lot-sizing problem (MCLSP) with setup times is strongly NP-hard. Additionally, Bitran and Yanasse (1982) present the multiple items capacitated lot size model with independent setups without back orders and resource capacities that change through time. Contrary to our model that assumes non-negative integer values for  $u_{it}$ ,  $x_{it}^+$ , and  $x_{it}^-$ , the model in Bitran and Yanasse (1982) contains no integrality constraints for these variables. The introduction of an integrality constraint leads to a generally NP-complete integer programming problem.

#### 4. Formulation of the decentralised MCLSP-BOSI problem

Problem  $P$  (1)–(8) is a classic and centralised mathematical formulation of the MCLSP-BOSI problem. The decentralised variant of the MCLSP-BOSI problem should consider that both the providers of capacitated and homogeneous robots and their customers (the owners of the produced items) are interested in the transformation of heterogeneous raw materials into heterogeneous final products and thereby both of them should be considered active participants in the production process; no one is willing to disclose its complete information but will share a part of it if this facilitates achieving its local objective. Therefore, they must negotiate resource allocation while exchanging relevant (possibly obsolete) information.

In the following, we propose the formulation of the decentralised MCLSP-BOSI problem.

First, we create a so-called Lagrangian relaxation (*e.g.*, Lemaréchal (2001); Fisher (1981)) of problem ( $P$ ), which we name  $L(P, \Lambda)$ , for which we assume to be tractable. This new modelling allows us to decompose the original problem  $P$  into item-period pair  $(i, t)$  subproblems, where  $i \in I$  and  $t \in T$  (section 4.1). The decomposition based on item-period  $(i, t)$  pairs allows for the agentification of the demand of product type  $i$  at period  $t$  and decentralisation of problem ( $P$ ).

The Lagrangian relaxation is achieved by turning the complicating global capacity constraints (4) into constraints that can be violated at price  $\Lambda = \{\lambda_t | t \in T\}$ , while keeping the remaining (easy) constraints for each product  $i \in I$  and period  $t \in T$ . We dualize the complicating constraints (4), *i.e.*, we drop them while adding their slacks to the objective function with weights  $\Lambda$  (vector of dual variables for constraints (4)-Lagrangian multipliers) and thus create the *Lagrangian relaxation*  $L(P, \Lambda)$  of  $P$ .

Here, we let  $\Lambda$  be the (current) set of nonnegative Lagrange multipliers (resource conflict prices).

The solution of the Lagrangian problem provides a lower bound, while the upper bound is obtained by first fixing the setup variables given by the dual solution and secondly, obtaining the solution from the resulting transportation problem.

Then the Lagrangian relaxation of the MCLSP-BOSI problem can be mathematically modelled as follows:

$(L(P, \Lambda))$ :

$$z^*(\mathbf{u}) = \min \sum_{i \in I, t \in T} (h_{it}x_{it}^+ + b_{it}x_{it}^- + c_{it}u_{it} + s_{it}y_{it}) + \sum_{i \in I} (b_{i,|T|+1}x_{i,|T|+1}^- + h_{i,|T|+1}x_{i,|T|+1}^+) + \sum_{i \in I, t \in T} \lambda_t (u_{it}r_i - R_t) \quad (11)$$

subject to:

$$x_{it}^+ - x_{it}^- = x_{i,t-1}^+ - x_{i,t-1}^- + u_{it} - d_{it}, \quad \forall i \in I, \forall t \in T \quad (12)$$

$$x_{i,|T+1|}^+ - x_{i,|T+1|}^- = x_{i,|T|}^+ - x_{i,|T|}^-, \quad \forall i \in I \quad (13)$$

$$u_{it} \leq y_{it} \sum_{k \in T} d_{ik}, \quad \forall i \in I, \forall t \in T \quad (14)$$

$$u_{it} \in \mathbb{Z}_{\geq 0}, \quad \forall i \in I, \forall t \in T \quad (15)$$

$$y_{it} \in \{0, 1\}, \quad \forall i \in I, \forall t \in T \quad (16)$$

$$x_{it}^+, x_{it}^- \in \mathbb{Z}_{\geq 0}, \quad \forall i \in I, \forall t \in \{1, \dots, |T| + 1\} \quad (17)$$

where the variable, parameter, and constraint descriptions remain the same as in  $(P)$  except for constraints (4) that are relaxed in (11).

#### 4.1. Item-period agent modelling

We decompose the overall production planning problem to subproblems solvable by item-period  $(i, t)$  pair agents. Given a vector  $\Lambda$ , we substitute indices  $a = (i, t)$ . The mathematical program of the problem  $L(P_A, \Lambda)$  reformulated for  $(i, t)$  pair agent  $a \in A$  is given in the following; related sets, indices, parameters, and decision variables are summarised in Table 2.

$(L(P_A, \Lambda))$ :

$$z^*(\mathbf{u}) = \min \sum_{a \in A} \sum_{k \in T} \left( h_{ak} x_{ak}^+ + b_{ak} x_{ak}^- + c_a u_a^k + s_{ak} y_{ak} \right) + \sum_{a \in A} \left( b_{a,|T|+1} x_{a,|T|+1}^- + h_{a,|T|+1} x_{a,|T|+1}^+ \right) + \sum_{k \in T} \lambda_k \left( \sum_{a \in A} u_a^k r_a - R_k \right) \quad (18)$$

subject to:

$$x_{ak}^+ - x_{ak}^- = x_{a,k-1}^+ - x_{a,k-1}^- + u_a^k - d_{ak}, \quad \forall a \in A, \forall k \in T \quad (19)$$

$$x_{a,|T+1|}^+ - x_{a,|T+1|}^- = x_{a,|T|}^+ - x_{a,|T|}^-, \quad \forall a \in A \quad (20)$$

$$u_a^k \leq d_a y_{ak}, \quad \forall a \in A, \forall k \in T \quad (21)$$

$$u_a^k \in \mathbb{Z}_{\geq 0}, \quad \forall a \in A, \forall k \in T \quad (22)$$

$$y_{ak} \in \{0, 1\}, \quad \forall a \in A, \forall k \in T \quad (23)$$

**Table 2**

List of symbols used in the item-period pair agent modelling

Sets and indices	
$T$	planning time horizon, set of consecutive periods $k, t \in T$ of equal length
$A$	set of pairs $a = (i, t)$ , where $i \in I$ and $t \in T$ whose elements $a \in A$ have positive demand value, <i>i.e.</i> , $d_{it} > 0$
Parameters	
$d_a$	demand for pair $a = (i, t)$ , <i>i.e.</i> , $d_a = d_{it}$
$d_{ak}$	demand for pair $a = (i, t)$ at period $k$ , where $d_{ak} = d_{it}$ when $k = t$ , and $d_{ak} = 0$ otherwise
$r_a$	resource requirement for production of a unit of pair $a$ (item $i$ at period $t$ )
$c_a$	unitary production cost for pair $a$
$h_{ak}$	unitary holding cost for pair $a$ at period $k$
$b_{ak}$	unitary back order cost for pair $a$ at period $k$
$s_{ak}$	fixed setup cost for pair $a$ at period $k$
$R_k$	production capacity at period $k$
Decision variables	
$u_a^k$	number of items $i$ demanded at time $t$ ( $a = (i, t)$ ) that are produced at time $k$ . Note that we use $k$ as superindex to avoid confusion between $u_a^k$ and $u_{it}$ from (P), which are not the same; $u_a^k = u_{itk}$
$x_{ak}^+, x_{ak}^-$	holding and back order buffer content of pair $a$ at the beginning of period $k$ equals 1 if pair $a$ is produced at period $k$ , 0 otherwise.
$y_{ak}$	
$\mathbf{u}_a$	vector $\{u_a^k   \forall a \in A, k \in T\}$
Objective function	
$z_a(\mathbf{u}_a)$	generalized cost objective function of solution $\mathbf{u}_a$ , $a \in A$

$$x_{ak}^+, x_{ak}^- \in \mathbb{Z}_{\geq 0}, \quad \forall a \in A, k \in T \cup \{|T| + 1\}, \quad (24)$$

where  $\mathbf{u} = \{u_a^k | a \in A \text{ and } k \in T\}$ , for all  $a \in A$ . Similarly to problem (P), the values of  $x_{a0}^+$  and  $x_{a0}^-$  represent the initial conditions, *i.e.*, the stock and back order level of pair  $a = (i, t)$  at the beginning of the planning time horizon. Constraints (19) are flow-balance constraints among stock level  $x_{ak}^+$ , back order level  $x_{ak}^-$ , production level  $u_a^k$ , and product demand  $d_{ak}$  for each period in time horizon  $k \in T$ . Note that  $d_{ak} = d_a$  when  $k = t$ , *i.e.*, at the period at which demand  $a$  is released, and  $d_{ak} = 0$  for all other periods  $k \in T \setminus \{t\}$ .

