Central European Journal of Operations Research manuscript No.
(will be inserted by the editor)

Combinatorially Simple Pickup and Delivery Paths

Marco E. Lübbecke

Technische Universität Berlin
Institut für Mathematik, Sekr. MA 6-1
Straße des 17. Juni 136, D-10623 Berlin, Germany
e-mail: `M.Luebbecke@math.tu-berlin.de`

September 9, 2003

Abstract Pickup and delivery problems discussed in the literature are often constrained to particularly simple solutions in terms of the sequence of visited locations. We study the very simplest pickup and delivery paths which are concatenations of short patterns visiting one or two requests. This restricted variant, still $\mathcal{NP}$-hard, is close to the traveling salesman problem with the additional choice of what patterns to visit. We compare the number of restricted and unrestricted paths, and evaluate their respective path lengths. We conclude with two polynomially solvable cases.

Key words Pickup and delivery problem; mini-cluster; traveling salesman problem; counting of paths

MSC (2000) 90B06, 90C27, 05A05

1 Motivation

Generically, when items are to be transported from origins to destinations by a capacitated vehicle we speak of the *pickup and delivery problem* (PDP). In particular, variants with multiple vehicles and time window constraints have received considerable attention in the operations research literature [3, 6, 8–11, 14, 21, 22]. Remarkably, many computational experiments are performed on problem instances which allow only a restricted structure of solutions, that is, the feasible sequences of pickups and deliveries are constrained. There are several reasons for such limitations: In dial-a-ride systems for the transportation for the handicapped and the elderly, temporal constraints strongly restrict the total vehicle load at any time [7]; clustering approaches produce on average mini-clusters of small size [9, 14]; the ratio of the largest vehicle capacity and the smallest weight may be small [10, 11, 22]; in [17–19] simplicity of solutions is explicitly required in the problem definition. The aim of this paper is to demonstrate that well-defined restricted pickup and delivery paths also have theoretical, possibly algorithmically useful benefits.

For a comprehensive review of the relevant approximative and exact solution approaches to the PDP in its variants see [21,22]. A catalog of practical applications is provided in [6]. We consider a single vehicle only without time window constraints. This setting arises as a subproblem for solution approaches to more complex variants.

2 Pickup and Delivery Paths

Given a set $\mathcal{R}$ of n requests we represent the PDP by a directed graph $G=(N,A)$. The node set N comprises all pickup and delivery locations r^+ and r^-, $r\in\mathcal{R}$. For each arc $(i,j)\in A=N\times N$ a weight c_{ij} reflects the cost for traveling from i to j. For each $i\in N$ we are given the size ℓ_i of the load to be picked up ($\ell_i\geq 0$) or delivered ($\ell_i\leq 0$), where $\ell_{r^+}=-\ell_{r^-}$, $r\in\mathcal{R}$. The vehicle capacity is denoted by L.

Definition 1 *Let $R=(i_1,i_2,\dots,i_K)$ with $i_p\in N, p=1,\dots,K$, represent a directed simple path in G. R is called a* pickup and delivery path (PD path) : $\Longleftrightarrow$

(i) Either $\{r^+,r^-\}\subseteq R$ or $\{r^+,r^-\}\cap R=\emptyset$, for all $r\in\mathcal{R}$
(ii) If $i_p=r^+$ and $i_q=r^-$ for an $r\in\mathcal{R}$ then $p<q$
(iii) $\sum_{p=1}^{k}\ell_p\leq L$, for all $k\leq K$

Conditions (i) through (iii) are called *pairing*, *precedence*, and *capacity* constraints, in that order. Note that a PD path must not repeatedly visit the same request.

3 Combinatorially Simple Pickup and Delivery Paths

Define $\mathcal{P}_1=\bigcup_{r\in\mathcal{R}}\{(r^+,r^-)\}$, $\mathcal{P}_2=\bigcup_{r\neq s\in\mathcal{R}}\{(r^+,s^+,s^-,r^-),(r^+,s^+,r^-,s^-)\}$, and $\mathcal{P}=\mathcal{P}_1\cup\mathcal{P}_2$. We use the simple node sequences $P\in\mathcal{P}$, or *patterns*, as building blocks for more complicated paths. Note that $\sum_{p\in P}\ell_p=0$ for all $P\in\mathcal{P}$.

Definition 2 *A PD path $R=(i_1,\dots,i_K)$ is called a $\mathcal{P}$-*concatenation : $\Longleftrightarrow$ *indices $1=p_0\leq\cdots\leq p_k=K$ exist such that*

$$(i_{p_0},\dots,i_{p_1}),(i_{p_1+1},\dots,i_{p_2}),\dots,(i_{p_{k-1}+1},\dots,i_{p_k})\in\mathcal{P}\quad.$$

In simple words, a pattern concatenation is a concatenation of patterns. In a sense, we constrain each request's *context* in admissible pickup and delivery paths. When restricting ourselves to $\mathcal{P}_1$ only, we obtain a traveling salesman problem.

