Conveyor operations in distribution centers: modeling and optimization

Article *in* Optimization Letters · August 2022

DOI: 10.1007/s11590-022-01912-7

CITATIONS
5

READS
351

3 authors, including:

Michel Fathi
University of North Texas
113 PUBLICATIONS **762** CITATIONS
SEE PROFILE

Panos Pardalos
University of Florida
1,728 PUBLICATIONS **48,337** CITATIONS
SEE PROFILE

Noname manuscript No.
(will be inserted by the editor)

Conveyor operations in distribution centers: modeling and optimization

Farzaneh Karami · Mahdi Fathi · Panos M. Pardalos

Abstract The optimal operation of a conveyor network used in distribution centers is critical to delivery service quality. Optimizing conveyor operations entails routing a given number of items from loading locations to unloading locations in the shortest possible time, called makespan. Developing an efficient model for conveyor operations is necessary to aid in routing processes. In this regard, this paper proposes two models while taking conveyor operational constraints into account. The first model determines the best single path-based item routing for each distinct loading-unloading while avoiding operation interruptions. The second model with the same goal focuses on utilizing the conveyor's full capacity by routing items of each loading-unloading on multiple paths. This model requires the complete set of available paths for each loading-unloading as an input. This model is solved using a heuristic algorithm given the requirement for high-quality solutions within a short computation time. The computational experience was conducted on a set of benchmark instances to provide proof of concept for our original models and to assess their quality and applicability. The results confirm both models' effective representation and resolution and provide evidence for how controlling the number of used paths improves routing and handles a complex real-world problem. Therefore, an operational manager can weigh the gains against the model and operation complexity expense.

F. Karami
Department of Computer Science, KU Leuven, CODeS, Gebroeders De Smetstraat 1, 9000 Gent, Belgium;
Department of Electromechanical, Systems and Metal Engineering, Ghent University, EEDT-DC, Flanders Make, 9052 Zwijnaarde, Belgium E-mail: karami.farzaneh@UGent.be

M. Fathi (Corresponding Author)
Department of Information Technology & Decision Sciences, G. Brint Ryan College of Business, University of North Texas, Denton, TX, USA E-mail: mahdi.fathi@unt.edu

P. M. Pardalos
Department of Industrial and Systems Engineering, University of Florida, Gainesville, FL, USA E-mail: pardalos@ufl.edu

Keywords Conveyor operations · Dynamic flows · Flows over time · Logistics · Matheuristics · Quickest flows · Quickest path

1 Introduction

Due to increasing consumer expectations, services such as same-day delivery have become one of the fastest-rising logistics components in the e-commerce industry. The same-day delivery demand cannot be overstated; however, most service providers fall short of such an expectation, with only slightly more than half offering same-day delivery. When this statistic is compared to the fact that over 75% of consumers want same-day shipping, there is a significant disparity (eFu, 2017). As a result, this service has led to an ever-increasing freight volume, all of which require sorting, packing, and transporting steps under tight delivery schedules (Wahba, 2015).

Distribution centers (DCs) serve as intermediary storage centers for goods between two supply chain stages. In a DC, the daily throughput can be hundreds of thousands of items to be sorted and packed in parcels. More than half of all DC operating costs can be attributed to manual sorting and packing (De Koster et al., 2007). With the recent advances in technology, automated sorting within conveyors has largely replaced manual sortation. Such automation facilitates logistics companies to meet the same-day delivery requirements, and as a result, more and more DCs are being automated for increasing throughput. Such DCs consist of several storage areas. Each area is equipped with a robot that automatically collects and loads items onto their corresponding conveyor loading location. DCs also typically employ a conveyor network to transfer loaded items to the packing areas. The higher the number of parcels is and the longer their delivery distances are, the shorter time is available for a DC for sorting and packing. Therefore, a high capacity and efficient DC is necessary for genuinely realizing same-day delivery. Such aspects are unquestionably possible when decisions are optimized within a DC at all stages. Overall, it appears at least three decision problems need to be addressed, which may be summarized as:

DC and its conveyor layout design: The construction cost of a large automated DC with a conveyor network is expensive and varies in the $10-50 million range (Fedtke and Boysen, 2017). Many researchers have investigated conveyor layout design, such as its network shape or segment lengths (see the comprehensive survey by Boysen et al. (2018)). For example, Fedtke and Boysen (2017) compare different design alternatives and conclude that a conveyor with multiple loading and unloading locations provides better performance. They also investigate which vehicle assignments to unloading locations can facilitate fast delivery. Despite numerous studies on such design problems, the conveyor operations in the literature are presumed to be efficient.

Vehicle routing and scheduling: Vehicle routing and scheduling account for nearly 53% of shipping costs DHL (2019), and extensive research

has been already conducted and will continue to address such a problem (see the recent survey by Mor and Speranza (2022)).

Conveyor operations: Conveyor operations hourly expenses frequently exceed $1,000 (Johnson, 1997) and typically are bottlenecks to meet the fast delivery goal. A substantial portion of the literature has almost entirely ignored conveyor operations optimization, which demands optimal route scheduling. Instead, they focus on designing a conveyor-based sorting system (for example, see (Chen et al., 2021) and the comprehensive survey by Boysen et al. (2018)). Determining how efficiently a conveyor conducts item routing is as crucial as its layout design. Optimizing conveyor operations is to route a given number of items from loading locations to unloading locations on the conveyor as a capacitated network in minimum makespan. Makespan denotes the time between the starting time of the routing process and the arrival time of the last item at its unloading location.

Suppose a conveyor network with capacities and transit times on its edges, loading locations, unloading locations, and time horizon are given. It is also given how many items must be routed between each distinct loading-unloading pair. As a powerful decision support tool, network flows theory can provide a solid foundation for conveyor operations modeling. More specifically, it has a great potential to model the conveyor operations suitable to be represented as items' routing on its network while considering given conveyor network structural limitations, such as finite capacities on edges. An advancement of network flows theory has been achieved by introducing dynamic flows, which intend to integrate the temporal dimension into the model of items routing on networks (Ford and Fulkerson, 1962). Dynamic flows modeling allows to identify the exact position of items at each point in time and, thus, closely manage and control their routing over the given time horizon. For example, the dynamic network flows provide an efficient way to model data routing in telecommunication networks (Clímaco et al., 2007) as well as evacuation plans in emergency scenarios (Kappmeier, 2015; Melchiori and Sgalambro, 2015), where supervised routing increases safety.

Therefore, conveyor operations modeling based on dynamic flows may enable precise tracking of items, thus enabling one to manage and control their routing over time. A more efficient routing may be achievable by imposing more control on the number of paths used, thus preventing interruptions or overload in operations. As a result, conveyor operations optimization based on dynamic flows may yield dramatically preferable results in overall process performance at the expense of a significantly higher level of complexity in performing operations.

For the first time, Ford and Fulkerson (1962) introduced the problem of the dynamic flow without considering edge capacities and demonstrated that such a problem could be solved in polynomial time. Latter, Fleischer (2001) propose a model that takes edge capacities into account, and their model assumes edges with zero transit times. Therefore, their model is incapable of

articulating temporal dependency, in which the number of items in an edge at any given time depends on the number of items in that same edge at its last time. Together with their first assumption, such an independency prevents employing their proposed algorithm for many real-world-based problems such as conveyor operations.

A category of dynamic flow problems is the quickest flow problem (QFP), which minimizes the makespan subject to capacities and transit times on the edges (Burkard et al., 1993). Routing becomes more complicated when conveyors are outfitted with multiple loading and unloading locations. Quickest routing is sometimes associated with requiring the use of a large and unrestricted number of paths in the network. Such a requirement is unrealistic for practical purposes due to the limitations in the installed hardware and the additional need for tight route monitoring. From a modeling point of view, such a demand can be translated into the routing via a limited number of paths. As a relevant concept, the path limitations denote the maximum number of paths used for routing items of each loading-unloading. Such a specific case of the QFP is called the quickest path problem (QPP). The QPP forces items to be routed on a particular number of paths. For example, in the context of convoy-type traffic flow, a single path model for a network with single loading-unloading was first proposed by Chen and Chin (1990). Recently, Kolliopoulos (2018) also studied the same concept for classical network flow problems where items are routed on a single path. Melchiori and Sgalambro (2015) formulate emergency management with the same concept and show that when a large number of evacuees must survive, assigning one evacuation path to the entire population may be considered neglectful. As a result, dividing and routing items of each loading-unloading on multiple paths would be a preferable alternative. However, the number of paths cannot increase to infinity in the conveyor due to the designed layout and installed hardware. Therefore the number of paths demands to be bound. Such restrictions, thus, may impair routing quality.

