MISTA 2019

Facilitating same-day delivery through conveyor operations modeling and optimization in distribution centers

Farzaneh Karami · Wim Vancroonenburg · Greet Vanden Berghe

the date of receipt and acceptance should be inserted later

Abstract Optimizing conveyor operations concerns the problem of transferring a given number of items among multiple loading and unloading locations within a minimum makespan. Achieving this requires solving a multi-commodity quickest flow problem which is adapted to accommodate real-world conveyor operation constraints. Something which is currently missing from the literature. To address this gap, this study compares two path-based formulations for routing and scheduling items on the conveyor network. The first concerns the transshipment of items on a single path during a given time horizon with the objective being to minimize the makespan. The second formulation targets the same objective through utilizing multiple paths. A heuristic algorithm is proposed for solving the multi-path-based formulation given the need for high quality solutions within reasonable computational times. Experiments are conducted on a set of benchmark instances and the results provide evidence concerning how controlling the number of used paths enables handling real-world levels of complexity.

1 Introduction

Offering same-day delivery (SDD) concept is a recent trend in the e-commerce industry, and one enabled by some online logistic service providers. SDD is becoming very common and the service has led to an ever-increasing freight volume, all of which requires sorting, packing and transportation under tight deliver schedules (Wahba, 2015). The time horizon available to service all SDD requests is primarily devoted to the transportation, with only a very narrow time window left for sorting and packing. Therefore, a high capacity and efficient distribution center (DC) is a prerequisite for truly realizing SDD. DCs serve as intermediary storage centers for goods in-between two stages of a supply chain. Automation is a solution for

F. Karami[1] (corresponding author), W. Vancroonenburg[12], G. Vanden Berghe[1]
[1] KU Leuven, Department of Computer Science, CODeS & imec, Gebroeders De Smetstraat 1, 9000 Gent, Belgium
Tel.: +32 9 265 87 04
E-mail: {farzaneh.karami, wim.vancroonenburg, greet.vandenberghe}@cs.kuleuven.be
[2] Postdoctoral research fellow at Flanders Research Foundation - FWO

meeting SDD needs and increasingly more DCs are being automated to increase throughput. These DCs mainly consist of storage areas, each of which is equipped with a robot which automatically collects and loads items onto the conveyor. DCs typically employ conveyor networks to transport items from the storage areas to the packing areas. Each conveyor consists of a set of linked segments arranged in a closed-loop belt, along which there are multiple loading and unloading locations. Each segment has a limited capacity and takes a certain period of time to transport items. To optimize conveyor operations, the problem of transferring a given number of items from the loading to the unloading locations within a minimum makespan must be solved. Makespan denotes the time between the start of the planning horizon and the last items' arrival time at its unloading location.

Boysen et al. (2018a) conducted a comprehensive survey of the scientific literature concerning all kinds of fully-automated conveyor-based sorting systems from an operational research perspective. All of the proposed methods assume one of two major approaches: simulation-based and mathematical formulation-based. For example, Russell and Meller (2003) proposed a descriptive model for conveyor operations and conducted a simulation study which included stochastic aspects to verify their results. Jarrah et al. (2014) considered the distribution center of a postal service provider as a hierarchical system. By doing so, they decompose conveyor operations into smaller subsystems to handle. Boysen et al. (2018b) introduced a mathematical model for conveyor operations and proposed and validated priority rule- and dynamic programming-based methods. They investigated the relationship between sequencing and sorting operations and suggested that sequence optimization would help prevent congestion at loading locations. Banerjee et al. (1999) considered conveyor operations as a transshipment problem where the goal is to minimize the total travel time.

