DOI: 10.2195/lj_Proc_hormes_en_202012_01
URN: urn:nbn:de:0009-14-51420

Fleet Sizing of Dynamically Routed In-plant Milk-run Vehicles Based on a Genetic Algorithm

Dimensionierung von Routenzugsystemen mit dynamischen Routen auf Basis eines Genetischen Algorithmus

Fabian Hormes
Amin Siala
Christian Lieb
Johannes Fottner

Chair of Materials Handling, Material Flow, Logistics
Department of Mechanical Engineering
Technical University of Munich (TUM)

In-plant milk-run (MR) systems enable efficient supply of assembly lines in small lot sizes. One major challenge for MR systems are demand fluctuations and short-term changes within schedules. Dynamic control strategies aim at increasing flexibility and efficiency of MR systems in volatile environments. This paper presents an application-oriented approach for determining the fleet size of an MR system with dynamically controlled routes based on a genetic algorithm. The approach is evaluated and discussed using a case study from a commercial vehicle manufacturer. The results show that the approach enables effective analytical dimensioning of MR systems with dynamic routes. In addition, the case study indicates that the implementation of dynamic routes can lead to a reduction in fleet size.

[Keywords: In-plant milk-run, control strategies, routing problems, genetic algorithms]

Innerbetriebliche Routenzugsysteme (RZS) ermöglichen eine effiziente Materialversorgung von Montagelinien in kleinen Losgrößen. Eine Herausforderung für RZS sind Bedarfsschwankungen und kurzfristige Änderungen in Fahrplänen. Das Ziel von dynamischen Steuerungsstrategien ist die Erhöhung der Flexibilität und Effizienz von RZS in volatilen Umgebungen. Dieser Beitrag stellt einen anwendungsorientierten Ansatz zur Bestimmung der Flottengröße eines RZS mit dynamischen Routen basierend auf einem genetischen Algorithmus vor. Der Ansatz wird anhand einer Fallstudie eines Nutzfahrzeugherstellers evaluiert und diskutiert. Die Ergebnisse zeigen, dass der vorgestellte Ansatz eine effektive analytische Dimensionierung von RZS mit dynamischen Routen ermöglicht. Die Fallstudie legt zudem nahe, dass der Einsatz von dynamischen Routen zu einer Reduzierung der benötigten Anzahl an Fahrzeugen führen kann.

[Schlüsselwörter: Routenzugsysteme, Steuerungsstrategien, Routing-Probleme, Genetische Algorithmen]

1 INTRODUCTION

In-plant milk-runs (MR) enable efficient supply of assembly lines in small lot sizes [Tak06]. They deliver material on defined routes throughout the factory following predetermined schedules [Har03]. Routes and schedules are defined in the planning process of an MR system based on average material consumption and layout restrictions [Bru12]. Each route is assigned a certain number of operators and vehicles. The routes start and end at an MR station (depot) close to a warehouse or supermarket. During each MR cycle (tour), a certain number of parts are delivered to the line-side points of use (POU). A tour includes three or more stops at different POUs. However, the number of parts delivered on a tour is limited by the capacity of the MR vehicle [VDI5586a]. Although MR systems are widely used in the industry due to efficiency advantages over single or unit-load transport, challenges arise regarding fluctuating transport demands or short-term changes within schedules [Klu13, Lie18]. The dynamic control of MR systems is a concept to increase flexibility of in-plant MR systems and to ensure efficiency despite fluctuations [Kle15].

Existing methodologies for MR system dimensioning focus on MR systems with static, previously defined routes. In this paper we present a fleet sizing approach for MR systems with flexible, dynamically controlled routes. The approach extends the standard procedure defined in VDI 5586 by a genetic algorithm that generates dynamic routes and enables practitioners to determine the required number of MR vehicles under given constraints. The paper is structured as follows: in section 2, we describe the theoretical background regarding MR system planning including the standard fleet sizing approach and control strategies. In section 3, the development of the fleet sizing approach for MR systems with dynamic routes and the genetic algorithm are outlined. In section 4, we evaluate and discuss the approach using a case study example of a commercial vehicle manufacturer. Section 5 contains a summary and an outlook on future work.

DOI: 10.2195/lj_Proc_hormes_en_202012_01
URN: urn:nbn:de:0009-14-51420

2 In-Plant Milk-Run (MR) Systems Design

2.1 Planning of MR Systems

The basic planning approach for MR systems is defined in the VDI 5586 standard. The planning framework consists of four major planning steps [VDI5586b]. Step 1 addresses the collection of relevant data for the planning task. Relevant data includes, for example, the different types of parts, the required throughput and the layout restrictions. In step 2, planning alternatives are generated. A planning alternative is created by combining different processes and technology variants, e.g. with regard to the loading and unloading of MR vehicles [Keu18]. Step 3 includes the dimensioning and determination of the required resources, e.g. the number of MR vehicles (fleet size) or operators, for each planning alternative generated. Finally, in step 4, all planning alternatives are evaluated based on performance indicators and economic factors.

