On Solving Continuous-time Dynamic Network Flows

S. Mehdi Hashemi[*] Ebrahim Nasrabadi[*,†] Martin Skutella[†]

Abstract

Temporal dynamics is a crucial feature of network flow problems occurring in many practical applications. Important characteristics of real-world networks such as arc capacities, transit times, transit and storage costs, demands and supplies etc. are subject to fluctuations over time. Consequently, also flow on arcs can change over time which leads to so-called dynamic network flows. While time is a continuous entity by nature, discrete time models are often used for modeling dynamic network flows as the resulting problems are in general much easier to handle computationally.

In this paper we study a general class of dynamic network flow problems in the more challenging continuous time model and develop two algorithms based on a discretization approach that compute, or at least converge to optimum solutions. Both algorithms lead to approximate interior solutions, while one would like to have extreme point solutions as these usually have a considerably simpler structure than arbitrary feasible solutions and are more meaningful in practice. We therefore also present a purification algorithm for our dynamic flow problems, that is, an algorithm which given as input an arbitrary feasible solution produces as output an extreme point solution without degrading the objective function value.

Keywords: Dynamic network flows, continuous linear programming, discretization, duality, extreme points, purification.

1 Introduction

Network flows have applications in a wide range of fields, including chemistry, physics, most branches of engineering, manufacturing, scheduling and routing, telecommunication, transportation and logistics (see, for example, [1]). A crucial characteristic of network flows occurring in real-world applications is flow variation over time due to seasonally altering demands, supplies and arc capacities. This is not captured by classical *static* network flow models known from the literature. Here is where *dynamic network flows* come into play. In fact, Ford and Fulkerson [14, 15] introduce network flows by taking into account *transit times* on arcs to capture this important feature of many real-work networks in their seminal work on the subject. Since then, this topic has become an area of active research and many authors have extensively studied different features of dynamic network flows (see [34] and the references given there). The research in this area has pursued two main approaches with respect to time modeling, namely discrete time models and continuous time models. While discretization of time leads to problems that are considerably easier to solve computationally, the more challenging continuous time model reflects reality in more detail. The aim of this paper is to study a general class of dynamic flows in time-varying networks and develop algorithms for solving such problems.

1.1 Problem Description and Formulation

We suppose that there is a single commodity to be routed through a network G. Let the nodes of the network be denoted by the set N and suppose that there is a set of directed arcs $A \subseteq N \times N$, so that (i, j) represents the arc from node i to node j. We assume without essential loss of generality that every pair of nodes is connected by at most one arc. Each arc (i, j) has a *transit cost* $c_{i,j}$, a *transit capacity* $a_{i,j}$

[*]Department of Computer Science, Amirkabir University of Technology, 424 Hafez Ave., Tehran, Iran

[†]Institut für Mathematik, Technische Universität Berlin, Straße des 17. Juni 136, 10623 Berlin, Germany. The work of these authors was supported by Berlin Mathematical School and by DFG RESEARCH CENTER MATHEON "Mathematics for key technologies" in Berlin.

and a *transit time* $\lambda_{i,j}$. Similarly, each node i has a *supply/demand* r_i, a *storage cost* d_i and a *storage capacity* b_i. All of these parameters are supposed to be functions of time. The aim of the *Dynamic Network Flow Problem (DNFP)* is to seek a flow over time that satisfies all supplies/demands and obeys all the transit and storage capacity constraints over time, while minimizing total transit and storage costs. This problem can be formulated in two ways, as we shall discuss below, depending on whether we use a discrete or continuous representation of time.

Discrete-time Model. We first consider the case where time is discretized into steps of unit length. We suppose that the time variable t varies in the set $\{0, 1, \ldots, T\}$ of time steps, where $T > 0$ is a given *time horizon.* Here $c_{i,j}(t)$ is the cost for sending one unit of flow through arc (i, j) at time t, $a_{i,j}(t)$ is an upper bound on the amount of flow that can enter arc (i, j) at time t and $\lambda_{i,j}(t)$ is the amount of time required for flow to traverse arc (i, j) at time t. More precisely, flow which is entering arc (i, j) at time t, arrives at node j at time $t + \lambda_{i,j}(t)$. Moreover, $d_i(t)$ is the cost for storing one unit of flow at node i from time $t - 1$ to t, $b_i(t)$ is an upper bound on the amount of flow that can be stored at node i from time $t - 1$ to t and $r_i(t)$ is the amount of supply or demand at node i at time t. All transit times $\lambda_{i,j}(t)$ as well as the time horizon T are assumed to be non-negative integer values. The *Discrete-time Dynamic Network Flow Problem (DDNFP)* can now be formulated as follows:

$$\begin{aligned}
\text{DDNFP}: \quad \min \quad & \sum_{t=0}^{T} \sum_{(i,j)\in A} c_{i,j}(t)x_{i,j}(t) + \sum_{t=0}^{T} \sum_{i\in N} d_i(t)y_i(t) \\
\text{s.t.} \quad & \sum_{j:(i,j)\in A} x_{i,j}(t) - \sum_{j:(j,i)\in A} \sum_{t':t'+\lambda_{j,i}(t')=t} x_{j,i}(t') + y_i(t+1) - y_i(t) \\
& \qquad = r_i(t), \quad i \in N,\ t \in \{0, \ldots, T\}, \\
& 0 \le x_{i,j}(t) \le a_{i,j}(t), \quad (i, j) \in A,\ t \in \{0, \ldots, T\}, \\
& 0 \le y_i(t) \le b_i(t), \quad i \in N,\ t \in \{0, \ldots, T\}.
\end{aligned}$$

The problem is only defined on $\{0, 1, \ldots, T\}$ and in order to simplify notation we use $x_{i,j}(t)$ for $t < 0$, implicitly assuming that $x_{i,j}(t) = 0$ in this case. In the above formulation $x_{i,j}(t)$ gives the amount of flow sent at time t into arc (i, j) and $y_i(t)$ gives the amount of flow stored at node i from time $t - 1$ to t. Note that $y_i(0)$ is given, not a variable, and represents an initial storage at node i.

DDNFP can be solved by applying classical min-cost flow algorithms on a time-expanded network. However, the size of this network is typically very large for realistic problems and a polynomial time min-cost flow algorithm will in general only yield a pseudo-polynomial time algorithm for DDNFP since the size of the time-expanded network is linear in T (and therefore exponential in $\log T$). In fact, it follows from the work of Klinz and Woeginger [16] that already a very restricted subproblem of DDNFP with constant arc capacities, transit costs and transit times, zero storage costs, and infinite storage capacity is $\mathcal{NP}$-hard. The situation becomes even much more difficult when time is modeled as a continuous quantity rather than discrete.

Continuous-time Model. We now consider the more challenging continuous time model in which the time variable t can take any point in time in the interval $[0, T]$. In contrast to the discrete-time model, $a_{i,j}(t)$ limits the flow rate (i.e., amount of flow per time unit) into arc (i, j) at time t, $d_i(t)$ is the cost per time unit for storing one unit of flow at node i at time t, $b_i(t)$ denotes the maximum storage allowed at node i at time t and $r_i(t)$ represents the supply or demand rate at node i at time t. We require that the components of $\lambda(t)$ are piecewise polynomial, and that the components of $c(t)$, $d(t)$, $r(t)$, $a(t)$, and $b(t)$ are bounded measurable functions on $[0, T]$.

A *flow over time* in the network G with time horizon T is a function $x : A \times [0, T] \longrightarrow \mathbb{R}^+ \cup \{0\}$ that assigns a flow rate $x_{i,j}(t)$ to every point in time t and every arc (i, j). We assume that an initial storage of $y_i(0)$ is associated with each node i. The flow over time x and initial storage $y(0)$ induce a storage function $y : N \times [0, T] \longrightarrow \mathbb{R}^+ \cup \{0\}$ that assigns a storage $y_i(t)$ to every node i at each point in time t. More specifically, $y_i(t)$ measures the amount of flow stored at node i at time t and $x_{i,j}(t)$, which is required to be a bounded measurable function on $[0, T]$, gives the rate of flow entering arc (i, j) at time t.

Mathematically, the *Continuous-time Dynamic Network Flow Problem (CDNFP)* can now be expressed as follows:

$$\begin{aligned}
\text{CDNFP}: \quad \min \quad & \int_0^T c(t)^T x(t)dt + \int_0^T d(t)^T y(t)dt \\
\text{s.t.} \quad & \int_0^t \sum_{j:(i,j)\in A} x_{i,j}(s)ds - \int_0^t \sum_{j:(j,i)\in A} \sum_{s':s'+\lambda_{j,i}(s')=s} x_{j,i}(s')ds + y_i(t) \\
& \qquad = y_i(0) + \bar{r}_i(t), \quad i \in N,\ t \in [0,T], \qquad (1) \\
& 0 \le x(t) \le a(t), \quad t \in [0,T], \qquad (2) \\
& 0 \le y(t) \le b(t), \quad t \in [0,T]. \qquad (3)
\end{aligned}$$

For the ease of notation, we assume that $x_{i,j}(t) = 0$, for $t < 0$. In the *flow conservation constraints* (1), the first integral represents the total amount of flow that is leaving node i up to time t. Analogously, the second integral represents the total amount of flow that is entering node i up to time t. Notice that the flow which is leaving node j at time s' via arc (j,i), will enter node i at time $s' + \lambda_{j,i}(s')$.

As mentioned above, any choice of flow $x(s)$, $s \in [0,t]$, will uniquely determine a storage function $y(t)$ by the *flow conservation constraints* (1). If $x(t)$ satisfies *transit capacity constraints* (2) and generates storage $y(t)$ satisfying the *storage capacity constraints* (3) for all $t \in [0,T]$, then we say that $x(t)$ (with corresponding storage $y(t)$) is *feasible* for CDNFP. Hence the feasible region for CDNFP, denoted by F, can be defined as

$$F \quad = \quad \left\{ x(t) \in L_\infty^{|A|}[0,T] : x(t), y(t) \text{ is feasible for CDNFP with } y(t) \text{ derived from (1)} \right\}.$$

1.2 Results from the Literature

Continuous-time models for network flow problems were first introduced by Philpott [23] and further studied by Anderson, Nash and Philpott [5]. They consider the problem of sending as much flow as possible from a source node s to a sink node t by horizon time T in a network with zero transit times and time-varying transit and storage capacities. They also introduce the concept of s-t-cuts over time and establish a max-flow min-cut theorem (see also [4]). This result was later extended to arbitrary transit times by Philpott [25].

Anderson [9] studies CDNFP and presents a characterization of the extreme point solutions for the special case of constant and rational transit times. Later, Anderson and Philpott [10] survey results relating to continuous-time network flows as well as CDNFP. They define a dual problem for CDNFP and prove a weak duality result.

In the absence of transit times on the arcs and storage capacities and costs at the nodes, CDNFP is reduced to the so-called *Continuous-time Network Flow Problem (CNFP)*. This problem was introduced by Philpott [24] and studied in more details by Anderson and Philpott [7]. They develop a continuous-time version of the simplex method for CNFP under the assumption that the cost functions on the arcs are piecewise linear. All remaining functions are assumed to be piecewise constant, that is, supply and demand rates and arc capacities. There are no guarantees for the convergence of this algorithm and it often produces a sequence of solutions which converge to a suboptimal instead of an optimum solution. Philpott and Craddock [26] develop an adaptive discretization algorithm for solving CNFP under the same assumptions as in [7] and present encouraging computational results.

CNFP is a special type of the *Separated Continuous Linear Program (SCLP)*, which takes the following form:

$$\begin{aligned}
\text{SCLP}: \quad \min \quad & \int_0^T c(t)^T x(t)dt \\
\text{s.t.} \quad & \int_0^t Kx(s)ds + y(t) = a(t),\ t \in [0,T], \qquad (4) \\
& Hx(t) \le b(t),\ t \in [0,T], \qquad (5) \\
& x(t) \ge 0, y(t) \ge 0, \quad t \in [0,T]. \qquad (6)
\end{aligned}$$

Here $c(t), a(t)$ and $b(t)$ are column vector valued functions, defined on the time interval $[0, T]$, of dimensions n_1, n_2 and n_3, respectively, and K and H are fixed-matrix of dimensions $n_2 \times n_1$ and $n_3 \times n_1$, respectively. The decision variables are given by $x(t)$ and $y(t)$, which are vector valued functions of dimensional n_1 and n_2. Moreover, $x(t)$, $c(t)$ and $b(t)$ are bounded measurable functions, and $y(t)$ and $a(t)$ are absolutely continuous functions. The term "separated" refers to the fact that the constraints are partitined into two sets, the integral constraints (4) and the instantaneous constraints (5) and (6).

The SCLP problem was first introduced by Anderson [2] in order to model job-shop scheduling problems. This problem has attracted most of the attention in the class of continuous-time linear programs (CLP) due to its applications and has been investigated by several authors. In particular, Anderson, Nash, and Perold [3] characterize the extreme point solutions of SCLP and show the existence of optimum solutions with a finite number of breakpoints in certain cases. Anderson and Philpott [8] discuss the form of solutions for SCLP. They prove the existence of a piecewise analytic optimum solution under certain assumptions on the problem data and establish a strong duality result.

