[Schö05b] Schönberger, J.; Kopfer, H.: Freight Flow Consolidation in Presence of Time Windows. In: Fleuren, H.; den Hertog, D.; Kort, P. (eds.): Proceedings of Operations Research 2004. Springer, Berlin, Heidelberg, 2005, pp. 184-191

Freight Flow Consolidation in Presence of Time Windows

Jörn Schönberger and Herbert Kopfer

Chair of Logistics, Department of Business Administration and Economics
University of Bremen, Wilhelm-Herbst-Straße 5, 28359 Bremen, Germany
{sberger,kopfer}@logistik.uni-bremen.de

Abstract. This contribution addresses the consideration of time windows in the optimization of multi-commodity network flows. For each node, one interval is specified in which the visitation is allowed. Applications in freight flow consolidation let this problem become interesting. An optimization model is proposed and a construction heuristic is presented. For improving the generated solutions, a genetic algorithm framework including several hill climbing procedures for local optimization, is configured.

1 Introduction

Multi-Commodity Network Flow Problems (MCBFP) are subject of a large number of scientific investigations. They are often consulted if a least cost flow of physical goods through a given transport network is searched. Typically, the costs represent the consumption of resources like time, fuel or budgets.

In logistics, the transportation of goods is only one particular step in the value creating process of a product. This step has to be synchronized with previous and subsequent processing steps. Therefore, time windows are specified for each single transport task in order to ensure a temporal coordination of the processing steps.

This article is about the optimization of time window-constrained flow of goods. In Section 2 the problem is stated in detail. Section 3 describes the used construction heuristic, Section 4 contains the description of a memetic improvement algorithm. The results of several numerical experiments are presented and discussed in Section 5.

2 Multi-Commodity Flow with Time Windows

A logistics service provider (LSP) is responsible for the reliable fulfillment of N pickup and delivery requests. Each request r_i expresses the need for the movement of a commodity i with capacity c_i. It has to be picked up at location p_i within the time window T_i^{pick} and unloaded within the time window $T_i^{delivery}$ at location q_i. Since the LSP does not own any vehicles it pays a forwarding company for the physical execution of the requests. The minimization of the execution costs for the complete portfolio is required.

Literature. MCNFPs are targeted in several contributions. Here, it is referred to the comprehensive article [1]. The most famous special case is the shortest path problem [2].

Applications of MCNFP related to the optimization of the flow of goods in a transport network are described in [3], termed Freight Optimization.

The consideration of time windows has been described only in the context of the single-commodity shortest path problem [4].

Problem Statement. The set $\mathcal{V} := \{p_1, \ldots, p_N, q_1, \ldots, q_N\}$ of all involved locations and the set $\mathcal{A} := \mathcal{V} \times \mathcal{V}$ of arcs between pairs of the involved locations lead to the graph $\mathcal{G} = (\mathcal{V}, \mathcal{A}, \gamma, \tau)$ representing the network available to fulfill the transport demands. The function γ is defined on the set of arcs and assigns the travel distance γ_{ij} for processing from i to j to each arc (i, j) and the time for traversing (i, j) is τ_{ij}.

Each loading activity and each unloading activity is represented by a triple $a(o) := (o, t_o^{start}, t_o^{end})$. At location o, the corresponding loading or unloading activity takes place. It starts at time t_o^{start} and is completed at time $t_o^{end} := t_o^{start} + d_o$ where d_o refers to the dwell time associated with o. The earliest allowed starting time of the operation associated with o is denoted by t_o^{min} and the latest allowed finishing time is named by t_o^{max}.

The way of commodity i through the graph $\mathcal{G}$ originating from $o_1 := p_i$ and terminating in $o_{N_i} := q_i$ is determined in the *origin/destination-path (o/d-path)* $\mathcal{P}_i := (a(o_1), a(o_2), \ldots, a(o_{N_i}))$ *of commodity* i.

Along an o/d-path for commodity i, the operations are scheduled recursively starting from the earliest allowed execution time of the corresponding pickup-operation $a(o_1)$. The following starting times for $i = 1, \ldots, N_i - 1$ are computed by $t_{o_i+1}^{start} = \max\{t_{o_i}^{min}, t_{o_{i+1}}^{start} + \tau_{o_i, o_{i+1}}\}$ and the finishing times are calculated by $t_{o_i}^{end} := t_{o_i}^{start} + d_{o_i}$.