Lemma 1 *Finding (a collection of) shortest $\mathcal{P}$-concatenations which visit all requests is $\mathcal{NP}$-hard in the strong sense.*

Remark 1 The introduction of patterns allows for considering sequence dependent travel cost. That is, we are able to account for the extent of work required at a particular location which depends on the item picked up previously.

Applications Even though $\mathcal{P}$-concatenations heavily constrain the general PDP, we encouter applications which require solutions which are similarly simple in structure to ours. This includes the scheduling of switching locomotives [18,19], ship scheduling [12], and the extraction of logs in forestry [20]. We discuss these applications in more detail in [17] and sketch here only one idealized example.

The *HHLA CTA GmbH*, Hamburg, Germany, plans an almost fully automated container terminal. Container vessels are loaded/unloaded, and containers are transported by automated guided vehicles (AGVs) to and from the intermediate stowage areas. AGVs navigate via electronic marks in the ground, thus, the planning focus is on assigning and scheduling the transports. An AGV may lift either one forty feet container or two twenty feet container simultaneously. Therefore, each AGV trip to the stowage area corresponds to serving either a $\mathcal{P}_1$ pattern or a $\mathcal{P}_2$ pattern.

4 Algorithmic Consequences

Ideas similar to pattern concatenation are well established in the operations research literature. Cluster-first route-second heuristics group requests into mini-clusters and concatenate *all* of them into routes. In airline crew pairing [1,15] often a subset of all possible pairings to work with is enumerated *a priori*. In contrast, because of the small number of patterns which is $O(n^2)$, our intention is to select a subset of $\mathcal{P}$ *simultaneously* to the construction of concatenations. Provided that $\mathcal{P}$ is inherently given by the problem our approach is designed to be exact. We may adapt e.g., existing labeling algorithms for resource constrained shortest paths [8], see [19] for a suggestion.

Note however, that although derived from the PDP, pattern concatenation is of a different nature. The essential constraints of Def. 1 are explicitly controlled by the definition of $\mathcal{P}$. We may rather consider our problem as the simplest conceivable pickup and delivery flavored extension to the TSP. Additionally, we have to select patterns that appropriately partition the node set N. On the one hand, this strongly suggests to focus algorithmic design on this selection. On the other hand, it requires some care when adapting the above mentioned node oriented construction methods.

Due to the hardness of finding a shortest $\mathcal{P}$-concatenation, we cannot expect to discover an exact algorithm which is *principally* better than a complete enumeration of feasible solutions, either implicit or explicit. In order to get an impression of the size of the solution space of such algorithms, an interesting question is how many $\mathcal{P}$-concatenations exist, especially in comparison to the general situation of unrestricted pickup and delivery paths. We settle this issue in the next section.

Remark 2 Unlike two-stage strategies, in presence of time windows patterns do not fix the temporal relation of nodes, i.e., travel times within a pattern. For instance, in [3] a vehicle is forced not to wait within a cluster, and the total cluster duration is known in advance.

5 The Number of Concatenations

The aim of this section is to find an upper bound on the size of the solution space of enumeration algorithms for $\mathcal{P}$-concatenation. We would like to make the point that, when compared in a fair way, our restriction entails *much* fewer feasible solutions but still guarantees an acceptable approximation guarantee.

For the sake of simplicity let us assume $\ell_{r^+} = 1$, $r \in \mathcal{R}$ and $L = 2$, in particular *every* pattern is feasible with respect to vehicle capacity. We denote by $\tau^n_{\mathcal{P}}$ the number of $\mathcal{P}$-concatenations which visit all n requests. The number of *all* PD paths is $\tau^n = (2n)!/2^n$, which is simply the number of all directed Hamiltonian paths on $2n$ nodes, discarding those which do not respect the precedence constraints.

Lemma 2 *The number of $\mathcal{P}$-concatenations on $2n$ nodes is*

$$\tau^n_{\mathcal{P}} \;=\; n! \cdot \sum_{i=0}^{\lfloor \frac{n}{2} \rfloor} \frac{2^i}{i!} \cdot \prod_{j=i}^{2i-1} (n-j) \quad . \tag{1}$$

Proof Each summand represents the number of different combinations of patterns for a fixed number $0 \le i \le \lfloor \frac{n}{2} \rfloor$ of $\mathcal{P}_2$ patterns involved in the concatenation. We assume that the empty product evaluates to one. For a fixed i there are $(n-2i)$ $\mathcal{P}_1$ patterns, and therefore $(n-2i+1)$ possibilities to position the first $\mathcal{P}_2$ pattern. For the second $\mathcal{P}_2$ pattern there are $(n-2i+2)$ possible positions, and finally $(n-2i+i)$ possibilities to place the last. This yields the stated product. Symmetry in this counting is accounted for by dividing by the number $i!$ of permutations of the $\mathcal{P}_2$ patterns. Multiplication by 2^i considers the fact that there are two different $\mathcal{P}_2$ patterns for a given ordered pair of requests. Having thus calculated the number of configurations of $\mathcal{P}_1$ and $\mathcal{P}_2$ patterns, multiplication by $n!$ assigns requests to each particular pattern. This gives (1). $\square$