2.2 Fleet Sizing according to VDI 5586

The average MR cycle time (t_{cyc}) can be calculated using the average travel time (t_T), the time for acceleration and deceleration at the stops (t_S), the average train loading time at the depot (t_L), and the average unloading and handling time at a stop (t_U):

$$t_{cyc} = t_T + t_S + t_L + t_U$$

Given the required throughput per part k to be delivered at stop m ($T_{k,m}$), the throughput of the total route (T_R) with the total number of stops (n_M) and parts (n_K) can be calculated as follows:

$$T_R = \sum_{n_M} \sum_{n_K} T_{k,m}$$

Based on the total route throughput, the tour start interval (t_{TS}) can be calculated using the train capacity (C_{max}) and its degree of utilization (η):

$$t_{TS} = \frac{\eta * C_{max}}{T_R}$$

The required number of MR vehicles (n_{MR}) to deliver the average throughput T_R is then calculated as follows:

$$n_{MR} = \frac{t_{cyc}}{t_{TS}}$$

The presented fleet sizing approach for MR systems enables simple and effective system dimensioning in the early planning stages. However, as stated by the authors, the approach only refers to single-route MR systems with previously determined and static routes [VDI5586b].

2.3 Control Strategies for MR Systems

One major challenge for traditional MR systems with static routes and schedules are demand fluctuations. Demand fluctuations arise, for example, due to a high number of product variants, changes within the production program or deviations in process times [Lie18].

One strategy for dealing with demand fluctuations is the dynamic control of MR systems [Emd12, Aln15, Hor17, Kle19]. Typical decision problems in in-plant vehicle control include i) the selection and assignment of orders to vehicles (dispatching), ii) the determination of routes between start and end points (routing), and iii) the definition of departure and arrival times (scheduling) [Co91, Kle19].

A transport control system allows for dynamic control of MR vehicles based on actual demand. Therefore, no fixed routes or schedules are required. The control system calculates routes and schedules in real-time in order to optimize MR transports under given constraints [Hor17].

3 Fleet Sizing Approach for Dynamically Routed MR Systems

3.1 Routing Problems

Due to dynamic control, routes and travel times (t_T) in dynamic MR systems vary between each MR cycle. However, average stop times (t_S), loading times (t_L), and unloading times (t_U) can be assumed to be constant and deterministic. In order to determine the number of MR vehicles (n_{MR}) in a dynamic MR system, an approach is therefore needed that allows the calculation of average travel times assuming dynamic routes for each MR cycle.

To determine an average travel time *Urru et al.* propose an exact approach that determines all possible combinations of routes between the depot and the customers of the system under consideration. However, due to the complexity of the decision problem, the exact approach is only applicable to a very limited number of POUs [Urr18].

The underlying decision problem corresponds to the well-known Vehicle Routing Problem (VRP) [Dan59].The VRP aims to minimize transport costs associated with transporting goods between a depot and several customers using a fleet of vehicles with limited capacity [Cla64]. Since the complexity of the VRP is to be regarded as NP-hard, i.e. cannot be solved optimally for arbitrary instances in polynomial time, heuristics or metaheuristics in particular are used in addition to exact procedures in order to find solutions for the problem instance [Kil13].

Genetic algorithms (GA) are a form of metaheuristics based on biologically inspired methods that are used in optimization problems [Kra17]. GA are none other than the equivalent of the biological evolution in algorithmic form.

DOI: 10.2195/lj_Proc_hormes_en_202012_01
URN: urn:nbn:de:0009-14-51420

GA have been successful at finding decent solutions to complex routing problems [Faz15]. In the following we present a genetic algorithm that allows the determination of dynamic routes and average travel times ($t_{T,d}$) assuming dynamic routes. Based on the average travel times ($t_{T,d}$), the number of MR vehicles can be calculated following the approach for static routes according to VDI 5586.

3.2 Development of the Genetic Algorithm

3.2.1 Introduction

GA are composed of a set of candidate solutions which individually provide a solution to our optimization problem. The set of candidate solutions is called "population." In our case, each solution represents a vector containing a set of stops and tours. At the beginning, an initial population is randomly generated, which allows to cover a large solution space at an early stage of the optimization (Figure 1).