The *o/d-path* $\mathcal{P}_i$ *is feasible for commodity* i if it satisfies the time window conditions for its associated pickup operation $a(p_i)$ and its associated delivery operation $a(q_i)$.

A fee $F^{ij}(c)$ has to be transferred to the cooperating forward company for the movement of a commodity with a given capacity c along the arc (i, j).

Typically, F is degressive with respect to increasing capacity c, so that it is more profitable to move one large commodity with capacity αc along (i, j) than moving α commodities each with capacity c along this arc. Thus the consolidation of several commodities associated with several requests that are shipped along an arc (i, j) starting at time t into a shipment $S^{ij}(t) \subseteq \{r_1, \ldots, r_N\}$ is profitable in certain cases. It leads to a reduced amount to be paid as long as the saved amount of fees dominates the additional feeder and distribution costs.

The overall sum of fees to be paid for realizing the feasible o/d-paths $\mathcal{P}_1, \ldots, \mathcal{P}_N$ is calculated as follows. At first, all shipments $S^{ij}(t)$ occurring in $\mathcal{P}_1, \ldots, \mathcal{P}_N$ are identified. Then, the capacities $C(S^{ij}(t))$ of the shipments $S^{ij}(t)$ are computed by summing up the capacities of the included com-

modities. Next, the freight fees $F^{ij}(C(S^{ij}(t)))$ to be paid for the execution of $S^{ij}(t)$ are calculated. Finally, the overall sum of fees $F(\mathcal{P}_1, \ldots, \mathcal{P}_N)$ to be paid is computed by summing up the freight fees calculated for the particular shipments.

It is aimed to determine a set of o/d-paths $\mathcal{P}_1, \ldots, \mathcal{P}_N$, so that (A1): The o/d-path $\mathcal{P}_i$ of commodity i is feasible, (A2): If two or more paths are consolidated into the shipment $S^{ij}(t)$ at location i then i is associated with the pickup operation of a request r_l and the commodity l is contained in this shipment, (A3): If the shipment $S^{ij}(t)$ is resolved at location j then j is associated with the delivery operation of a request r_k and the associated commodity is contained in $S^{ij}(t)$.

The set $\mathcal{P}_1, \ldots, \mathcal{P}_N$ satisfying (A1)-(A3) is called an *o/d-path-family*. It is aimed to generate a least cost o/d-path-family.

Test Cases. The construction of artificial pickup and delivery transport requests is described in [5]. Following these proposals, instances are generated for all six proposed problem classes with tight or relaxed time windows combined with spatially scattered, semi-clustered or clustered locations.

The fee $f^{ij}(c)$ to be paid for the capacity c along the arc (i, j) is defined as $f^{ij}(c) = \gamma_{ij}$ for $c > 0$ and $f^{ij}(c) = 0$ for $c = 0$.

3 Construction Heuristic

Originally, the used construction procedure has been proposed for the construction of vehicle routes serving time window-constrained customer locations [6]. Three subsequent stages are controlled by a permutation $\sigma = (\sigma_1, \ldots, \sigma_N)$ of the commodities.

Preprocessing. The relevant part of the time axis is partitioned into m equidistant time slots $S_1, \ldots, S_m$. Each operation is sorted into the slot in which its latest allowed execution time falls. For commodity i the expression s_i^p refers to its pickup slot (slot of the pickup operation) and s_i^d to the delivery slot (slot of the delivery location).

An example with six commodities is used to support the presentation of the following construction step. The control permutation $\sigma = (2, 5, 4, 1, 6, 3)$ is applied. The customer specified time windows lead to the following slot assignments: $s_1^p = 2$, $s_1^d = 3$, $s_2^p = 3$, $s_2^d = 4$, $s_3^p = 2$, $s_3^d = 5$, $s_4^p = 2$, $s_4^d = 4$, $s_5^p = 1$, $s_5^d = 6$, $s_6^p = 3$ and $s_6^d = 6$.