Quickest commodity flow-over-time approaches are among the most suited to provide a modelling representation of conveyor operations. Each commodity is defined as a triple of its loading location, unloading location and amount of flow demanded. For example, the quickest path problem (QPP) requires loading-unloading flow to be routed on a given network while taking into account the capacity of arcs, while minimizing the makespan. Since a conveyor has multiple loading and unloading locations, its operations are closely related to the multi-commodity QPP and solving it is equivalent to solving the maximum flow-over-time problem. While there is substantial literature on the static multi-commodity QPP, hardly any results on multi-commodity QPP with flow-over-time are available. Melchiori and Sgalambro (2018) have illustrated that the multi-commodity QPP with flow-over-time is NP-hard. Therefore, equipping a conveyor with multiple loading and unloading locations necessarily introduces an additional layer of complexity to the routing and scheduling of its flows.

The concept of flow-over-time was originally introduced by Ford and Fulkerson (1958). They extended the static flows problem to a flow-over-time context by demanding the completion of an optimal routing within a given fixed time horizon. They introduced a general procedure to translate a flow-over-time problem to a static one by means of a time-expanded graph. This enabled taking advantage of a polynomial algorithm developed for the static problem. Ford and Fulkerson

showed that an optimal solution for the maximum flow-over-time problem was easily attained by decomposing it into flow on paths. While this seems practical in the case of the single commodity flow problem, it demands some adjustment for the multi-commodity flow problem.

The QPP forces flow to be routed on a single path (Pascoal et al., 2006). Since this constraint will impair solution quality in many real-world applications, the literature has suggested increasing the number of paths to a reasonable fixed number. The problem of ranking k-quickest paths has been addressed by Pascoal et al. (2007) and applied to the routing of data packets across the internet by Clímaco et al. (2007). The literature on multiple paths has demonstrated that adjusting the fixed number of paths helps in many real-world applications. For example, the emergency transportation problem has been solved by Melchiori and Sgalambro (2018) by minimizing the makespan of transshipment operations while imposing restrictions on the number of paths for each commodity. Such a modeling approach is new for conveyor operations. Furthermore, assuming a fixed number of paths for each commodity without considering the flow demands of each is not realistic in a conveyor operation context.

A few of the unaddressed question have motivated the present paper. Is it better to employ a single path for each commodity or split each commodity's flow across several paths? Does controlling the number of paths used per commodity help? How does a realistic restriction on the number of paths utilized affect solution quality? And, finally, is imposing a realistic number of paths demands an essential modeling requirement which is not captured by the multi-commodity QPP? To address these, two path-based formulations for the routing and scheduling of commodities' flow are proposed and compared: the single-path-based multi-commodity QPP and the multi-path-based multi-commodity QPP. Finally, a local search-based matheuristic is proposed for solving the multi-path-based multi-commodity QPP.

2 Problem description

Figure 1 depicts a simple example of a DC's conveyor belt and provides an illustrative overview of items' flow. This conveyor has two loops and consists of two loading locations, L_1 and L_2, and two unloading locations, U_1 and U_2.

The conveyor network can be represented as a directed graph $G = (V, A)$ where V is the set of nodes and A is the set of arcs. G can be constructed by considering each junction as a node and the conveyor segment that links any two connected nodes as an arc. These arcs have limited capacities and take some time to transit on each of them. For each arc $(i, j) \in A$, parameters c_{ij} and λ_{ij} denote its capacity and travel time, respectively. Loading and unloading locations of a conveyor correspond to sources and sinks in this corresponding graph. It is assumed that a set of commodities $\mathbb{C}$ is given, where a commodity $c \in \mathbb{C}$ is defined by a triple (L_c, U_c, σ_c) which corresponds to its loading location, unloading location and flow demands. A path p from $i \in V$ to $j \in V$ in G is defined as a sequence of the form $p = \langle i = v_0, v_2, ..., v_{l_p} = j \rangle$ and $(v_k, v_{k+1}) \in A$ for any $k \in \{0, ..., l_p\}$. The set of

paths for commodity c will be denoted by $\mathcal{P}_c$. Paths do always begin at a loading location and end at an unloading location.