Most of the progress in the field of SCLP has been achieved by Pullan. In a series of papers [27, 28, 29, 30, 31, 32, 33], he extensively studies the SCLP. In particular, he develops a detailed duality theory, conditions under which an optimum solution exists with a finite number of breakpoints as well as a convergent algorithm for solving SCLP under certain assumptions on the problem data. Fleischer and Sethuraman [13] present polynomial-time approximation algorithms for solving SCLP under certain assumptions on the problem data. In contrast to previous approaches [20, 26, 27], their algorithm uses a fixed partition of $[0, T]$, specifically designed to meet the accuracy requirement on the solution. Weiss [35] examines SCLP under the assumption of linear problem data. He characterizes the form of optimum solutions, establishes a strong duality result and develops a solution algorithm using simplex pivot operations.

Anderson and Pullan [11] develop a purification algorithm for SCLP, that is, the process of turning an arbitrary feasible solution into an extreme point solution whose objective function value is no worse. They also demonstrate by numerical examples that the use of a purification algorithm can significantly enhance the performance of the algorithm developed by Pullan [27] for the solution of SCLP.

Luo and Bertsimas [19, 20] examine a larger subclass of CLP so-called *State-Constrained Separated Continuous Linear Programs (SCSCLP)*, which is motivated by real-world applications in areas such as communication, manufacturing, and urban traffic control. They use quadratic programming techniques in conjunction with discretization and develop an algorithm for the solution of SCSCLP under some assumptions on the form of the problem data.

Pullan [31] studies a more general class of SCLP to include time-delays in the following form, so-called *Separated Continuous Linear Programs with Time-Delays (SCLPTD)*:

$$\begin{aligned}
\text{SCLPTD}: \quad \min \ & \int_0^T c(t)^T x(t) dt \\
\text{s.t.} \ & \int_0^t \big(Kx(s)\big)_i ds + \sum_{j=1}^{n_1} \int_0^t \gamma_{i,j} x_j(s - \lambda_{ij}) ds + y_i(t) \\
& \qquad\qquad\qquad\qquad = a_i(t), \quad i = 1, \dots, n_2, \ t \in [0, T], \\
& Hx(t) \le b(t), \ t \in [0, T], \\
& x(t) \ge 0, y(t) \ge 0, \quad t \in [0, T].
\end{aligned}$$

This problem serves as a useful model for a variety of dynamic network problems with zero transit times on the arcs. In fact, it is not difficult to see that SCLPTD includes CDNFP as a special case if there is no upper bound on the storage at nodes, the storage costs are zero and the transit times are constant. This can be seen by letting n_1 denote the number of arcs (numbered 1 to n_1) and n_2 denote the number of nodes (numbered 1 to n_2), and taking $K_{i,j} = 1$ if node i is the tail of arc j and $K_{i,j} = 0$ otherwise, H the identity matrix, $a_i(t)$ the total supply/demand at node i up to time t, and by setting $\gamma_{i,j} = -1$ if node i is the head of arc j and $\gamma_{i,j} = 0$ otherwise. Notice that here $\lambda_{i,j}$ denotes the transit time on arc j whose tail is node i.

Pullan [31] assumes that the transit times $\lambda_{i,j}$ and the time horizon T are all rational. He then characterizes the extreme point solutions and proves the existence of piecewise analytic optimum extreme point solutions for the case where input functions are piecewise analytic. Furthermore, he uses the ideas of [27] to present an algorithm for solving SCLPTD. To this end, Pullan [31] first transforms the SCLPTD, in two stages, into a much larger problem which is very close to a special class of SCLP and then uses the theory of SCLP to derive similar results for SCLPTD. In the first stage, rational time-delays and the time horizon are converted into integers by scaling time by some common denominator. Notice that the time horizon T increases enormously. In the second stage, the dimensions of the variables are scaled by T such that also the dimensions of the variables become very large. As mentioned already by Pullan [31], the resulting problem will be huge in general and his algorithm may thus be difficult to use. Moreover, his transformation technique can no longer be used for the case of time-varying or arbitrary transit times.

1.3 Contribution and Organization of the Paper

The aim of this paper is developing solution algorithms as well as a purification algorithm for CDNFP under certain assumptions on the problem data. We make use of techniques from the fascinating area of continuous-time linear programming to derive network related results. In contrast to Pullan's approach [31], we work directly on the original problem and thus avoid facing a much larger problem. The section-by-section description and contribution of the rest of this paper follow.

In Section 2 we consider CDNFP with piecewise linear transit costs and storage capacities, while supply and demand rates, storage costs, transit capacities, and transit times are supposed to be piecewise constant. By partitioning the time interval $[0, T]$ into finitely many subintervals, we construct two different discretizations of CDNFP. One discretization is based on averaging the network properties over each subinterval and another one corresponds to a dual problem CDNFP*. It is worth nothing that both discretized problems can be solved by any of the well-known min-cost flow algorithms on a time-expanded network to give a lower and an upper bound, respectively, on the optimum solution value of CDNFP. We show that the gap between these bounds converges to zero when the time discretization gets arbitrarily fine. Indeed, not only the optimum value of CDNFP but also the optimum value of CDNFP* is nested between the upper and lower bounds provided by the two discretizations.

The results from Section 2 can be turned into an algorithmic approach for solving CDNFP that outputs a sequence of feasible solutions converging to an optimum solution value. The main idea is to solve the two discretized problems for successively finer partitions of time. The quality of the solution can be controlled by the gap between the current upper and lower bound. There are a variety of possible implementations of this algorithm based on how fine the discretization at each iteration. The typical method, which is called *uniform discretization*, is to divide the time interval into a series of subintervals of equal length. Here the time interval is arbitrarily divided and no information from solutions of discrete approximations is used to fine the discretization.

In Section 3 we present an algorithm which solves the two discretized problems on successively finer (nonuniform) discretizations. In contrast to the uniform discretization, the algorithm uses the properties of the second discretized problem (the one corresponding to the the dual problem) in order to insert new breakpoints at appropriate places and consequently does not discretize arbitrarily the time interval. It is worth noting that this idea was introduced by Pullan [27] for SCLP. We also present an alternative algorithm that proceeds by adding and removing points at favorable places, thereby adapting the discretization used in the current iteration. This is built upon ideas of Philpott and Craddock [26] that turns out to be more efficient in practice.

We give an empirical analysis of the developed algorithms for small example instances. Computational results show that already after few iterations the number of breakpoints in the discretizations and thus the encoding size of the solutions and the time required to compute them is huge. Moreover, it keeps growing at a fast rate while the improvement of the objective function value is only marginal. On the other hand, the structure of an optimum solution is often much simpler than the structure of those produced by the presented algorithms. A good solution to this problem is purification as extreme point solutions usually have a considerably simpler structure than arbitrary feasible solutions.

In Section 4 we give a characterization of extreme point solutions and present a purification algorithm for the general case of piecewise analytic problem data and constant transit times. The purification algorithm can be incorporated into the algorithms for solving CDNFP to enhance the solution procedure and develop a simplex-like algorithm.

2 Time-expanded Networks and Discretizations

As mentioned earlier, DDNFP can be solved by computing a min-cost flow in a time-expanded network. But this is not longer true for CDNFP and an optimum solution cannot be obtained by this approach in general situations. However, we can compute an upper bound and a lower bound on the optimum value of CDNFP by using two non-uniform time-expanded networks if the input data is supposed to be piecewise constant/linear. In this section, we describe the construction of these two time-expanded networks followed by their mathematical formulations and a discussion of their properties. Before beginning our discussion, we give some definitions and notation that we will use throughout this and the following section.

Definition 2.1.

1. *A set $P = \{t_0, \ldots, t_m\}$ is said to be a partition of the time interval $[0, T]$ if*
$$0 = t_0 \leq t_1 \leq \ldots \leq t_m = T.$$
The term breakpoints *will be used to refer to the points in P. The* norm *of the partition P, denoted by $||P||$, is defined by*
$$||P|| := \max\{t_k - t_{k-1} : \ k = 1, 2 \ldots, m\}.$$

2. *Let f be a real-valued function defined on the time interval $[0, T]$ and $P = \{t_0, \ldots, t_m\}$ be a partition of $[0, T]$. We say that f is* piecewise constant (linear) *with respect to the partition P, if it is constant (linear) on $[t_{k-1}, t_k)$ for $k = 1, \ldots, m$. We say that f is piecewise constant (linear) on $[0, T]$ if it is piecewise constant (linear) with respect to some partition of $[0, T]$. The* breakpoints *of a piecewise linear or piecewise constant function are the discontinuity points in the function and its derivatives.*

3. *Let f be a real-valued function. We will use the notation*
$$f(t-) = \lim_{s \to t-} f(s), \quad f(t+) = \lim_{s \to t+} f(s),$$
when the above limits exist.

4. *We shall use the notation $V[OP]$ to denote the optimum value of an optimization problem (OP), where it is understood that the value ∞ if OP is an infeasible minimization problem and $-\infty$ if OP is an infeasible maximization problem. Moreover, the notation $V[OP, x]$ will be used to denote the objective function value of OP for a given feasible solution x.*

We now make the following assumption on the form of the problem data.

Assumption 1. *The functions $c(t)$, $\bar{r}(t)$, and $b(t)$ are piecewise linear (with continuous $\bar{r}(t)$ and $b(t)$), and the functions $d(t)$, $a(t)$ and $\lambda(t)$ are piecewise constant (with nonnegative $\lambda(t)$).*

This assumption is supposed to hold throughout the rest of this and the following section. We give the following definition due to the existence of transit times in our model.

Definition 2.2. *Let β denote the set of all breakpoints of $c(t), d(t), \bar{r}(t), a(t), b(t)$ and $\lambda(t)$. A partition $P = \{t_0, \ldots, t_m\}$ of $[0, T]$ is said to be* valid *if it contains the set β, and for each arc (i, j) and any breakpoint $t_k \in P$, the following statements hold:*

(i) if $t_k + \lambda_{i,j}(t_k+) \leq T$, then $t_k + \lambda_{i,j}(t_k+) \in P$,

(ii) if $t_k - \lambda_{i,j}(t+) \geq 0$, then $t_k - \lambda_{i,j}(t+) \in P$, for $t \in \{t' : t' + \lambda_{i,j}(t'+) = t_k\}$.

The set of all valid partitions of $[0, T]$ is denoted by $\mathcal{P}$.

2.1 Computing an Upper Bound

Given a partition $P = \{t_0, t_1, \ldots, t_m\} \in \mathcal{P}$, a *non-uniform time-expanded network* of G, denoted by $G(P)$, is created as follows: $G(P)$ contains $m+1$ copies of N, denoted by $N_0, N_1, \ldots, N_m$, in which N_{k-1} corresponds to the time interval $[t_{k-1}, t_k)$, and N_m to the time horizon T. Here and throughout the rest of this section, unless mentioned otherwise, index k varies between 1 and m. The copy of node $i \in N$ in N_{k-1} is denoted by i_{k-1}. For each arc $(i,j) \in A$ and each time interval $[t_{k-1}, t_k)$ with $0 \le t_{k'} = t_{k-1} + \lambda_{i,j}(t_{k-1}+) \le t_{m-1}$, there is an arc $(i_{k-1}, j_{k'})$. The amount of flow that passes through this arc corresponds to a flow over time entering arc (i,j) in the time interval $[t_{k-1}, t_k)$. Since arc $(i_{k-1}, j_{k'})$ corresponds to a time interval of length $t_k - t_{k-1}$, we set $(t_k - t_{k-1}) a_{i,j}(t_{k-1}+)$ as the capacity of arc $(i_{k-1}, j_{k'})$. Moreover, for each node i, there is a holdover arc from node i_{k-1} to node i_k. The amount of flow that passes through (i_{k-1}, i_k) corresponds to the amount of flow stored at node i at time t_k. Thus, an arc capacity equal to $b_i(t_k)$ is associated with the holdover arc (i_{k-1}, i_k). Supply/demand at node i_{k-1} is set to the total amount of supply/demand at node i during $[t_{k-1}, t_k)$, i.e., $\int_{t_{k-1}}^{t_k} r_i(s)ds = \bar{r}_i(t_k) - \bar{r}_i(t_{k-1})$. There is no supply/demand at node i_m, for $i \in N$. In the model with initial storage at nodes, an additional copy of N, denoted by N_{-1}, is added to $G(P)$. A supply of $y_i(0)$ is associated with each node i_{-1}, in which i_{-1} is the label of node i in N_{-1}. Moreover, for each node i, a holdover arc from i_{-1} to i_0 is introduced with infinite capacity and zero cost.