Remark 3 Using $\binom{n}{k} = \frac{n!}{(n-k)! \cdot k!}$ we may rewrite (1) as

$$\tau^n_{\mathcal{P}} \;=\; n! \cdot \sum_{i=0}^{\lfloor \frac{n}{2} \rfloor} 2^i \cdot \binom{n-i}{i} \quad . \tag{2}$$

An interpretation of (2) is that the binomial coefficient gives the number of possibilities to arrange $n-2i$ $\mathcal{P}_1$ and i $\mathcal{P}_2$ patterns (in their two forms): For i fixed requests we only have to decide which of the remaining $n-i$ requests are used to build $\mathcal{P}_2$ patterns.

Remark 4 Using another interpretation we obtain a recursive formula for $\tau^n_{\mathcal{P}}$, for which a different *closed form* can be given [16]. Similar to Fibonacci numbers, define $l_1 = l_2 = 1$ and $l_n = l_{n-1} + 2 \cdot l_{n-2}$ for $2 < n \in \mathbb{N}$.

Lemma 3 $\tau^n_{\mathcal{P}} = n! \cdot l_{n+1} \quad .$ (3)

Proof We use an inductive argument. Lemma 3 holds for $n = 1,2$. Now suppose the claim be valid for arbitrary but fixed $n-1, n \in \mathbb{N}$ with n even. We omit the analogue proof for odd n.

$$\begin{aligned}\tau_{\mathcal{P}}^{n+1} &\overset{(2)}{=} (n+1)! \sum_{i=0}^{\lfloor \frac{n+1}{2} \rfloor} 2^i \binom{n+1-i}{i} = (n+1)! \sum_{i=0}^{\lfloor \frac{n+1}{2} \rfloor} 2^i \left[\binom{n-i}{i} + \binom{n-i}{i-1} \right] \\ &\overset{(\star)}{=} (n+1)n! \sum_{i=0}^{\lfloor \frac{n}{2} \rfloor} 2^i \binom{n-i}{i} + (n+1)n(n-1)! \cdot 2 \sum_{i=0}^{\lfloor \frac{n-1}{2} \rfloor} 2^i \binom{n-1-i}{i} \\ &\overset{(3)}{=} (n+1) \cdot n! l_{n+1} + 2 \cdot (n+1)n \cdot (n-1)! l_n = (n+1)!(l_{n+1} + 2 \cdot l_n) \quad .\end{aligned}$$

The last term equals $(n+1)! \cdot l_{n+2}$. The index transformation $(\star)$ uses n even. □

Although $\tau_{\mathcal{P}}^n$ is exponential in n, its contribution to the total number of PD paths is negligible.

Lemma 4 $\lim_{n\to\infty} \tau_{\mathcal{P}}^n / \tau^n = 0$.

Proof

$$\begin{aligned}\frac{\tau_{\mathcal{P}}^n}{\tau^n} &= \frac{n!}{(2n)!} \cdot \sum_{i=0}^{\lfloor \frac{n}{2} \rfloor} \frac{2^{n+i}}{i!} \cdot \prod_{j=i}^{2i-1} (n-j) \leq \frac{n!}{(2n)!} \cdot n \cdot 2^{1.5n} \cdot n! \\ &= n \cdot 2^{1.5n} \cdot \frac{1 \cdot 2 \cdot \dots \cdot n}{(n+1) \cdot (n+2) \cdot \dots \cdot 2n} < n \cdot \frac{2^{1.5n}}{2^n} = \frac{n}{2^{0.5n}} \longrightarrow 0\end{aligned}$$

for $n \to \infty$. □

Already for very small n the ratio is almost zero, but this was to be expected: The comparison is not fair. More meaningfully, let us compare $\mathcal{P}$-concatenations against PD paths of an almost identically constrained vehicle, one with $L = 2$. This condition is weaker; it allows $i^+, i_1^+, i_1^-, i_2^+, i_2^-, \dots, i_k^+, i_k^-, i^-$ with $i, i_1, \dots, i_k \in \mathcal{R}$, $k > 1$ as a feasible path segment. Let $\tau_{L=2}^n$ denote the number of PD paths which are feasible in this situation.

Lemma 5 $\tau_{L=2}^n = 3^{n-1} n!$.

Proof We prove this result by induction. For $n = 1$ we have $\tau_{L=2}^1 = 3^0 \cdot 1! = 1$ which is clearly true. Now let $\tau_{L=2}^n = 3^{n-1} n!$ hold for an arbitrary but fixed $n \in \mathbb{N}$. We will iteratively construct a path respecting $L = 2$. For each stage, given that $i \in \{0,1,2\}$ items are in the vehicle, and k requests are not yet completed, i.e., k destination nodes are not yet visited, denote by τ_i^k the number of different possibilities to complement the current path from the current node. We seek $\tau_{L=2}^{n+1} = \tau_0^{n+1}$: In the beginning no request is completed and the vehicle is empty.