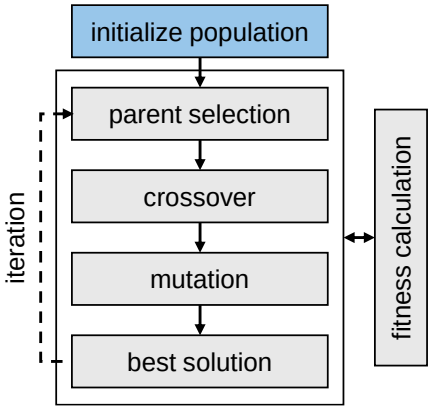

Figure 1. General flowchart for the genetic algorithm

Similar to nature, GA mix the genetic material of two selected parent solutions to create child solutions (crossover). The one-point crossover is the most popular crossover approach [Kra17]. It involves splitting up two parent solutions at an arbitrary point and reassembling them to get two child solutions (Figure 2). Another nature-inspired approach is mutation. It makes selective changes within the child solutions. A possible mutation is illustrated in Figure 2. The mutation operator randomly swaps two indices within the child solutions. The aim of mutation is to search a larger solution space and to escape local optima.

The example in Figure 2 includes five parts (1, 2, 3, 4, 5). The required throughput of part 1 is two units. The throughput of parts 2, 3, 4, and 5 is one unit. Therefore, the total throughput T_T adds up to six units. The delivery vector d_s for parent solution s contains the delivery order of all units (d_1 = [3, 2, 1, 4, 5, 1], d_2 = [2, 3, 5, 4, 1, 1]). The crossover point is at index two of the delivery vector. Child solutions are created by combining parent solutions 1 and 2. The tour vector $t_{s,t}$ is generated after mutation and contains the units delivered on tour t for child solution s ($t_{1,1}$ = [4, 2, 5], $t_{1,2}$ = [3, 1, 1], $t_{2,1}$ = [2, 3, 1], $t_{2,2}$ = [1, 5, 4]). The number of parts to be delivered during a tour depends on the capacity of the MR vehicle. In our example, the maximum capacity C_{max} is three units.

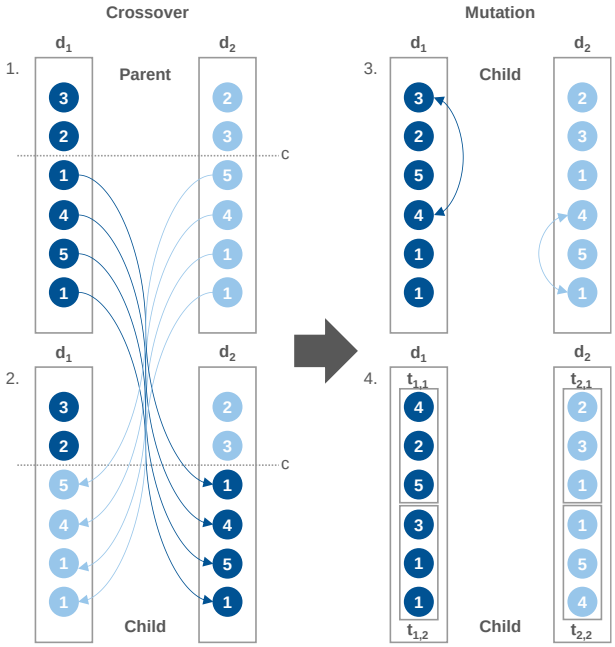

Figure 2. Crossover and mutation

After creating a new population of solutions using crossover and mutation, it is necessary to evaluate the solutions. This is done with the help of a fitness function. Fitness functions assess the quality of the solutions according to the optimization problems. There are several ways to create a fitness function. Depending on the problem, it can impose costs or penalties for solutions that are not feasible and do not meet the constraints and requirements. In the case of multiple-objective optimization, where the solution is optimized to meet two or more objectives, the fitness function is the weighted sum of the individual fitness functions of the optimization. The fitness-based evaluation can be used in several steps during the algorithm, e.g. for the selection of parent solutions or the selection of child solutions (Figure 1).

Finally, a criterion for stopping the algorithm must be determined. One approach is to stop after a certain number of iterations (generations). Another approach is to stop when there are no more significant improvements in the fitness function. These types of approaches, however, carry the risk of being terminated at a local optimum. Restart strategies are therefore required.

3.2.2 Fitness Function and Constraints

The fitness function allows to evaluate the generated solutions of a population P_P. The number of solutions of the population n_P is set at the beginning. A higher n_P allows a higher diversity of solutions. However, this also has a negative impact on the computation time. Each solution s is