Any static flow in $G(P)$ corresponds to a flow over time in G: consider a static flow in $G(P)$. Let $\hat{x}(t_{k-1}+)$ denote the amount of flow passing through N_{k-1} and $\hat{y}(t_k)$ the amount of flow on holdover arcs from N_{k-1} to N_k. A flow over time $x(t)$ is obtained by interpreting $\hat{x}(t_{k-1}+)/(t_k - t_{k-1})$ as the flow rate on arc set A in the interval $[t_{k-1}, t_k)$ and by interpreting $\hat{y}(t_k)$ as the amount of flow stored at node set N at time t_k. Specifically, the flow over time $x(t)$ and storage $y(t)$ are determined as follows:

$$x(t) = \begin{cases} \frac{\hat{x}(t_{k-1}+)}{t_k - t_{k-1}}, & t \in [t_{k-1}, t_k), \quad k = 1, \ldots, m, \\ \frac{\hat{x}(t_{m-1}+)}{t_m - t_{m-1}}, & t = T, \end{cases} \tag{7}$$

$$y(t) = \left(\frac{t_k - t}{t_k - t_{k-1}}\right) \hat{y}(t_{k-1}) + \left(\frac{t - t_{k-1}}{t_k - t_{k-1}}\right) \hat{y}(t_k), \; t \in [t_{k-1}, t_k], \quad k = 1, \ldots, m. \tag{8}$$

Conversely, any flow over time $x(t)$ with corresponding storage $y(t)$ corresponds to a static flow in $G(P)$ by averaging $x(t)$ on any arc in each time interval $[t_{k-1}, t_k)$ and by interpreting $y(t_k)$ as the amount of flow on the holdover arcs from N_{k-1} to N_k.

Now we discuss how to assign costs to the arcs in $G(P)$ so that the cost of the static flow in $G(P)$ is the same as the cost of the corresponding flow over time in G. Since the flow over time $x(t)$ obtained by using $G(P)$ and the transit costs $c(t)$ are constant and linear on $[t_{k-1}, t_k)$, respectively, the total transit cost is equal to

$$\int_0^T c(t)^T x(t) dt = \sum_{k=1}^m \int_{t_{k-1}}^{t_k} c(t)^T x(t) dt = \sum_{k=1}^m c\left(\frac{t_k + t_{k-1}}{2}\right)^T \hat{x}(t_{k-1}+).$$

We therefore assign a cost of $c\left(\frac{t_k + t_{k-1}}{2}\right)$ to the arcs whose tails are in N_{k-1}.

Moreover, since $y(t)$ and $d(t)$ are linear and constant over $[t_{k-1}, t_k)$, respectively, the total storage cost is equal to

$$\int_0^T d(t)^T y(t) dt = \sum_{k=1}^m \int_{t_{k-1}}^{t_k} d(t)^T y(t) dt = \sum_{k=1}^m \frac{t_k - t_{k-1}}{2} d(t_{k-1}+)^T (\hat{y}(t_k) + \hat{y}(t_{k-1})).$$

Hence, we assign a cost of $\frac{t_1 - t_0}{2} d_i(t_0+)$ to the holdover arcs from N_{-1} to N_0, a cost of

$$\frac{t_k - t_{k-1}}{2} d_i(t_{k-1}+) + \frac{t_{k+1} - t_k}{2} d_i(t_k+)$$

to the holdover arcs from N_{k-1} to N_k for $k = 1, \ldots, m-1$, and a cost of

$$\frac{t_m - t_{m-1}}{2} d_i(t_{m-1}+)$$

to the holdover arcs from N_{m-1} to N_m.

We denote $G(P)$ with the associated costs, as discussed above, by $G_U(P)$. We can now establish the following theorem.

Theorem 2.3. *Let P be an arbitrary partition in $\mathcal{P}$. Any flow over time in G corresponds to a static flow in $G_U(P)$. Moreover, any static flow in $G_U(P)$ corresponds to a flow over time in G of equal cost.*

Mathematically, the static min-cost flow problem in $G_U(P)$ can be formulated as follows:

$$\begin{aligned}
\min\ & \sum_{k=1}^{m} c\left(\frac{t_k + t_{k-1}}{2}\right)^T \hat{x}(t_{k-1}+) + \sum_{k=1}^{m} \frac{t_k - t_{k-1}}{2} d(t_{k-1}+)^T (\hat{y}(t_k) + \hat{y}(t_{k-1})) \\
\text{s.t.}\ & \sum_{j:(i,j)\in A} \hat{x}_{i,j}(t_0+) - \sum_{j:(j,i)\in A, \lambda_{j,i}(t_0+)=0} \hat{x}_{j,i}(t_0+) + \hat{y}_i(t_1) = \bar{r}_i(t_1), \quad i \in N, \\
& \sum_{j:(i,j)\in A} \hat{x}_{i,j}(t_{k-1}+) - \sum_{j:(j,i)\in A} \sum_{t:t+\lambda_{j,i}(t+)=t_{k-1}} \hat{x}_{j,i}(t+) + \hat{y}_i(t_k) - \hat{y}_i(t_{k-1}) \\
& \qquad = \bar{r}_i(t_k) - \bar{r}_i(t_{k-1}), \quad k = 2, \ldots, m,\ i \in N, \\
& 0 \le \hat{x}(t_{k-1}+) \le (t_k - t_{k-1}) a(t_{k-1}+), \quad k = 1, \ldots, m, \\
& 0 \le \hat{y}(t_k) \le b(t_k), \quad k = 1, \ldots, m.
\end{aligned}$$

This problem is only defined on $P = \{t_0, t_1, \ldots, t_m\}$, and so it is an implicit constraint that $\hat{x}_{i,j}(t+) = 0$, for $t < 0$. We will refer to the above problem as $DP(P)$ that can be viewed as a discrete approximation of CDNFP. Having established Theorem 2.3, we can immediately conclude the following result.

Corollary 1. *Let P be an arbitrary partition in $\mathcal{P}$. $DP(P)$ is feasible if and only if CDNFP is feasible. Moreover $V[CDNFP] \le V[DP(P)]$.*

Corollary 1 shows that the optimum value of $DP(P)$ gives an upper bound on the optimum value of CDNFP. Moreover, if it is known that there is some optimum solution $x(t), y(t)$ in which $x(t)$ is piecewise constant with breakpoints in some partition $P \in \mathcal{P}$, then an optimum solution for CDNFP can be obtained by solving $DP(P)$. Unfortunately, we cannot use this result to solve CDNFP, since the set of all breakpoints of an optimum solution is unknown in advance. However, we can obtain a lower bound on the optimum value of CDNFP by introducing a slightly different time-expanded network from $G_U(P)$. This is the context of next subsection.

2.2 Computing a Lower Bound

To compute a lower bound on the optimum value of CDNFP, we consider a dual problem and introduce a corresponding discrete approximation.

The dual problem of CDNFP, denoted by $CDNFP^*$, is defined as follows:

$$\begin{aligned}
\text{CDNFP}^*:\ \max\ & -\int_0^T \{y(0) + \bar{r}(t)\}^T d\pi(t) - \int_0^T a(t)^T \rho(t) dt - \int_0^T b(t)^T d\eta(t) \\
\text{s.t.}\ & \pi_i(t) - \sum_{t':t'-\lambda_{i,j}(t)=t} \pi_j(t') - \rho_{i,j}(t) \le c_{i,j}(t), \quad (i,j) \in A,\ t \in [0,T], \\
& \pi(t) \text{ of bounded variation and right continuous on } [0,T] \text{ with } \pi(T) = 0, \\
& \eta(t) \text{ monotonic increasing and right continuous on } [0,T] \text{ with } \eta(T) = 0, \\
& \rho(t) \text{ bounded measurable and nonnegative on } [0,T], \\
& \pi(t) + \eta(t) - \int_t^T d(s) ds \text{ monotonic increasing on } [0,T].
\end{aligned}$$

The form of this dual is based on that given by Pullan [27] for the dual of SCLP. A detailed discussion of the above problem and its equivalent formulations can be found in [21]. The following weak duality result is easily established.

Lemma 1 (Weak duality)**.** *Let $x(t), y(t)$ and $\pi(t), \eta(t), \rho(t)$ be feasible solutions for CDNFP and CDNFP*, respectively. Then*

$$V[CDNFP^*, \pi(t), \eta(t), \rho(t)] \leq V[CDNFP, x(t), y(t)].$$

Given a partition $P = \{t_0, t_1, \ldots, t_m\}$, we now construct another non-uniform time-expanded network, denoted by $G_L(P)$, as follows: $G_L(P)$ has the same arc set and node set as $G_U(\bar{P})$, where $\bar{P}$ includes P with each interval split in half, i.e., $\bar{P} = \{t_0, \frac{t_0+t_1}{2}, t_1 \ldots, \frac{t_{m-1}+t_m}{2}, t_m\}$. Thus, copies of N in $G_L(P)$ correspond to time intervals defined by the breakpoints in $\bar{P}$. We denote the copy of N corresponding to the time intervals $[t_{k-1}, \frac{t_{k-1}+t_k}{2})$ and $[\frac{t_{k-1}+t_k}{2}, t_k)$ by N_{k-1}^+ and N_k^-, respectively, and the copy of N corresponding with t_m by N_m^+. For modeling initial storage at nodes, a copy of N, denoted by N_0^- is introduced by adding holdover arcs from N_0^- to N_0^+. The capacity of arcs and supply/demand at nodes in $G_L(P)$ are defined in the same way as in $G_U(\bar{P})$. Thus, any static flow in $G_L(P)$ corresponds to a flow over time in G and vise versa.

The only difference between $G_L(P)$ and $G_U(\bar{P})$ is in the assignment of costs to the arcs. We wish to assign costs to the arcs in such a way so that the static flow in $G_L(P)$ gives a lower bound on the optimum value of CDNFP. To this end, the costs of the arcs passing N_{k-1}^- and N_k^+ are set to $c(t_{k-1}-)$ and $c(t_k+)$, respectively. Furthermore, the costs of the holdover arcs from N_{k-1}^+ to N_k^- and from N_k^- to N_k^+ are set to zero and $(t_k - t_{k-1})d(t_{k-1}+)$, respectively.

We shall show that a min-cost flow in $G_L(P)$ gives a lower bound on the cost of the optimum solution in the original network G. The static min-cost flow in $G_L(P)$ can be expressed as follows:

$$\begin{aligned}
\min\ & \sum_{k=1}^{m} \left(c(t_{k-1}+)^T \hat{x}(t_{k-1}+) + c(t_k)^T \hat{x}(t_k-) \right) + \sum_{k=1}^{m} (t_k - t_{k-1}) d(t_{k-1}+)^T \hat{y}\left(\frac{t_k + t_{k-1}}{2}\right) \\
\text{s.t. } & \sum_{j:(i,j)\in A} \hat{x}_{i,j}(t_0+) - \sum_{j:(j,i)\in A, \lambda_{j,i}(t_0+)=0} \hat{x}_{j,i}(t_0+) + \hat{y}_i\left(\frac{t_1+t_0}{2}\right) = \bar{r}_i\left(\frac{t_1+t_0}{2}\right), \ i \in N, \\
& \sum_{j:(i,j)\in A} \hat{x}_{i,j}(t_{k-1}+) - \sum_{j:(j,i)\in A} \sum_{t:t+\lambda_{j,i}(t+)=t_{k-1}} \hat{x}_{j,i}(t+) + \hat{y}_i\left(\frac{t_k+t_{k-1}}{2}\right) - \hat{y}_i(t_{k-1}) \\
& \qquad = \bar{r}_i\left(\frac{t_k+t_{k-1}}{2}\right) - \bar{r}(t_{k-1}), \quad k = 2, \ldots, m, \quad i \in N, \\
& \sum_{j:(i,j)\in A} \hat{x}_{i,j}(t_k-) - \sum_{j:(j,i)\in A} \sum_{t:t+\lambda_{j,i}(t-)=t_k} \hat{x}_{j,i}(t-) + \hat{y}_i(t_k) - \hat{y}_i\left(\frac{t_k+t_{k-1}}{2}\right) \\
& \qquad = \bar{r}_i(t_k) - \bar{r}_i\left(\frac{t_k+t_{k-1}}{2}\right), \quad k = 1, \ldots, m, \ i \in N, \\
& 0 \leq \hat{x}(t_{k-1}+) \leq \left(\frac{t_k - t_{k-1}}{2}\right) a(t_{k-1}+), \quad k = 1, \ldots, m, \\
& 0 \leq \hat{x}(t_k-) \leq \left(\frac{t_k - t_{k-1}}{2}\right) a(t_k-), \quad k = 1, \ldots, m, \\
& 0 \leq \hat{y}(t_k) \leq b(t_k), \quad k = 1, \ldots, m, \\
& 0 \leq \hat{y}\left(\frac{t_k+t_{k-1}}{2}\right) \leq b\left(\frac{t_k+t_{k-1}}{2}\right), \quad k = 1, \ldots, m.
\end{aligned}$$

This problem is only defined on $\bar{P} = \{t_0, \frac{t_0+t_1}{2}, t_1 \ldots, \frac{t_{m-1}+t_m}{2}, t_m\}$, and so it is an implicit constraint that $\hat{x}_{i,j}(t+) = 0, \hat{x}_{i,j}(t-) = 0$, for $t < 0$. In this formulation, $\hat{x}(t_{k-1}+)$ and $\hat{x}(t_k-)$ denote the amount of flow passing through N_{k-1}^+ and N_k^-, respectively, and $\hat{y}(\frac{t_{k-1}+t_k}{2})$ and $\hat{y}(t_k)$ denote the amount of flow along the holdover arcs from N_{k-1}^+ to N_k^- and from N_k^- to N_k^+, respectively. We will refer to the above problem as $AP(P)$ that can be viewed as another discrete approximation of CDNFP.

The time-expanded network $G_L(P)$ (or equivalently the discrete approximation $AP(P)$) has many interesting properties that enable us to develop an algorithm for solving CDNFP. In what follows, we present some of them needed for the purposes of this paper.