Optimal routing in conveyor operations requires a large and unrestricted number of paths in the network. Due to limitations of the conveyor layout and installed hardware, this situation may be unrealistic. Therefore, a restriction concerning the number of paths must be activated for each loading-unloading routing. Such an additional characteristic can be translated into integrating the concept of k-splittable flows (Baier et al., 2005), which ensures a routing process through bounded at most k_{max} paths within the dynamic setting of network flows. Despite the number of papers that focus on the theoretical side (see (Salimifard and Bigharaz, 2022) and the reference therein), the number of contributions addressing a combination of k-splittable and dynamic flows in the real-word setting is insignificant. Only Martens and Skutella (2006) have conducted some preliminary research. The problem of ranking the k_{max} quickest paths has been addressed by Pascoal et al. (2007) and applied to route data packets across the internet by Clímaco et al. (2007). They developed a generalized strategy for ranking k-shortest paths instead of one. Limiting the maximum number of paths can assist in efficient routing in real-world scenar-

ios (with hardware and layout constraints). This paper addresses the routing problem in conveyor operations using the QPP constrained by a maximum number of employed paths. The literature on multiple paths has demonstrated that adjusting the maximum number of paths may help many applications. Therefore, to model conveyor operations efficiently, QPP, together with tight controls concerning the number of paths used, is the subject of this research.

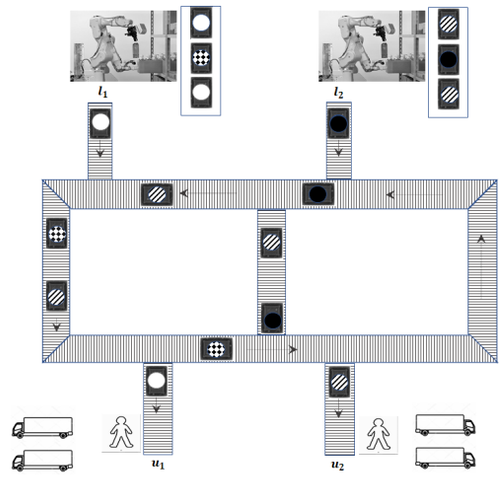
In this paper, we define the conveyor operations as a dynamic flows problem (Section 2). The required network flows setting is presented, specifically introducing the so-called dynamic digraph structure and its time-related features. In particular, two mathematical formulations are proposed in Section 3 to be able to bound the maximum number of paths between each loading-unloading pair. This number of path is defined as *one* and k_{max}, respectively. Therefore, we propose two novel Mixed Integer Linear Programming (MILP) formulations for optimizing conveyor operations to (i) determine edge flow values when routing is restricted to employ a single path during a time horizon to avoid any interruptions in the conveyor operations, and (ii) utilize the conveyor's full capacity by routing items of each loading-unloading on multiple paths over a time horizon. Therefore, we investigate the concept of path limitations, expressed as the maximum number of paths that can be employed. The exponential number of path-related variables raises several questions when we aim to optimize conveyor operations employing the second formulation. Those questions are often translated into large instances requirements related to designing an algorithm with low computation time that calculates high-quality routing. In Section 3.3, therefore, a metaheuristic algorithm is developed to find efficiently reasonable quality solutions. The algorithm employs a path-based MLIP in its exploration routine. At each iteration, a neighborhood is constructed by identifying a subset of paths for each distinct loading-unloading pair and then explored to optimality by solving via a MILP-solver. So at each iteration, the MILP formulation is restricted to the currently selected paths.

2 Problem definition

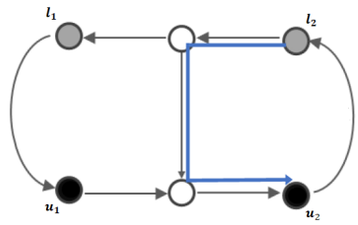
2.1 Conveyor network

Figure 1-a depicts an example of a conveyor and provides an illustrative overview of items' routing. This conveyor consists of a set of linked segments arranged in a closed-loop belt with a constant speed. It takes a particular time to route items along each segment from one end to another, namely transit time. Each segment has two capacity constraints. The maximum number of items that can enter a segment simultaneously, namely its inflow capacity, is limited. In addition, the maximum number of items that can be routed on a segment simultaneously, namely holding capacity, is also limited.

We can consider each junction as a node for constructing each conveyor's corresponding graph. Each junction models the connection between two segments. A conveyor segment that links together any two connected junctions



(a)



(b)

Fig. 1: Schematic layout of a conveyor with (a) two loading locations (l_1 and l_2) and two unloading locations (u_1 and u_2), and (b) its corresponding graph with an example $l_2 \rightarrow u_2$ path

shows an edge. The graph of example conveyor, Figure 1-a, is illustrated in Figure 1-b. Items' journey along a conveyor can be summarized as follows: Items are first brought from storage to their corresponding loading locations. A switching system successively loads them at the loading location onto the conveyor. They are then routed on a sequence of edges towards predefined unloading locations. Each item is collected at its unloading location and packed. The conveyor in Figure 1-a also consists of two loading locations (l_1 and l_2) and two unloading locations (u_1 and u_2).

A digraph $G_D(\mathcal{V}, \mathcal{A}, \mathcal{T})$ can represent conveyor network, where $\mathcal{V}$ denotes a set of nodes, $\mathcal{A}$ denotes a set of directed edges and $\mathcal{T}=\{0, 1, \ldots, T\}$ is a finite set of time steps through which a certain time horizon is discretized. A set of loading, $\mathcal{L} \subset \mathcal{V}$, and unloading, $\mathcal{U} \subset \mathcal{V}$, locations correspond to sources and sinks in the conveyor network. Node i of a given edge (i, j) is referred to as the

edge's tail, while node j is the edge's head. Three time-independent labels are associated with each edge $(i, j) \in \mathcal{A}$. A strictly positive inflow capacity, Ic_{ij}, a strictly positive holding capacity, Hc_{ij} and non-negative transit time, t_{ij}.

Figure 2 illustrates an edge with four-time steps transit time and one unit inflow capacity. The edge's holding capacity is limited to four items. One item, shown in black, traverses the edge by entering its tail at t=0 and exiting from its head at t=4. A white item is also routed through the same edge at the first available time instant, t=1, and will clear the edge at t=5.

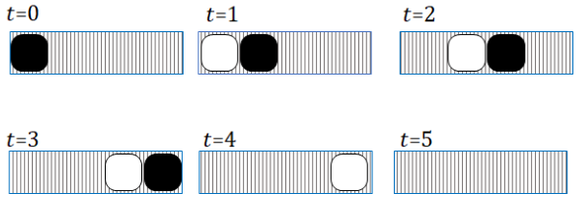


Fig. 2: items traversing an edge in a conveyor network

It is assumed that a set of loading-unloading transport pairs, $\mathcal{C}$, is given where each $c \in \mathcal{C}$ is defined by a triple $(l_c,\ u_c,\ \sigma_c)$ which corresponds to its loading location, $l_c \in \mathcal{L}$, unloading location, $u_c \in \mathcal{U}$, and item demands σ_c.

A l_c-u_c path, p, from $l_c \in \mathcal{L}$ to $u_c \in \mathcal{U}$ is defined as a sequence of the form p=$\langle l_c$=$i_0, i_1, i_2, ..., i_k$=$u_c\rangle$ and an edge given by $(i_n, i_{n+1}) \in \mathcal{A}$, $\forall n$=$0, ..., k-1$. Paths do always begin at a loading location $l_c \in \mathcal{L}$ and end at an unloading location $u_c \in \mathcal{U}$. For the graph of Figure 1-a an example $l_2 \rightarrow u_2$ path is illustrated in Figure 1-b in blue. The set of paths for loading-unloading pair c is denoted by $\mathcal{P}_c$. For a given path p, there is an associated inflow capacity u_p which denotes the minimum of its edges' inflow capacities, u_p= $\min_{(i,j)\in p} Ic_{ij}$.