Constraints (20) model the backlog and surplus stock after the end of a given time horizon, at period  $|T| + 1$ . By constraints (21), we model the fact that if the production is launched in period  $k$ , the quantity produced should not be larger than the demand  $d_a$  for pair  $a = (i, t)$ . Constraints (22) are nonnegativity and integrality constraints on production  $u_a^k$  decision variables; constraints (23) limit setup variables  $y_{ak}$  to binary values and constraints (24) model the domain of storage  $x_{ak}^+$  and back order  $x_{ak}^-$  variables to  $\{1, \dots, |T| + 1\}$ .

Since the formulation  $L(P_A, \Lambda)$  is separable for a given set of multipliers, it is possible to extract a local optimisation subproblem  $P_a$  addressed by each  $(i, t)$  pair agent  $a \in A$  that includes its production quantities  $u_a^k$ , setup decisions  $y_{ak}$ , holding  $x_{ak}^+$  and back order  $x_{ak}^-$  levels for each period  $k \in T$ , as well as backlog  $x_{a,|T|+1}^-$  and surplus stock  $x_{a,|T|+1}^+$  after the end of planning time horizon  $T$ .

In the following, we present the problem  $P_a$  to be solved by each  $(i, t)$  pair agent  $a \in A$ .

( $P_a$ ):

$$z_a(\mathbf{u}_a) = \min \sum_{k \in T} \left( h_{ak} x_{ak}^+ + b_{ak} x_{ak}^- + s_{ak} y_{ak} + c_a u_a^k \right) + b_{a,|T|+1} x_{a,|T|+1}^- + h_{a,|T|+1} x_{a,|T|+1}^+ + \sum_{k \in T} \lambda_k r_a u_a^k \quad (25)$$

subject to:

$$x_{ak}^+ - x_{ak}^- = x_{a,k-1}^+ - x_{a,k-1}^- + u_a^k - d_{ak}, \quad \forall k \in T \quad (26)$$

$$x_{a,|T+1|}^+ - x_{a,|T+1|}^- = x_{a,|T|}^+ - x_{a,|T|}^- \quad (27)$$

$$u_a^k r_a \leq R_k, \quad \forall k \in T \quad (28)$$

$$u_a^k \leq d_a y_{ak}, \quad \forall k \in T \quad (29)$$

$$u_a^k \in \mathbb{Z}_{\geq 0}, \quad \forall k \in T \quad (30)$$

$$y_{ak} \in \{0, 1\}, \quad \forall k \in T, \quad (31)$$

$$x_{ak}^+, x_{ak}^- \in \mathbb{Z}_{\geq 0}, \quad \forall a \in A, k \in T \cup \{|T| + 1\}, \quad (32)$$

where  $\mathbf{u}_a = \{u_a^k | k \in T\}$  and  $\lambda_k$  is Lagrangian multiplier (conflict price) for resources available at period  $k$ . Here, except for the constraints already explained previously, additional constraints (28) limit the overall resource usage for the production of agent  $a$  to be not greater than the amount of resources available at each period  $k$ . In this way, each  $(i, t)$  pair agent resolves its local optimisation problem having available only its local variables and without the need to communicate with other agents to know their demands or production decisions.

The decentralised MCLSP-BOSI problem allows for multiple robot owner agents and multiple competing item-period pair agents requesting the production of the same item type at different periods, and asynchrony in decision making. For each  $(i, t)$  pair agent  $a \in A$ , the solution of problem  $P_a$  is a local production plan  $u_a = \{u_a^k | k = 1, \dots, |T|\}$  (in response to production demand  $d_a$  and the (current) set of nonnegative Lagrange multipliers *i.e.*, resource conflict prices  $\lambda_k$ ) with related setup  $y_{ak}$ , storage  $x_{ak}^+$ , and back order  $x_{ak}^-$  decisions in each period  $k \in T \cup \{|T| + 1\}$ . Plan  $u_a = \{u_a^k | k = 1, \dots, |T|\}$  might not be globally feasible (*i.e.*, not complying with constraints (4)) and thus should be negotiated with other agents  $a \in A$ . Therefore, item-period pair agents must negotiate resource allocation while exchanging relevant (possibly obsolete) information. Resource allocation here should be achieved by the means of a decentralised protocol where fairness plays a major role.

## 5. Liquid flow network model and the Spillover Algorithm

In this section, we propose a decentralised multi-agent based coordination model and the Spillover Algorithm for the MCLSP-BOSI problem. In Table 3, we present symbols used in the Spillover Algorithm in addition to the ones already presented in Table 2. We describe these symbols in the following.

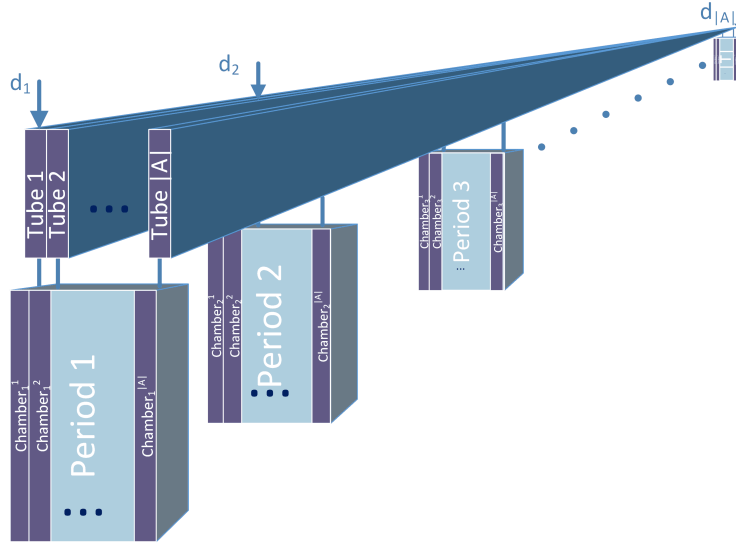
We model each pair  $(i, t)$ , where  $i \in I$  and  $t \in T$  as a rational self-concerned agent  $a \in A$  responsible for finding a production plan  $u_a = \{u_{ak} | k = 1, \dots, |T|\}$  for the demand  $d_a = d_{it}$ , where  $A$  is a set of pairs  $(i, t)$  of cardinality  $|A| = |I| \cdot |T|$ . Since the heuristics that will be proposed in the following imitates the behaviour of liquid flows in tube networks (pipelines) with buffers, we call each agent  $a \in A$  a *liquid agent*.