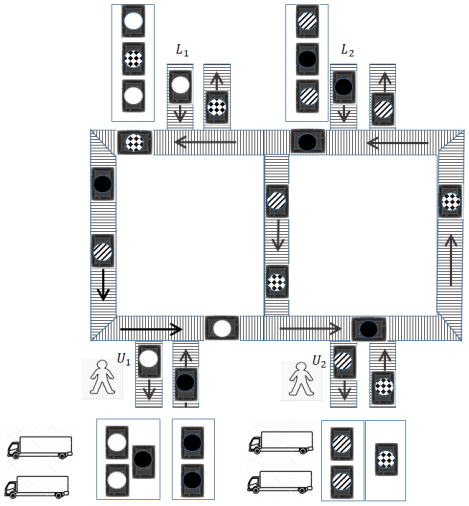

Figure 1: Schematic layout of a conveyor belt as a hatched area

A commodity flow's journey along a conveyor can be summarized as follows: First, they are brought from storage to their respective loading locations based on their predefined sequence. Second, a switch system successively loads them at the loading location onto the conveyor. Commodity flow then travel to predefined unloading locations. Finally, the flow which has left the conveyor is collected and packed by human workers at unloading locations.

As introduced by Ford and Fulkerson (1958), a conveyor commodity flow-over-time problem can be translated into a static flow problem on a time-expanded graph. To generate the time-expanded graph, first the entire scheduling horizon is divided into a finite number of intervals. The interval is calculated based on the shortest conveyor arc, l_{min}, and the conveyor velocity v ($\frac{m}{sec}$), as $t_{int} = \frac{l_{min}}{v}$. Capacity c_{ij} and transmission time λ_{ij} for each arc (i,j) with length l_{ij} can be calculated by: $c_{ij} = \lceil \frac{l_{ij}}{l_{min}} \rceil$ and $\lceil \frac{l_{ij}}{v} \rceil$, respectively. The time-expanded graph is a directed graph $G_{\hat{T}} = (V^{\hat{T}}, A^{\hat{T}})$ where $V^{\hat{T}}$ contains $\hat{T} = \lceil \frac{T}{t_{int}} \rceil$ replicas of the original graph's nodes, where $V_{\hat{T}} = \{(i,t) | i \in V,\ t \in \hat{T}\}$. This graph has two sets of arcs: $A_H = \{((i,t),(i,t+t_{int}))|i \in V, t = 0,...,T - t_{int}\}$, and $A_M = \{((i,t),\ (j,\hat{t}))|(i,j) \in A,\ \hat{t} = t + \lambda_{ij} \leq T\}$. A_H contains the holdover arcs which allow flow to wait at a single node. Meanwhile, A_M contains the set of arcs which connect separate nodes in the time-expanded graph, respecting the associated travel time. Note that $A_{\hat{T}} = A_M \cup A_H$. Since the only nodes with a capacity greater than zero are loading locations, the holdover arcs are only allowed at these locations. The time-expanded graph corresponding to the conveyor graph of Figure 2 which has seven time intervals (seven steps of expansion) is illustrated in Figure 3.

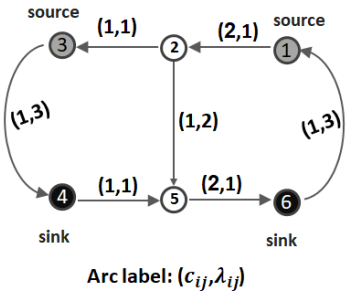

Figure 2: A conveyor network

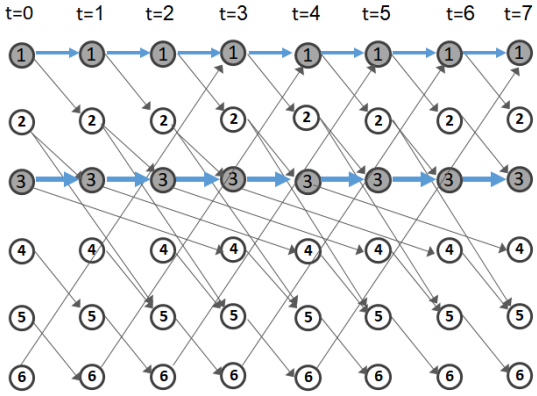

Figure 3: The conveyor graph expanded in time. The holdover arcs are highlighted in blue.