Each path p's length, le_p, shows the sum of its edge transit times, le_p= $\sum_{(i,j)\in\mathcal{A}} \delta_{ij}^p t_{ij}$, where δ_{ij}^p is a binary parameter. Its value is 1 for all edges (i, j) that are part of path p, and 0 otherwise. The path is feasible for a loading-unloading if a positive amount of demand can arrive at the unloading location within the considered time horizon.

2.2 Routing alternatives

Figure 3 depicts a simple conveyor network with three items that must be routed between nodes 1 and 3 and one item that must be routed between nodes 2 and 3. Two alternatives of possible item routing while fulfilling edge constraints are given in Figures 4 and 5. Figure 4 shows routing takes three-time units when each set of items with the same loading-unloading is restricted to employ a single path. However, Figure 5 shows that it takes only two-time units to route all those items while each set of items with the same loading-unloading is not restricted to employing a single path. Removing restrictions

on the number of paths used for each set of items with the same loading-unloading improves the routing quality. Such alleviation demands investment in hardware installed for the routing, such as monitoring and switching systems. Conveyor operations, therefore, may benefit from the obtained results as a simultaneous control regarding both the number of the support paths that would be enabled and the time spent for routing each set of items with the same loading-unloading.

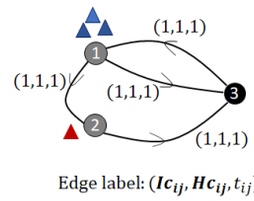


Fig. 3: A simple conveyor network with edges that have unit inflow and holding capacity as well as transit times and two set of items with the same loading-unloading: (1,3,3) and (2,3,1). The edge label $(Ic_{ij}, Hc_{ij}, t_{ij})$ indicates its inflow capacity, holding capacity, and transit time.

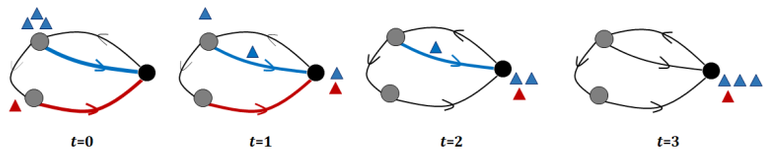


Fig. 4: Snapshots of a feasible items routing of the Figure 3. The items of each loading-unloading pair are restricted to employ a single path. Edges used by each set are colored to distinguish routes.

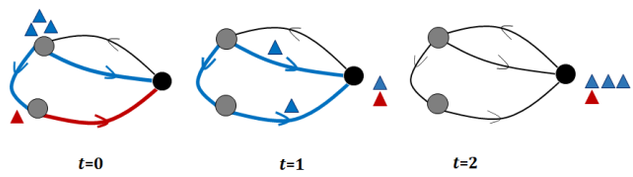


Fig. 5: Snapshots of a feasible item routing of the Figure 3. The items of each loading-unloading pair are not restricted to employing a single path. Edges used by each set are colored to distinguish routes.

2.3 Condensed time-extended network

Ford and Fulkerson (1958) developed the first solution method for dynamic flow problems based on the so-called time-extension procedure. This procedure generates an equivalent time-extended network (TEN) from the original dynamic network G_D. To construct this network for a conveyor, first, we assume an integral transit time for each conveyor segment. We should copy the underlying set of nodes for each discrete time step within the time horizon. Then we must connect those copies through edges according to their transit times. Moreover, nodes with storage characteristics and loading locations must be connected through edges in consecutive time steps. Therefore, the dynamic flow problem is an equivalent static problem in which the algorithms developed for classical network flow algorithms can be utilized. Recently this procedure has been employed in a few applications such as train timetabling (Fischer and Helmberg, 2014), water supply management (Raayatpanah et al., 2013) and fleet management (Bsaybes et al., 2019).

Although TEN can model temporal dependencies such as transit times, high dimensionality restricts their application in large time horizons. To overcome such an issue, we propose constructing the condensed TEN for the conveyor network, where the unit time step is replaced with a larger step. Without diminishing accuracy, the entire routing horizon can be divided into a smaller number of steps. This time step, t_{int}, is calculated based on the shortest conveyor network's edge length, l_{min}, and the conveyor's belt speed v (m/sec), as $t_{int}{=}l_{min}/v$. Therefore, the transit time for each edge (i,j) with length l_{ij} can be calculated by $t_{ij}{=}\lceil l_{ij}/l_{min}\rceil t_{int}$.

A conveyor's equivalent condensed TEN is a digraph $G_E(\mathcal{V}_e,\mathcal{A}_e)$, where $\mathcal{V}_e{=}\{(i,t) \mid i \in \mathcal{V},\ t \in \mathcal{T}_e\}$, where $\mathcal{T}_e{=}\{0,t_{int},2t_{int},...,(T_e-1)t_{int}\}$, and $T_e{=}\lceil T/t_{int}\rceil$. G_E contains T_e copies of the original graph's nodes. Total edges $\mathcal{A}_e$ are made by two type of edges namely, (i) $\mathcal{A}_{e_1}{=}\{((i,t),(i,t+t_{int})) \mid i \in \mathcal{V}, t \in \mathcal{T}_e\}$, and (ii) $\mathcal{A}_{e_2}{=}\{((i,t),\ (j,\hat{t})) \mid (i,j) \in \mathcal{A},\ t \in \mathcal{T}_e,\ \hat{t}{=}t{+}t_{ij} \leq T\}$. $\mathcal{A}_{e_1}$ contains *holdover* edges which allow items to wait at a single node. $\mathcal{A}_{e_2}$ contains a set of edges that connect separate nodes in the TEN, respecting the associated transit times.

Any feasible static flow in G_E can be interpreted as an equivalent feasible dynamic flow in the original digraph G_D. Flow on edge, $(i,j)_t$, corresponds to a flow of the same amount passing through the edge (i,j) at time t in the dynamic digraph. Conversely, given a dynamic flow in G_D with T as the time horizon, the equivalent flow can be constructed by sending the same amount of flow along edge $(i,j)_t$ that is assigned to the edge $(i,j) \in \mathcal{A}$ at time t. Figure 7 illustrates the proposed condensed TEN corresponding to the conveyor network of Figure 6 with seven-time steps of extension. The original six nodes are placed in the first vertical column on the right and are reproduced along the horizontal axis, making a separate copy for every discrete time step. Edges are drawn according to their transit time attribute at each time step whenever their head falls within the considered time horizon. To represent source nodes with capacity, edges are added to connect them in the TEN with

its copy at the next layer. In this paper, this graph clarifies some concepts and provides examples in the sections that follow.

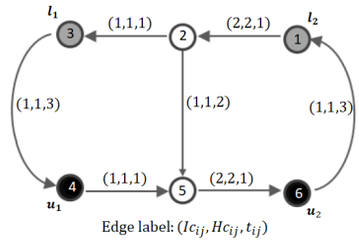


Fig. 6: A conveyor network with edge label $(Ic_{ij}, Hc_{ij}, t_{ij})$

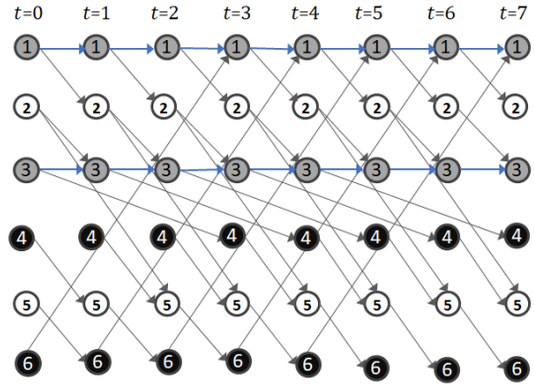


Fig. 7: The conveyor graph (Figure 6) extended in time. There is one copy of each node for each time step $[t, t+t_{int})$, for t=0, t_{int}, $2t_{int}$, ..., (T_e-1)t_{int}, where t_{int}=1. Holdover edges are highlighted in blue.

3 Mathematical formulations

This section introduces two formulations to route items through single or multiple paths. Both formulations respect the conveyor's operational constraints, such as edge's inflow and holding capacities.

3.1 Single-path-based formulation

The single-path-based formulation is developed based on edge flow variables. These variables for a graph G_D can be defined as:

$$\begin{aligned}\mathbf{x} : ((\mathcal{C}, \mathcal{A}), \mathcal{T}_e) &\rightarrow N \cup \{0\},\\ ((c, (i, j)), t) &\rightarrow x_{ij}^{ct},\end{aligned}$$

which associates to each edge (i, j) and time t, number of items of loading-unloading pair c entering its tail at the given time instant t and exiting from the its head at time $t+t_{ij}$.