We model each period  $k \in T$  as a limited resource allocation agent that we call a *buffer agent*, responsible for the allocation of its production capacity  $R_k$  that is proportional to the volume of the buffer (inventory level), Figure 1. We

**Table 3**

Additional symbols in the Spillover Algorithm

$A$	set of liquid agents $a \in A$ , where $a = (i, t)$ , $i \in I$ and $t \in T$
$\delta_{ak}^+$ $\delta_{ak}^-$	valves controlling flow of liquid agent $a$ to posterior and anterior periods
$\rho_{ak}$	number of resources requested by liquid agent $a$ from buffer agent $k \in T$
$\mathbf{u}_k$	vector $\{u_a^k \mid \forall a \in A\}$ of production accommodated by buffer $k$
$Z_a(\mathbf{u}_a)$	approximated cost found by Spillover Algorithm solution $\mathbf{u}_a$ , $a \in A$

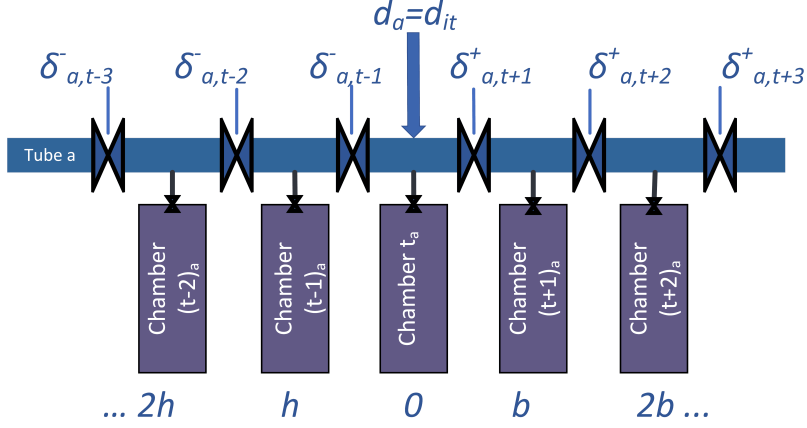


**Figure 1:** Liquid flow network model with  $|T|$  buffers, one per each period; each buffer with  $|A|$  impermeable chambers of interdependent volumes, one chamber per each liquid agent  $a = (i, t)$

assume that available shared resources in period  $k \in T$  are owned by a single resource owner.

Initially, each liquid agent  $a$  releases liquid volume proportional to its demand  $d_a$  into the horizontal tube above buffer  $t$ , Figure 1. This volume, while in the horizontal tube, is awaiting allocation (transfer) to one or more buffers. A liquid agent  $a$  negotiates producing in time, postponing or advancing the production of its demand with the buffers representing present, later or earlier periods with respect to buffer  $t$  by means of two types of horizontal tube valves: valves  $\delta_{ak}^+$ , that control the flow of liquid (demand) of agent  $a$  in the horizontal tube to the posterior periods  $k \in [t + 1, \dots, |T|]$  and valves  $\delta_{ak}^-$ ,  $k \in [t - 1, \dots, 1]$ , that control the flow of liquid  $(i, t)$  to the previous periods  $k \in [t - 1, \dots, 1]$  with respect to period  $t$ , as seen in Figure 2.

The volume of each buffer  $k \in T$  is composed of  $|I| \cdot |T|$  (initially empty) impermeable chambers, one chamber per each liquid agent  $a = (i, t)$ . Each chamber is attached from above to a horizontal tube exclusively dedicated to liquid  $(i, t)$  through a valve controlled by the buffer agent, Figure 2. Buffer  $k$  is modelled as a rational collaborative agent that controls the valves regulating the flow between the horizontal tube of each liquid agent  $a \in A$  that requests allocation at period  $k$  and buffer  $k$ . So even if liquid  $a$  is present above buffer  $k$ , it does not flow into the buffer if the valve of chamber  $k_a$  of buffer  $k$  does not open. The valve will open to let the inflow to chamber  $k_a$  of the liquid volume corresponding to production  $u_a^k$ . Initially, all buffers' valves and all liquids' horizontal tube valves are closed. The volume of liquid  $a = (i, t)$  equal to demand  $d_a$  enters into the tube network above buffer  $k = t$  ( $t$  being the period at which the demand is released). Valves  $\delta_{ak}^+$  and  $\delta_{ak}^-$  for every  $k \in T$  are all closed and do not allow the liquid flow in the horizontal tube, all liquid volume being concentrated between the valves  $\delta_{a,t+1}^+$  and  $\delta_{a,t-1}^-$ , Figure 2. The order of openings of these valves will control the direction of the flow of the unallocated liquid demand in the horizontal tube. This order is influenced by the relation between accumulated back order and holding costs for the liquid and follows linear increase in their values as will be further explained. Liquid agent  $a = (i, t)$  can direct the flow of its demand



**Figure 2:** Side view of a generic tube  $a$  with related buffers' chambers in Figure 1

from its dedicated tube  $(i, t)$  into the chamber  $k_a$  of buffer  $k \in T$  if and only if all the control valves of liquid agent  $a$  from period  $t$  towards period  $k$  are open and the valve of chamber  $k_a$  of buffer  $k$  is open, Figure 2. The volume of flow from tube  $(i, t)$  into buffer  $k$  is equal to the production  $u_a^k$ . This value is inversely proportional to the liquid volumes of other chambers in the same buffer since their overall sum is limited from above by the buffer capacity, i.e.,  $\sum_{a \in A} u_a^k \leq R_k$ . When two or more liquid agents demand more resources than the production capacity  $R_k$ , they compete for them. Since no central resource allocation entity exists, each liquid agent needs to negotiate its production with buffer agents through a negotiation mechanism.

Next, we present the auction-based mechanism for the allocation of the demand requested by liquid agents to be produced by buffer agents that we name the *Spillover Algorithm*. In each iteration, liquid and buffer agents negotiate for the allocation of the liquid demands in the buffers through multiple auctions (one auction per each buffer) in which each buffer agent announces its available resources and liquid agents bid for available buffers with locally lowest cost. Then, each buffer agent allocates liquid agents' demand that locally maximizes its social welfare.

### 5.1. Spillover Algorithm: Liquid Agent

We propose next a decentralised anytime algorithm for liquid agents. Note that an anytime algorithm should be stoppable at any time, providing a feasible solution to the problem (as an approximation to the optimal solution). Also, the provided solution should monotonically improve with the runtime, contrary to Lagrangian relaxation that may oscillate between two points. This is why we explore a heuristic approach that follows given rules in iteratively allocating liquid agent demands to buffers based on the spillover effect. The basic idea here is that the liquid demand that cannot be allocated in a buffer where it appears due to its limited capacity, spills over through its dedicated tube towards other neighbouring buffers. The direction and quantity of spillage will depend on the accumulated values of the production, setup, holding and back order cost throughout the planning time horizon and the backlog cost after the end of the time horizon and their relative values in respect to other concerned liquids.

We introduce the definitions that are the building blocks of the algorithm.

**Definition 5.1.** Accumulated unit production cost  $UPC_{ak}$  of liquid agent  $a = (i, t) \in A$  for each  $k \in T \cup \{|T| + 1\}$ :

$$UPC_{ak} = \begin{cases} c_{ik} + s_{ik} + \sum_{m=k}^{t-1} h_{im}, & \forall k \in T | k < t \\ c_{ik} + s_{ik}, & \text{for } k = t \\ c_{ik} + s_{ik} + \sum_{m=t+1}^k b_{im}, & \forall k \in T | k > t \\ M \cdot \sum_{m=t+1}^k b_{im}, & \text{for } k = |T| + 1 \end{cases} \quad (33)$$

That is,  $UPC_{ak}$  for period  $k$  is composed of a setup ( $s_{ik}$ ) and production cost ( $c_{ik}$ ) and the accumulated holding and

back order costs (if any) depending on the relation of period  $k$  with the demand period  $t$  and with the time horizon  $T$ . Even though  $c_a$  (agent production cost) and  $s_{ak}$  (agent's set-up cost) enter the objective function (18) very differently (one is multiplied with  $u_a^k$ , the other with  $y_{ak}$ ), to achieve a straight-forward computation of  $UPC$  and avoid iterative convergence issues, we model it as the overall cost of a unitary item production in period  $k$  for agent  $a$ .