When the vehicle is empty and k requests are not completed, we have k possibilities to immediately continue the path, i.e., $\tau_0^k = k \cdot \tau_1^k$. When one item is in the vehicle, we can immediately deliver the item, reducing k by one and making the vehicle empty, or we can pick up another item, for which we have $k-1$ possibilities. In the latter case, two items are in the vehicle, thus we must deliver precisely one of them. Then, one item remains in the vehicle, and k is reduced by one, i.e., $\tau_2^k = 2\cdot\tau_1^{k-1}$ for any k. More formally, we have

$$\tau_0^{n+1} = (n+1)\cdot\tau_1^{n+1} = (n+1)\cdot(\tau_0^n + n\cdot\tau_2^{n+1}) =$$
$$(n+1)\cdot(\tau_0^n + 2n\cdot\tau_1^n) = (n+1)\cdot(\tau_0^n + 2n\cdot\frac{\tau_0^n}{n}) = 3\cdot(n+1)\cdot\tau_0^n \quad .$$

Hence, by the induction hypothesis and $\tau_0^n = \tau_{L=2}^n$ we have $\tau_{L=2}^{n+1} = 3^n(n+1)!$ as claimed. □

Lemma 6 $\lim_{n\to\infty}\tau_{\mathcal{P}}^n/\tau_{L=2}^n = 0 \quad .$

Proof Upper bounding $\tau_{\mathcal{P}}^n$ by eliminating the binomial coefficients via

$$\sum_{i=0}^{\lfloor\frac{n}{2}\rfloor} 2^i\binom{n-i}{i} \;\le\; \sum_{i=0}^{\lfloor\frac{n}{2}\rfloor} 2^i\binom{n}{i} \;\le\; 2^{\frac{n}{2}}\sum_{i=0}^{\lfloor\frac{n}{2}\rfloor}\binom{n}{i} \;\le\; 2^{\frac{n}{2}}\sum_{i=0}^{n}\binom{n}{i} \;=\; 2^{\frac{n}{2}}\cdot 2^n$$

we obtain $2^{\frac{n}{2}}\cdot 2^n/3^{n-1} = 3\cdot\sqrt{8}^n/3^n \to 0$ as claimed. □

When the vehicle capacity is two by definition of the problem situation, the restriction to $\mathcal{P}$-concatenations is not that severe. Still, asymptotically, their fraction among PD paths with $L=2$ vanishes.

6 Error Analysis

What is the influence of our restrictions on the quality of solutions in terms of the path length? One easily sees that a vehicle restricted to $\mathcal{P}$-concatenations versus a vehicle driving general PD paths performs arbitrarily bad. On the other hand, when restricting the latter vehicle to $L=2$ the situation changes.

Lemma 7 *Given the optimal length $l_{L=2}$ of a PD path with $L=2$, and the optimal length $l_{\mathcal{P}}$ of a $\mathcal{P}$-concatenation in G, then $l_{\mathcal{P}} \le 3\cdot l_{L=2}$ and this bound is tight.*

Proof For a vehicle with $L=2$, consider an optimal PD path $R=(i_1,\dots,i_K)$ in G of length $l_{L=2}$. We think of unit size items loaded into one of two storage holds in the vehicle. Then, we construct an approximating $\mathcal{P}$-concatenation as follows.

Given R, we know for each item whether it is loaded into storage hold 1 or 2. Further, from R we have the sequence in which items are optimally served by the vehicle. In our $\mathcal{P}$-concatenation we serve only one item at a time. Start in i_1, and pickup and *immediately* deliver the first item which was loaded into storage hold 1. Continue serving all items from hold 1 in the order given by R. Return to i_1, and repeat this procedure for all items which were loaded into storage hold 2.

In the worst case $r^+ = i_1$ and $r^- = i_K$ for some $r\in\mathcal{R}$, and $c_{i_1 i_2} = c_{i_{K-1} i_K} = 0$. Then, the length of this $\mathcal{P}$-concatenation is exactly three times $l_{L=2}$. □

Note that we construct an approximating $\mathcal{P}_1$-concatenation, so that, unfortunately, this gives no better approximation guarantee than a TSP approximation.

7 Polynomially Solvable Cases

A natural question in connection with hard optimization problems is to ask for polynomially solvable special cases. We assume again that we have to visit all n requests in a concatenation. In the context of the TSP, two research directions are classical: Special attributes of the cost matrix, and restrictions to special (typically sparse) graph structures. Balas [2] considers a third class similar in spirit to our approach, *viz.* constraining the combinatorial variability of a tour. We emphasize the TSP polynomial time special cases—which are numerous—because they immediately give rise to an analogous statement for $\mathcal{P}_1$-concatenation. Interestingly, when we drop the TSP component of our problem, the remaining selection of patterns is easy, and could be used in a heuristic cluster-first, route-second approach.