In this formulation, we want to prevent any possible interruptions in the routing process while it is initiated and facilitate the packaging at the unloading location. Therefore we assume items collection and packaging associated with each loading-unloading must be conducted consecutively. Therefore once routing of each loading-unloading has started, items are routed through the same unique path over consecutive time steps. Given this condition, items of each loading-unloading at each loading location can be split into subsets. The first item leaving the loading location determines the path through which the entire items must be scheduled at consecutive uninterrupted time steps.

We need to define the following two sets, which will be employed in our formulation. The nodes at the tails of edges entering node i, Figure 8-a, are stored in set $\mathcal{I}^i$=$\{j \in \mathcal{V} | (j, i) \in \mathcal{A}\}$ while the nodes at the heads of outgoing edges, Figure 8-b, are stored in set $\mathcal{O}^i$=$\{j \in \mathcal{V} | (i, j) \in \mathcal{A}\}$.

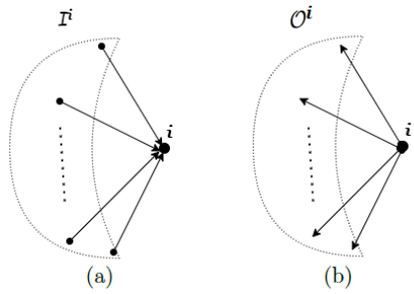


Fig. 8: $\mathcal{I}^i$ and $\mathcal{O}^i$ sets illustration

In what follows, the notation, parameters, and decision variables are first defined. Next, the model is introduced and explained to enable a complete presentation of conveyor operations. We have also assumed (i) corresponding to the conveyor network, the unloading locations have no outgoing edges, (ii) any item storage at nodes except loading nodes are forbidden, and (iii) edges of the type (i, i) are added to $\mathcal{A}$ only for loading locations $i \in \mathcal{L}$ with one time step transit time and infinite inflow and holding capacities.

Notation and parameters

$G_D(\mathcal{V}, \mathcal{A}, \mathcal{T}_e)$: the conveyor network
$\mathcal{L}$: set of loading locations
$\mathcal{U}$: set of unloading locations
$\mathcal{N}$: set of intermediate locations
$\mathcal{V}$=$\{\mathcal{L} \cup \mathcal{U} \cup \mathcal{N}\}$: set of all graph nodes
$\mathcal{A}$=$\{(i,j) : i,j \in \mathcal{V}\}$: set of all graph edges
$\mathcal{T}_e$=$\{0, t_{int}, 2t_{int}, ..., (T_e\text{-}1)t_{int}\}$: set of time steps
$\mathcal{C}$: set of all loading-unloading transport pairs
$\mathcal{O}_i$=$\{j : (i,j) \in \mathcal{A}\}$: set of head nodes of outgoing edges from node i
$\mathcal{I}_i$=$\{j : (j,i) \in \mathcal{A}\}$: set of tail nodes of edges entering node i
Ic_{ij} : inflow capacity of the edge (i,j)
Hc_{ij} : holding capacity of the edge (i,j)
t_{ij} : transit time of the edge (i,j)

Decision variables

$x_{ij}^{ct} \geq 0$: items of loading-unloading transport pair c leave node i at time t heading to node j

$$y_{ij}^{ct} : \begin{cases} 1 & \text{iff items of loading-unloading transport pair } c \text{ leave node } i \text{ at time } t \text{ heading to node } j \\ 0 & \text{otherwise} \end{cases}$$

The objective is to minimize makespan. Makespan is represented in the objective function (1) by ζ. It is defined as the total time required to route all given items on the conveyor network completely.

$$\textbf{Minimize} \quad \zeta \tag{1}$$

$$(t + t_{ju_c})(y_{ju_c}^{ct} - y_{ju_c}^{c(t+t_{int})}) - \zeta \leq 0 \quad \forall c \in \mathcal{C}, \quad t \in \mathcal{T}_e, \quad u_c \in \mathcal{U}, \quad j \in \mathcal{I}_{u_c} \tag{2}$$

Constraints (2) determine the time instant at which the last item arrives at its unloading location, which provides an estimate of the lower bound of makespan.

$$\sum_{j \in \mathcal{O}_{l_c}} x_{l_c j}^{c0} = \sigma_c \qquad \forall c \in \mathcal{C}, \quad l_c \in \mathcal{L} \tag{3}$$

$$\sum_{j \in \mathcal{I}_{u_c}} \sum_{t=0}^{T - t_{ju_c}} x_{ju_c}^{ct} = \sigma_c \qquad \forall c \in \mathcal{C} \tag{4}$$

Constraints (3) impose flow conversion for each loading location at time zero. Routing of each loading-unloading transport pair is not forced to start at time zero, given that loading locations are connected via holdover edge. Flow

conversion Constraints (4) force to route the given number of items within the time horizon from the loading location of each loading-unloading transport to its corresponding unloading location.

$$\sum_{j \in \mathcal{O}_{l_c}, j \neq l_c} y_{l_c j}^{c0} \leq 1 \qquad \forall c \in \mathcal{C}, \quad l_c \in \mathcal{L} \tag{5}$$

$$y_{l_c l_c}^{c(t+t_{int})} \leq y_{l_c l_c}^{ct} \qquad \forall c \in \mathcal{C}, \quad t \in \mathcal{T}_e, \quad l_c \in \mathcal{L} \tag{6}$$

$$\sum_{j \in \mathcal{O}_{l_c}, j \neq l_c} y_{l_c j}^{c(t+t_{int})} \leq y_{l_c l_c}^{ct} \qquad \forall c \in \mathcal{C}, \quad t \in \mathcal{T}_e, \quad l_c \in \mathcal{L} \tag{7}$$

$$\sum_{j \in \mathcal{O}_i} y_{ij}^{ct} \leq 1 \qquad \forall c \in \mathcal{C}, \quad t \in \mathcal{T}_e, \quad i \in \mathcal{N} \tag{8}$$

Constraints (5)-(7) are the path conservation constraints at loading locations, which enable the pause of each loading-unloading routing by en-routing through the holdover edge of each loading location. The routing activates whenever the entire items of loading-unloading are routed through a single path. Therefore, constraints (8) ensure loading-unloading items are not split into subsets at intermediate locations.

$$y_{l_c l_c}^{ct} + y_{l_c j}^{ct} - y_{l_c j}^{c(t+t_{int})} \leq 1 \quad \forall c \in \mathcal{C}, \quad t \in \mathcal{T}_e, \quad l_c \in \mathcal{L}, \quad j \neq l_c, \quad j \in \mathcal{O}_{l_c} \tag{9}$$

$$y_{ij}^{ct} + y_{mi}^{c(t-t_{mi})} + y_{mi}^{c(t+t_{int}-t_{mi})} - y_{ij}^{c(t+t_{int})} \leq 2 \quad \forall c \in \mathcal{C},\ i \in \mathcal{N},\ t \in \mathcal{T}_e,\ m \in \mathcal{I}_i,\ j \in \mathcal{O}_i \tag{10}$$

Constraints (9) and (10) ensure that once routing of the first item of each loading-unloading transport pair begins, the parallel and uninterrupted path is assigned for all items of that loading-unloading transport until all items have been routed.