3 Solution approach

During the SDD horizon, customers' requests are continuously registered and initially listed based on their arrival time. In order to achieve optimal routing and scheduling and meet the SDD goal, two path-based formulations are introduced in this section. The aim of these formulations is to find routes whereby commodity flow items are shipped through either a single or multiple paths. Paths differ from each other with respect to at least one arc. In the multi-path-based formulation, paths are therefore not required to be completely different. The number of different paths used by each commodity c must not exceed a given parameter k_c^{max}. Both proposed formulations respect the conveyor's practical limitation: shared arc capacities must be respected and holdovers are only allowed at source nodes.

The single-path-based formulation In this formulation, the entire flow of each commodity must be transshipped through a single path. More specifically, it is assumed that a commodity's flow may wait at the loading location, but once the transshipment is started it is deterministically routed through the same unique path over consecutive time intervals. It is assumed that once a commodity's item is flowing on

the path, no other commodity's items can use this path. The path will be available only after the entire flow of the current commodity has been transported. Given these conditions, the commodities' flow at each loading location can be sequenced and split into subsets, each of which is scheduled at a distinct point in time. The first unit of flow leaving the loading location determines the path through which the entire commodity flow must be scheduled on consecutive uninterrupted time intervals. This prevents any possible interruptions and perturbations in the process while it is underway. Table 1 presents a list of constraints for the single-path-based formulation.

Table 1: The single-path-based formulation constraints

Constraint	Description
C1	Each commodity flow is not forced to start at time zero
C2	Each commodity flow transshipment is postponed by routing it through a holdover arc
C3	Each commodity flow will not split into subsets
C4	Flow arrivals at each junction must not take place simultaneously
C5	Flow conservation at all nodes
C6	Arc capacity constraints
C7	The flow demand of each commodity constrains total amount of items to be transshipped

When several flow items assigned to different paths share the same arc at the same time, this corresponds to a node in the time-expanded graph where flows coming from several arcs collide. This situation is a source of interruption in conveyor operations and one which violates an arc capacity. In this way, interruption of the conveyor operation is prevented. Figures 4 and 5 show the routing of commodity flow while considering these conveyor operational constraints. It is clearly seen that by considering that flow arrivals at each junction must not take place simultaneously the makespan will extended by two time intervals.

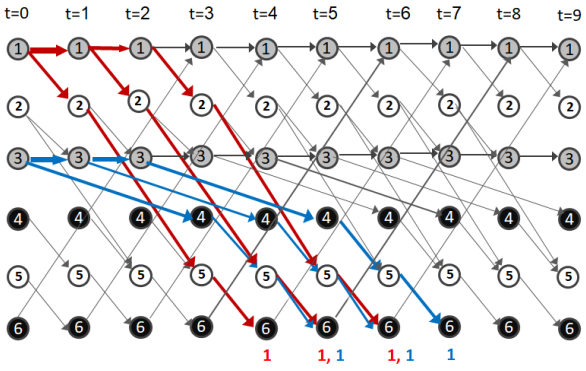

Figure 4: Routing c_1(1,6,3) on path 1-2-5-6 and c_2(3,6,3) on path 3-4-5-6

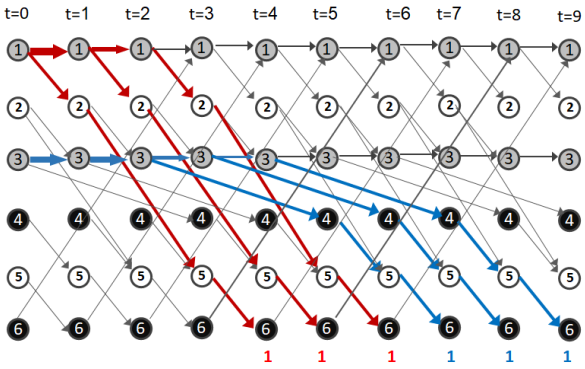

Figure 5: Routing $c_1(1,6,3)$ on path 1-2-5-6 and $c_2(3,6,3)$ on path 3-4-5-6 non-simultaneously