DOI: 10.2195/lj_Proc_hormes_en_202012_01
URN: urn:nbn:de:0009-14-51420

described by the solution vector $P_s \in P_P$. Each solution vector P_s contains the delivery vector d_s, the tour vector t_s and the fitness value F_s. The fitness value can be calculated according to the following equation:

$$max\, F = \left((1-\alpha) \sum_{t}^{n_T} \sum_{i}^{n_N} \sum_{j}^{n_N} x_{tij} d_{ij} + \alpha \sum_{t}^{n_T} \sum_{k}^{n_K} y_{tk} c_{tk} \bar{d} \right)^{-1}$$

$$x_{tij} = \begin{cases} 1 & \text{, if tour t contains arc (i, j)} \\ 0 & \text{, otherwise} \end{cases}$$

$$y_{tk} = \begin{cases} 1 & \text{, } c_{tk} > 0 \\ 0 & \text{, otherwise} \end{cases}$$

$$c_{tk} = \left(\sum_{m}^{n_M} z_{tkm} \right) - \left\lceil \frac{T_{k,T}}{n_T} \right\rceil$$

$$z_{tkm} = \begin{cases} 1 & \text{, if part k is delivered to stop m of tour t} \\ 0 & \text{, otherwise} \end{cases}$$

As stated in 3.2.1, the algorithm aims at minimizing the distance travelled and the travel time by the MR vehicles to serve all stops. Therefore, the first part of the fitness function covers the calculation of the total distance travelled of the corresponding solution. It adds up all distances d_{ij} between all nodes N that are part of a tour t for the total number of nodes (n_N) and tours (n_T) of the solution. The formulation of the first part of the equation is based on existing VRP formulations [Vai99]. The higher the total distance travelled, the lower the fitness value F_s of solution s.

The second part of the equation serves as a penalty function and ensures that the deliveries of a number of units of part k are spread over multiple tours (split delivery routing). The function ensures a continuous supply of parts and low line-side inventories. The variable z_{tkm} counts the number of deliveries of part k on tour t for all parts K, stops M and tours T. Under the assumption of continuous supply, the optimal number of deliveries of part k for the set of tours T can be calculated by dividing the total throughput of part k ($T_{k,T}$) by the number of tours n_T. The penalty costs c_{tk} only occur if the actual number of deliveries of part k on tour t exceeds the optimal number of deliveries of part k per tour ($c_{tk} > 0$). In this case, y_{tk} is 1; in all other cases, y_{tk} is 0 (no penalty costs occur).

Following our example from 3.2.1, the actual number of deliveries per tour of part 1 of child solution 2 is one unit. The optimal number of deliveries of part 1 per tour is also one unit ($T_{1,T} = 2$ divided by $n_T = 2$). As both units of part 1 are delivered on two different tours, no penalty costs occur (c_{t1}, $y_{t1} = 0$). The same applies to parts 2, 3, 4, and 5 of child solution 2. However, for child solution 1 (after mutation) both units of part 1 are delivered on tour 2. Therefore, in this solution, a penalty for tour 2 and part 1 occurs ($c_{21} = 1$). The penalty costs are weighted by the average distance $\bar{d}$ between two nodes of the MR system. The penalty is added to the total distance travelled and reduces the total fitness value of the solution. The factor α is used to weight the influence of both parts of the fitness function.

The generation of solutions must comply with the following set of constraints:

- All tours t ∈ T start and end at a single depot
- The number of units delivered of part k ∈ K must be equal to the required throughput per part $T_{k,T}$
- The total number of units delivered must be equal to the total required throughput T_T
- The deliveries of part k ∈ K to stop m ∈ M can be split between several tours
- The number of deliveries per tour is limited by the maximum capacity C_{max} of the MR vehicle

3.2.3 Modeling of the Genetic Algorithm

In the first step, the algorithm (Figure 3) uses the data specified by the user to create n_P random solutions for the initial population P. As stated in 3.2.2, the generation of the first population of solutions must meet the predefined set of constraints. Once the first generation of solutions is generated, the quality of these solutions is assessed. The fitness function evaluates the generated solutions according to the total distance travelled and the distribution of the deliveries of the same type of parts over the different tours. The shorter the distance driven and the more spread and varied the part deliveries are, the better the fitness value will be.

After the generation of the initial population P_P, the optimization process starts (Figure 3). In each iteration, several parent solutions P_{parent} are selected from the previous generation and build the basis for the next generation. The number of selected solutions depends on the parent quotient p, which determines the number of parent solutions n_p from the number of total solutions in the population n_P. Which solutions are selected as parent solutions, depends on their fitness value F_s compared to the total fitness of the population F_P, which determines the parent selection probability p_p. Therefore, solutions with a higher fitness have a higher probability of being selected as parents.

In the next step, the crossover operator is used to form child solutions. First of all, two parent solutions are selected from the set of parents P_{parent}. The selection is based on p_p, which means that solutions with a higher fitness are selected more often to form a child solution. The parent solutions are combined so that the child solutions are still valid and respect all constraints. Therefore, a modified order crossover operator is used. The algorithm is displayed in Figure 4.