Any interruption in conveyor operations should be prevented. When several items assigned to different paths share the same edge simultaneously, this corresponds to a situation where items may collide. This situation is a source of interruptions in conveyor operations and may violate an edge inflow capacity. Figure 9 shows items routing for the example Figure 6. Figure 10 shows routing items of each loading-unloading transport for the same example with c_1: (1,6,3) and c_2: (3,6,3) while considering Ic_{12}=1, Ic_{56}=1. It is seen that at time steps 4 and 5 items collide while passing to/ entering edge $(5, 6)$. Because the arrival items at each node must not exceed the inflow capacity, such an issue will resolve while the makespan will be worsened by two-time steps (27.6%). Therefore, constraints (11) ensure that when multiple items enter the same edge simultaneously, they do respect inflow capacity.

$$\sum_{c \in \mathcal{C}} y_{ij}^{ct} \leq Ic_{ij} \qquad \forall (i,j) \in \mathcal{A}, \quad i \neq j, \quad t \in \mathcal{T}_e \tag{11}$$

The number of items that traverse an edge is obtained by summing its inflow

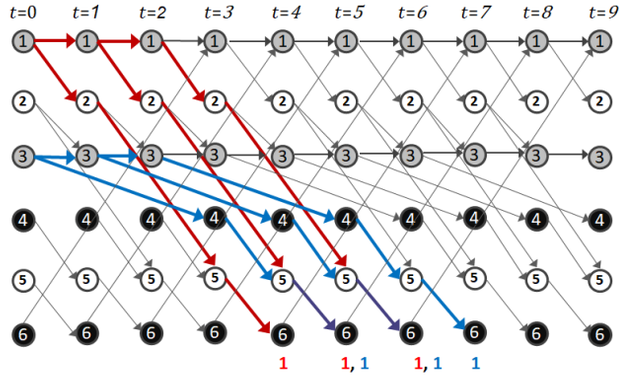


Fig. 9: Routing c_1: (1,6,3) on path $\langle 1, 2, 5, 6\rangle$ and c_2: (3,6,3) on path $\langle 3, 4, 5, 6\rangle$.. Purple edges contain items form both c_1 and c_2

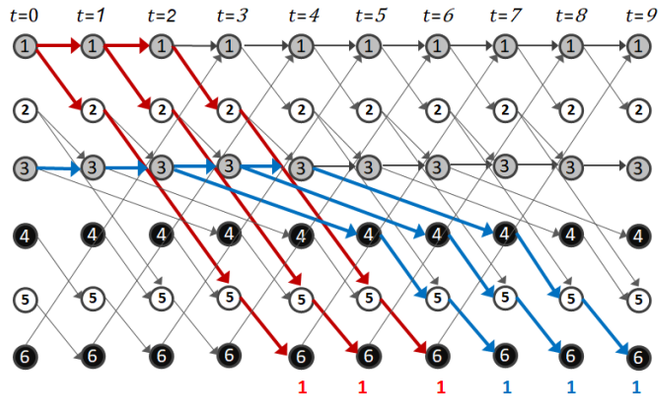


Fig. 10: Routing c_1: (1,6,3) on path $\langle 1, 2, 5, 6\rangle$ and c_2: (3,6,3) on path $\langle 3, 4, 5, 6\rangle$ non-simultaneously, where inflow capacities are considered as Ic_{12}=1 and Ic_{56}=1

over time. Therefore, holding capacity constraints on edges are respected by (12).

$$\sum_{c\in C}\sum_{t}^{t+t_{ij}} x_{ij}^{ct} \leq H c_{ij} \qquad \forall (i,j)\in\mathcal{A},\quad t\in\mathcal{T}_e,\quad t+t_{ij}\leq T_e \qquad (12)$$

Constraints (13) enforce the flow conservation for each loading-unloading transport at intermediate and loading locations. Constraints (14) enforce the path

conservation at intermediate locations.

$$\sum_{j \in \mathcal{I}_i} x_{ji}^{c(t-t_{ji})} - \sum_{j \in \mathcal{O}_i} x_{ij}^{ct} = 0 \quad \forall c \in \mathcal{C}, \quad t \in \mathcal{T}_e, \quad i \in (\mathcal{N} \cup \mathcal{L}), \quad t - t_{ji} \geq 0 \tag{13}$$

$$\sum_{j \in \mathcal{I}_i} y_{ji}^{c(t-t_{ji})} - \sum_{j \in \mathcal{O}_i} y_{ij}^{ct} = 0 \quad \forall c \in \mathcal{C}, \quad t \in \mathcal{T}_e, \quad i \in \mathcal{N}, \quad t - t_{ji} \geq 0 \tag{14}$$

$$x_{ij}^{ct} \leq c_{ij} y_{ij}^{ct} \quad \forall c \in \mathcal{C}, \quad t \in \mathcal{T}_e, \quad (i,j) \in \mathcal{A} \tag{15}$$

$$y_{ij}^{ct} \in \{0,1\} \quad \forall c \in \mathcal{C}, \quad t \in \mathcal{T}_e, \quad (i,j) \in \mathcal{A} \tag{16}$$

$$x_{ij}^{ct} \geq 0 \quad \forall c \in \mathcal{C}, \quad t \in \mathcal{T}_e, \quad (i,j) \in \mathcal{A} \tag{17}$$

A feasible solution in the single-path-based formulation corresponds to a temporary repetition items routing on a single path. The number of repetitions depends on the number of demanded items for loading-unloading transport c, σ_c. Figure 11, for the same network from Figure 6, demonstrates when c_1: (1,6,**7**) and c_2: (3,6,3) how due to capacity constraints the makespan can increase in the single-path-based formulation. Figure 12 showcases how the makespan can be improved from 13 to 12 (8%) if items of each loading-unloading transport are not restricted to using only a single, unique path but allowed c_1 to employ $\langle 1,2,5,6 \rangle$ and $\langle 1,2,3,4,5,6 \rangle$ paths.

Removing single path routing constraints for optimizing conveyor operations opens up the possibility of improved routing. According to Martens and Skutella (2006) preliminary research, no efficient methods can tackle realistic situations unless more than one path is utilized. Therefore, we can gain further insight by investigating deeper in a direction that intensifies more control on routing. Hence, the multi-path-based formulation in the next section seeks to account for several path limitations while adhering to conveyor operational constraints.

3.2 Multi-path-based formulation

This section provides a path-flow-based MILP formulation and its adaptation to conveyor operations before presenting a metaheuristic-based algorithm to solve it. The path-flow-based formulation requires as input a complete set of available paths for each loading-unloading transport pair. Paths differ from each other with respect to at least one edge. The aim is to find multi-path-based items routing to minimize the makespan while the number of different paths used by each loading-unloading transport $c \in \mathcal{C}$, must not exceed a given maximum k_{max}^c. The notation, parameters, and decision variables are first defined in what follows. Next, the model is introduced and explained to enable a complete presentation of routing in conveyor operations. We have also assumed (i) corresponding to the conveyor network, the loading and unloading

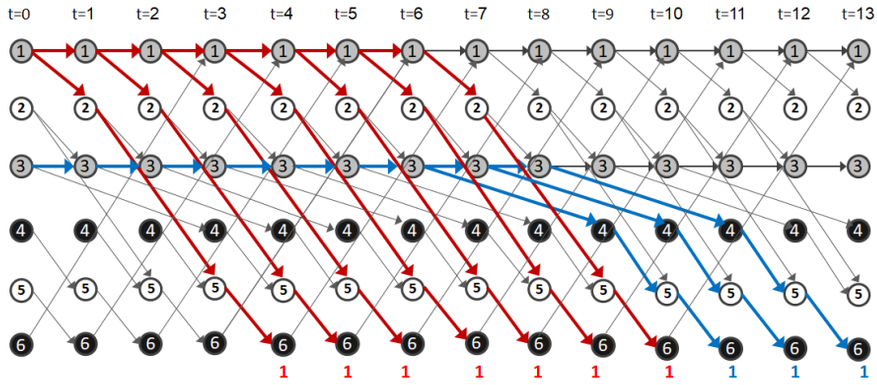


Fig. 11: Single-path-based items routing of the Figure 6. Routing c_1(1,6,7) on path $\langle 1, 2, 5, 6\rangle$ and c_2(3,6,3) on path $\langle 3, 4, 5, 6\rangle$.

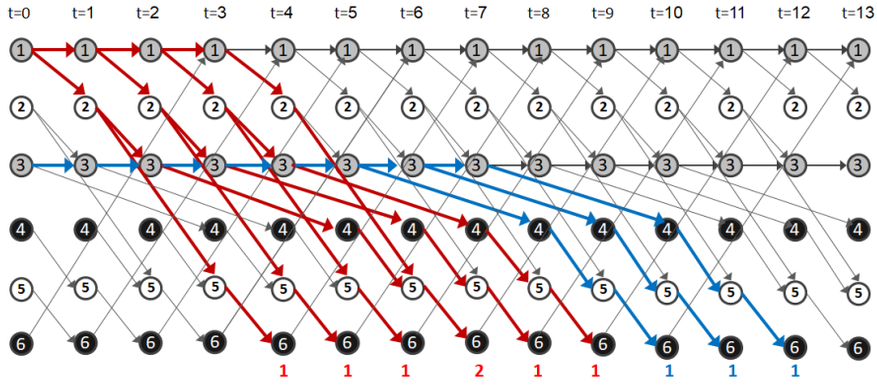


Fig. 12: Improved result of Figure 11 by splitting the c_1(1,6,7) flow items across $\langle 1, 2, 5, 6\rangle$ and $\langle 1, 2, 3, 4, 5, 6\rangle$ paths.

locations have no incoming and no outgoing edges, respectively, (ii) no edges of the type (i, i) are included in $\mathcal{A}$, and (iii) any item storage at nodes except loading nodes are forbidden.