Lemma 2. *Let P be an arbitrary partition in $\mathcal{P}$. Then CDNFP is feasible if and only if $AP(P)$ is feasible.*

Proof. Let $P = \{t_0, \ldots, t_m\} \in \mathcal{P}$ and $\hat{x}, \hat{y}$ be a feasible solution for $AP(P)$. It is clear that this solution forms a feasible solution $x(t), y(t)$ for CDNFP defined by

$$x(t) = \begin{cases} \frac{2\hat{x}(t_{k-1}+)}{t_k - t_{k-1}}, & t \in \left[t_{k-1}, \frac{t_k - t_{k-1}}{2}\right), \quad k = 1, \ldots, m, \\ \frac{2\hat{x}(t_k-)}{t_k - t_{k-1}}, & t \in \left[\frac{t_k - t_{k-1}}{2}, t_k\right), \quad k = 1, \ldots, m, \\ \frac{2\hat{x}(t_m-)}{t_m - t_{m-1}}, & t = T, \end{cases} \tag{9}$$

with $y(t)$ derived from the constraints (1). For the other direction, we assume that $x(t), y(t)$ is a feasible solution for CDNFP and we define $\hat{x}, \hat{y}$ by

$$\hat{x}(t_{k-1}+) = \int_{t_{k-1}}^{\frac{t_{k-1}+t_k}{2}} x(t)dt, \quad k = 1, \ldots, m, \tag{10}$$

$$\hat{x}(t_k-) = \int_{\frac{t_{k-1}+t_k}{2}}^{t_k} x(t)dt, \quad k = 1, \ldots, m, \tag{11}$$

$$\hat{y}(t_k) = y(t_k), \quad k = 1, \ldots, m, \tag{12}$$

$$\hat{y}\left(\frac{t_k + t_{k-1}}{2}\right) = y\left(\frac{t_k + t_{k-1}}{2}\right), \quad k = 1, \ldots, m. \tag{13}$$

It then is not hard to check that $\hat{x}, \hat{y}$ is feasible for $AP(P)$. □

Lemma 2 motivates the following definition.

Definition 2.4. *Let $P = \{t_0, \ldots, t_m\} \in \mathcal{P}$ and $\hat{x}, \hat{y}$ be a feasible solution for $AP(P)$. We will refer to the feasible solution $x(t), y(t)$ defined by (9) as the natural solution for $AP(P)$ (constructed from $\hat{x}, \hat{y}$). Similarly, assume that $x(t), y(t)$ is a feasible solution for CDNFP in which $x(t)$ is piecewise constant with breakpoints in P. We will refer to the feasible solution $\hat{x}, \hat{y}$ defined by (10)-(13) as the natural solution for $AP(P)$ (constructed from $x(t), y(t)$).*

Theorem 2.5. *For any partition $P \in \mathcal{P}$, we have $V[AP(P)] \leq V[CDNFP]$.*

Proof. The dual problem $AP^*(P)$ for $AP(P)$ can be written as

$$\begin{aligned} \max \ & \sum_{k=1}^{m} \left(\bar{r}(t_k) - r\left(\frac{t_k + t_{k-1}}{2}\right)\right)^T (\hat{\pi}(t_{k-1}+) + \hat{\pi}(t_k-)) \\ & - \sum_{k=1}^{m} \frac{t_k - t_{k-1}}{2} a(t_k-)^T (\hat{\rho}(t_{k-1}+) + \hat{\rho}(t_k-)) \\ & + \sum_{k=1}^{m} \left(b(t_k)^T \hat{\eta}(t_k) + b\left(\frac{t_k + t_{k-1}}{2}\right)^T \hat{\eta}\left(\frac{t_k + t_{k-1}}{2}\right)\right) \\ \text{s.t. } & \hat{\pi}_i(t_k-) - \sum_{t' : t' - \lambda_{i,j}(t'-) = t_k} \hat{\pi}_j(t'-) - \hat{\rho}_{i,j}(t_k-) \leq c_{i,j}(t_k-), \ k = 1, \ldots, m, \ (i,j) \in A, \\ & \hat{\pi}_i(t_{k-1}+) - \sum_{t' : t' - \lambda_{i,j}(t'+) = t_{k-1}} \hat{\pi}_j(t'+) - \hat{\rho}_{i,j}(t_{k-1}+) \leq c_{i,j}(t_{k-1}+), \ k = 1, \ldots, m, \ (i,j) \in A, \\ & \hat{\pi}(t_{k-1}+) - \hat{\pi}(t_k) + \hat{\eta}\left(\frac{t_k + t_{k-1}}{2}\right) \leq (t_k - t_{k-1}) d(t_{k-1}), \quad k = 1, \ldots, m, \\ & \hat{\pi}(t_k-) - \hat{\pi}(t_k) + \hat{\eta}(t_k) \leq 0, \quad k = 1, \ldots, m-1, \\ & \hat{\pi}(t_m-) + \hat{\eta}(t_m) \leq 0, \\ & \hat{\rho}(t_{k-1}+), \ \hat{\rho}(t_k-) \geq 0, \quad k = 1, \ldots, m, \\ & \hat{\eta}(t_k), \ \hat{\eta}\left(\frac{t_k + t_{k-1}}{2}\right) \leq 0, \quad k = 1, \ldots, m. \end{aligned}$$

Now suppose that $\hat{x}, \hat{y}$ is an optimum solution for $AP(P)$. By the strong duality theorem of linear programming, there is some $\hat{\pi}, \hat{\rho}, \hat{\eta}$ that solves $AP^*(P)$ with

$$V[AP(P), \hat{x}, \hat{y}] = V[AP^*(P), \hat{\pi}, \hat{\rho}, \hat{\eta}]. \tag{14}$$

The proof is completed by showing that the optimum value of $AP^*(P)$ is a lower bound on the optimum value of $CDNFP^*$. Let

$$\pi(t) = \begin{cases} \hat{\pi}(t+), & t = t_0, t_1, \dots, t_{m-1}, \\ 0, & t = T, \\ \left(\frac{t_k - t}{t_k - t_{k-1}}\right) \hat{\pi}(t_{k-1}) + \left(\frac{t - t_{k-1}}{t_k - t_{k-1}}\right) \hat{\pi}(t_k), & t \in (t_{k-1}, t_k),\ k = 1, \dots, m, \end{cases} \tag{15}$$

$$\rho(t) = \begin{cases} \hat{\rho}(t+), & t = t_0, t_1, \dots, t_{m-1}, \\ 0, & t = T, \\ \left(\frac{t_k - t}{t_k - t_{k-1}}\right) \hat{\rho}(t_{k-1}) + \left(\frac{t - t_{k-1}}{t_k - t_{k-1}}\right) \hat{\rho}(t_k), & t \in (t_{k-1}, t_k),\ k = 1, \dots, m, \end{cases} \tag{16}$$

$$\eta(t) = \begin{cases} \sum_{l=k+1}^{m} \left\{ \hat{\eta}\left(\frac{t_k - t_{k-1}}{2}\right) + \hat{\eta}(t_l) \right\}, & t = t_k, k = 0, \dots, m-1, \\ 0, & t = T, \\ \left(\frac{t_k - t}{t_k - t_{k-1}}\right) \hat{\eta}(t_{k-1}) + \left(\frac{t - t_{k-1}}{t_k - t_{k-1}}\right) \hat{\eta}(t_k), & t \in (t_{k-1}, t_k),\ k = 1, \dots, m. \end{cases} \tag{17}$$

It is not difficult to check that $\hat{\pi}, \hat{\rho}, \hat{\eta}$ is a feasible solution for $AP^*(P)$. Moreover, it can be shown that the objective function value of this solution is equal to $V[AP(P)]$ by a similar argument as in the first part of the proof of Theorem 1 in [20]. Hence,

$$V[AP^*(P), \hat{\pi}, \hat{\rho}, \hat{\eta}] \leq V[CDNFP^*]. \tag{18}$$

The result now follows from (14), (18) and Lemma 1. $\square$

Combining Corollary 2.3 and Theorem 2.5, we obtain the following important result.

Corollary 2. *For any two partitions P and Q in $\mathcal{P}$, we have*

$$V[AP(Q)] \leq V[CDNFP^*] \leq V[CDNFP] \leq V[DP(P)].$$

2.3 Absence of a Duality Gap

A crucial property of finite-dimensional linear programming problems is that the objective value of the given problem is equal to the objective function of the dual problem. But this is not always the case for continuous-time linear programming problems (see [4]). If the problem and its dual have the same objective function value, then it said that there is *no duality gap* between the problem and its dual. Here we show that there is no duality gap between CDNFP and CDNFP*.

Given a partition $P = \{t_0, t_1, \dots, t_m\} \in \mathcal{P}$ and an optimum solution $\hat{x}, \hat{y}$ for $AP(P)$. Let $x(t), y(t)$ be the corresponding natural solution for CDNFP constructed from $\hat{x}, \hat{y}$. We define

$$\alpha[\hat{x}, \hat{y}] := \alpha[\hat{x}] + \alpha[\hat{y}], \tag{19}$$

where

$$\alpha[\hat{x}] := \int_0^T c(t)^T x(t) dt - \sum_{k=1}^{m} \left\{ c(t_{k-1}+)^T \hat{x}(t_{k-1}+) + c(t_k)^T \hat{x}(t_k-) \right\},$$

$$\alpha[\hat{y}] := \int_0^T d(t)^T y(t) dt - \sum_{k=1}^{m} (t_k - t_{k-1}) d(t_{k-1}+)^T \hat{y}\left(\frac{t_k - t_{k-1}}{2}\right).$$

The value $\alpha[\hat{x}, \hat{y}]$ gives the difference in the objective function values yielded by $x(t), y(t)$ for CDNFP and $\hat{x}, \hat{y}$ for $AP(P)$. By Theorem 2.5, we know that $\alpha[x, y] \geq 0$, and $\alpha[\hat{x}, \hat{y}] = 0$ implies that $x(t), y(t)$ is optimum for CDNFP. Now we give some properties of $\alpha[\hat{x}, \hat{y}]$.

Lemma 3. *The values of $\alpha[\hat{x}]$ and $\alpha[\hat{y}]$ can be computed by the following formulas:*

$$\alpha[\hat{x}] = \sum_{k=1}^{m} \left(\frac{t_k - t_{k-1}}{4}\right) \dot{c}(t_k-)^T \left\{\hat{x}(t_{k-1}+) - \hat{x}(t_k-)\right\},$$
$$\alpha[\hat{y}] = \sum_{k=1}^{m} \left(\frac{t_k - t_{k-1}}{4}\right) d(t_{k-1}+)^T \left\{\hat{y}(t_{k-1}) - 2\hat{y}\left(\frac{t_k + t_{k-1}}{2}\right) + \hat{y}(t_k)\right\}.$$

Proof. The result follows from simple algebra. □

Lemma 4. *There is a constant $\mathcal{K}$ such that for any partition $P \in \mathcal{P}$, we have*

$$\alpha[\hat{x}_P, \hat{y}_P] \leq ||P||\mathcal{K},$$

where $(\hat{x}_P, \hat{y}_P)$ is an optimum solution for $AP(P)$.

Proof. Assume that $||\dot{c}(t)||_2 \leq C, ||d(t)||_2 \leq D$, and $||a(t)||_2 \leq M$ for $t \in [0,T]$. Let $P = \{t_0, t_1, \ldots, t_m\}$ be an arbitrary partition in $\mathcal{P}$. We then have

$$\begin{aligned}\alpha[\hat{x}_P] &= \sum_{k=1}^{m} \left(\frac{t_k - t_{k-1}}{4}\right) \dot{c}(t_k-)^T \left\{\hat{x}(t_{k-1}+) - \hat{x}(t_k-)\right\}\\ &\leq \frac{MC}{4} \sum_{k=1}^{m} (t_k - t_{k-1})^2 \leq \frac{ACT}{4}||P||,\end{aligned}$$

and

$$\begin{aligned}\alpha[\hat{y}_P] &= \sum_{i\in N}\sum_{k=1}^{m} \left(\frac{t_k - t_{k-1}}{4}\right) d_i(t_{k-1}+) \Bigg\{ \sum_{j:(i,j)\in A} (\hat{x}_{i,j}(t_{k-1}+) - \hat{x}_{i,j}(t_k-))\\ &\qquad + \sum_{j:(j,i)\in A} \Bigg(\sum_{t:t+\lambda_{j,i}(t)=t_k} \hat{x}_{j,i}(t-) - \sum_{t:t+\lambda_{j,i}(t)=t_{k-1}} \hat{x}_{j,i}(t+) \Bigg) \Bigg\}\\ &\leq D \sum_{k=1}^{m} \left(\frac{t_k - t_{k-1}}{2}\right) \Bigg\{ \sum_{(i,j)\in A} (\hat{x}_{i,j}(t_{k-1}+) + \hat{x}_{i,j}(t_k-)) \Bigg\}\\ &\leq \frac{|A|MD}{2} \sum_{k=1}^{m} (t_k - t_{k-1})^2 \leq \frac{|A|MDT}{2}||P||,\end{aligned}$$

The result now follows by setting $\mathcal{K} = \frac{1}{4}MT(C + 2|A|D)$. □

Corollary 3. *Let $\{P_n\}_{n=1}^{\infty}$ be any sequence of partitions in $\mathcal{P}$ such that $\lim_{n\to\infty} ||P_n|| = 0$, and $\hat{x}_n, \hat{y}_n$ be an optimum solution for $AP(P_n)$. Then*

$$\lim_{n\to\infty} \alpha[\hat{x}_n, \hat{y}_n] = 0.$$

Corollary 3 implies that the optimum values of discrete approximations $DP(\bar{P}_n)$ and $AP(P_n)$ close up to the same value as the norm of sequence $\{P_n\}$ tends to zero value. This leads to the following theorem.