Proposition 1 *A shortest $\mathcal{P}$-concatenation can be found in polynomial time if the sequence of patterns is irrelevant, i.e., all arcs joining two patters have zero weight.*

Proof Consider an undirected graph with vertex set $V_1 \cup V_2 := \{i \mid i \in \mathcal{R}\} \cup \{i' \mid i \in \mathcal{R}\}$ and edge set $E_1 \cup E_2 \cup E_3 := \{(i,j) \mid i \neq j \in \mathcal{R}\} \cup \{(i',j') \mid i \neq j \in \mathcal{R}\} \cup \{(i,i') \mid i \in \mathcal{R}\}$, i.e., for each request there exists a node in V_1 and a copy in V_2 joined by an edge in E_3, and (V_1,E_1) and (V_2,E_2) are complete, c.f. Fig. 1. Edges $(i,j) \in E_1$ and $(i',j') \in E_2$ are weighted with the cost of a cheapest of the four patterns in $\mathcal{P}_2$ involving requests i and j. Similarly, twice the cost of the $\mathcal{P}_1$ pattern for request i defines the weight for $(i,i') \in E_3$.

Let M be a perfect matching in this graph. The edge sets $M_1 = M \cap (E_1 \cup E_3)$ and $M_2 = M \cap (E_2 \cup E_3)$, respectively, each define a selection of patterns in the obvious way. Furthermore, if M is a minimum weight perfect matching, the edge weight of M_1 equals that of M_2. Otherwise, either $M \cap E_1$ or $M \cap E_2$ would have greater weight, contradicting the optimality of M. Thus, the weight of M is twice the cost of an optimal selection of patterns. Serving these patterns in any order gives an optimal solution to the $\mathcal{P}$-concatenation problem. Polynomial time computability of M, *see* e.g., [4], completes the proof. □

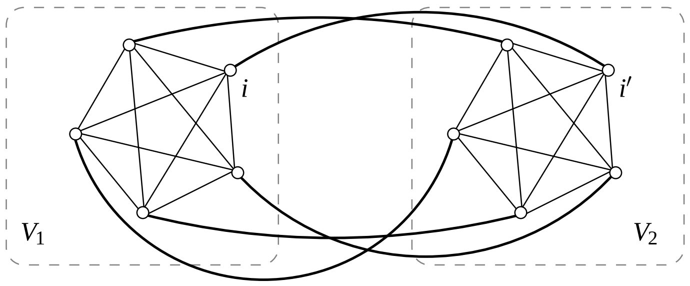

Fig. 1 Sketch of the undirected graph constructed in the proof of Proposition 1

We will now restrict the position of a request within a concatenation, adopting an interesting idea for the TSP. We follow the lines of the original presentation [2].

For a given concatenation, we say that request i *precedes* request j if and only if i^- is visited before j^-, i.e., i is completed not later than j is. Given an integer $1 \leq k < n$, and a linear ordering $(1,\ldots,n)$ of the set $\mathcal{R}$ of requests, a concatenation is required to fulfill the following condition:

$$\text{For all } i \neq j \in \{1,\ldots,n\},\ j \geq i+k \implies i \text{ precedes } j\ . \tag{4}$$

In other words, the *position* of request i within a concatenation is required to be in the interval $[i-k+1, i+k-1]$. Since for $k=1$ the ordering $(1,\ldots,n)$ is already optimal, (4) holds for a relatively small k only when the ordering is sufficiently *close* to the optimal one. We state the main result of [2] in our terminology.

Theorem 1 *Finding an optimal $\mathcal{P}_1$-concatenation with* (4) *needs $O(k^2 2^{k-2} n)$ time.*

An outline of the proof is as follows. The basic idea is to exploit the restricted candidate set for a given position in the concatenation in the well known dynamic programming approach to the TSP. This leads to the construction of a layered graph $G^* = (N^*, A^*)$ with (in our case) $n+2$ layers, the layers being the node sets N_l^*, $l = 0,\ldots,n+1$, where N_0^* and N_{n+1}^* only contain virtual start and end nodes of paths. Each node in N_l^* represents a $\mathcal{P}_1$ pattern in position l together with the knowledge about which requests were already visited. Two nodes are adjacent if and only if the corresponding requests i and j can be served consecutively according to this knowledge and (4). The respective arc is weighted with $c_{i^- j^+} + c_{j^+ j^-}$. Balas [2] shows that $|N_l^*| \leq (k+1)2^{k-2}$, $l = 0,\ldots,n+1$, and that the in-degree of every node is bounded by k. This gives an upper bound on the number of arcs in G^*. The claimed result then follows from the one-to-one correspondence between shortest paths in G^* and optimal $\mathcal{P}_1$-concatenations in G, and the computability of such paths in a layered graph in $O(|A^*|)$. Repeating all the technicalities here would be unduly. However, Balas' idea allows for an immediate generalization.