Notation and parameters

$G_D(\mathcal{V}, \mathcal{A}, \mathcal{T}_e)$: the original conveyor network
$\mathcal{V}$: set of all nodes
$\mathcal{A}$: set of all edges
$\mathcal{T}_e$=$\{0, t_{int}, 2t_{int}, ..., (T_e - 1)t_{int}\}$: set of time steps
$\mathcal{C}$: set of all loading-unloading transports
$\mathcal{P}_c$: set of paths for loading-unloading transport c
$u_p = Min_{(i,j)\in p} Ic_{ij}$: the path p's inflow capacity

δ_{ij}^{p}: binary parameter, its value is 1 iff $(i,j) \in p$, and 0 otherwise.
$le_p = \sum_{(i,j)\in p} \sigma_{ij} t_{ij}$: length of the path p
t_i^p: time required to reach node i following path p
$C_{ct} = Min\{\sigma_c, \sum_{p\in\mathcal{P}_c:(le_p\leq t)} u_p\}$: maximal number of items of loading-unloading transport c that allowed to arrive unloading location before t

Decision variables

$x_p^{ct} \geq 0$: the number of items of loading-unloading transport c leaving its loading location at time t through path p

$$y^{ct}: \begin{cases} 1 & \text{iff some items of loading-unloading transport } c \text{ arrives at unloading location at time } t \\ 0 & \text{otherwise} \end{cases}$$

$$z^{cp}: \begin{cases} 1 & \text{iff path } p \text{ is used by loading-unloading transport } c \\ 0 & \text{otherwise} \end{cases}$$

$$\textbf{Minimize} \quad \zeta \tag{18}$$

The proposed model aims to minimize the makespan (ζ) subject to the following constraints.

$$ty^{ct} - \zeta \leq 0 \qquad \forall c \in \mathcal{C}, \quad t \in \mathcal{T}_e \tag{19}$$

Constraints (19) determine the time step at which the last item arrives at its unloading location, which estimates the lower bound to the makespan.

$$\sum_{p\in\mathcal{P}_c} x_p^{c(t-le_p)} \leq C_{ct} y^{ct} \qquad \forall c \in \mathcal{C}, \quad t \in \mathcal{T}_e \tag{20}$$

Constraints (20) couple variables x_p^{ct} and y_c^t. If at time t there is no item of loading-unloading transport c arrives at its destination, then the items which are routed on any path $p \in \mathcal{P}_c$ at time $t - le_p$ must be equal to zero.

$$\sum_{p\in\mathcal{P}_c} z^{cp} \leq k_{max}^c \qquad \forall c \in \mathcal{C} \tag{21}$$

A feasible routing for the conveyor is characterized by a set of at most k_{max}^c paths for each loading-unloading transport c, namely $\mathcal{P}_c$=$\{p_1^c,\ p_2^c, \ldots,\ p_{k_{max}^c}^c\}$. Constraints (21) force each loading-unloading transport c have at most k_{max}^c different paths.

$$\sum_{t\in\mathcal{T}_e} \sum_{p\in\mathcal{P}_c} x_p^{ct} = \sigma_c \qquad \forall c \in \mathcal{C} \tag{22}$$

Constraints (22) state the number of items that must be routed for each loading-unloading transport c.

$$\sum_{c\in\mathcal{C}}\sum_{p\in\mathcal{P}_c}\delta_{ij}^{p}x_{p}^{c(t-t_i^p)}\leq Ic_{ij}\qquad\forall(i,j)\in\mathcal{A},\quad t\in\mathcal{T}_e \tag{23}$$

Constraints (23) enforce edge inflow capacities at each time step.

$$\sum_{c\in\mathcal{C}}\sum_{p\in\mathcal{P}_c}\sum_{t=t}^{t=t+t_{ij}}\delta_{ij}^{p}x_{p}^{c(t-t_i^p)}\leq Hc_{ij}\quad\forall(i,j)\in\mathcal{A},\quad t\in\mathcal{T}_e,\ t+t_{ij}\leq T_e \tag{24}$$

Constraints (24) enforce edge holding capacities at each time step.

$$x_p^{ct}\leq u_p z^{cp}\qquad\forall c\in\mathcal{C},\quad t\in\mathcal{T}_e,\quad p\in\mathcal{P}_c \tag{25}$$

Constraints (25) restrict the items routing on a given path at each time step t. If the path has not been chosen, nothing will be routed through it.

$$y^{ct}\in\{0,1\}\qquad\forall c\in\mathcal{C},\quad t\in\mathcal{T}_e \tag{26}$$

$$z^{cp}\in\{0,1\}\qquad\forall c\in\mathcal{C},\quad p\in\mathcal{P}_c \tag{27}$$

$$x_p^{ct}\geq 0\qquad\forall c\in\mathcal{C},\quad t\in\mathcal{T}_e,\quad p\in\mathcal{P}_c \tag{28}$$

3.3 A local-search-based matheuristic approach

For each loading-unloading transport c, the multi-path-based formulation necessitates the challenging enumeration of a large number of paths. As the number of paths increases, the multi-path-based size/difficulty increases. Therefore, solving the large-scale multi-path-based formulation becomes computationally challenging. Furthermore, Melchiori and Sgalambro (2018) investigates the NP-hardness in the strong sense for a similar problem with only one loading and unloading. Therefore, we develop a simple and efficient metaheuristic capable of handling the massive amount of data associated with conveyor operations. Algorithm 1 provides the pseudo-code for the suggested local search-based algorithm.

Algorithm 1 has two main steps: *initialization* and *improvement*. In the initialization step, a set of paths for each loading-unloading transport c is randomly selected, and their exploration is realized by solving the related multi-path-based formulation. The multi-path-based formulation is restricted to these initial paths and solved to optimality with a MILP solver, thus providing a local optimum. Since k_{max} number of paths are given as input for each loading-unloading transport c, Constraints (21) should be removed. Then, at each improvement step, a different neighborhood is constructed by randomly selecting for each loading-unloading transport c a collection of k_{max} candidate paths from its path list. The MILP then solves the related restricted multi-path-based formulation, which will return a solution with an optimum

Algorithm 1 The local search-based algorithm

Require: I_{max} ▷ maximum number of non-improving iterations
Require: k^c_{max} ▷ maximum number of paths for loading-unloading pair c
Require: tl ▷ computation time limit
Require: T ▷ time horizon
Require: $\mathcal{P}_c$ ▷ set of paths for loading-unloading pair c
Ensure: (x,val(x)) ▷ the best loading-unloading pair flow and its makespan

Initialization
$x \leftarrow 0$
$val(x) \leftarrow T$
$UB_{makespan} \leftarrow T$
$j \leftarrow 0, n \leftarrow 1$
for $c \in \mathcal{C}$ **do**
  $P_0 \leftarrow \cup_c p_c$ ▷ p_c randomly selects k^c_{max} paths for loading-unloading pair c from $\mathcal{P}_c$
end for
$(x_0, val(x_0)) \leftarrow \text{solveMILP}_{(P_0, val(x), tl)}$
$UB_{makespan} \leftarrow val(x_0)$
Improvement
while $n \leq I_{max}$ **do**
  for $path \in \mathcal{P}_c$ **do**
    if $path$ termination time $\geq UB_{makespan}$ **then**
      remove $path$ from $\mathcal{P}_c$
    end if
  end for
  $P_j \leftarrow \cup_c p_c$ ▷ p_c randomly selects k^c_{max} paths for loading-unloading pair c from $\mathcal{P}_c$
  $(x_{j+1}, val(x_{j+1})) \leftarrow \text{solveMILP}_{(P_j, val(x_j), tl)}$
  if $val(x_{j+1}) \leq UB_{makespan}$ **then**
    $UB_{makespan} \leftarrow val(x_{j+1})$
  else
    $n \leftarrow n + 1$
  end if
end while
return (x,val(x))

makespan. Note that this improved makespan prunes the path list by discarding those paths whose transmission times (i.e., when the last item that is routed on that path arrives at its corresponding unloading location) are greater than or equal to the best makespan obtained so far.