Assuming non-negative cost parameters, surplus (unsold) stock  $x_{i,|T|+1}^+$  at period  $|T| + 1$  in problem  $P$  (1)–(8) will be zero. Since we want to avoid backlog at period  $|T| + 1$ , accumulated unit production costs for period  $|T| + 1$  in the proposed Spillover Algorithm are found by summing back order costs through a given time horizon multiplied by a very high number  $M$ .  $M$  is a design parameter, whose value is given by the owner of a liquid agent  $a \in A$ . If its value is relatively low, an optimal solution may contain some unmet demand in a given time horizon.

**Definition 5.2.** *Estimated accumulated cost  $EAC_a$  of liquid agent  $a = (i, t)$  is accumulated unitary cost made of unitary production, setup, holding, and back order costs estimated over available periods  $k \in \Gamma_a \cup \{|T| + 1\}$ , where  $\Gamma_a \subset T$  is the set of available buffers that have capacity to produce at least one unit of its product:*

$$EAC_a = \sum_{k \in \Gamma_a \cup \{|T|+1\}} UPC_{ak} \quad (34)$$

The decision-making procedure for each liquid agent  $a \in A$  is presented in Algorithm 1. The algorithm initiates

---

**Algorithm 1:** Spillover Algorithm: liquid agents  $a = (i, t) \in A$

---

```

1 compute  $UPC_{ak}$  for all  $k \in T \cup \{T + 1\}$ 
2 transmit  $d_a, r_i$  to all  $k \in T$ 
3 receive  $R_k$  from all  $k \in T$ 
4 while  $d_a \geq 1$  and  $\{ \lfloor (\max_{k \in T} R_k) / r_i \rfloor \} \geq 1$  do
5   create set  $\Gamma_a \subset T$  of available buffers such that  $k \in \Gamma_a$  iff  $\lfloor (\max_{k \in T} R_k) / r_i \rfloor \geq 1$ 
6   sort  $\Gamma_a$  based on non-decreasing  $UPC_{ak}$  value
7   compute estimated accumulated cost  $EAC_a$ 
8   foreach  $k \in \Gamma_a$  do
9      $\rho_{ak} = \begin{cases} \min(\lfloor R_k / r_i \rfloor \cdot r_i, d_a \cdot r_i) & \text{if } k = t, \\ \min(\lfloor R_k / r_i \rfloor \cdot r_i, d_a \cdot r_i - \sum_{m=\Gamma_{a1}}^{\Gamma_{a,k-1}} \rho_{am}), & \text{otherwise;} \end{cases}$ 
10    transmit bid  $B_{ak} = (\rho_{ak}, EAC_a)$  to  $k$ 
11  end
12  receive  $u_a^k$  for all  $k \in \Gamma_a$ 
13   $d_a = d_a - \sum_{k \in \Gamma_a} u_a^k$ 
14  transmit  $d_a$  to all  $k \in T$ 
15  receive  $R_k$  from all  $k \in T$ 
16 end
17  $x_{a,|T|+1}^- = d_a$ ;
18 compute  $Z_a(\mathbf{u}_a)$ ;
19 return  $Z_a(\mathbf{u}_a)$  and  $x_{a,|T|+1}^-$ 

```

---

with transmitting unaccommodated demand  $d_a$  and resource requirement per product ( $r_i$ ) to buffer agents  $k \in T$ , and receiving buffers' capacities  $R_k$ . If none of the buffers has available capacity (line 4), the procedure terminates after computing and returning its unmet demand  $x_{a,|T|+1}^-$  and heuristic approximate cost  $Z_a(\mathbf{u}_a)$ , where  $\mathbf{u}_a = \{u_a^k | k \in T\}$ , given by:

$$Z_a(\mathbf{u}_a) = \sum_{k \in T} UPC_{ak} \cdot u_a^k + UPC_{a,|T|+1} \cdot x_{a,|T|+1}^- \quad (35)$$

By summing heuristic costs  $Z_a(\mathbf{u}_a)$  in (35) over all liquid agents  $a \in A$ , we can easily obtain the overall system approximate heuristic cost made of the heuristic costs of the allocation of the demands of all items over all periods.

Otherwise, liquid agent with positive demand ( $d_a \geq 1$ ) creates a set of available buffers  $\Gamma_a$  (line 5). Then, it computes  $UPC_{ak}$  for each buffer  $k \in \Gamma_a$  and orders the buffers with available capacity in  $\Gamma_a \subset T$  based on the non-decreasing  $UPC_{ak}$  value (line 6).  $EAC_a$  is computed on line 7 and is used as a part of bid  $B_{ak}$  sent to each of the buffer agents  $k \in \Gamma_a$  considering resource demand  $\rho_{ak}$  and available capacity  $\lfloor R_k/r_i \rfloor$  until the extinction of the unaccommodated demand  $d_a$  or the available capacity (line 9), while  $\Gamma_{a1}$  and  $\Gamma_{a,k-1}$  are the 1<sup>st</sup> and  $(k-1)$ <sup>st</sup> element of  $\Gamma_a$ . Liquid agent sends this information as a part of bid  $B_{ak} = (\rho_{ak}, EAC_a)$  to each buffer  $k \in \Gamma_a$  (line 10). The direction and quantity of spillage will depend on  $EAC_a$ .

By communicating its bid in terms of resource demand  $\rho_{ak}$  for each one of the elements  $k \in \Gamma_a$  and the value of  $EAC_a$ , a liquid agent does not have to disclose its sensitive private information regarding the values of unitary production, holding, setup and back order cost in each time period of the planning time horizon as well as the unitary cost of backlog after the end of the planning time horizon nor reasons for its decision-making. We choose this greedy strategy of producing as much as possible starting with the first buffer agent in set  $\Gamma_a$  since we aim to minimize agent's estimated accumulated cost  $EAC_a$  for the rest of available periods in the horizon. Then, on receiving production response  $u_a^k$  (line 12) from buffers  $k \in \Gamma_a$ , liquid agent  $a$  updates its unaccommodated demand (line 13) and transmits it to all buffers  $k \in \Gamma_a$  (line 14). After receiving available production capacities  $R_k$  from all  $k \in T$  (line 15), if there still remains an unaccommodated demand  $d_a$  and available capacities  $\lfloor R_k/r_i \rfloor$ , new iteration starts; otherwise, the algorithm terminates.

## 5.2. Spillover Algorithm: Buffer Agent

Each buffer agent  $k \in T$  knows its capacity and acts as an auctioneer for accommodation of liquid agent bidders' demand for period  $k$ . It orders all liquid agents bids in a non-increasing order and greedily accommodates for production  $\mathbf{u}_k = \{u_a^k | a \in A\}$  allocating as much production resources as possible to the liquid agent bidder with the highest  $EAC_a$  value up to the depletion of its available production capacity  $R_k$ .