DOI: 10.2195/lj_Proc_hormes_en_202012_01
URN: urn:nbn:de:0009-14-51420

Input	n_P, n_i, p
Output	P_{fit}
1	generate initial population $P_P := \{P_1, \ldots, P_{n_P}\}$
2	**while** iteration count $i \leq n_i$
3	// select parent solutions $P_{parent} \subseteq P_P$
4	$n_p \leftarrow p * n_P$
5	$p_{p,j} \leftarrow F_j / F_P, \forall j \in \{1, n_P\}$
6	$P_{parent,k} \leftarrow P_j (\max\{p_p\}), \forall k \in \{1, n_p\}$
7	generate child solutions $P_{child} \leftarrow$ **crossover** (P_{parent})
8	mutate child solutions $P_{child} \leftarrow$ **mutation** (P_{child})
9	add P_{parent}, P_{child} to new population P_P
10	choose fittest solution $P_{fit} \leftarrow P_j (\max\{F_j\})$
11	$i \leftarrow i + 1$
12	**endwhile**

Figure 3. General procedure

Input	n_c, n_P, n_p, p_p, p_m, P_{parent}, T_T
Output	P_{child}
1	// crossover (P_{parent})
2	**while** $n_c < n_P - n_p$
3	select $P_{parent,1}$, $P_{parent,2} \subseteq P_{parent}$ based on p_p
4	generate random crossover index $c \in \{1, T_T\}$
5	// add first part of $d_{parent,1}$ to $d_{child,1}$
6	**for** $j \in 1$ to c
7	$d_{child,1,j} \leftarrow d_{parent,1,j}$
8	**endfor**
9	$c \leftarrow c + 1$
10	// add first part of $d_{parent,2}$ to $d_{child,1}$
11	**for** $j \in 1$ to T_T
12	**if** $T_j < T_{j,T}$
13	$d_{child,1,c} \leftarrow d_{parent,2,j}$
14	$c \leftarrow c + 1$
15	**endif**
16	**endfor**
17	**repeat** for $d_{child,2}$ (start with $d_{parent,2}$)
18	// mutation ($d_{child,1}$, $d_{child,2}$)
19	generate random number $r \in \{0, 1\}$
20	**if** $r < p_m \in [0, 1]$
21	generate random mutation index $m_{1,2} \in \{1, T_T\}$
22	swap $d_{child,1,m_1}$ and $d_{child,1,m_2}$
23	**endif**
24	**repeat** for $d_{child,2}$
25	generate tour vector $t_{child,i}$
26	calculate fitness $F_{child,i}$
27	// select child with higher fitness
28	$P_{child,fit} \leftarrow P_{child,i} (\max\{F_{child,i}\})$
29	// add fittest child $P_{child,fit}$ to P_{child}
30	$P_{child} \leftarrow P_{child,fit}$
31	$n_c \leftarrow n_c + 1$
32	**endwhile**

Figure 4. Crossover and mutation operators

After selecting two parents, a random crossover index c is generated. The crossover index refers to the delivery vectors $d_{parent,1}$ and $d_{parent,2}$ of the selected parent solutions, which contain all deliveries in a specific order. Up to this index, the first part of the delivery vector $d_{child,1}$ of the first child solution is built from the first parent and the rest from the second parent. The same process is repeated for the delivery vector $d_{child,2}$ of the second child, with the only difference being that the first part is given by the second parent and the second part by the first parent (Figure 2).

To diversify the solution space and to avoid getting stuck in local optima, a mutation algorithm is used (Figure 4). The mutation function decides with a defined probability (p_m) whether a child solution is modified or not. If a child solution is selected for modification, the operator swaps two deliveries of the delivery vector. The operator is executed for both delivery vectors of the two child solutions. After mutation of the delivery vectors $d_{child,1}$ and $d_{child,2}$, the tour vectors $t_{child,1}$ and $t_{child,2}$ are generated based on the maximum capacity C_{max} of the MR vehicles. Finally, the operator calculates the fitness values for both child solutions and selects the child solution with the highest fitness $P_{child,fit}$. The solution is added to the child population P_{child}.

The crossover and mutation operators are repeated until the number of parent (n_p) and child solutions (n_c) equals the total number of solutions of a population n_P. When the new population is complete, the best solution of the population P_{fit} is determined based on the highest fitness value. This entire process is repeated for i iterations (generations). After n_i iterations, the algorithm returns the best solution found P_{fit}, which contains the best delivery and tour vector found in terms of the total distance travelled and the distribution of parts on different tours. It should be noted that P_{fit} is not necessarily the optimal solution of the optimization problem.