4 Computational study

Computational experiments are conducted to evaluate single-path-based formulation and the metaheuristic proposed for the multi-path-based formulation. Their evaluation is performed against the QFP on a set of instances. The QFP solves the routing problems with no upper bounds on the number of paths. The set of instances generated by Melchiori and Sgalambro (2015) are employed as a benchmark to conduct these experiments. These instance graphs have two different sizes: s_1 (50 nodes, 186 edges) and s_2 (75 nodes, 292 edges). Each set contains five instances with five different loading-unloading transports $c \in \{1, 2, 3, 4, 5\}$. For each size and loading-unloading transport combination, two demand levels are considered where the demands of *level b*

are double the demands of *level a*. This results in a comprehensive set of 20 instances.

For the multi-path-based formulation, the proposed heuristic is employed to solve each instance when k^c_{max}=$\{1, 2, 3, 4, 5\}$, with the results averaged over all instances. Tables 3 and 4 provide detailed results for each value of k^c_{max}. The time horizon for performing the routing is defined as 16 time steps. Gurobi time limit (tl) is set to 1h. Tables 1 and 2 report the computational results for levels a and b, respectively. Columns $|\mathcal{V}|$ (number of nodes), $|\mathcal{A}|$ (number of edges), $|\mathcal{C}|$ (number of loading-unloading transports) denote the structure of the network. LB^* corresponds to the QFP makespan when no upper bounds on the number of usable paths is applied. These value are reported by Melchiori and Sgalambro (2015). Each value provides a valid lower bound for the makespan. $Gap_S(\%)$ and $Gap_M(\%)$ provide the makespan gaps of the proposed single-path-based and multi-path-based formulations to the LB^*, respectively.

Table 1: Aggregated results for the *level a* instances

Instance	$\|\mathcal{V}\|$	$\|\mathcal{A}\|$	$\|\mathcal{C}\|$	LB^*	Single-path makespan	CPU Time (sec)	$Gap_S(\%)$	Multi-path makespan (avg.)	CPU Time (sec)	$Gap_M(\%)$
s_1-1	50	186	1	6	8	8	33.3	6.4	0.2	6.7
s_1-2	50	186	2	5	6	7	20.0	5.2	0.4	4.0
s_1-3	50	186	3	6	8	60	33.3	7.2	1.8	20.0
s_1-4	50	186	4	5	8	828	60.0	7.0	2.2	40.0
s_1-5	50	186	5	6	9	1050	50.0	7.4	2	23.3
s_2-1	75	292	1	6	9	80	50.0	7.0	0.8	16.7
s_2-2	75	292	2	6	10	741	66.7	7.2	2.4	20.0
s_2-3	75	292	3	6	7	1104	16.7	7.0	5.4	16.7
s_2-4	75	292	4	6	9	3489	50.0	8.0	5.2	33.3
s_2-5	75	292	5	7	tl	–	–	8.0	3.4	14.2

Table 2: Aggregated results for the *level b* instances

Instance	$\|\mathcal{V}\|$	$\|\mathcal{A}\|$	$\|\mathcal{C}\|$	LB^*	Single-path makespan	CPU Time (sec)	$Gap_S(\%)$	Multi-path makespan (avg.)	CPU Time (sec)	$Gap_M(\%)$
s_1-1	50	186	1	6	10	5	66.7	6.8	0.2	13.3
s_1-2	50	186	2	6	10	11	66.7	7.0	0.4	16.7
s_1-3	50	186	3	6	11	53	83.3	7.4	2.6	23.3
s_1-4	50	186	4	6	9	550	50.0	7.0	3.6	16.7
s_1-5	50	186	5	6	10	2351	66.7	7.8	2.4	30.0
s_2-1	75	292	1	6	11	42	83.3	7.0	2.1	16.7
s_2-2	75	292	2	6	12	885	100	7.6	2.6	26.7
s_2-3	75	292	3	6	8	487	33.3	7.8	7.2	30.0
s_2-4	75	292	4	6	16	3421	176.7	10.2	7.8	70.0
s_2-5	75	292	5	8	tl	–	–	10.8	4.8	35.0

The complete results of the metaheuristics are provided in Tables 3 and 4, where the first and eighth columns identify instances. The next four columns

after each instance denote its number of nodes $|\mathcal{V}|$, number of edges $|\mathcal{A}|$, number of loading-unloading transports $|\mathcal{C}|$, and the k_c^{max} column. The makespan and computational time t are reported in the next two columns. The proposed metaheuristic provides, as expected, makespans that are smaller than or equal to the single-path-based formulation for $k_c^{max} \geq 1$. For k_c^{max}=1, metaheuristic results in the same makespan as the single-path-based formulation.

These results show that the single-path-based formulation can solve all but one instance to optimality within the time limit (tl) of 1 hour for both level a and level b. As expected, introducing additional constraints (8)-(10) in the single-path-based formulation increases the makespan for all instances. The metaheuristic also provides solutions with a makespan always greater than or equal to the lower bound (LB^*). The results get closer to this lower bound as the value of the k_c^{max} parameter is increased. These results show that the multi-path-based formulation's average makespan is far shorter than the single-path-based makespan for all instances, providing evidence that controlling the number of used paths enables efficient routing in real-world conveyor operations. For both formulations, $Gap_S(\%)$ and $Gap_M(\%)$ increase as the item demands increase. More specifically, the average $Gap_S(\%)$ deteriorates largely, on average (38%), from level a to level b, which proves the sensitivity of the single-path-based formulation to changes in the number of item demands that must be routed between each loading-unloading pair.

5 Conclusions and future work

Distribution centers (DCs) as intermediaries receive and store goods before they are delivered to customers. Due to increasing fast delivery demands, DCs have limited time to sort and pack parcels as the time horizon available for delivery service is primarily devoted to delivery transports. The conveyor operations affect the sorting and packing the most. Therefore, the present paper facilitates the sorting and packing processes by reducing the time required for a conveyor to route items from storage to packing areas. It has been demonstrated that conveyor operations optimization corresponds to solving items routes scheduling on the conveyor network.

The optimal routing in complex conveyor operations can be identified by leveraging the benefits of dynamic network flows-based formulations while accounting for conveyors' operational and structural restrictions, such as finite edge capacities and the limited number of paths possible for routing. This research developed two single-path-based and multi-path-based formulations for efficient routing while avoiding any interruptions in conveyor operations.

We find that improving the routing performance is accessible at the expense of increased complexity in solving multi-path-based formulation and managing operations. To resolve such complexity, a heuristic approach was also proposed to obtain high-quality solutions in a reasonable computational time. Computational studies demonstrate a proof-of-concept of both formula-

tions and evaluate the quality of the proposed local-search-based metaheuristic. Furthermore, the results confirm the suitability and the superiority of the multi-path-based formulation.

There are currently no available real-world datasets capable of providing researchers with the opportunity to evaluate their algorithm's efficiency. Therefore, a practical research direction for future research would be to generate real-world-based data or introduce a dataset generator that enables researchers to accurately assess their models and algorithms' (relative) performance.