Theorem 2.6 (No duality gap)**.** *If CDNFP has an optimum solution, then there is no duality gap between CDNFP and CDNFP*, i.e. $V[CDNFP] = V[CDNFP^*]$.*

Proof. Let P be an arbitrary partition in $\mathcal{P}$ and $x(t), y(t)$ be an optimum solution for CDNFP. By Lemma 2, this solution can be turned into a feasible solution for $AP(P)$, and as a consequence for $DP(\bar{P})$. Furthermore, by Corollary 2.3, the optimum value of CDNFP is a lower bound on the optimum value of $DP(\bar{P})$. Thus $DP(\bar{P})$ has an optimum solution. On the other hand, the optimum value of CDNFP is an upper bound on the optimum value of the maximization problem $AP^*(P)$. Since $AP(P)$ is feasible and the objective function value of its dual is bounded, by theory of duality in linear programming, both $AP(P)$ and $AP^*(P)$ have optimum solutions.

Now let $\{P_n\}_{n=1}^{\infty}$ be any sequence of partitions in $\mathcal{P}$ with $\lim_{n\to\infty} ||P_n|| = 0$. From the above argument, all three problems $DP(\bar{P}_n)$, $AP(P_n)$ and $AP^*(P_n)$ have optimum solutions for any n. Let $\hat{x}_n, \hat{y}_n$, $\bar{\hat{x}}_n, \bar{\hat{y}}_n$, and $\hat{\pi}_n, \hat{\rho}_n, \hat{\eta}_n$ represent optimum solutions for $DP(\bar{P}_n)$, $AP(P_n)$ and $AP^*(P_n)$, respectively, and

$$a_n = V[DP(\bar{P}_n), \hat{x}_n, \hat{y}_n], \quad b_n = V[AP^*(P_n), \hat{\pi}_n, \hat{\rho}_n, \hat{\eta}_n], \quad \text{for } n = 1, 2, \ldots.$$

By Corollary 2 and the fact that $\alpha[\bar{\hat{x}}_n, \bar{\hat{y}}_n]$, given by (19), is an upper bound on the difference between a_n and b_n, we have

$$0 \le b_n - a_n \le \alpha[\bar{\hat{x}}_n, \bar{\hat{y}}_n], \quad \text{for } n = 1, 2, \ldots. \tag{20}$$

Corollary 3 and relation (20) imply that

$$\lim_{n\to\infty} (b_n - a_n) = 0.$$

We also have $b_n - a_m \ge 0$ for all m and n, by Corollary 2. Thus, both $\lim_{n\to\infty} a_n$ and $\lim_{n\to\infty} b_n$ exist (see Lemma 3.8 in [27]) and

$$\lim_{n\to\infty} a_n = \lim_{n\to\infty} b_n.$$

On the other hand, by Corollary 2, we have

$$V[AP^*(P_n)] \le V[CDNFP^*] \le V[CDNFP] \le V[DP(\bar{P}_n)], \quad \text{for } n = 1, 2, \ldots.$$

We can thus deduce

$$\begin{aligned}
\lim_{n\to\infty} V[AP^*(P_n)] &= \lim_{n\to\infty} \left\{ -\int_0^T \bar{r}(t)^T d\pi_n(t) - \int_0^T a(t)^T \rho_n(t) dt - \int_0^T b(t)^T d\eta_n(t) \right\} \\
&= V[CDNFP^*] = V[CDNFP] \\
&= \lim_{n\to\infty} \int_0^T \left(c(t)^T x_n(t) + d(t)^T y_n(t) \right) dt \\
&= \lim_{n\to\infty} V[DP(\bar{P}_n)],
\end{aligned}$$

where $x_n(t), y_n(t)$ is obtained from $\hat{x}_n, \hat{y}_n$ by (7)-(8), and $\pi_n(t), \rho_n(t), \eta_n(t)$ from $\hat{\pi}_n, \hat{\rho}_n, \hat{\eta}_n$ by (15)-(17). This shows the desired result. $\square$

3 Discretization-based Algorithms for CDNFP

So far we have introduced two different discretizations $AP(P)$ and $VDP(P)$ of CDNFP with respect to a given valid partition P of $[0, T]$. The two problems $AP(P)$ and $VDP(P)$ yield a lower and an upper bound, respectively, on the optimum solution value. Moreover, we have shown that these bounds converge to the optimum value of CDNFP as the partition P becomes finer. This motivates developing a class of algorithms for solving CDNFP. The typical method is the *Uniform Discretization (UD) Algorithm*, which starts with a valid partition P of equal size intervals. With respect to this fixed partition, $DP(P)$ and $AP(P)$ are both solved to yield an upper bound and a lower bound on the optimum value of CDNFP, respectively. Thus, by solving these two discretizations, one gets an estimate of the duality gap. If the gap is not small enough, the number of breakpoints is doubled by introducing new bleakpoints at the midpoints of the current partition.

A serious drawback of the algorithm outlined above is that the time interval is arbitrarily divided and no information from the solutions of discrete approximations $AP(P)$ and $VDP(P)$ is used to fine the partition P. Moreover, a straightforward implementation of this algorithm soon becomes impractical since the number of breakpoints grows exponentially. In this section we present two others algorithms which solve both $DP(P)$ and $AP(P)$ on successively finer (nonuniform) discretizations, but in contrast to the uniform discretization approach, they use the information from the current solution for $AP(P)$ in order to insert new breakpoints at favorable places to improve the solution for CDNFP.

3.1 The Descent Algorithm

Here we show that the algorithm developed by Pullan [27] for SCLP can be extended to CDNFP. This algorithm that we call the *Descent Algorithm* moves between feasible solutions of CDNFP and successively improves the objective function value until an optimum solution is obtained. The main task is then to construct an improved feasible solution from a given (non-optimum) feasible solution.

Let $x(t), y(t)$ be a feasible solution for CDNFP in which $x(t)$ is piecewise constant with respect to some partition $P = \{t_0, t_1, \dots, t_m\} \in \mathcal{P}$. Suppose that $\hat{x}, \hat{y}$ is the corresponding natural solution for $AP(P)$. By Theorem 2.5, if $\hat{x}, \hat{y}$ is optimum for $AP(P)$, then $x(t), y(t)$ is also optimum for CDNFP. Otherwise there exists a feasible solution $\tilde{\hat{x}}, \tilde{\hat{y}}$ for $AP(P)$ with strictly improved objective function value, i.e.,

$$\delta[x(t), y(t)] := V[AP(P), \tilde{\hat{x}}, \tilde{\hat{y}}] - V[AP(P), \hat{x}, \hat{y}] < 0. \tag{21}$$

We can write

$$\delta[x(t), y(t)] = \delta[x(t)] + \delta[y(t)],$$

where

$$\delta[x(t)] := \sum_{k=1}^{m} \Big(c(t_{k-1}+)^T \{\tilde{\hat{x}}(t_{k-1}+) - \hat{x}(t_{k-1}+)\} + c(t_k)^T \{\tilde{\hat{x}}(t_k-) - \hat{x}(t_k-)\} \Big),$$

$$\delta[y(t)] := \sum_{k=1}^{m} (t_k - t_{k-1}) d(t_{k-1}+)^T \left\{ \tilde{\hat{y}} \left(\frac{t_k + t_{k-1}}{2} \right) - \hat{y} \left(\frac{t_k + t_{k-1}}{2} \right) \right\}.$$

The problem that we address here is how to construct a new feasible solution for CDNFP that has a better objective function value than the current solution $x(t), y(t)$. To end this, the first step is to construct a new partition P^ϵ of $[0, T]$ by adding two new breakpoints $t_{k-1} + \epsilon_k$ and $t_k - \epsilon_k$ in each interval $[t_{k-1}, t_k]$, where $\epsilon_k = \frac{(t_{k-1} - t_k)\epsilon}{2}$ and ϵ is a fixed value in $[0, 1]$. Formally, we define P^ϵ as

$$P^\epsilon = \{t_0, t_0 + \epsilon_1, t_1 - \epsilon_1, t_1 + \epsilon_2, \dots, t_k - \epsilon_k, t_k, t_k + \epsilon_{k+1}, \dots, t_m - \epsilon_m, t_m\}.$$

For the general case, partition P^ϵ may not be a valid partition and we require to make the following assumption.

Assumption 2. *For each arc $(i, j) \in A$, the transit time $\lambda_{i,j}(t)$ is constant and will be denoted by $\lambda_{i,j}$.*

The above assumption is assumed to hold throughout the rest of this subsection (notice that our algorithm presented in the next section does not rely on this assumption).

Lemma 5. *If $P \in \mathcal{P}$ and Assumption 2 holds, then $P^\epsilon \in \mathcal{P}$.*

Proof. Since P contains the set of breakpoints of the problem data, so does P^ϵ. Let t_{k-1} and t_k be two consecutive breakpoints in P and (i, j) be an arbitrary arc in A. Then, P^ϵ includes breakpoints $t_{k-1}, t_{k-1} + \epsilon_k, t_k - \epsilon_k$ and t_k. Since $P \in \mathcal{P}$ and $P \subseteq P^\epsilon$, if $t_{k-1} + \lambda_{i,j} \leq T$ and $0 \leq t_k - \lambda_{i,j}$, then $t_{k-1} + \lambda_{i,j} \in P^\epsilon$, and $t_k - \lambda_{i,j} \in P^\epsilon$, respectively. Now assume that $t_{k-1} + \epsilon_k + \lambda_{i,j} \leq T$. We can conclude that $t_{k-1} + \lambda_{i,j}$ and $t_k + \lambda_{i,j}$ are two consecutive breakpoints in P. Hence,

$$t_{k-1} + \lambda_{i,j} + \frac{(t_k + \lambda_{i,j} - t_{k-1} - \lambda_{i,j})\epsilon}{2} = t_{k-1} + \epsilon_k + \lambda_{i,j}$$

is a member of P^ϵ. Similarly, we can show that $t_k - \epsilon_k - \lambda_{i,j}$ is a member of P^ϵ, if $0 \leq t_k - \epsilon_k - \lambda_{i,j}$, which establishes the lemma. □

The next step is to construct a solution $\bar{x}^\epsilon(t), \bar{y}^\epsilon(t)$ for CDNFP with breakpoints in P^ϵ. This can be done by patching together the current feasible solution $x(t), y(t)$ and the natural solution $\tilde{x}(t), \tilde{y}(t)$ constructed from $\tilde{\hat{x}}, \tilde{\hat{y}}$. Specifically, we define $\bar{x}^\epsilon(t), \bar{y}^\epsilon(t)$ by

$$\bar{x}^\epsilon(t) = \begin{cases} \tilde{x}(t), & t \in [t_{k-1}, t_{k-1} + \epsilon_i) \cup [t_k - \epsilon_i, t_k),\ k = 1, \dots, m, \\ x(t), & \text{otherwise,} \end{cases}$$

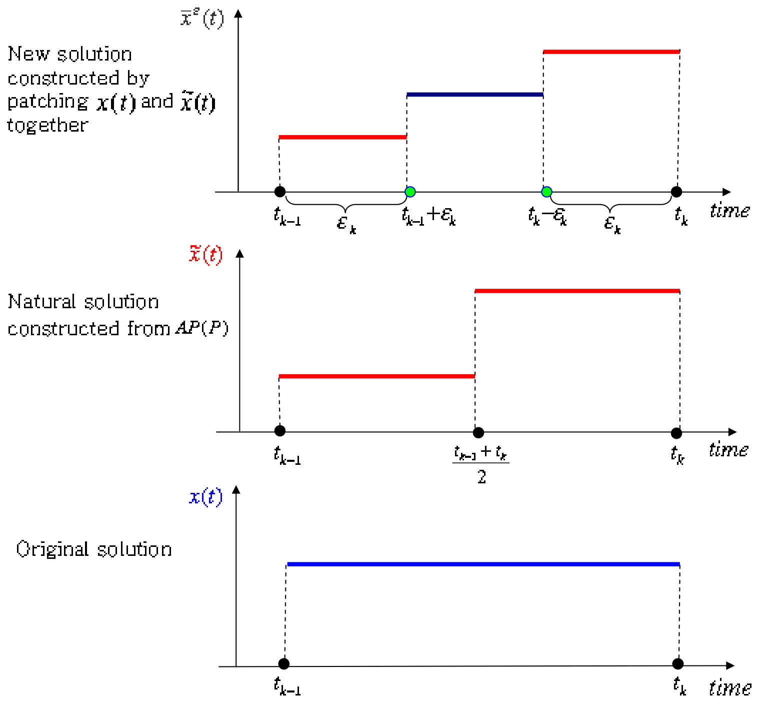

Figure 1: Construction of new solution $\bar{x}^{\epsilon}(t)$ for interval $[t_{k-1}, t_k]$.

with $y^{\epsilon}(t)$ derived from the flow conservation constraints (1). The construction of $\bar{x}^{\epsilon}$ for interval $[t_{k-1}, t_k]$ is illustrated in Figure 1. As with [27], it is referred to as a "patching" together process. We have the following result.