---

### Algorithm 2: Spillover Algorithm: buffer agents $k \in T$

---

```

1 transmit  $R_k$  to all  $a \in A$ ;
2 receive  $d_a, r_i$  for all  $a \in A$ 
3  $A^u = \{a \in A | d_a > 0\}$ 
4 while  $|A^u| \geq 1$  and  $\{\lfloor R_k / \min_{a \in A^u} r_i \rfloor\} \geq 1$  do
5   receive  $B_{ak} = (\rho_{ak}, EAC_a)$  for all  $a \in A^u$ 
6   create ordered set  $BA_k = \{(a, \rho_{ak}) | a \in A^u\}$  based on non-increasing  $EAC_a$  value
7   // allocate liquid agents in  $BA_k$  for production
8   foreach  $a \in BA_k$  do
9      $u_a^k = \min(\rho_{ak}/r_i, \lfloor R_k/r_i \rfloor)$ 
10     $R_k = R_k - u_a^k \cdot r_i$ 
11  end
12  transmit  $u_a^k$  and  $R_k$  to all  $a \in A^u$ 
13  receive  $d_a$  from all  $a \in A^u$ 
14   $A^u = \{a \in A | d_a > 0\}$ 
15 end
16 return  $u_a^k, \forall a \in A$ 

```

---

The decision-making of each buffer agent runs in iterations (Algorithm 2). Initially, it transmits its available capacity  $R_k$  (line 1) and receives demands  $d_a$  and resource requirements per product  $r_i$  of all liquid agents  $a = (i, t) \in A$  (line 2). Then, it obtains the set  $A^u$  of liquid agents with unmet demand (line 3). If there is unmet demand and sufficient production capacity (line 4), after receiving bids from bidding liquid agents (line 5), it creates an ordered tuple of bidding agents  $BA_k$  based on non-increasing  $EAC_a$  value (line 6). This policy minimizes the maximum cost of the liquid agents bidding for a buffer and can be seen as a fairness measure to increase egalitarian social welfare among liquid agents. Then, it allocates the highest possible production level to liquid agents in  $BA_k$  considering resource demand  $\rho_{ak}$  and remaining capacity (line 9) and transmits the production values and remaining capacity to concerned agents  $a \in A^u$  (line 12). On receiving unaccommodated demand  $d_a$  from  $a \in A^u$  (line 13), the buffer



updates  $A^u$  and checks whether the termination conditions on unaccommodated demand or its remaining capacity are fulfilled (line 4). If so, it terminates; otherwise, it repeats.

### 5.3. Spillover Algorithm analysis

This section analyses the algorithm's termination, optimality, complexity and the protection of sensitive private information.

*Termination.* The Spillover Algorithm terminates in at most  $|I| \cdot |T|^2$  iterations since it iterates through  $|I| \cdot |T|$  liquid agents and each of them does at most  $|T|$  iterations. It stops for a liquid agent  $a \in A$  in the worst case when all the periods of the time horizon  $T$  have been bid for. This is guaranteed to occur in  $|T|$  steps, if the time horizon is bounded. If agent set is bounded, in the worst case when applied serially, this will happen in  $|A| \cdot |T|$  steps for all agents. If the overall demand through the given time horizon may be eventually allocated for production, i.e., if  $\sum_{a \in A} d_a \leq \sum_{k \in T} \lfloor R_k / r_i \rfloor$ , no unmet demand (big M) cost will be accrued.

*Optimality.* Whether the complete allocation obtained upon termination of the auction process is optimal depends strongly on the method for choosing the bidding value. The heuristic of ordering local  $EAC_a$  values does not have a guarantee of a global optimum solution but guarantees local optimum. Thus, in case of an unpredicted setback or change in capacities or demands, instead of updating the plan for the whole system, as is the case in the centralised control, it is sufficient to modify only the plans of the directly involved agents.

*Complexity of Algorithm 1 (Liquid agent).* A liquid agent does at most  $|T|$  iterations in the worst case (line 4) with the aim to accommodate its demand in any of the buffer agents. A bid from liquid agent  $a$  is either (i) completely satisfied, (ii) partially satisfied or (iii) completely unsatisfied. In case (i), it may be possible that a liquid agent with still remaining unallocated demand bids again for the same buffer (if its demand allocation request was refused in other buffers). The buffer may i) allocate all the liquid agent's demand up to the buffer's capacity or up to the extinction of the liquid agent's demand or ii) reject the allocation since it is already full. Thus, although liquid agent  $a$  may bid at most two times for each buffer, in each iteration, at least one buffer is filled up and thus discarded for the next round of bids of the same liquid agent. The calculation of  $EAC_a$  (line 7) requires  $|T|$  iterations in the worst case as well as sending bids to buffer agents (foreach loop in line 8). Thus, in each loop in line 4, liquid agents send  $2 \cdot |T|$  messages to and receive  $2 \cdot |T|$  messages from buffer agents (lines 12 and 15); exchanged messages  $O(|T|^2)$ .

*Complexity of Algorithm 2 (Buffer agent).* The main loop (line 4) of a buffer agent is repeated  $2 \cdot |A|$  times in the worst case, since in every iteration a new bid (different from a previous one) from a liquid agent could be received.

In each iteration, a buffer agent sends (lines 12) and receives  $2 \cdot |A|$  messages (lines 5 and 13) in the worst case, respectively. Thus, the complexity in number of messages exchanged is  $O(|A|)$ . Note that sorting the received bids  $BA_k$  (line 6) can be done in  $O(|A| \cdot \log|A|)$ . However, this is done locally, not being as complex as exchanging  $O(|A|)$  messages. In total, the complexity of the decision-making algorithm of each buffer agent is  $O(|A|^2)$ .

*Protection of sensitive private information.* In the Spillover Algorithm, the most sensitive private information includes the values of unitary production  $c_{it}$ , holding  $h_{it}$ , setup  $s_{it}$  and back order cost  $b_{it}$  in each time period  $t$  of the planning time horizon  $T$  as well as the unitary cost of backlog  $b_{i,|T|+1}$  after the end of the planning time horizon. None of this information is shared among liquid agents representing users of production resources. To be able to learn deterministically the cost values of a liquid agent by a buffer agent from the Estimated Accumulated Cost  $EAC_a$ , at least  $(4|T| + 1)$  linearly independent  $EAC_a$  values must be available (a solution to  $n$  unknowns can be found deterministically by solving  $n$  linearly independent equations). Here,  $4|T|$  refers to the number of cost parameters  $(c_{it}, h_{it}, b_{it}, s_{it})$  over  $|T|$  time periods and we need 1 additional equation for finding the value of  $M$ . Note that there are at most  $|T|$  possible bids in each iteration and at most 2 bids per buffer made by a liquid agent. Since deterministic inference of the cost parameter values and a very large number  $M$  in the Spillover Algorithm is possible only if  $4|T| + 1$  linearly independent  $EAC_a$  values are available, it is impossible to obtain the sensitive private cost parameter values of liquid agents by buffer agents in the Spillover Algorithm.

## 6. Simulation experiments

In this section, we compare the performance of the proposed Spillover Algorithm and the centralised and optimal CPLEX solution considering randomly generated and diversified problem set for large-scale capacitated lot-sizing. To the best of our knowledge, there are no other decentralised solutions to the MCLSP-BOSI problem to compare with. We tested a general DCOP ADOPT-n (Modi, Shen, Tambe and Yokoo (2005)) but due to its modelling for n-ary constraints and the presence of mostly global constraints in MCLSP-BOSI, with only 4 periods and 4 liquid agents, it didn't find a solution in a reasonable time.

### 6.1. Experiment setup

The experiment setup was performed following the principles in Diaby et al. (1992) and Giordani et al. (2013). We consider the case of *Wagner-Within costs*, i.e., the costs that prohibit speculative motives for early and late production resulting in speculative inventory holding and back-ordering, respectively. Considering constant production costs  $c_a$  for all  $a = (i, t) \in A$ , setup, holding, and back order costs comply with the assumption of Wagner-Within costs, i.e.,

$$-b_{ik} < s_{i,k+1} - s_{ik} < h_{ik}, \quad \forall k \in T \quad (36)$$

The value of  $s_{i1}$  for each item  $i \in I$  is chosen randomly from the uniform distribution in the range [50, 100], i.e.  $s_{i1} \sim U(50, 100)$ , and the value of the setup cost of every following period  $\{s_{i2}, \dots, s_{i,|T|}\}$  is computed by the following formula  $s_{i,k+1} = s_{ik} + X$ , where  $X \sim U(-1, 1)$ .