4 Case Study

4.1 Description of the MR System

The case study was conducted in the final assembly of a commercial vehicle manufacturer. The investigated MR system consists of three predefined routes serving a total of 17 POUs with 17 different parts. One manually-operated MR vehicle is assigned to each route. The maximum capacity C_{max} of each MR is three units. The average travelling speed is 2 m/s. The routes are illustrated in Figure 6. Table 1 displays the critical fleet sizing parameters for the MR system with static routes. For the calculation of the cycle time, the following times for t_S, t_L and t_U are used:

- stop time $t_S = 40$ s
- loading time (depot) $t_L = 60$ s
- unloading time (stop) $t_U = 300$ s

DOI: 10.2195/lj_Proc_hormes_en_202012_01
URN: urn:nbn:de:0009-14-51420

Table 0. Parameters of the MR system with static routes

K	R	$T_{k,T}$ [1/h]	n_T [1/h]	L_R [m]	D_R [m]	t_T [s]	t_{cyc} [s]	t_{TS} [s]	n_{MR}
1	1	1.0							
2	1	1.0							
3	1	3.0	3.0	636	1908	318	718	1200	0.60
4	1	3.0							
5	1	1.0							
Total		9.0							1.0
6	2	1.0							
7	2	3.0							
8	2	3.0							
9	2	3.0	5.0	399	1995	200	600	720	0.83
10	2	2.0							
11	2	2.0							
12	2	1.0							
Total		15.0							1.0
13	3	1.0							
14	3	4.0							
15	3	3.0	4.0	719	2876	360	760	900	0.84
16	3	3.0							
17	3	1.0							
Total		12.0							1.0
Total		**36.0**	**12.0**		**6779**				**3.0**

As displayed in Table 1, the throughput T_R, the number of tours per hour n_T and the route length L_R differ between the different routes. Route 3 is the longest route leading to the longest travel time t_T per tour and the longest cycle time t_{cyc}. As route 3 has to deliver 12 parts per hour and the capacity is limited to three parts per tour ($C_{max} = 3$), the number of tours per hour n_T is 4. Therefore, the required tour start interval t_{TS} is 15 minutes (= 900 s) and the total distance travelled D_R is 2876 meters. However, the total cycle time of route 3 is 760 s leading to an exact required number of 0.84 MR vehicles (n_{MR}) for route 3. As the assignment of vehicles to routes is fixed, the MR system with static routes requires a total number of three MR vehicles. The average utilization rate η_T of the MR system is 0.76.

4.2 Fleet Sizing for Dynamic Routes

Table 2 displays the results generated by the developed GA. For the execution of the algorithm, we used the following parameters:

- population size $n_P = 1000$
- number of iterations $n_i = 100$
- parent quotient $p = 0.25$
- mutation probability $p_m = 0.7$
- fitness weight $\alpha = 0.85$

Table 1. Parameters of the MR system with dynamic routes

R	N	L_R [m]	t_T [s]	t_{cyc} [s]
1	13, 4, 14	476.0	238	638
2	11, 9, 8	350.0	175	575
3	1, 15, 14	345.2	173	573
4	14, 16, 3	371.6	186	586
5	9, 8, 10	350.0	175	575
6	2, 4, 5	336.6	169	569
7	14, 15, 4	485.8	243	643
8	10, 8, 7	306.8	154	554
9	6, 7, 12	280.4	141	541
10	7, 11, 9	350.0	175	575
11	15, 16, 3	371.6	186	586
12	3, 16, 17	481.0	241	541
Average		**375.4**	**187.7**	**587.7**

The results include a set of 12 routes with an average route length L_R of 375.4 meters. The average travel time $t_{T,d}$ for dynamic routes is 187.7 seconds, resulting in an average cycle time t_{cyc} of 587.7 seconds based on the constant stop, loading and unloading times. Table 3 summarizes the fleet size calculation for the MR system with dynamic routes. The exact number of required MR vehicles n_{MR} is 1.96.

Table 2. Calculation of the fleet size for dynamic routes

T_T [1/h]	n_T [1/h]	L_R [m]	D [m]	t_T [s]	t_{cyc} [s]	t_{TS} [s]	n_{MR}
36.0	12.0	375.4	4505	188	588	300	1.96
Total							**2.0**

4.3 Discussion of Results

Figure 5 displays the optimization results of the GA. The algorithm was implemented in a Microsoft Visual Basic for Applications (VBA) programming environment to create an application-oriented tool that is easy to use by practitioners in logistics planning. The optimization was executed on an Intel i7 CPU (2.6 GHz, 16 GB RAM). The required CPU time was 56 seconds. After 50 iterations no significant improvements in distance and fitness were observed. In the case study described, the total distance travelled D per hour is reduced from 6779 to 4505 meters when comparing static to dynamic routes. Given the optimized distance travelled, the fleet size can be reduced from 3 to 2 MR vehicles.