Lemma 6. *$\bar{x}^{\epsilon}(t), \bar{y}^{\epsilon}(t)$ is a feasible solution for CDNFP.*

Proof. The feasibility of $\bar{x}^{\epsilon}(t), \bar{y}^{\epsilon}(t)$ follows from Lemma 5 and the fact that $\bar{y}^{\epsilon}$ is piecewise linear with respect to the partition P^{ϵ}. If fact, $\bar{y}^{\epsilon}$ is as the following:

$$\begin{aligned}
\bar{y}^{\epsilon}(t_{k-1}) &= (1-\epsilon)y(t_{k-1}) + \epsilon\tilde{y}(t_{k-1}), \quad k = 1, \ldots, m,\\
\bar{y}^{\epsilon}(t_{k-1}+\epsilon_i) &= (1-\epsilon)y(t_{k-1}) + \epsilon\tilde{y}\left(\frac{t_k + t_{k-1}}{2}\right), \quad k = 1, \ldots, m,\\
\bar{y}^{\epsilon}(t_k-\epsilon_i) &= (1-\epsilon)y(t_k) + \epsilon\tilde{y}\left(\frac{t_k + t_{k-1}}{2}\right), \quad k = 1, \ldots, m.
\end{aligned}$$

□

We can obtain not only a new feasible solution for CDNFP, but also an improved solution if ϵ is appropriately chosen. This is a direct consequence of the following theorem.

Theorem 3.1. *We have*

$$V\left[CDNFP, \bar{x}^{\epsilon}(t), \bar{y}^{\epsilon}(t)\right] - V[CDNFP, x(t), y(t)] \quad = \quad \epsilon(\delta + \epsilon\alpha),$$

where $\alpha = \alpha\left[\tilde{\hat{x}}, \tilde{\hat{y}}\right]$ *and* $\delta = \delta[x(t), y(t)]$ *are given by (19) and (21), respectively.*

Proof. It can be shown by a simple calculation (see [21] for details) that

$$\int_0^T c(t)^T \bar{x}^{\epsilon}(t)dt - \int_0^T c(t)^T \bar{x}^{\epsilon}(t)dt = \epsilon\left(\delta[x(t)] + \epsilon\alpha\left[\tilde{\hat{x}}\right]\right),$$
$$\int_0^T d(t)^T \bar{y}^{\epsilon}(t)dt - \int_0^T d(t)^T y(t)dt = \epsilon\left(\delta[y(t)] + \epsilon\alpha\left[\tilde{\hat{y}}\right]\right).$$

The lemma now easily follows. □

We can now conclude the following:

Corollary 4. *For* ϵ *small enough,* $\bar{x}^{\epsilon}(t), \bar{y}^{\epsilon}(t)$ *has a strictly smaller objective function value than* $x(t), y(t)$. *Moreover,*

$$\min_{\epsilon} V[CDNFP, \bar{x}^{\epsilon}(t), \bar{y}^{\epsilon}(t)] - V[CDNFP, x(t), y(t)] = \begin{cases} -\frac{\delta^2}{4\alpha}, & \alpha > 0 \text{ and } \delta < -2\alpha, \\ \delta + \alpha, & \textit{otherwise,} \end{cases}$$

which occurs at

$$\epsilon^* = \begin{cases} -\frac{\delta}{2\alpha}, & \alpha > 0 \text{ and } \delta < -2\alpha, \\ 1, & \textit{otherwise.} \end{cases} \tag{22}$$

We will refer to patching $x(t), y(t)$ and $\tilde{x}(t), \tilde{y}(t)$ together with $\epsilon = \epsilon^*$, given by (22), as the patching together optimality.

Theorem 3.2. *Let* $x(t), y(t)$ *be an optimum solution for CDNFP in which* $x(t)$ *is piecewise constant with respect to some partition* $P \in \mathcal{P}$. *Then the natural solution* $\hat{x}, \hat{y}$ *is optimum for* $AP(P)$.

Proof. If $\hat{x}, \hat{y}$ is not optimum, then by patching together optimality process, we can construct a feasible solution for CDNFP with strictly smaller objective function value $x(t), y(t)$. □

Summarizing the above argument, we can establish a strong duality result and present an algorithm for solving CDNFP.

Theorem 3.3 (Strong duality)**.** *Suppose that CDNFP has an optimum solution* $x(t), y(t)$, *in which* $x(t)$ *is piecewise constant with respect to some* $P \in \mathcal{P}$. *Then there exists a piecewise linear optimum solution* $\pi(t), \rho(t), \eta(t)$ *with breakpoints in* P *with*

$$V[CDNFP^*, \pi(t), \rho(t), \eta(t)] = V[CDNFP, x(t), y(t)].$$

Proof. By Theorem 3.2, the natural solution $\hat{x}, \hat{y}$ constructed from $x(t), y(t)$ is optimum for $AP(P)$. Thus by the strong duality theorem for linear programming, there exist an optimum solution $\hat{\pi}, \hat{\rho}, \hat{\eta}$ for $AP^*(P)$ with the same objective function value. By Theorem 2.5, this solution yields a piecewise linear solution $\pi(t), \rho(t), \eta(t)$ with breakpoints in P and the same objective function value. The result now follows from Corollary 2. □

The formal description of the algorithm for solving CDNFP is now given as follows:

Descent Algorithm

Step 0 Let $P_1 \in \mathcal{P}$ be an initial partition. Construct an initial solution $x_0(t), y_0(t)$ for CDNFP by constructing a feasible solution for $DP(P_1)$. If $DP(P_1)$ is infeasible, then so is CDNFP (by Lemma 2). Set $n = 1$.

Step 1 Let $\hat{x}_{n-1}, \hat{y}_{n-1}$ be the natural solution for $AP(P_n)$ constructed from $x_{n-1}(t), y_{n-1}(t)$. If $\hat{x}_{n-1}, \hat{y}_{n-1}$ is optimum for $AP(P_n)$, then stop as $x_{n-1}(t), y_{n-1}(t)$ is optimum for CDNFP (by Corollary 2).

Step 2 Optimize $AP(P_n)$ to produce $\tilde{\tilde{x}}_n, \tilde{\tilde{y}}_n$. Let $\tilde{x}_n(t), \tilde{y}_n(t)$ be the natural solution for CDNFP.

Step 3 Patch $\tilde{x}_n(t), \tilde{y}_n(t)$ and $x_{n-1}(t), y_{n-1}(t)$ together optimality to produce the solution $\bar{x}_n^\epsilon(t), \bar{y}_n^\epsilon(t)$. Let P_{n+1} be the constructed partition from the patching optimality together process.

Step 4 Optimize $DP(P_{n+1})$ to generate a solution $x_n(t), y_n(t)$ to CDNFP. Set $n = n + 1$ and return to Step 1.

The convergence of the Descent Algorithm follows by Lemma 4 and a similar argument as in [32]. We thus have the following theorem.

Theorem 3.4. *Either* Descent Algorithm *terminates at the optimum solution for CDNFP or $V[DP\,(P_n)]$ and $V[AP(P_n)]$ both converge to $V[CDNFP]$.*

3.2 The Adaptive Discretization Algorithm

The current form of the Descent Algorithm presented above may be not practical because at each iteration of the algorithm the number of breakpoints is increased by a factor of 2 or 3. Here we present another algorithm based on the ideas of Philpott and Craddock [26]. We first need to give a definition.

Definition 3.5. *Given a valid partition $P = \{t_0, t_1, \ldots, t_m\}$, we refer to a member of $N \times \{t_0, t_1, \ldots, t_{m-1}\}$ as a* node-time pair *(NTP). We say that NTP (i, α) is* connected *to NTP (j, β) if there is a sequence of distinct NTPs as*

$$(j, \alpha) = (i_1, t'_1), (i_2, t'_2), \ldots, (i_q, t'_q) = (j, \beta),$$

in which for each $k = 1, \ldots, q-1$ either $(i_k, i_{k+1}) \in A$ and $t'_{k+1} = t_k + \lambda_{i_k, i_{k+1}}$, or $i_k = i_{k+1}$ and $t'_k \le t'_{k+1}$.

If we assume that every NTP is connected to itself, then connectedness is an equivalence relation, and as such partitions $N \times \{t_0, t_1, \ldots, t_{m-1}\}$ into disjoint subsets, denoted by $NTP_1, NTP_2, \ldots, NTP_p$. Now assume that $\hat{x}, \hat{y}$ is an optimum solution for $AP(P)$. For each NTP_ℓ, $\ell = 1, \ldots, p$, we define

$$\begin{aligned} \alpha_\ell[\hat{x}, \hat{y}] := & \sum_{(i,t_{k-1}) \in NTP_\ell} \sum_{j:(i,j)\in A} \left(\frac{t_k - t_{k-1}}{4}\right) \dot{c}_{i,j}(t_k -) \left\{\hat{x}_{i,j}(t_{k-1}+) - \hat{x}_{i,j}(t_k -)\right\} \\ & + \sum_{(i,t_{k-1}) \in NTP_\ell} \left(\frac{t_k - t_{k-1}}{4}\right) d_i(t_{k-1}+) \left\{\hat{y}_i(t_{k-1}) - 2\hat{y}_i\left(\frac{t_k + t_{k-1}}{2}\right) + \hat{y}_i(t_k)\right\}. \end{aligned} \tag{23}$$

Theorem 3.6. *Let $P \in \mathcal{P}$, $\hat{x}, \hat{y}$ be an optimum solution for $AP(P)$, and NTP_1, NTP_2, $\ldots$, NTP_p be a partition of $N \times \{t_0, t_1, \ldots, t_{m-1}\}$ defined by the connectedness relation. Then*

$$\alpha[\hat{x}, \hat{y}] = \sum_{\ell=1}^{p} \alpha_\ell[\hat{x}, \hat{y}],$$

where $\alpha[\hat{x}, \hat{y}]$ and $\alpha_\ell[\hat{x}, \hat{y}]$ are given by (19) and (23), respectively. Moreover, we have $\alpha_\ell[\hat{x}, \hat{y}] \ge 0$, for every $\ell = 1, 2, \ldots, p$.

Proof. We only prove the second part; the first part is straightforward. Assume that for some q we have $\alpha_q[\hat{x}, \hat{y}] < 0$. We define $\hat{x}^o, \hat{y}^o$ by

$$\begin{aligned} \hat{x}^o_{i,j}(t_{k-1}+) &:= \begin{cases} \frac{1}{2}\left\{\hat{x}_{i,j}(t_{k-1}+) + \hat{x}_{i,j}(t_k -)\right\}, & \text{if } (i, t_{k-1}) \in NTP_\ell, \\ \hat{x}^*_{i,j}(t_{k-1}+), & \text{otherwise,} \end{cases} \\ \hat{x}^o_{i,j}(t_k -) &:= \begin{cases} \frac{1}{2}\left\{\hat{x}_{i,j}(t_{k-1}+) + \hat{x}_{i,j}(t_k -)\right\}, & \text{if } (i, t_{k-1}) \in NTP_\ell, \\ \hat{x}^*_{i,j}(t_k -), & \text{otherwise,} \end{cases} \\ \hat{y}^o_i\left(\frac{t_{k-1} + t_k}{2}\right) &:= \begin{cases} \frac{1}{2}\left\{\hat{y}_i(t_{k-1}) + \hat{y}_i(t_k)\right\}, & \text{if } (i, t_{k-1}) \in NTP_\ell, \\ \hat{y}_i\left(\frac{t_{k-1}+t_k}{2}\right), & \text{otherwise,} \end{cases} \\ \hat{y}^o_i\,(t_k) &:= \hat{y}(t_k), \end{aligned}$$

where $i \in N$, $(i,j) \in A$ and $k = 1, \ldots, m$. It is easy to check that $\hat{x}^o, \hat{y}^o$ is feasible for $AP(P)$. Moreover, by comparing the objective value of $AP(P)$ for $\hat{x}^o, \hat{y}^o$ and $\hat{x}, \hat{y}$, we obtain

$$V[AP(P), \hat{x}^o, \hat{y}^o] - V[AP(P), \hat{x}, \hat{y}] = \alpha_\ell[\hat{x}, \hat{y}] < 0 \ .$$

This contradicts the assumed optimality of $\hat{x}, \hat{y}$. □

Now we present an outline of the *Adaptive Discretization (AD) Algorithm.* In iteration n, starting from an initial valid partition P_n, both $DP(P_n)$ and $AP(P_n)$ are solved to produce $\tilde{\hat{x}}_n, \tilde{\hat{y}}_n$ and $\hat{x}_n, \hat{y}_n$, respectively. Then the duality gap and $\alpha[\hat{x}_n, \hat{y}_n]$ are computed accordingly. If the duality gap is zero, then the solution $x(t), y(t)$ corresponding to $\tilde{\hat{x}}_n, \tilde{\hat{y}}_n$ is an optimum solution to CDNFP and the algorithm terminates. Otherwise, we have $\alpha[\hat{x}_n, \hat{y}_n] > 0$ and, by Theorem 3.6, $\alpha_\ell[\hat{x}_n, \hat{y}_n] > 0$ for some ℓ. We keep ℓ fixed and introduce new breakpoints at the midpoints of those intervals $[t_{k-1}, t_k]$ so that there is some node $i \in N$ with $(i, t_{k-1}) \in NTP_\ell$. Formally, a new valid partition P_{n+1} is defined as follows:

$$P_{n+1} := P \cup \left\{ \left(\frac{t_k + t_{k-1}}{2} \right) : \ (i, t_{k-1}) \in NTP_\ell \text{ for some } i \right\} . \qquad (24)$$

We are now ready to state a formal description of the AD Algorithm.