Path-Construction. Initially, an exclusive path $\mathcal{P}_i = (a(p_i), a(q_i))$ is set up for each commodity. These paths are modified in the following steps with the goal of concatenating exclusive paths to more complex paths leading to larger shipments.

Two paths $\mathcal{P}_k$ and $\mathcal{P}_l$ are called *incompatible* if and only if at least one of the following two conditions is satisfied: (1) at least one operation o_k in $\mathcal{P}_k$ and one operation $o_l \neq o_k$ in $\mathcal{P}_l$ fall into the same time slot or (2) all

operations in one path fall in time slots larger than the slot of the final operation of the other o/d-path.

In the case of satisfaction of (1), two different operations have to be scheduled approximately at the same time, which is assumed to lead to a time window constraint violation as soon as both are served in the same path. The validity of condition (2) implicates, that one o/d-path cannot be started after the other on is terminated, so that no physical bundling of the associated commodities is possible.

Two paths that are not incompatible are called *compatible*. Only two compatible paths can be merged with the goal to generate larger shipments.

The available paths are checked successively for pairwise compatibility with each other paths. As soon as two compatible paths are detected, they are updated dynamically by absorbing the operations from the other path.

Let $\mathcal{P}_{\sigma_k}$ be the first so far unconsidered path in the order determined by σ. It is checked successively whether $\mathcal{P}_{\sigma_j}$ is compatible with $\mathcal{P}_{\sigma_k}$. If $\mathcal{P}_{\sigma_k}$ and $\mathcal{P}_{\sigma_j}$ are compatible then $\mathcal{P}_{\sigma_k}$ is updated by inserting all operations from $\mathcal{P}_{\sigma_j}$ that fall into slots s with $s^p_{\sigma_k} < s < s^d_{\sigma_k}$ and which are not included so far in $\mathcal{P}_{\sigma_k}$. The other path $\mathcal{P}_{\sigma_l}$ is updated in the same manner.

This update-strategy is demonstrated for the example introduced above. The first path is $\mathcal{P}_2$. This path is compatible with $\mathcal{P}_5$, so that $\mathcal{P}_5$ is updated to $\mathcal{P}_5 := (a(p_5), a(p_2), a(q_2), a(q_5))$ but $\mathcal{P}_2$ remains unchanged since no operations fall in a time slot between $a(p_2)$ and $a(q_2)$. Next it is found, that $\mathcal{P}_4$, $\mathcal{P}_1$ and $\mathcal{P}_6$ are incompatible with $\mathcal{P}_2$. Finally, the compatibility of $\mathcal{P}_3$ with $\mathcal{P}_2$ is detected but again no additional operations can be inserted into $\mathcal{P}_2$. The next path to be combined with other paths is $\mathcal{P}_5$. It is incompatible with $\mathcal{P}_4$, $\mathcal{P}_1$ and $\mathcal{P}_6$ but compatible $\mathcal{P}_3$ so that $\mathcal{P}_5$ is updated to $\mathcal{P}_5 := (a(p_5), a(p_3), a(p_2), a(q_2), a(q_3), a(q_5))$ and $\mathcal{P}_3$ is updated to $\mathcal{P}_3 = (a(p_3), a(p_2), a(q_2), a(q_3))$. Next, $\mathcal{P}_4$ is compared with $\mathcal{P}_1$ (incompatible), $\mathcal{P}_6$ (compatible, updating $\mathcal{P}_4 := (a(p_4), a(p_6), a(q_4))$ and $\mathcal{P}_6 := (a(p_6), a(q_4), a(q_6))$) and $\mathcal{P}_3$ (incompatible). The path $\mathcal{P}_1$ is incompatible with $\mathcal{P}_6$ and, finally, $\mathcal{P}_3$ and $\mathcal{P}_6$ are incompatible.

Ensuring time window feasibility. The generated o/d-paths are successively checked for time window constraint violations. In doing so, two cases are distinguished.

If path $\mathcal{P}_{\sigma_k}$ has been checked and if no violations have been detected, then the path $\mathcal{P}_{\sigma_k}$ is confirmed. This means, the calculated arrival and departure times at the associated locations visited in $\mathcal{P}_{\sigma_k}$ are propagated into the so far unconsidered paths $\mathcal{P}_{\sigma_{k+1}}, \ldots, \mathcal{P}_{\sigma_N}$ and can only be modified as long as they fall into the associated time windows.