For simplicity, holding costs  $h_{ik}$  are assumed constant in time, i.e.,  $h_{ik} = h_i$ , and are generated randomly from  $U(20, 100)$  for each item  $i \in I$ . Backorder costs are computed from the holding costs considering a multiplication factor of 10, 2, 0.5, and 0.1 (the obtained values are rounded to have all integer values). When considering backlog costs in (33), to produce as much as possible during the time horizon, we model  $M$  as a very large number whose value is 10000. We assume unitary production cost  $c_{it}$  for each item  $i \in I$  to be constant in time, i.e.  $c_{it} = c_i$ , and  $c_i \sim U(1000, 10000)$ .

In the experiments, we consider a large production system scenario producing from 50 to 150 items with increase 10, over a time horizon composed of 100 periods with the value of mean item demand per period  $\bar{d}_i \sim U(100, 1000)$ . This sums up to overall demand from 2.75 to 8.25 million units on average in the given time horizon.

For each item  $i \in I$ , demand  $d_{it}$  is generated from the normal distribution with mean  $\bar{d}_i$  and standard deviation  $\sigma = \bar{d}_i/\kappa$ , with  $\kappa > 1$ , i.e.  $(d_{it} \sim N[\bar{d}_i, (\bar{d}_i/\kappa)^2])$ . Standard deviation of the distribution of the item-period demands controls the demand variability. We experiment two levels of demand variability: high level whose  $\sigma = \bar{d}_i/2$  ( $\kappa = 2$ ) and low level with  $\sigma = \bar{d}_i/4$  ( $\kappa = 4$ ). The larger the value of  $\kappa$ , the more "lumpy" the demand, i.e., the greater the differences between each period's demand and the greater the number of periods with zero or fixed base level demand (e.g., Wemmerlöv and Whybark (1984)). For simplicity, but w.l.o.g, the values of  $x_{i0}^+$ ,  $x_{i0}^-$ , and  $y_{i0}$  are assumed to be equal to zero. Resource requirement  $r_i$  for production of a unit of  $i \in I$  is generated from  $U(1, 3)$ . We consider variable production capacities generated from  $(R_k \sim N[110000, (110000/4)^2])$ . This value is chosen to satisfy the overall demand of  $|I| = 100$  items with average demand  $\bar{d}_i = 550$  units with average resource requirements  $\bar{r}_i = 2$  for all  $i \in I$  over 100 periods.

As the key performance indicators, we consider average computational time and optimality gap compared between the solution of the Spillover Algorithm implemented in Matlab and the optimal solution computed with IBM ILOG CPLEX Studio 12.8 with CPLEX solver, both run on an Intel Core i5 with 2.4 GHz and 16 GB RAM. We tested scenarios, from very decongested ones with the overall demand representing 50% of the available production resources in the horizon, to very congested scenarios where this percentage increases to 150%. We combine the 11 cases of demand varying from 50 to 150 items with increment of 10 over 100 periods with 2 cases of variability and 4 cases of back order costs related to the holding costs, resulting in a total of 88 experiment setups.

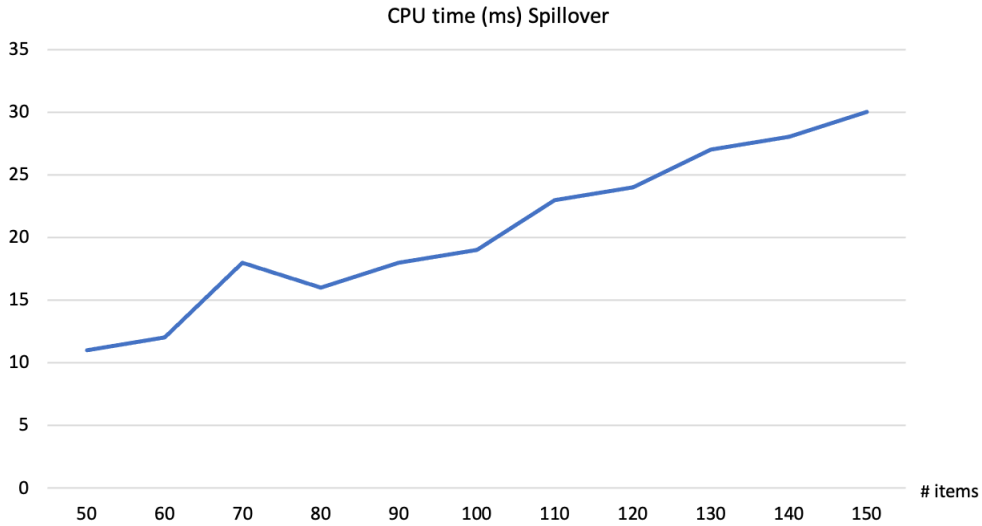
### 6.2. Results

The simulation results of the Spillover Algorithm with the approximation scheme are shown in Table 4. The table contains the optimality gap (average of 8 experiments each, as described previously) in relation to the solution found in CPLEX together with their computational times. The optimality gap is the one used by CPLEX and is obtained as  $(z_x - z_o)/z_o$ , where  $z_x$  is the cost of the solution found by the Spillover Algorithm and  $z_o$  is the cost of the CPLEX optimal solution. The mean gap value is 25% and the individual gap value throughout the instances is rather constant and independent of the item number. This gap value, for such a heuristic approximation approach, is a very good result

**Table 4**

Summary of the computational results

# items	50	60	70	80	90	100	110	120	130	140	150
Avg. Gap (%)	28	28	30	16	16	28	27	28	27	27	27
Spillover (ms)	11	12	18	16	18	19	23	24	27	28	30
Min CPLEX (s)	0.05	0.56	1.1	1.53	1.92	2.49	3.1	2.83	3.35	3.2	3.75
Max CPLEX (s)	0.69	1.07	1.39	11	47	74	157	391	451	571	869

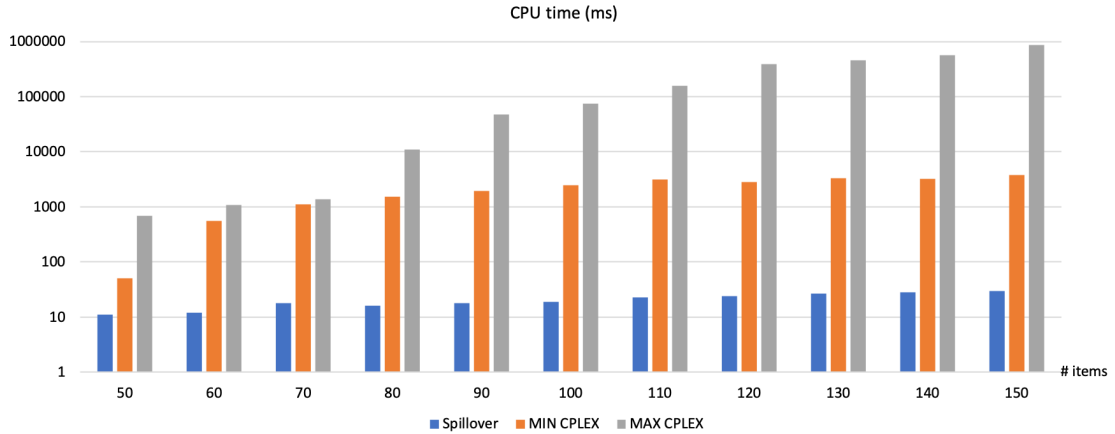
**Figure 3:** Average computational time of the Spillover Algorithm for different numbers of items

considering that heuristic approaches generally do not have quality of solution guarantees. The computational time of CPLEX has high variability for the same problem size. Thus, Table 4 shows the minimum and maximum execution times (in seconds) for each number of items. The Spillover Algorithm execution time is less than 0.1 sec for all the experiments and grows lineally with the number of items (see Figure 3). This confirms its good performance in relation to CPLEX, whose computational time increases exponentially with the increase of the item set size and thus scales poorly. Figure 4 shows graphically, using a logarithmic scale, the minimum and maximum times taken by CPLEX for different problem sizes (number of items) and the average Spillover Algorithm time. However, the emphasis here is not on proposing an optimal method in a centralised environment, but to decentralise the coordination decisions of production planning in environments where the exposure of private and sensitive information is desirably minimized while still having a reasonably good global solution away from the control of a centralised decision maker.