The multi-path-based formulation A feasible solution for given commodity flow in the single-path-based formulation corresponds to a temporary repetition of the path, where the number of repetitions depends on commodity flow demand. Figure 6 demonstrates how the makespan can increase markedly in the single-path-based formulation when the flow demand of a commodity increases. This clearly hinders the overall goal of SDD. Figure 7 showcases how the makespan can be shrunk from 13 to 12 (8%) if commodity flow items are not restricted to using only a single, unique path.

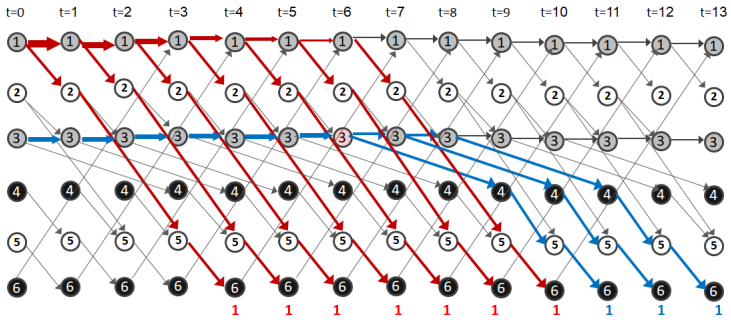

Figure 6: Routing $c_1(1,6,7)$ on path 1-2-5-6 and $c_2(3,6,3)$ on path 3-4-5-6

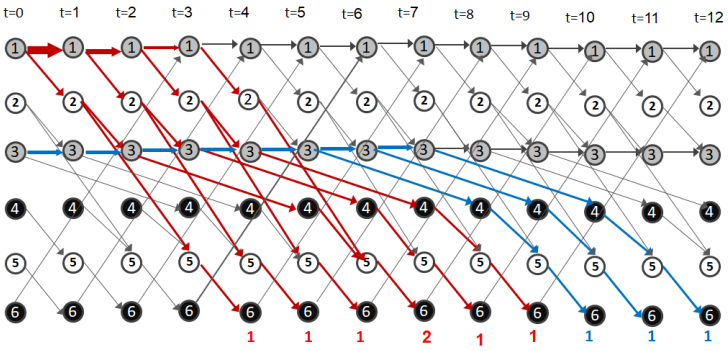

Figure 7: Improved result of Figure 6 by splitting the c_1(1,6,7) flow across 1-2-5-6 and 1-2-3-4-5-6 paths

In the multi-path-based formulation, which is based on a multi-commodity QPP formulation, the aim is to find a set of at most k_c^{max} paths through which commodity c's flow can be transshipped. Table 2 provides the list of constraints for the multi-path-based formulation.

Table 2: The multi-path-based formulation constraints

Constraint	Description
C1	Flow variables which must be coupled with their associated path-related variables
C2	The maximum number of different paths for each commodity
C3	Arc capacity constraints
C4	Path capacity constraints
C5	Flow demand constraints

A Local search-based matheuristic approach The multi-path-based formulation demands the enumeration of all commodity paths. Since this enumeration in the time-expanded graph is challenging, solving the multi-path-based formulation is also a difficult task. These characteristics motivated the design of an efficient matheuristic capable of accommodating the immense amount of data associated with conveyor operations. A local search-based algorithm is proposed to solve this formulation whose pseudo-code provided in Algorithm 1.