If a time window constraint violation is detected in path $\mathcal{P}_{\sigma_k}$ at a certain location, then the associated pickup operation $a(o_1)$ and the corresponding delivery operation $a(o_2)$ of this commodity are deleted in the paths $\mathcal{P}_{\sigma_k}, \ldots, \mathcal{P}_{\sigma_N}$ and the o/d-path of this commodity is determined as the exclusive o/d-path $(a(o_1), a(o_2))$. This o/d-path is confirmed immediately.

4 Memetic Algorithm Path-Improvement

The memetic algorithm (MA) framework introduced in [7] is adapted for evolving o/d-path-families towards the least costs-goal. An initial population consisting of K different o/d-path-families is generated by calling the construction heuristic, described in Section 3, K times. After each call, the control permutation σ is re-determined randomly. Each existing population is replaced according to the $\mu + \lambda$-replacement strategy [8] by recombining new sequences of intermediate stops between the pickup and delivery location of each commodity. Mutation inserts a so far unconsidered operation into randomly selected offspring paths.

The offspring o/d-path-families generated by mutation and crossover do not comply with the conditions (A2) and (A3) in every case. Generally, it exists a commodity i, so that the generated offspring path $\mathcal{P}_i$ contains a pickup operation (delivery operation) of another commodity k and there is no shipment in with both i and k are carried away from (brought to) the associated location.

As a remedy, the o/d-paths in an o/d-path-family are modified and detected deficiencies of the kind mentioned just above are corrected in a straightforward way. Therefore, the o/d-paths in an offspring are checked successively according to the order σ. After potential violations of (A2) or (A3) are corrected, the path is confirmed and cannot be modified anymore.

Let $\mathcal{P}_k = (a(o_1^k), \ldots, a(o_{N_k}^k))$ be an o/d-path. For two included operations $a(o_i^k)$ and $a(o_j^k)$ with $(i \leq j)$ the $a(o_i^k)$-$a(o_j^k)$-*subpath* of $\mathcal{P}_k$ is defined as $(a(o_i^k), a(o_{i+1}^k), \ldots, a(o_{j-1}^k), a(o_j^k))$. An o/d-path $\mathcal{S}$ is called a subpath of another o/d-path $\mathcal{P}$ if there exists operations $a(o_1)$ and $a(o_2)$ in $\mathcal{P}$ so that $\mathcal{S}$ equals the $a(o_1)$-$a(o_2)$-subpath of $\mathcal{P}$.

Let $a(o_j^k)$ be an operation in $\mathcal{P}_k$. The $a(o_j^k)$-*tail of* $\mathcal{P}_k$ is defined as the subpath $(a(o_j^k), \ldots, a(o_{N_k}^k))$. Additionally, the $a(o_j^k)$-*beginning of* $\mathcal{P}_k$ is given by $(a(o_1^k), \ldots, a(o_j^k))$. Two paths $\mathcal{P}_j$ and P_k are called *consistent* if and only if at least one of the following conditions is satisfied: (i) $\mathcal{P}_j$ and $\mathcal{P}_k$ do not have any operations in common, (ii) $\mathcal{P}_j$ is a subpath of $\mathcal{P}_k$ or vice-versa, (iii) the $a(o_1^j)$-tail of $\mathcal{P}_k$ equals the $a(o_{N_k}^k)$-beginning of $\mathcal{P}_j$ or (iv) the $a(o_1^k)$-tail of $\mathcal{P}_j$ equals the $a(o_{N_j}^j)$-beginning of $\mathcal{P}_k$

In all other cases $\mathcal{P}_j$ and P_k are called *inconsistent*. An o/d-path-family in which all included o/d-paths are pairwise consistent satisfy the conditions (A2) and (A3).

The following scheme is used to check the pairwise consistency, to correct inconsistencies and to propagate the decisions into so far unconfirmed o/d-paths. Initially, all o/d-paths in the considered o/d-path-family are unconfirmed. Now let $\mathcal{P}_k$ be the first so far unconfirmed o/d-path according to the order induced by σ.