## 7. Discussion and conclusions

Open and shared factories are becoming popular in Industry 4.0 as another component of today's global economy. In such facilities, production resources and product owners coexist in a shared environment. One of the main issues faced by product (item) owners that compete for limited production resources held by multiple resource owners in such factories is the exposure of their private and sensitive information including the values of unitary production, holding, setup and back order cost in each time period of the planning time horizon as well as the unitary cost of backlog after the end of the planning time horizon.

The scope of this paper was not to propose a more computationally efficient heuristic for centralised MCLSP-BOSI problem, but to study a decentralised version of the MCLSP-BOSI problem and develop a heuristic approach applicable to intrinsically decentralised shared and open factories. Centralised state-of-the-art heuristics cannot be



**Figure 4:** Comparison of the average computational time (in logarithmic scale) of the Spillover Algorithm with the minimum and maximum computational times of CPLEX solver

applied to this problem with self-concerned and individually rational resource and item owners since these heuristics require complete exposure of everyone’s private and sensitive information.

Therefore, in this paper, we presented a decentralised and dynamic variant of the classic MCLSP-BOSI problem with time-dependent costs. To reach a decentralised variant of the problem and to control locally the linear increase of product owners’ costs, we decomposed the problem based on *item-period* pairs.

As a solution approach to the decentralised and dynamic MCLSP-BOSI problem, we proposed the *Spillover Algorithm*, to the best of our knowledge, the first heuristic decentralised algorithm for this problem that complies with the intrinsically decentralised nature of large and shared open factories. A heuristic solution was needed to cope with the NP-hard nature of the (decentralised) MCLSP-BOSI problem. The spillover heuristic is formulated as a multi-agent algorithm in a liquid flow network model with buffers: each item-period pair is represented by a *liquid agent* responsible for obtaining production resources (robots) to manufacture its demand, *i.e.*, product of type *i* requested to be produced by the means of bidding for resource allocation at time *t*. Likewise, production capacity (number of available robots) in each period is represented by a *buffer agent* responsible for allocating the capacity to bidding *liquid agents*.

An auction-based algorithm leveraging spillover effect has been designed for a one-on-one negotiation between the liquid agents requesting item production and their buffer agents of interest. Each liquid agent sends greedy bids (consisting of the amount of the item to be produced, and the agent’s estimated accumulated cost for available buffers) to the buffer agents in order of non-decreasing accumulated unit production cost, and each buffer agent greedily accepts bids in order of the bidding liquid agents’ non-increasing estimated accumulated cost for all buffers. This is repeated until all demands are allocated for production throughout the planning time horizon.

The proposed multi-agent auction-based approach has the advantage that agents do not need to reveal all their private and sensitive information (*i.e.*, unitary costs of production, setup, back order, and holding) and that the approach is decentralised. Distributed problem solving here includes sharing of estimated accumulated costs for each item-period pair, *i.e.*, a heuristic unitary demand cost accumulated over time periods that are still available for resource allocation. The latter can be interpreted as an approximate “resource conflict price” paid if a unit of demand is not allocated for production in the requested period.

Note that the state-of-the-art heuristics are centralised solution approaches that are not applicable in intrinsically decentralised open and shared factories with self-interested and individually rational resource and product owners. Thus, the comparison of the performance of the Spillover Algorithm with these solution approaches is meaningless.

Since all agents in the Spillover Algorithm have only a local view of the system, a disadvantage is that global optima are not necessarily achieved with such a decentralised control.

We presented an experimental evaluation with randomly generated instances that nonetheless shows that the solutions obtained using the spillover effect heuristic are only about 25% more expensive than the optimal solutions

calculated with CPLEX. The average optimality gap values are between 16% and 30% for the 11 tested item type numbers (from 50 to 150 different items (products)), where the average has been taken over 8 randomly generated instances for each choice of the number of item types. However, the Spillover Algorithm gives an anytime feasible solution running in close-to real time, under 30 milliseconds, while CPLEX requires between 4 and 869 seconds for the largest tested instances.

The low computational complexity of the Spillover Algorithm facilitates the implementation in shared and open factories with competitive stakeholders who want to minimize their sensitive and private information exposure.

The Spillover algorithm can be applicable, with no substantial changes, to other domains where multiple independent self-concerned agents compete for limited shared resources distributed over a time horizon of a limited length. This is the case, for example, in flight arrival/departure scheduling to maximize runway utilization and the allocation of electric vehicles to a charging station. In the former, flight scheduling case, liquid agents (representing flights) compete for runway time slots, which are managed by buffer (runway) agents responsible for allocating their landing/take-off capacity in each time period. At peak hours, sometimes it is inevitable that some flights change their schedules because of the limited capacities of the runways. Thus, dynamic (re-)allocation is required. In the application of the Spillover algorithm to this case, the production cost is the time slot cost for an aircraft, back order costs reflect the consequences of flight delays (e.g. compensations to clients, damage of reputation, etc.), and holding costs represent the monetary effect of early departures/arrivals. Setup costs are the operational costs and other charges for an aircraft using the runway at certain time period.

The Spillover Algorithm is resilient to crashes as the buffer agents continue to assign their resources as long as there is at least one liquid agent bidding for them. This topic will be treated in future work. In future work, we will also study more accurate spillover rules that iteratively approximate estimated accumulated costs as the bidding progresses while guaranteeing convergence and the protection of sensitive private information. Furthermore, in the Spillover algorithm, buffer agents prioritize bids from liquid agents based on their estimated accumulated cost *EAC* value included in each bid. The higher the value, the higher the priority for resource allocation of a liquid agent. Thus, a strategic liquid agent may want to report a falsely high *EAC* value to get the production resources at requested time periods. Open issues of strategic agents, trust and incentives not to lie about production demand, estimated cost parameters, and production capacities will also be studied in future work. Examples of lines to explore are game theory and mechanism design (e.g. Vickrey-Clarke-Groves mechanism).

## Acknowledgements

This work has been partially supported by the “E-Logistics” project funded by the French Agency for Environment and Energy Management (ADEME) and by the “AGRIFLEETS” project ANR-20-CE10-0001 funded by the French National Research Agency (ANR) and by the STSM Grant funded by the European ICT COST Action IC1406, “cHiPSeT”, and by the Spanish MINECO projects RTI2018-095390-B-C33 (MCIU/AEI/FEDER, UE) and TIN2017-88476-C2-1-R.

## References

- Aggarwal, A., Park, J.K.. Improved algorithms for economic lot size problems. *Operations Research* 1993;41(3):549–571.
- Arora, A., Bokhari, F.A.. Open versus closed firms and the dynamics of industry evolution. *The Journal of Industrial Economics* 2007;55(3):499–527.
- Bijvank, M., Vis, I.F.. Lost-sales inventory theory: A review. *European Journal of Operational Research* 2011;215(1):1–13.
- Bitran, G.R., Yanasse, H.H.. Computational complexity of the capacitated lot size problem. *Management Science* 1982;28(10):1174–1186.
- Brahimi, N., Absi, N., Dauzère-Pérès, S., Nordli, A.. Single-item dynamic lot-sizing problems: An updated survey. *European Journal of Operational Research* 2017;263(3):838–863.
- Buschkühl, L., Sahling, F., Helber, S., Tempelmeier, H.. Dynamic capacitated lot-sizing problems: a classification and review of solution approaches. *OR Spectrum* 2010;32(2):231–261.
- Chen, W.H., Thizy, J.M.. Analysis of relaxations for the multi-item capacitated lot-sizing problem. *Annals of Operations Research* 1990;26(1):29–72.
- Cheng, C.H., Madan, M.S., Gupta, Y., So, S.. Solving the capacitated lot-sizing problem with backorder consideration. *Journal of the Operational Research Society* 2001;52(8):952–959.
- Diaby, M., Bahl, H.C., Karwan, M.H., Zionts, S.. A lagrangean relaxation approach for very-large-scale capacitated lot-sizing. *Management Science* 1992;38(9):1329–1340.
- Dixon, P.S., Silver, E.A.. A heuristic solution procedure for the multi-item, single-level, limited capacity, lot-sizing problem. *Journal of Operations Management* 1981;2(1):23–39.