Table 3: Results for each value of k^c_{max} for the *level a* instances

Instance					Matheuristic		Instance					Matheuristic	
Size	$\|\mathcal{V}\|$	$\|\mathcal{A}\|$	$\|\mathcal{C}\|$	k_c^{max}	Multi-path makespan	CPU time (sec)	Size	$\|\mathcal{V}\|$	$\|\mathcal{A}\|$	$\|\mathcal{C}\|$	k_c^{max}	Multi-path makespan	CPU time (sec)
s_1-1	50	186	1	1	8	0	s_2-1	75	292	1	1	9	0
s_1-1	50	186	1	2	6	0	s_2-1	75	292	1	2	7	1
s_1-1	50	186	1	3	6	0	s_2-1	75	292	1	3	7	1
s_1-1	50	186	1	4	6	0	s_2-1	75	292	1	4	6	1
s_1-1	50	186	1	5	6	1	s_2-1	75	292	1	5	6	1
s_1-2	50	186	2	1	6	0	s_2-2	75	292	2	1	10	0
s_1-2	50	186	2	2	5	0	s_2-2	75	292	2	2	8	2
s_1-2	50	186	2	3	5	0	s_2-2	75	292	2	3	6	2
s_1-2	50	186	2	4	5	1	s_2-2	75	292	2	4	6	3
s_1-2	50	186	2	5	5	1	s_2-2	75	292	2	5	6	5
s_1-3	50	186	3	1	9	0	s_2-3	75	292	3	1	7	2
s_1-3	50	186	3	2	8	1	s_2-3	75	292	3	2	7	2
s_1-3	50	186	3	3	7	1	s_2-3	75	292	3	3	7	4
s_1-3	50	186	3	4	6	1	s_2-3	75	292	3	4	7	7
s_1-3	50	186	3	5	6	6	s_2-3	75	292	3	5	7	12
s_1-4	50	186	4	1	8	0	s_2-4	75	292	4	1	9	2
s_1-4	50	186	4	2	8	1	s_2-4	75	292	4	2	8	5
s_1-4	50	186	4	3	8	2	s_2-4	75	292	4	3	8	6
s_1-4	50	186	4	4	6	3	s_2-4	75	292	4	4	8	5
s_1-4	50	186	4	5	5	5	s_2-4	75	292	4	5	7	8
s_1-5	50	186	5	1	9	0	s_2-5	75	292	5	1	9	2
s_1-5	50	186	5	2	9	0	s_2-5	75	292	5	2	8	4
s_1-5	50	186	5	3	8	1	s_2-5	75	292	5	3	8	3
s_1-5	50	186	5	4	7	5	s_2-5	75	292	5	4	8	3
s_1-5	50	186	5	5	6	4	s_2-5	75	292	5	5	7	5

Table 4: Results for each value of k_{max}^{c} for the *level b* instances

Instance					Matheuristic		Instance					Matheuristic	
Size	$\|\mathcal{V}\|$	$\|\mathcal{A}\|$	$\|\mathcal{C}\|$	k_c^{max}	Multi-path makespan	CPU time (sec)	Size	$\|\mathcal{V}\|$	$\|\mathcal{A}\|$	$\|\mathcal{C}\|$	k_c^{max}	Multi-path makespan	CPU time (sec)
s_1-1	50	186	1	1	10	0	s_2-1	75	292	1	1	11	0
s_1-1	50	186	1	2	6	0	s_2-1	75	292	1	2	6	1
s_1-1	50	186	1	3	6	0	s_2-1	75	292	1	3	6	1
s_1-1	50	186	1	4	6	0	s_2-1	75	292	1	4	6	5
s_1-1	50	186	1	5	6	1	s_2-1	75	292	1	5	6	4
s_1-2	50	186	2	1	10	0	s_2-2	75	292	2	1	12	0
s_1-2	50	186	2	2	7	0	s_2-2	75	292	2	2	8	2
s_1-2	50	186	2	3	6	0	s_2-2	75	292	2	3	6	3
s_1-2	50	186	2	4	6	1	s_2-2	75	292	2	4	6	3
s_1-2	50	186	2	5	6	1	s_2-2	75	292	2	5	6	5
s_1-3	50	186	3	1	11	0	s_2-3	75	292	3	1	8	2
s_1-3	50	186	3	2	7	1	s_2-3	75	292	3	2	8	4
s_1-3	50	186	3	3	7	5	s_2-3	75	292	3	3	8	11
s_1-3	50	186	3	4	6	1	s_2-3	75	292	3	4	8	10
s_1-3	50	186	3	5	6	6	s_2-3	75	292	3	5	7	9
s_1-4	50	186	4	1	9	0	s_2-4	75	292	4	1	16	2
s_1-4	50	186	4	2	7	1	s_2-4	75	292	4	2	11	15
s_1-4	50	186	4	3	7	4	s_2-4	75	292	4	3	10	6
s_1-4	50	186	4	4	6	8	s_2-4	75	292	4	4	8	5
s_1-4	50	186	4	5	6	5	s_2-4	75	292	4	5	6	11
s_1-5	50	186	5	1	10	0	s_2-5	75	292	5	1	16	2
s_1-5	50	186	5	2	8	0	s_2-5	75	292	5	2	12	4
s_1-5	50	186	5	3	8	1	s_2-5	75	292	5	3	12	3
s_1-5	50	186	5	4	7	5	s_2-5	75	292	5	4	8	3
s_1-5	50	186	5	5	6	6	s_2-5	75	292	5	5	8	12

References

(2017). The State of Ecommerce Order Fulfillment & Shipping .

(2019). 5 ways to shorten last mile delivery.

Baier, G., Köhler, E., and Skutella, M. (2005). The k-splittable flow problem. *Algorithmica*, 42(3-4):231–248.

Boysen, N., Briskorn, D., Fedtke, S., and Schmickerath, M. (2018). Automated sortation conveyors: A survey from an operational research perspective. *EJOR*.

Bsaybes, S., Quilliot, A., and Wagler, A. K. (2019). Fleet management for autonomous vehicles using flows in time-expanded networks. *Top*, 27(2):288–311.

Burkard, R. E., Dlaska, K., and Klinz, B. (1993). The quickest flow problem. *Zeitschrift für Operations Research*, 37(1):31–58.

Chen, T.-L., Chen, J. C., Huang, C.-F., and Chang, P.-C. (2021). Solving the layout design problem by simulation-optimization approach–a case study on a sortation conveyor system. *Simulation Modelling Practice and Theory*, 106:102192.

Chen, Y. L. and Chin, Y. H. (1990). The quickest path problem. *COR*, 17(2):153–161.

Clímaco, J. C. N., Pascoal, M. M. B., Craveirinha, J. M. F., and Captivo, M. E. V. (2007). Internet packet routing: Application of a k-quickest path algorithm. *European Journal of Operational Research*, 181(3):1045–1054.

De Koster, R., Le-Duc, T., and Roodbergen, K. J. (2007). Design and control of warehouse order picking: A literature review. *European journal of operational research*, 182(2):481–501.

Fedtke, S. and Boysen, N. (2017). Layout planning of sortation conveyors in parcel distribution centers. *Transportation Science*, 51(1):3–18.

Fischer, F. and Helmberg, C. (2014). Dynamic graph generation for the shortest path problem in time expanded networks. *Mathematical Programming*, 143(1-2):257–297.

Fleischer, L. K. (2001). Universally maximum flow with piecewise-constant capacities. *Networks: An International Journal*, 38(3):115–125.

Ford, J. L. R. and Fulkerson, D. R. (1958). Constructing maximal dynamic flows from static flows. *Operations research*, 6(3):419–433.

Ford, J. L. R. and Fulkerson, D. R. (1962). Flows in networks princeton university press. *Princeton, New Jersey*, 276:22.

Johnson, M. E. (1997). The impact of sorting strategies on automated sortation system performance. *IIE transactions*, 30(1):67–77.

Kappmeier, J.-P. (2015). *Generalizations of flows over time with applications in evacuation optimization*. epubli.

Kolliopoulos, S. G. (2018). Disjoint paths and unsplittable flow.

Martens, M. and Skutella, M. (2006). Length-bounded and dynamic k-splittable flows. In *Operations Research Proceedings 2005*, pages 297–302. Springer.

Melchiori, A. and Sgalambro, A. (2015). Optimizing emergency transportation through multicommodity quickest paths. *Transportation Research Procedia*, 10:756–765.

Melchiori, A. and Sgalambro, A. (2018). A matheuristic approach for the quickest multicommodity k-splittable flow problem. *Computers & Operations Research*, 92:111–129.

Mor, A. and Speranza, M. G. (2022). Vehicle routing problems over time: a survey. *Annals of Operations Research*, pages 1–21.

Pascoal, M., Captivo, M., and Clímaco, J. (2007). Computational experiments with a lazy version of a k quickest simple path ranking algorithm. *Top*, 15(2):372–382.

Raayatpanah, M., Ghasvari, H., Ebrahimnejad, A., et al. (2013). Generalized water supply management over time. *Middle East Journal of Scientific Research*, 15(3):383–388.

Salimifard, K. and Bigharaz, S. (2022). The multicommodity network flow problem: state of the art classification, applications, and solution methods. *Operational Research*, 22(1):1–47.

Wahba, P. (2015). Amazon waives same-day delivery fees for prime members. http://fortune.com/2015/05/28/amazon-same-day-delivery-prime.