First phase (consistency check): It is checked whether $\mathcal{P}_k$ is pairwise consistent with all so far confirmed o/d-paths. If $\mathcal{P}_k$ and a confirmed o/d-path $\mathcal{P}_j$

are inconsistent then all error-causing operations are successively removed from $\mathcal{P}_k$ together with the associated pickup or delivery operation (if included). Finally, the modified $\mathcal{P}_k$ is confirmed. For all commodities i associated with a removed operation, the o/d-path $\mathcal{P}_i := (a(p_i), a(q_i))$ is generated and confirmed.

Second phase (subpath propagation): All subpaths of $\mathcal{P}_k$ are propagated into the paths of the commodities associated with so far unconfirmed o/d-paths. Consider successively all operations $a(o)$ in $\mathcal{P}_k$. Let $c(a(o))$ be the commodity related to operation $a(o)$. If $\mathcal{P}_{c(a(o))}$ is unconfirmed so far then one of the following three options is executed.

If $\mathcal{P}_k$ contains both operations belonging to $c(a(o))$ then $\mathcal{P}_{c(a(o))}$ is set to the $a(p_{c(a(o))})$-$a(q_{c(a(o))})$-subpath of $\mathcal{P}_k$. If $\mathcal{P}_k$ contains only the pickup-operation $a(p_{c(a(o))})$ then the associated delivery operation $a(q_{c(a(o))})$ is appended to the $a(p_{c(a(o))})$-tail of $\mathcal{P}_k$ and this extended tail becomes the new o/d-path for commodity $c(a(o))$. If $\mathcal{P}_k$ contains only the delivery-operation $a(q_{c(a(o))})$ then the associated pickup operation $a(p_{c(a(o))})$ is prefixed to the $a(q_{c(a(o))})$-beginning of $\mathcal{P}_k$ and this extended beginning becomes the new o/d-path for commodity $c(a(o))$.

The mode of operation of this procedure is demonstrated in the following example. Parameterized with the sequence $\sigma = (2, 5, 4, 1, 6, 3)$, the procedure modifies the erroneous o/d-path-family $\mathcal{P}_1 = (a(p_1),\ a(q_1),\ \mathcal{P}_2 = (a(p_2),\ a(q_2)),\ \mathcal{P}_3 = (a(p_3),\ a(p_2),\ a(q_2),\ a(q_3)),\ \mathcal{P}_4 = (a(p_4),\ a(p_6),\ a(q_4)),\ \mathcal{P}_5 = (a(p_5),\ a(p_2),\ a(p_3),\ a(q_2),\ a(q_4),\ a(q_3),\ a(p_1),\ a(q_5))$ and $\mathcal{P}_6 = (a(p_6),\ a(q_4),\ a(p_3),\ a(q_6))$ into the o/d-path-family $\mathcal{P}_1 = (a(p_1),\ a(q_5),\ a(q_1),\ \mathcal{P}_2 = (a(p_2),\ a(q_2)),\ \mathcal{P}_3 = (a(p_3),\ a(q_3)),\ \mathcal{P}_4 = (a(p_4),\ a(p_5),\ a(p_2),\ a(q_2),\ a(q_4)),\ \mathcal{P}_5 = (a(p_5),\ a(p_2),\ a(q_2),\ a(q_4),\ a(p_1),\ a(q_5))$ and $\mathcal{P}_6 = (a(p_6),\ a(q_4),\ a(q_6))$ which is free of errors (see also Fig. 1).

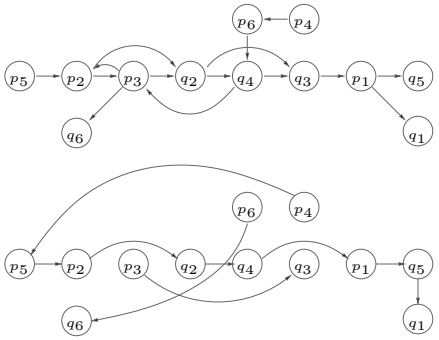

Fig. 1. Hill climbing repair of (A2) and (A3) constraint violations. In the figure above, q_2 has two predecessors and p_2 has two successors. The hill climbing procedure is applied to this family, which is shown in the lower figure. It is free of constraint violations.