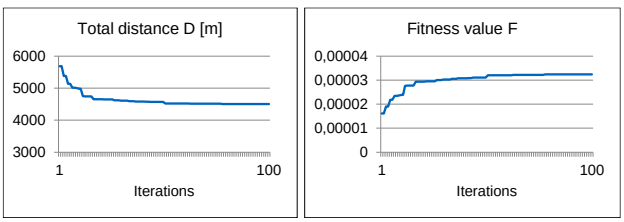

Figure 5. Optimization results of the genetic algorithm

DOI: 10.2195/lj_Proc_hormes_en_202012_01
URN: urn:nbn:de:0009-14-51420

The developed approach enables fleet sizing for MR systems with dynamic routes. In contrast to MR systems with static routes, the route length L_R and the travel time t_T vary between each tour. Therefore, the existing approach of the VDI 5586 standard cannot be applied. The genetic algorithm finds a set of tours that fulfills the constraints and optimizes route lengths and travel times. Due to the heuristic approach, the algorithm solves even larger problems in a reasonable time. The implementation in VBA ensures easy use of the fleet sizing tool. However, an implementation in another programming environment (e.g. C++) could further improve computation times [Erd17].

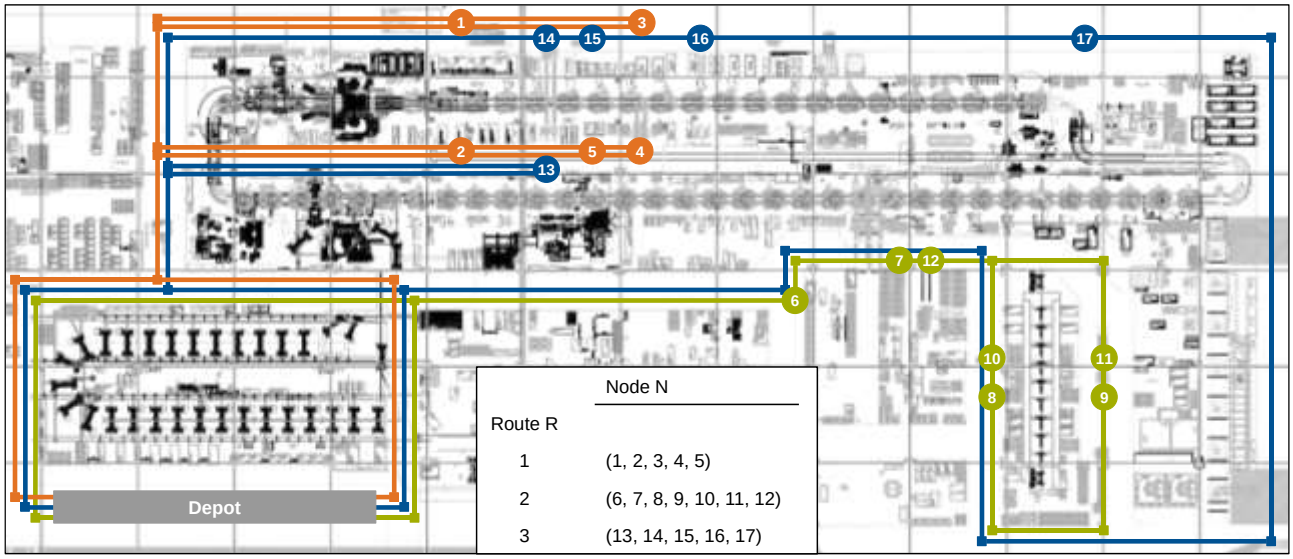

Figure 6. Layout of the MR system

5 Summary and Future Research

This paper presents a metaheuristic-based fleet sizing approach for MR systems with dynamic routes. The developed GA optimizes the total distance travelled under given constraints and determines the average travel time $t_{T,d}$ per MR cycle for dynamic routes. Based on $t_{T,d}$, the average cycle time and fleet size can be determined according to existing dimensioning approaches for MR systems.

In conclusion, the approach enables a dimensioning of MR systems with dynamic routes in early planning stages and supports practitioners in the decision-making process whether to implement an MR system with dynamic routes. As the case study of the commercial vehicle manufacturer shows, an MR system with dynamic routes is more flexible and can offer efficiency advantages by reducing unnecessary travel times and the number of vehicles. However, the implementation of an MR system with dynamic routes is usually more complex and requires additional information and communication technology and availability of data.

Future work will focus on the inclusion of further constraints in the algorithm, such as variable capacity consumptions for different parts and throughput variations, to cover even more scenarios. Furthermore, the approach will be applied to various industry use cases to further evaluate the concept of dynamic routes in MR systems.

Literature

[Tak06] Takeda, H. (2006). *The synchronized production system – Going beyond just-in-time through kaizen.* Kogan Page, London.

[Har03] Harris, R., Harris, C., & Wilson, E. (2003). *Making materials flow – A lean material-handling guide for operations, production-control, and engineering professionals.* Lean Enterprise Institute, Inc., Cambridge, MA.