Algorithm 1 has two main steps: initialization and improvement. In the initialization step, a random set of paths for each commodity is selected and their exploration is realized by solving the related multi-path-based formulation. This involves calling the Gurobi MILP solver, thus providing a local optimum. The improvement step then randomly selects the first k_{max}^c paths from the feasible path list for each commodity c. The Gurobi MILP is then called, which will return a solution with optimum makespan. This improved makespan prunes the path list to the paths whose transshipment times less than or equal to the best solution obtained so far.

Algorithm 1 The local search-based algorithm

Input: I_{max} ▷ maximum number of non-improving iterations
Input: k_c^{max} ▷ maximum number of paths for commodity c
Input: tl ▷ computation time limit
Input: T ▷ time horizon
Input: $\mathcal{P}_c$ ▷ set of paths for commodity c
Output: (x,val(x)) ▷ the best commodity flow and its makespan

Initialization
$x \leftarrow 0$
$val(x) \leftarrow T$
$UB_{makespan} \leftarrow T$
$j \leftarrow 0$, $n \leftarrow 1$
for $c \in \mathbb{C}$ **do**
  $P_0 \leftarrow \cup_c p_c$ ▷ p_c randomly selects k_c^{max} paths for commodity c from $\mathcal{P}_c$
end for
$(x_0, val(x_0)) \leftarrow solveMILP(P_0, val(x), tl)$
$UB_{makespan} \leftarrow val(x_0)$
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 commodity c from $\mathcal{P}_c$
  $(x_{j+1}, val(x_{j+1})) \leftarrow solveMILP(P_j, val(x_j), tl)$
  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

4 Computational experiments

Computational experiments have been conducted to evaluate both the single-path-based formulation and the heuristic proposed for the multi-path-based formulation. To conduct these experiments, a set of benchmark instances generated by Melchiori and Sgalambro (2015) is considered. These benchmark instance graphs have two different sizes: $s1$ (50 nodes, 186 arcs) and $s2$ (100 nodes, 292 arcs). There are 5 instances with different number of commodities $c = \{1, 2, 3, 4, 5\}$ associated with each of these sets. For each combination, two demand levels are considered where the flow demands of *level b* is double that of *level a*. This results in a total 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. The time horizon for performing the transshipment is defined as 16 time intervals. For both graphs it is assumed that $c_{ij} = 1$, $\lambda_{ij} = 1$, with a Gurobi time limit of 1h.

4.1 Experimental results

Tables 3 and 4 report the computational results for levels a and b, respectively. Columns $|V|$ (number of nodes), $|A|$ (number of arcs), $|c|$ (number of commodities)

denote the structure of the network. LB^* corresponds to the makespan when a complete enumeration of all the available paths for each commodity is possible and the resulting dimension of the MILPs can be handled by Gurobi. The multi-path-based formulation results are averaged over all values for k_c^{max}. $Gap_S(\%)$ and $Gap_M(\%)$ provide the makespan gaps of the single-path and multi-path-based formulations to the LB^*, respectively.

Table 3: Results for the *level a* instances

Instances size	$\|V\|$	$\|A\|$	$\|c\|$	LB^*	Single-path-based makespan	$Gap_S(\%)$	Multi-path-based makespan	$Gap_M(\%)$
s1	50	186	1	6	8	25.0	6.2	3.2
s1	50	186	2	5	8	37.5	5.2	3.8
s1	50	186	3	6	9	33.3	7.2	16.7
s1	50	186	4	5	8	37.5	7.0	28.6
s1	50	186	5	6	9	33.3	7.4	18.9
s2	100	292	1	6	9	33.3	7.0	14.3
s2	100	292	2	6	10	40.0	7.2	16.7
s2	100	292	3	6	7	16.7	7.0	16.7
s2	100	292	4	6	9	33.3	8.0	25.0
s2	100	292	5	7	tl	–	8.0	37.5