Adaptive Discretization (AD) Algorithm

Step 0 Let $P_1 \in \mathcal{P}$ be an initial partition. Set $n := 1$.

Step 1 Solve $DP(P_n)$ to produce $\tilde{\hat{x}}_n, \tilde{\hat{y}}_n$.

Step 2 Solve $AP(P_n)$ to give $\hat{x}_n, \hat{y}_n$.

Step 3 Calculate the current duality gap, i.e., $\delta_n := V[DP(P_n)] - V[AP(P_n)]$. If $\delta_n = 0$, then stop as $\tilde{\hat{x}}_n, \tilde{\hat{y}}_n$ yields an optimum solution to CDNFP. Otherwise, construct a new partition P_{n+1} according to (24).

Step 4 Set $n := n + 1$ and return to **Step 1**.

The convergence property of the AD Algorithm is easily established.

Theorem 3.7. *Either AD Algorithm terminates at the optimum solution for CDNFP or $V[DP(P_n)]$ and $V[AP(P_n)]$ both converge to $V(CDNFP)$.*

Proof. By Corollary 2, termination of the AD Algorithm yields an optimum solution for CDNFP. So assume that the AD Algorithm does not terminate. Consider the sequence $\{\delta_n\}$. It is not hard to see that $\{\delta_n\}$ is a decreasing sequence, bounded below by 0. Further, by a similar way as the proof of Lemma 6 in [26], it can be shown that for every $\epsilon > 0$, there is some partition P_n generated by the AD Algorithm so as $\delta_n < \epsilon$. We can now deduce that $\{\delta_n\}$ tends to zero and the result follows immediately. □

Now assume that $x_n(t), y_n(t)$ denotes a solution for CDNFP generated by $\tilde{\hat{x}}_n, \tilde{\hat{y}}_n$ in iteration n of the AD Algorithm. Usually $x_n(t)$ remains constant over some consecutive intervals of P_n such that some breakpoints are redundant. Specifically, a breakpoint t_k in P_n is said to be *redundant* if

$$\frac{\tilde{\hat{x}}_n(t_{k-1}+)}{t_k - t_{k-1}} = \frac{\tilde{\hat{x}}_n(t_k+)}{t_{k+1} - t_k} \ .$$

It is desirable to remove the redundant breakpoints as they increase the size of the subproblems to be solved at each iteration. On the other hand, if redundant breakpoints are removed at every iteration, then the algorithm may cycle and not converge. However, a quicker AD Algorithm which maintains the convergence property is obtained if redundant breakpoints are removed only if a sufficient reduction in δ_n is observed. This leads to the following algorithm:

Adaptive Discretization Algorithm with Removal of Breakpoints (ADR)

Step 0 Let $\delta_1 := \infty, \gamma_1 := \infty$ and $P_1 \in \mathcal{P}$ be an initial partition. Choose θ such that $0 \le \theta < 1$. Set $n := 1$.

Step 1 Solve $DP(P_n)$ to produce $\tilde{\hat{x}}_n, \tilde{\hat{y}}_n$.

Step 2 If $\delta_n < \theta\gamma_n$, then

- **2.1** Remove redundant breakpoints from P_n resulting in a new partition $\hat{P}_n$.
- **2.2** Add required breakpoints to $\hat{P}_n$ to produce a valid partition P_n.
- **2.3** Set $\gamma_{n+1} := \delta_n$.

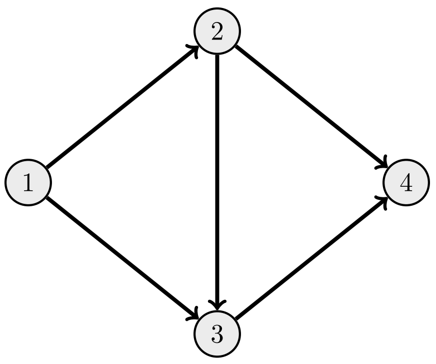

Figure 2: Network for Examples 3.1 and 3.2.

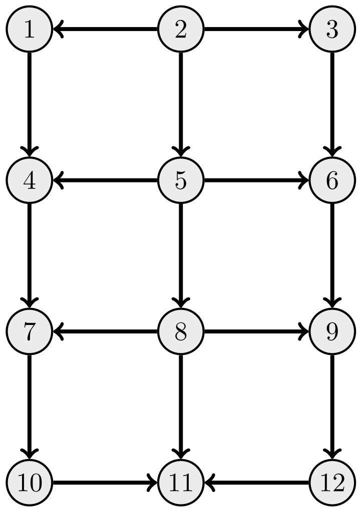

Figure 3: Network for Example 3.3.

otherwise

2.4 Set $\gamma_{n+1} := \gamma_n$.

Step 3 Solve $AP(P_n)$ to give $\hat{x}_n, \hat{y}_n$.

Step 5 Calculate the current duality gap, i.e., $\delta_n := V[DP(P_n)] - V[AP(P_n)]$. If $\delta_n = 0$, then stop as $\tilde{\hat{x}}_n, \tilde{\hat{y}}_n$ yields an optimum solution to CDNFP. Otherwise, construct a new partition P_{n+1} according to (24).

Step 6 Set $n := n + 1$, and return to **Step 1**.

The parameter θ determines the proportion by which the duality gap must be reduced before the removal of redundant breakpoints is allowed. We can now extend Theorem 3.7.

Theorem 3.8. *Either the ADR Algorithm terminates at an optimum solution to CDNFP or* $V[DP\,(P_n)]$ *and* $V[AP(P_n)]$ *both converge to* $V[CDNFP]$.

Proof. If the algorithm terminates, then by Theorem 2 an optimum solution for CDNFP is obtained. We can assume that $0 < \theta < 1$ since for $\theta = 0$ the ADR Algorithm is identical to the AD Algorithm. The sequence $\{\gamma_n\}$ is decreasing, bounded from below by 0, and has the property that for every $\epsilon > 0$ there is some partition P_n such that $\gamma_n < \epsilon$. Thus, sequence $\{\gamma_n\}$ tends to zero and this concludes the proof. □

3.3 Computational Results for Small Example Instances

In this section three simple example instances of CDNFP are solved by using the Uniform-discretization Algorithm, the Descent Algorithm and the ADR Algorithm. We wrote a preliminary version of all algorithms in MATLAB. This program records and updates the partition, generates the time-expanded networks associated to $DP(P)$ and $AP(P)$, and calls the TOMLAB simplex network solver at each iteration. For each algorithm, we give the results of the first five iterations, including optimum values of $DP(P_n)$ and $AP(P_n)$, the duality gap δ_n, the number of breakpoints at partition P_n, denoted by "# BP", and the number of breakpoints after removing redundant ones at optimum solution of $DP(P_n)$, denoted by "# BPR".

Example 3.1. Here, we consider the example designed by Philpott and Craddock [26]. The network has four nodes and five arcs connecting those nodes as shown in Figure 2. The transit costs and transit capacities are as follows:

$$c_{1,2}(t) = 1 + 0.6t, \quad c_{1,3}(t) = 12 - t, \quad c_{2,3}(t) = 1 + 1.4t, \quad c_{2,4}(t) = 20 + t, \quad c_{3,4}(t) = 6 - 0.2t,$$
$$a_{1,2}(t) = 0.6, \quad a_{1,3}(t) = 0.8, \quad a_{2,3}(t) = 0.8, \quad a_{2,4}(t) = 2.2, \quad a_{3,4}(t) = 1.6,$$

Table 1: Computational results for Example 3.1

n	$DP(P_n)$	$AP(P_n)$	δ_n	# BP	# BPR
		UD Algorithm			
1	124.16000	123.36000	0.799999	5	5
2	123.76000	123.52000	0.239999	10	8
3	123.64000	123.60000	0.039999	20	9
4	123.62000	123.61000	0.009999	40	9
5	123.61500	123.61250	0.002499	80	13
		Descent Algorithm			
1	123.76000	123.52000	0.239999	10	8
2	123.62166	123.60777	0.013895	42	14
3	123.61926	123.60849	0.010770	99	36
4	123.61560	123.61120	0.004402	173	74
5	123.61509	123.61165	0.003440	309	126
		ADR Algorithm			
1	124.16000	123.36000	0.799999	5	5
2	123.76000	123.52000	0.239999	10	8
3	123.64000	123.60000	0.040000	15	9
4	123.62000	123.61000	0.010000	15	9
5	123.61500	123.61250	0.002500	20	10

The storage capacities and storage costs are assumed to be infinity and zero, respectively. Moreover, we assign a transit time of 2 to each arc. An initial storage of 8 units must be sent from source node 1 to sink node 4 within the time interval $[0, 10]$.

The optimum solution has value $123.63\bar{3}$ with breakpoints in $\{0, 2, 3.5, 4, \frac{16}{3}, 6, \frac{22}{3}, 8, 10\}$. Computational results are reported in Table 1.

Example 3.2. We consider Example 3.1 with the following modified transit times:

$$\lambda_{1,2} = \lambda_{2,3} = \lambda_{2,4} = \lambda_{3,4} = 1,\ \lambda_{1,3} = 2.$$

The exact optimum solution to this problem is not known. The computational results are shown in Table 2.

Example 3.3. The third example is posed in the network shown in Figure 3. The transit times, transit costs and transit capacities are as follows:

$$\begin{aligned}
&\lambda_{1,4} = \lambda_{3,6} = \lambda_{4,7} = \lambda_{6,9} = \lambda_{7,10} = \lambda_{9,12} = 2.5,\\
&\lambda_{2,1} = \lambda_{2,3} = \lambda_{5,4} = \lambda_{5,6} = \lambda_{8,7} = \lambda_{8,9} = \lambda_{10,11} = \lambda_{12,11} = 5,\\
&\lambda_{2,5} = 2.5,\ \lambda_{5,8} = 5,\ \lambda_{8,11} = 10,
\end{aligned}$$

$$\begin{array}{rrrr}
c_{1,4}(t) = 5 + 2.8t, & c_{2,1}(t) = 7 + 2.4t, & c_{2,3}(t) = 1 + 2.4t, & c_{2,5}(t) = 2 + 2.8t,\\
c_{3,6}(t) = 50 - 1.5t, & c_{4,7}(t) = 72 - 2t, & c_{5,4}(t) = 12 + 1.2t, & c_{5,6}(t) = 9 + 1.6t,\\
c_{5,8}(t) = 3 + 1.6t, & c_{6,9}(t) = 1 + 2.5t, & c_{7,10}(t) = 1 + 2.2t, & c_{8,7}(t) = 9 + 1.2t,\\
c_{9,12}(t) = 4 + 2.6t, & c_{10,11}(t) = 3 + 1.2t, & c_{12,11}(t) = 3 + 1.6t, &
\end{array}$$

$$\begin{aligned}
&a_{1,4}(t) = a_{3,6} = a_{4,7}(t) = a_{6,9} = a_{7,10}(t) = a_{9,12} = 2,\\
&a_{2,1}(t) = a_{2,3} = a_{5,4}(t) = a_{5,6} = a_{8,7}(t) = a_{8,9} = 1,\\
&a_{2,5}(t) = a_{5,8} = a_{8,11}(t) = 1.5,\ a_{10,11} = a_{11,12}(t) = 3.
\end{aligned}$$

Table 2: Computational results for Example 3.2

n	$DP(P_n)$	$AP(P_n)$	δ_n	# BP	# BPR
		UD Algorithm			
1	102.96000	102.78000	0.180000	10	6
2	102.96000	102.87000	0.090000	20	7
3	102.92625	102.91250	0.013750	40	10
4	102.92312	102.91906	0.004062	80	11
5	102.92156	102.91992	0.001640	160	24
		Descent Algorithm			
1	102.96000	102.78000	0.180000	10	6
2	102.93300	102.85100	0.082000	25	9
3	102.92219	102.91400	0.008190	87	22
4	102.92172	102.91200	0.001669	240	64
5	102.92127	102.92040	0.000871	527	161
		ADR Algorithm			
1	102.96000	102.78000	0.180000	10	6
2	102.96000	102.87000	0.090000	20	7
3	102.92625	102.91250	0.013750	40	10
4	102.92312	102.91906	0.004062	30	11
5	102.92156	102.91992	0.001640	60	18

The supplies/demands, storage costs and storage capacities at nodes are as follows:

$$r_1(t) = \begin{cases} 0.5, & t \in [0,5], \\ 0, & t \in (5,30], \end{cases} \qquad r_2(t) = \begin{cases} 0.25, & t \in [0,10], \\ 0, & t \in (10,30], \end{cases}$$

$$r_3(t) = \begin{cases} 0.4, & t \in [0,5], \\ 0, & t \in (5,30], \end{cases} \qquad r_4(t) = \begin{cases} 0.6, & t \in [0,5], \\ 0, & t \in (5,30], \end{cases}$$

$$r_5(t) = \begin{cases} 0.3, & t \in [0,10], \\ 0, & t \in (10,30], \end{cases} \qquad r_6(t) = \begin{cases} 0.4, & t \in [0,5], \\ 0, & t \in (5,30], \end{cases}$$

$$r_7(t) = 0, \qquad r_8(t) = \begin{cases} 1, & t \in [0,10], \\ 0, & t \in (10,30], \end{cases}$$

$$r_9(t) = 0, \qquad r_{10}(t) = 0,$$

$$r_{11}(t) = \begin{cases} 0, & t \in [0,20], \\ -2.5, & t \in (20,30], \end{cases} \qquad r_{12}(t) = 0,$$

$$d_1(t) = 0, \quad i = 1, \dots, 12,$$

$$b_1(t) = b_3(t) = 2, \; = b_2(t) = b_4(t) = b_6(t) = 3,$$

$$b_5(t) = b_8(t) = b_{10}(t) = b_{12}(t) = 5, b_7(t) = b_9(t) = 4, b_{11}(t) = \infty.$$

The computational results are given in Table 3.