[Bru12] Brungs, F. (2012). *Der Milkrun in der Produktionslogistik.* Shaker, Aachen.

[VDI5586a] The Association of German Engineers (VDI) (2016). *VDI-Guideline 5586 Part 1: In-plant milk-run systems – Basics, design and practical examples.* Beuth, Berlin.

[Klu13] Klug, F. (2013). The internal bullwhip effect in car manufacturing. International Journal of Production Research, 51(1), 303-322.

[Lie18] Lieb, C., Hormes, F., Günthner, W. A., & Fottner, J. (2018). Modelling and

analysis of the demand volatility in in-plant milkruns serving scheduled mixed-model production facilities. Logistics Journal: Proceedings, 247-262.

[Kle15] Klenk, E., Galka, S., & Günthner, W. A. (2015). Operating strategies for in-plant milk-run systems. IFAC-PapersOnLine, 48(3), 1882-1887.

[VDI5586b] The Association of German Engineers (VDI) (2016). *VDI-Guideline 5586 Part 2: In-plant milk-run systems – Planning and dimensioning.* Beuth, Berlin.

[Keu18] Keuntje, C., Hormes, F., & Fottner, J. (2018). Considering technical details in the planning of tugger train systems. In: Proceedings of the 2nd International Conference on High Performance Compilation, Computing and Communications, ACM, 97-105.

[Emd12] Emde, S., & Boysen, N. (2012). Optimally routing and scheduling tow trains for JIT-supply of mixed-model assembly lines. European Journal of Operational Research, 217(2), 287-299.

[Aln15] Alnahhal, M., & Noche, B. (2015). Dynamic material flow control in mixed model assembly lines. Computers & Industrial Engineering, 85, 110-119.

[Hor17] Hormes, F., Lieb, C., Fottner, J., & Günthner, W. A. (2017). Steuerung von Routenzugsystemen. ZWF Zeitschrift für wirtschaftlichen Fabrikbetrieb, 112(11), 778-782.

[Kle19] Klenk, E., & Galka, S. (2019). Analysis of real-time tour building and scheduling strategies for in-plant milk-run systems with volatile transportation demand. IFAC-PapersOnLine, 52(13), 2110-2115.

[Co91] Co, C. G., & Tanchoco, J. M. A. (1991). A review of research on AGVS vehicle management. Engineering Costs and Production Economics, 21(1), 35-42.

[Urr18] Urru, A., Bonini, M., & Echelmeyer, W. (2018). Fleet-sizing of multi-load autonomous robots for material supply. In: IEEE International Conference on Service Operations and Logistics, and Informatics (SOLI), 244-249.

[Dan59] Dantzig, G.B., & Ramser, J.H. (1959). The Truck Dispatching Problem. Management Science, 6, 80-91.

[Cla64] Clarke, G., & Wright, J. W. (1964). Scheduling of vehicles from a central depot to a number of delivery points. Operations research, 12(4), 568-581.

[Kil13] Kilic, H. S., & Durmusoglu, M. B. (2013). A mathematical model and a heuristic approach for periodic material delivery in lean production environment. The International Journal of Advanced Manufacturing Technology, 69(5-8), 977-992.

[Kra17] Kramer, O. (2017). *Genetic algorithm essentials.* Springer, Cham, Switzerland.

[Faz15] Fazlollahtabar, H., & Saidi-Mehrabad, M. (2015). Methodologies to Optimize Automated Guided Vehicle Scheduling and Routing Problems: A Review Study. Journal of Intelligent & Robotic Systems, 77(3-4), 525–545.

[Vai99] Vaidyanathan, B. S., Matson, J. O., Miller, D. M., & Matson, J. E. (1999). A capacitated vehicle routing problem for just-in-time delivery. IIE Transactions, 31(11), 1083–1092.

[Erd17] Erdoğan, G. (2017). An open source Spreadsheet Solver for Vehicle Routing Problems. Computers & Operations Research, 84, 62–72.

Fabian Hormes, M. Sc., Research assistant at the Chair of Materials Handling Material Flow Logistics at the Technical University of Munich.

Amin Siala, B. Sc., Student assistant at the Chair of Materials Handling Material Flow Logistics at the Technical University of Munich.

Christian Lieb, M. Sc., Research assistant at the Chair of Materials Handling Material Flow Logistics at the Technical University of Munich.

Prof. Dr.-Ing. Johannes Fottner, Professor and head of the Chair of Materials Handling Material Flow Logistics at the Technical University of Munich.

Address: Technische Universität München, Lehrstuhl für Fördertechnik Materialfluss Logistik (fml), Boltzmannstr. 15, 85748 Garching, Germany, Phone: +49 89 289 15931, E-Mail: fabian.hormes@tum.de