Table 4: Results for the *level b* instances

Instances size	$\|V\|$	$\|A\|$	$\|c\|$	LB^*	Single-path-based makespan	$Gap_S(\%)$	Multi-path-based makespan	$Gap_M(\%)$
s1	50	186	1	6	10	40.0	6.4	6.25
s1	50	186	2	6	10	40.0	6.2	3.22
s1	50	186	3	6	11	45.4	7.4	18.90
s1	50	186	4	6	9	33.3	7.0	14.30
s1	50	186	5	6	10	40.0	7.8	23.07
s2	100	292	1	6	11	45.4	7.0	14.30
s2	100	292	2	6	12	50.0	7.6	21.05
s2	100	292	3	6	8	25.0	7.8	23.07
s2	100	292	4	6	16	62.5	10.2	41.17
s2	100	292	5	8	tl	–	10.8	25.92

These computational results show that the single-path-based formulation is able to solve all but one instance to optimality within the time limit for both level *a* and *b*. Furthermore, these results show that the average makespan of multi-path-based formulation is far shorter than the single-path-based makespan for all instances. For both formulations, $Gap_S(\%)$ and $Gap_M(\%)$ increase as the flow demand increases. Moreover, these results show that the proposed local search-based matheuristic outperforms the single-path-based approach thereby providing evidence on how controlling the number of used paths enables flow optimization in real-world conveyor problems and capable of fulfilling the goal of SDD.

5 Conclusions

Since the time horizon available to service all same-day delivery requests is primarily devoted to transportation, only a very narrow time window remains for distri-

bution centres to sort and pack items. This study attempts to minimize the total period it takes a distribution centre conveyor to transship items from storage to the packing areas. Conveyor operation is therefore modeled as a problem which demands the routing and scheduling of multi-commodity flow. This flow is routed and scheduled by employing single-path-based and multi-path-based formulations which respect a range of real-world-based operational constraints.

Due to the complexity of solving the multi-path-based formulation, a heuristic approach was also proposed which helps to provide high quality solutions within reasonable computational time. A set of benchmark instances are used to conduct the experiments. This set includes two network sizes with various numbers of commodities and two demand levels.

Computational experiments demonstrate a proof-of-concept of our proposed formulations and evaluate the quality of the local-search-based matheuristic. Furthermore, results confirm the suitability and the superiority of solution quality of both formulations. An important direction for future work is to collect real-world-based conveyor instances. Another promising direction is to improve the proposed heuristics with a view to generating higher quality solutions for instances of the real-world conveyor.

Acknowledgements

Wim Vancroonenburg is a postdoctoral research fellow at Research Foundation Flanders - FWO Vlaanderen. This research is also funded by the FWO as part of the Strategic Basic Research (SBO) program under the ORDinL (Operational Research and Data science in Logistics) project. Editorial consultation provided by Luke Connolly (KU Leuven).

References

Banerjee, P., Banerjee, A., and Tesic, R. (1999). Optimization-based planning heuristic for material flow congestion avoidance in conveyor network design. *Production Planning & Control*, 10(2):181–193.

Boysen, N., Briskorn, D., Fedtke, S., and Schmickerath, M. (2018a). Automated sortation conveyors: A survey from an operational research perspective. *European Journal of Operational Research*.

Boysen, N., Fedtke, S., and Weidinger, F. (2018b). Optimizing automated sorting in warehouses: The minimum order spread sequencing problem. *European Journal of Operational Research*, 270(1):386–400.

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.

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

Jarrah, A. I., Qi, X., and Bard, J. F. (2014). The destination-loader-door assignment problem for automated package sorting centers. *Transportation Science*, 50(4):1314–1336.

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.

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.

Pascoal, M. M., Captivo, M. E. V., and Clímaco, J. C. (2006). A comprehensive survey on the quickest path problem. *Annals of Operations Research*, 147(1):5–21.

Russell, M. L. and Meller, R. D. (2003). Cost and throughput modeling of manual and automated order fulfillment systems. *IIE Transactions*, 35(7):589–603.

Wahba, P. (2015). Amazon waives same-day delivery fees for prime members. *URL http://fortune. com/2015/05/28/amazon-same-day-delivery-prime/. [Online; accessed. 27-August-2015]*.