Proposition 2 *Construction of an optimal $\mathcal{P}$-concatenation with* (4) *can be done in $O(6 \cdot k^4 2^{2k-4} n)$ time.*

Proof Every path in G^* represents a feasible sequence of requests to be completed with respect to (4). Given $i \in N^*$, denote by $\mathrm{req}(i)$ the respective request. When all patterns in $\mathcal{P}$ are allowed, for every two adjacent nodes i and j in G^* we have to decide according to which pattern the corresponding requests $\mathrm{req}(i)$ and $\mathrm{req}(j)$ are to be served. Actually, given that $\mathrm{req}(i)$ precedes $\mathrm{req}(j)$, in addition to consecutively serving two $\mathcal{P}_1$ patterns, there are two possible $\mathcal{P}_2$-patterns.

To represent this, we extend G^*. We make four copies $N_{l,p,1}^*$ and $N_{l,p,2}^*$, $p = 1,2$, of the node sets N_l^*, $l = 2,\ldots,n-1$, two copies $N_{1,p,1}^*$, $p = 1,2$, of N_1^*, and two copies $N_{n,p,2}^*$, $p = 1,2$, of N_n^*. The nodes in $N_{l,p,1}^*$ and $N_{l,p,2}^*$, will represent beginning and ending, respectively, a $\mathcal{P}_2$ pattern in one of its two forms $p \in \{1,2\}$ in the l^{th} position of a concatenation. In each node set $N_{l,p,1}^*$, $p = 1,2$, $l = 1,\ldots,n-1$, we make $|\delta(i)|$ copies from each $i \in N_l^*$, where $\delta(i)$ denotes the set of arcs leaving node i. Denote by $i_{1,\alpha} \in N_{l,p,1}^*$, $\alpha \in \delta(i)$, and $i_2 \in N_{l,p,2}^*$, $p = 1,2$, the respective copies of $i \in N_l^*$, $l = 1, \ldots, n$. All arcs in A^* remain intact. Additionally, we introduce the following ten arc sets:

$$A_{a,p}^* = \bigcup_{l \in \{1,\ldots,n-1\}} \{(i_{1,\alpha}, j_2) \in N_{l,p,1}^* \times N_{l+1,p,2}^* \mid (i,j) \in A^*, \alpha \in \delta(i)\}, \qquad p = 1,2$$

$$A_{b,p}^* = \bigcup_{l \in \{2,\ldots,n-2\}} \{(i_2, j_{1,\alpha}) \in N_{l,p,2}^* \times N_{l+1,p,1}^* \mid (i,j) \in A^*, \alpha \in \delta(i)\}, \qquad p = 1,2$$

$$A_{c,p}^* = \bigcup_{l \in \{0,\ldots,n-2\}} \{(i, j_{1,\alpha}) \in N_l^* \times N_{l+1,p,1}^* \mid (i,j) \in A^*, \alpha \in \delta(i)\}, \qquad p = 1,2$$

$$A_{d,p}^* = \bigcup_{l \in \{2,\ldots,n\}} \{(i_2, j) \in N_{l,p,1}^* \times N_{l+1}^* \mid (i,j) \in A^*\}, \qquad p = 1,2$$

and

$$A_e^* = \bigcup_{l \in \{2,\ldots,n-2\}} \{(i_2, j_{1,\alpha}) \in N_{l,1,2}^* \times N_{l+1,2,1}^* \mid (i,j) \in A^*, \alpha \in \delta(i)\}$$

$$A_f^* = \bigcup_{l \in \{2,\ldots,n-2\}} \{(i_2, j_{1,\alpha}) \in N_{l,2,2}^* \times N_{l+1,1,1}^* \mid (i,j) \in A^*, \alpha \in \delta(i)\} \ .$$

Let us denote by i^+ and i^- the origin and destination location, respectively, of the request associated with node $i \in N^*$. We keep all arc weights w_{ij} for $(i,j) \in A^*$ unchanged, i.e., $w_{ij} = c_{i^- j^+} + c_{j^+ j^-}$. We lose no generality in assuming that

$p = 1$ represents (j^+, i^+, i^-, j^-), and
$p = 2$ represents (i^+, j^+, i^-, j^-).