- Drexl, A., Kimms, A.. Lot sizing and scheduling—survey and extensions. *European Journal of Operational Research* 1997;99(2):221–235.
- Faltings, B.. Distributed constraint programming. In: *Foundations of Artificial Intelligence*. Elsevier; volume 2; 2006. p. 699–729.
- Farrell, J., Monroe, H.K., Saloner, G.. The vertical organization of industry: Systems competition versus component competition. *Journal of Economics & Management Strategy* 1998;7(2):143–182.
- Fisher, M.L.. The lagrangian relaxation method for solving integer programming problems. *Management Science* 1981;27(1):1–18.
- Florian, M., Lenstra, J.K., Rinnooy Kan, A.. Deterministic production planning: Algorithms and complexity. *Management Science* 1980;26(7):669–679.
- Giordani, S., Lujak, M., Martinelli, F.. A decentralized scheduling policy for a dynamically reconfigurable production system. In: Mařík, V., Strasser, T., Zotl, A., editors. *Holonic and Multi-Agent Systems for Manufacturing*. Springer Berlin Heidelberg; volume 5696 of *Lecture Notes in Computer Science*; 2009. p. 102–113.
- Giordani, S., Lujak, M., Martinelli, F.. A distributed multi-agent production planning and scheduling framework for mobile robots. *Computers & Industrial Engineering* 2013;64(1):19–30.
- Gören, H.G., Tunali, S.. Fix-and-optimize heuristics for capacitated lot sizing with setup carryover and backordering. *Journal of Enterprise Information Management* 2018;31(6):879–890.
- Hao, Y., Helo, P., Shamsuzzoha, A.. Virtual factory system design and implementation: Integrated sustainable manufacturing. *International Journal of Systems Science: Operations & Logistics* 2018;5(2):116–132.
- Hindi, K.. Computationally efficient solution of the multi-item, capacitated lot-sizing problem. *Computers & Industrial Engineering* 1995;28(4):709–719.
- Hozdić, E.. Smart factory for industry 4.0: A review. *International Journal of Modern Manufacturing Technologies* 2015;7(1):28–35.
- Jans, R., Degraeve, Z.. Meta-heuristics for dynamic lot sizing: A review and comparison of solution approaches. *European Journal of Operational Research* 2007;177(3):1855–1875.
- Jans, R., Degraeve, Z.. Modeling industrial lot sizing problems: a review. *International Journal of Production Research* 2008;46(6):1619–1643.
- Jiang, P., Li, P.. Shared factory: A new production node for social manufacturing in the context of sharing economy. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture* 2019;234(1-2):285–294.
- Karimi, B., Ghomi, S.F., Wilson, J.. The capacitated lot sizing problem: a review of models and algorithms. *Omega* 2003;31(5):365–378.
- Karimi, B., Ghomi, S.M.T.F., Wilson, J.M.. A tabu search heuristic for solving the CLSP with backlogging and set-up carry-over. *Journal of the Operational Research Society (JORS)* 2006;57(2):140–147.
- Lemaréchal, C.. Lagrangian relaxation. In: *Computational Combinatorial Optimization*. Springer; 2001. p. 112–156.
- Liu, Y., Xu, X., Zhang, L., Wang, L., Zhong, R.Y.. Workload-based multi-task scheduling in cloud manufacturing. *Robotics and Computer-Integrated Manufacturing* 2017;45:3 – 20. Special Issue on Ubiquitous Manufacturing (UbiM).
- Lozano, S., Larraneta, J., Onieva, L.. Primal-dual approach to the single level capacitated lot-sizing problem. *European Journal of Operational Research* 1991;51(3):354–366.
- Lujak, M., Fernández, A., Onaindia, E.. A decentralized multi-agent coordination method for dynamic and constrained production planning. In: *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems, AAMAS '20*. International Foundation for Autonomous Agents and Multiagent Systems; 2020. p. 1913–1915.
- Maes, J., Wassenhove, L.V.. Multi-item single-level capacitated dynamic lot-sizing heuristics: A general review. *Journal of the Operational Research Society (JORS)* 1998;39(11):991–1004.
- Millar, H.H., Yang, M.. Lagrangian heuristics for the capacitated multi-item lot-sizing problem with backordering. *International Journal of Production Economics* 1994;34(1):1–15.
- Modi, P.J., Shen, W., Tambe, M., Yokoo, M.. ADOPT: asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence* 2005;161(1-2):149–180.
- Pochet, Y., Wolsey, L.A.. Lot-size models with backlogging: Strong reformulations and cutting planes. *Mathematical Programming* 1988;40(1-3):317–335.
- Pochet, Y., Wolsey, L.A.. *Production Planning by Mixed Integer Programming*. Springer Series in Operations Research and Financial Engineering, 2006.
- Qian, C., Zhang, Y., Sun, W., Rong, Y., Zhang, T.. Exploring the socialized operations of manufacturing resources for service flexibility and autonomy. *Robotics and Computer-Integrated Manufacturing* 2020;63:101912.
- Quadt, D., Kuhn, H.. Capacitated lot-sizing with extensions: a review. *4OR - A Quarterly Journal of Operations Research* 2008;6(1):61–83.
- Quadt, D., Kuhn, H.. Capacitated lot-sizing and scheduling with parallel machines, back-orders, and setup carry-over. *Naval Research Logistics (NRL)* 2009;56(4):366–384.
- Rand, T.A., Tylianakis, J.M., Tscharntke, T.. Spillover edge effects: the dispersal of agriculturally subsidized insect natural enemies into adjacent natural habitats. *Ecology Letters* 2006;9(5):603–614.
- Shrouf, F., Ordieres, J., Miragliotta, G.. Smart factories in industry 4.0: A review of the concept and of energy management approached in production based on the internet of things paradigm. In: *2014 IEEE International Conference on Industrial Engineering and Engineering Management*. 2014. p. 697–701.
- Süral, H., Denizel, M., Wassenhove, L.N.V.. Lagrangean relaxation based heuristics for lot sizing with setup times. *European Journal of Operational Research* 2009;194(1):51–63.
- Thizy, J., Van Wassenhove, L.N.. Lagrangean relaxation for the multi-item capacitated lot-sizing problem: A heuristic implementation. *IIE Transactions* 1985;17(4):308–313.
- Toledo, C.F.M., de Oliveira, R.R.R., França, P.M.. A hybrid multi-population genetic algorithm applied to solve the multi-level capacitated lot sizing problem with backlogging. *Computers & Operations Research* 2013;40(4):910–919.
- Wagner, H.M., Whitin, T.M.. Dynamic version of the economic lot size model. *Management Science* 1958;5(1):89–96.
- Wemmerlöv, U., Whybark, D.C.. Lot-sizing under uncertainty in a rolling schedule environment. *The International Journal of Production Research*

1984;22(3):467–484.

Xu, X.. From cloud computing to cloud manufacturing. *Robotics and Computer-Integrated Manufacturing* 2012;28(1):75 – 86.

Zangwill, W.I.. A deterministic multi-period production scheduling model with backlogging. *Management Science* 1966;13(1):105–119.