5 Computational Experiments

The proposed MA has been implemented and assessed for its suitability and is applied in five independent runs to the instances of the six problem classes.
Algorithm Configuration. A population consisting of K=200 individuals is evolved through 150 iterations so that overall 30000 o/d-path-families are tested. The crossover frequency is set to 100% and the mutation frequency is 90%. These intensities are very high but necessary, since the application of the hill climbers refuses a large part of the proposed modifications. The number of time slots is set to $m = 5N$.
Experiments. For each of the six problem classes three indicators are derived from the observed results. At first, the percentages ϵ of exclusively fulfilled requests, which are not contained in any bundle with at least one other request commodity, are calculated. Secondly, the improvement over the evolution stages is subsumed. Therefore, the improvement ι relative to the best observed fitness in the initial population is calculated. At last, the average deviation δ from the costs of the solution in which all requests are served by own vehicles (the VRPTW-case) is calculated in order to answer the question for which type of requests the freight consolidation-algorithm is best suited.
Presentation and Discussion of Results. The achieved values are presented in Tab. 1. It is observed that tight time windows (R1, RC1, C1) prevent the consolidation of a large fraction of requests. In almost half of the requests, the associated commodities cannot be consolidated with others. In case of relaxed time windows this quote ϵ varies between 12.2% and 26%.

The memetic search is able to find significantly better o/d-path-families compared to the initially generated families.

	R1	RC1	C1	R2	RC2	C2
ϵ	54.7%	58.5%	48.1%	12.2%	26.0%	26.0%
ι	30.4%	30.6%	62.0%	47.5%	46.6%	54.6%
δ	32.7%	39.6%	51.0%	-30.7%	4.8%	-60.6%

Table 1. Computational results achieved for the artificial test instances

The value of ι varies between 30.4% and 62.0% (with means the initially observed costs can be reduced by 30.4% up to 62.0%). Finally, it can be stated, that the overall algorithm performance is sufficient for the test cases with tight instances. Here significant improvements between 32.7% and 51.0% are achieved compared to the VRPTW-case. However, the performance for the problems with relaxed time windows is disappointing. In the R2 and in the C2-instances, the VRPTW-values are not reached. Since the observed improvement is sufficient (with respect to the ι-values), the families produced by the construction heuristic are of insufficient quality. It might be that the

time window oriented consolidation is not effective enough. Additionally, the scope of the problem should be extended and for each arcs, several different fee functions (representing alternative LSPs) should be provided.

6 Conclusions and Outlook

The extension of MCNFP by time windows for visiting nodes in the underlying network has been addressed in this article. An adequate optimization model has been proposed. Several specialized algorithms are composed to an MA that has proven its general applicability to solve artificial test instances. However, the proposed construction heuristic seems to be well suited only for instances with tight time windows, so that other path construction approaches should be assessed.

References

1. McBride R.D. (1998) Advances in Solving the Multicommodity-Flow Problem. Interfaces **28 (2)**, 32–41
2. Williams H.P. (1999) Model building in mathematical programming, 4th edition, Wiley
3. Kopfer H. (1992) Konzepte genetischer Algorithmen und ihre Anwendung auf das Frachtoptimierungsproblem im gewerblichen Güterfernverkehr. OR Spektrum **14**, 137–147
4. Desrochers M., Soumis F. (2001) A reoptimization algorithm for the shortest path problem with time windows. European Journal of Operational Research **35**, 242–254
5. Nanry W.B., Barnes J.W. (2000) Solving the pickup and delivery problem with time windows using reactive tabu search. Transportation Research Part B **34**, 107–121
6. Schönberger J., Kopfer H., Mattfeld, D.C. (2002) A Combined Approach to Solve the Pickup And Delivery Selection Problem in Leopold-Wildburger U., Rendl F., Wäscher G. (eds.) Operations Research Proceedings 2002
7. Schönberger J. (2004) Operational Freight Carrier Planning - Investigations on Basic Concepts, Optimization Models and Advanced Memetic Algorithms. PhD-Thesis, so far unpublished
8. Bäck T., Fogel D.B., Michalewicz Z. (2000) Evolutionary Computation 1 - Basic Algorithms and Operators. IoP Publishing