Observe, that each $i_{1,\alpha} \in N_{l,p,1}^*$, $\alpha \in \delta(i)$, $l = 1, \ldots, n-1$, has a *unique* successor in $N_{l+1,p,2}^*$. By analogy with the above, we denote by $\text{succ}^+(i_{1,\alpha})$ the origin location of the request associated with this successor. This enables us to assign the following weights to the additional arcs:

$(i_{1,\alpha}, j_2) \in A_{a,1}^*$:	$c_{j^+ i^+} + c_{i^+ i^-} + c_{i^- j^-}$	$(i_{1,\alpha}, j_2) \in A_{a,2}^*$:	$c_{i^+ j^+} + c_{j^+ i^-} + c_{i^- j^-}$
$(i_2, j_{1,\alpha}) \in A_{b,1}^*$:	$c_{i^-, \text{succ}^+(j_{1,\alpha})}$	$(i_2, j_{1,\alpha}) \in A_{b,2}^*$:	$c_{i^- j^+}$
$(i, j_{1,\alpha}) \in A_{c,1}^*$:	$c_{i^-, \text{succ}^+(j_{1,\alpha})}$	$(i, j_{1,\alpha}) \in A_{c,2}^*$:	$c_{i^- j^+}$
$(i_2, j) \in A_{d,1}^*$:	$c_{i^- j^+} + c_{j^+ j^-}$	$(i_2, j) \in A_{d,2}^*$:	$c_{i^- j^+} + c_{j^+ j^-}$
$(i_2, j_{1,\alpha}) \in A_e^*$:	$c_{i^- j^+}$		
$(i_2, j_{1,\alpha}) \in A_f^*$:	$c_{i^-, \text{succ}^+(j_{1,\alpha})}$		

Figure 2 gives an overall idea of our network construction. From the definition of our node sets we obtain an upper bound of $(k^2 2^{k-2})^2$ (the square of Balas' bound) for the number of arcs entering any $N_{l,p,1}^*$, $p = 1,2$, $l = 1, \ldots, n-1$. This dominates the original upper bound on the number of arcs entering the other node sets. There are six sets with arcs entering some $N_{l,p,1}^*$, *viz.* $A_{b,1}^*$, $A_{b,2}^*$, $A_{c,1}^*$, $A_{c,2}^*$, A_e^*, A_f^*, and the claim follows from Thm. 1. □

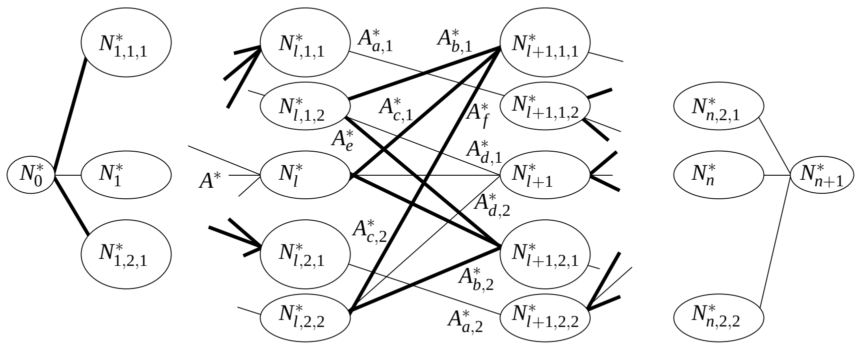

Fig. 2 Expansion of Balas' network constructed in the proof of Thm. 2. Each line represents the arcs connecting adjacent node sets. A thin line indicates that the corresponding arc set has cardinality $|A^*|$, whereas bold lines indicate arc sets with cardinality $|A^*|^2$.

It is interesting to see that the (j^+, i^+, i^-, j^-) patterns cause the costly expansion of the network. Without the expansion we would not have a unique successor of nodes in $i_{1,\alpha} \in N^*_{l,p,1}$, $\alpha \in \delta(i)$, $l = 1, \ldots, n-1$. Unfortunately, this successor determines which location is to be visited in the pattern first. Thus, we would not have been able to define arc weights e.g., for arcs in $A^*_{b,1}$. Note, however, that for (i^+, j^+, i^-, j^-) patterns the first location to be visited is always known when it is given which request is to be completed first.

Corollary 1 *An optimal concatenation of* $\mathcal{P}_1$ *and* (i^+, j^+, i^-, j^-) *patterns with* (4) *can be found in time* $O(5 \cdot k^2 2^{k-2} n)$.

Proof Again, we construct an expanded network. We let $\alpha \equiv 1$ and $p \equiv 2$ in the above construction for the proof of Prop. 2. This leads to four arc sets, $A^*_{a,2}$ through $A^*_{d,2}$, each of which has cardinality $|A^*|$. We are able to uniquely define the respective arc weights precisely as above, and the claim follows from Thm. 1. □

8 Conclusion

We investigate very simple pickup and delivery paths in an analytic way. We introduce small patterns serving exactly one or two requests and concatenate them into paths. This concept is successfully applied in [19]. In this paper we demonstrate that our restriction drastically reduces the search space of an implicit enumeration algorithm for finding an optimal PD path. In fact, all exact solution approaches to the PDP, we are aware of, rely on some dynamic programming scheme. By bounding the prolongation of the path length by our restriction to a factor of three, we have proven that $\mathcal{P}$-concatenations give, in a sense, a good representation of all PD paths where at most two requests are simultaneously served.

Acknowledgments This work was done while the author was with the Department of Mathematical Optimization, Braunschweig University of Technology. I wish to thank Thomas

Lindner, Tom Matsui, and Hannes Scheel for helpful discussions. I gratefully acknowledge the careful reading and constructive suggestions of an anonymous referee of an earlier version of this paper [17].

References

1. R. Anbil, J.J. Forrest, and W.R. Pulleyblank. Column generation and the airline crew pairing problem. In *Proceedings of the International Congress of Mathematicians Berlin*, Extra Volume ICM 1998 of *Doc. Math. J. DMV*, pages III 677–686, August 1998.
2. E. Balas. New classes of efficiently solvable generalized traveling salesman problems. *Ann. Oper. Res.*, 86:529–558, 1999.
3. R. Borndörfer, M. Grötschel, F. Klostermeier, and C. Küttner. Telebus Berlin: Vehicle scheduling in a dial-a-ride system. In N.H.M. Wilson, editor, *Computer-Aided Transit Scheduling*, volume 471 of *Lecture Notes in Economics and Mathematical Systems*, pages 391–422, Berlin, 1999. Springer.
4. W.J. Cook, W.H. Cunningham, W.R. Pulleyblank, and A. Schrijver. *Combinatorial Optimization*. John Wiley & Sons, Chichester, 1998.
5. J.R. Daduna and A. Wren, editors. *Computer-Aided Transit Scheduling*, volume 308 of *Lecture Notes in Economics and Mathematical Systems*. Springer, 1987.
6. G. Desaulniers, J. Desrosiers, A. Erdmann, M.M. Solomon, and F. Soumis. The VRP with pickup and delivery. In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*, SIAM Monographs on Discrete Mathematics and Applications, chapter 9. SIAM, Philadelphia, 2001.
7. M. Desrochers, J.K. Lenstra, M.W.P. Savelsbergh, and F. Soumis. Vehicle routing with time windows: Optimization and approximation. In Golden and Assad [13], pages 65–84.
8. J. Desrosiers, Y. Dumas, M.M. Solomon, and F. Soumis. Time constrained routing and scheduling. In M.O. Ball, T.L. Magnanti, C.L. Monma, and G.L. Nemhauser, editors, *Network Routing*, volume 8 of *Handbooks in Operations Research and Management Science*, pages 35–139. North-Holland, Amsterdam, 1995.
9. J. Desrosiers, Y. Dumas, and F. Soumis. The multiple vehicle dial-a-ride problem. In Daduna and Wren [5], pages 15–27.
10. J. Desrosiers, G. Laporte, M. Sauvé, F. Soumis, and S. Taillefer. Vehicle routing with full loads. *Comput. Oper. Res.*, 15(3):219–226, 1988.
11. Y. Dumas, J. Desrosiers, and F. Soumis. The pickup and delivery problem with time windows. *European J. Oper. Res.*, 54:7–22, 1991.
12. K. Fagerholt and M. Christiansen. A combined ship scheduling and allocation problem. *J. Opl. Res. Soc.*, 51(7):834–842, 2000.
13. B.L. Golden and A.A. Assad, editors. *Vehicle Routing: Methods and Studies*, volume 16 of *Studies in Management Science and Systems*. North-Holland, 2nd edition, 1991.
14. I. Ioachim, J. Desrosiers, Y. Dumas, M.M. Solomon, and D. Villeneuve. A request clustering algorithm for door-to-door handicapped transportation. *Transportation Sci.*, 29(1):63–78, 1995.
15. D. Klabjan, E.L. Johnson, G.L. Nemhauser, E. Gelman, and S. Ramaswamy. Solving large airline crew scheduling problems: Random pairing generation and strong branching. *Comput. Optim. Appl.*, 20(1):73–91, 2001.
16. D.E. Knuth. *The Art of computer programming, Vol. 1: Fundamental Algorithms*. Series in Computer Science and Information Processing. Addison-Wesley, Reading, 1969.

17. M.E. Lübbecke. *Engine Scheduling by Column Generation*. PhD thesis, Braunschweig University of Technology, Cuvillier Verlag, Göttingen, 2001.
18. M.E. Lübbecke and U.T. Zimmermann. Computer aided scheduling of switching engines. In W. Jäger and H.-J. Krebs, editors, *Mathematics—Key Technology for the Future: Joint Projects Between Universities and Industry*, pages 690–702. Springer-Verlag, Berlin, 2003.
19. M.E. Lübbecke and U.T. Zimmermann. Engine routing and scheduling at industrial in-plant railroads. *Transportation Sci.*, 37(2):183–197, 2003.
20. M. Rönnqvist, A. Westerlund, and D. Carlsson. Extraction of logs in forestry using operations research and geographical information systems. In *Proceedings of the 32nd Hawaii International Conference on System Sciences*. IEEE, 1999.
21. M.W.P. Savelsbergh and M. Sol. The general pickup and delivery problem. *Transportation Sci.*, 29(1):17–29, 1995.
22. M. Sol. *Column Generation Techniques for Pickup and Delivery Problems*. PhD thesis, Eindhoven University of Technology, 1994.