To compare the performance of the algorithms, our main focus lies on the growth of the number of breakpoints. This number is the critical parameter for the running time and memory requirement of the considered algorithms. As expected, the ADR Algorithm yields reasonable solutions and is clearly superior to the Uniform Discretization Algorithm and the Descent Algorithm for all three examples since the number of breakpoints grows considerably faster for the latter algorithms. On the other hand, although the Descent Algorithm guarantees to yield a strictly improved solution at each iteration, the

Table 3: Computational results for Example 3.3

n	$DP(P_n)$	$AP(P_n)$	δ_n	# BP	# BPR
UD Algorithm					
1	1225.68750	1225.34375	0.343750	12	8
2	1225.51562	1225.34375	0.171875	24	9
3	1225.42968	1225.34375	0.085937	48	9
4	1225.38671	1225.34375	0.042968	96	9
5	1225.36523	1225.36120	0.004028	192	9
Descent Algorithm					
n	$DP(P_n)$	$AP(P_n)$	δ_n	# BP	# BPR
1	1225.68750	1225.34375	0.343750	12	8
2	1225.51561	1225.35080	0.262323	64	14
3	1225.39080	1225.35080	0.040004	288	103
ADR Algorithm					
1	1225.68750	1225.34375	0.343750	12	8
2	1225.51562	1225.34375	0.171875	24	9
3	1225.42968	1225.34375	0.085937	48	9
4	1225.38671	1225.34375	0.042968	24	9
5	1225.36523	1225.36120	0.004028	48	9

solutions obtained by this algorithm have huge numbers of breakpoints, many more than necessary. Moreover, as the algorithm proceeds, it even produces many more breakpoints and obscures the form of optimum solution, while computational results for show that the structure of an optimum solution is often much simpler. The same serious problems have been already reported by Pullan [27, 33] in solving SCLP. The natural approach is to use a purification algorithm as extreme point solutions usually have a considerably simpler structure than arbitrary feasible solutions and are more meaningful in practice. Such an algorithm can be incorporated into the Descent Algorithm to construct an extreme point solution as the input solution for **Step 1** of the Descent Algorithm at each iteration.

It is worth noting that Anderson and Pullan [11] demonstrate by numerical examples that the use of purification may efficiently overcome the mentioned difficulties in solving SCLP. Moreover, purification is necessary if one is interested in extending the algorithm developed by Pullan [33] for SCLP to CDNFP with piecewise analytic costs. These observations are our motivation to develop a purification algorithm for CDNFP in the next section.

4 Purification for CDNFP

Purification for infinite-dimensional problems has been considered in the general context of linear programming by Lewis [18], for semi-infinite linear programming by Anderson and Lewis [6] and Leon and Vercher [17], and for SCLP by Anderson and Pullan [11]. The purification algorithm that we present later is a network specialization of the purification algorithm developed in [11] for SCLP, but including transit times on arcs and storage capacities at nodes.

4.1 Preliminary Results

The notion of extreme points plays a central role in the theory of linear programming, specifically for the simplex algorithm. Fundamental to this algorithm is that whenever the problem has an optimum solution, then one can be found among the extreme points of the feasible region. This result remains true for CDNFP.

Theorem 4.1. *If the feasible region F is nonempty, then there is an optimum solution for CDNFP at an extreme point of F.*

Proof. The proof is the same as the proofs of the similar results for continuous-time linear programming and SCLP given in [3, 22, 29]. We give an outline of the proof for the sake of completeness. The feasible region F is obviously convex and bounded. Moreover, we can show F is closed in the weak topology $\sigma(L_\infty^{n_1}[0,T], L_1^{n_1}[0,T])$. Then, by Alaoglua's Theorem (see [12]), it follows that F is compact in the $\sigma(L_\infty^{n_1}[0,T], L_1^{n_1}[0,T])$ topology and consequently is a convex hull of its extreme points by Krein-Milman's Theorem (see again [12]). Further, the objective function of CDNFP is $\sigma(L_\infty^{n_1}[0,T], L_1^{n_1}[0,T])$-continuous functional and hence will attain a minimum over F at such an extreme point. $\square$

By Theorem 4.1, we can restrict our attention to the set of all extreme points of the feasible region of CDNFP when looking for an optimum feasible solution and hence it is important to characterize the extreme points of F. In the context of finite-dimensional network flow, extreme points of the feasible region correspond to the flows which do not admit cycles, that is the arcs with flow strictly between their bounds form a forest. A similar characterization of the extreme points for CDNFP has been derived by Anderson [9]. To present this result, we need some concepts and notation.

Definition 4.2.

1. *We use the term* node-time pair *(NTP) to refer a particular node at a particular time, i.e., a member of $N \times [0,T]$.*
2. *We say that the NTP (i,α) is* arc-linked *to the NTP (j,β) if either*
   (a) $(i,j) \in A$ and $\beta = \alpha + \tau_{i,j}$, or
   (b) $(j,i) \in A$ and $\alpha = \beta + \tau_{j,i}$.
3. *We say that the NTP (i,α) is* node-linked *to the NTP (j,β) if $i = j$. In this case, it is assumed that $\alpha \neq \beta$.*
4. *A* continuous-time dynamic path *is defined as a sequence of distinct NTPs as*
$$(i,\alpha) = (i_1,t_1), (i_2,t_2), \ldots, (i_q,t_q) = (j,\beta),$$
with consecutive members either arc or node-linked.
5. *A sequence of distinct NTPs as*
$$(i,\alpha) = (i_1,t_1), (i_2,t_2), \ldots, (i_q,t_q) = (j,\beta),$$
with consecutive members either arc or node-linked is said to be a continuous-time dynamic cycle *if $q \geq 3$, $(i,\alpha) = (j,\beta)$ and all other NTPs are distinct. Hereafter we shall omit the term "continuous-time dynamic" to refer a continuous-time dynamic (cycle) path when the meaning is clear from context.*
6. *A (cycle) path $(i,\alpha) = (i_1,t_1), (i_2,t_2), \ldots, (i_q,t_q) = (j,\beta)$, is said to be an* (arc-cycle) arc-path *if every pair of consecutive NTPs is arc-linked. Such an (arc-cycle) arc-path can be written as*
$$(i,\alpha) = (i_1,\tau_1(\alpha)), (i_2,\tau_2(\alpha)), \ldots, (i_q,\tau_q(\alpha)) = (j,\beta),$$
where $\tau_1(\alpha) = \alpha$ and $\tau_k(\alpha) = \tau_{k-1}(\alpha) + \lambda_{i_{k-1},i_k}$, if $(i_{k-1},i_k) \in A$, and $\tau_k(\alpha) = \tau_{k-1}(\alpha) - \lambda_{i_k,i_{k-1}}$, otherwise, for $k = 2,\ldots,q$. We shall refer to α as the stating time *and to $\tau_k(\alpha)$ as the* arrival time *at node i_k through the arc-path.*

We then need to give the concept of augmenting paths and cycles with respect to a given flow $x(t)$ for CDNFP.

Definition 4.3.

1. *Given a flow over time $x(t)$ with corresponding storage $y(t)$, the* residual capacity *of the path (or cycle) $P : (i_1, t_1), (i_2, t_2), \ldots, (i_q, t_q)$ is defined by*

$$Cap[P] := \min_{1 \le k \le q-1} \delta_k,$$

   where for $k = 1, \ldots, q-1$, δ_k is given by

$$\delta_k = \begin{cases} a_{i_k,i_{k+1}}(t_k) - x_{i_k,i_{k+1}}(t_k), & \text{if } (i_k, i_{k+1}) \in A; \\ x_{i_{k+1},i_k}(t_{k+1}), & \text{if } (i_{k+1}, i_k) \in A; \\ \min\{b_{i_k}(t) - y_{i_k}(t) : \ t_k \le t \le t_{k+1} - 1\}, & i_k = i_k + 1, \ t_k < t_{k+1}; \\ \min\{y_{i_k}(t) : \ t_{k+1} \le t \le t_k - 1\}, & i_k = i_k + 1, \ t_{k+1} < t_k. \end{cases} \tag{25}$$

2. *The path (or cycle) P is called an* augmenting path *(or an* augmenting cycle*) under (or with respect to) $x(t)$ if $Cap[P] > 0$.*

3. *We refer to (cycle) path $P : (i_1, t_1), (i_2, t_2), \ldots, (i_q, t_q)$ as a* bi-augmenting (cycle) path *under $x(t)$ if both path P and its reserve (i.e., $(i_q, t_q), (i_{q-1}, t_{q-1}), \ldots, (i_1, t_1)$) are both augmenting (cycle) paths with respect to $x(t)$. We say that NTP (i, α) is* arc-connected *to NTP (j, β) if there is a bi-augmenting arc-path from (i, α) to (j, β).*

We also need to make the following assumption before give a characterization of extreme points.

Assumption 3. *The transit times $\lambda_{i,j}$ are all constant (time-independent) and rational. Moreover, the time horizon T is rational.*

Theorem 4.4 (Anderson [9])**.** *If $x(t)$ is a feasible solution and Assumption 3 holds, then $x(t)$ is an extreme point of the feasible region for CDNFP if and only if there is no node $i \in N$ and $S \subset [0, T]$ of nonzero-measure such that either*

(1) *for every $\alpha \in S$, (i, α) is part of a bi-augmenting arc-cycle under $x(t)$; or*

(2) *for every $\alpha \in S$, (i, α) is arc-connected to some (j, β) under $x(t)$, and moreover $0 < y_i(\alpha) < b_i(\alpha)$ and $0 < y_j(\beta) < b_j(\beta)$.*

The purification algorithm that we describe below requires the problem data to be piecewise analytic functions. In particular, we make the following assumption on the nature of the problem data.

Assumption 4. *The functions $c(t), d(t), r(t), a(t)$, and $b(t)$ are all piecewise analytic on $[0, T]$.*

This assumption guarantees the existence of an optimum solution which is piecewise analytic on $[0, T]$.

Theorem 4.5 (Pullan [31])**.** *If F is nonempty and Assumptions 3 and 4 hold, then CDNFP has a piecewise analytic optimum extreme point solution.*

We can now give an alternative characterization of extreme points for CDNFP in a similar way as in static network flows.

Theorem 4.6. *Suppose that Assumptions 3 and 4 hold and $x(t)$ is a piecewise analytic feasible solution for CDNFP. Then $x(t)$ is an extreme point of the feasible region F if and only if there exists no bi-augmenting cycle under $x(t)$.*

Proof. Having established Theorem 4.4, the proof is pretty straightforward and thus omitted here. We refer the reader to [21] for a detailed proof. □

4.2 A purification Algorithm

Here we present a generic form of the purification algorithm for CDNFP given piecewise analytic problem data. We suppose that Assumptions 3 and 4 hold throughout the rest of the paper. Thus by Theorem 4.5, CDNFP has a piecewise analytic optimum solution and we can restrict our attention to piecewise analytic solutions. The algorithm begins with a piecewise analytic solution $x(t)$ for CDNFP which is not an extreme point of F. Hence, by Theorem 4.4, there is some node i and some interval (u, v) such that either

(a) there is a bi-augmenting arc-cycle under $x(t)$ of the form

$$W(\alpha) : (i, \alpha) = (i_1, \tau_1(\alpha)), (i_2, \tau_2(\alpha)), \dots, (i_q, \tau_q(\alpha)) = (i, \alpha),$$

for each $\alpha \in (u, v)$; or

(b) there is a bi-augmenting arc-path from (i, α) to some (j, β) as

$$P(\alpha) : (i, \alpha) = (i_1, \tau_1(\alpha)), (i_2, \tau_2(\alpha)), \dots, (i_q, \tau_q(\alpha)) = (j, \beta),$$

with

$$0 < y_{i_1}(\tau_1(\alpha)) < b_i(\tau_1(\alpha)), \quad 0 < y_{i_q}(\tau_q(\alpha)) < b_{i_q}(\tau_q(\alpha)),$$

for each $\alpha \in (u, v)$.

The basic idea of the purification algorithm is to increase or decrease the flow rate on the arcs in either sequence $W(\alpha)$ or $P(\alpha)$ during $[u, v]$, depending on whether case (a) or (b) occurred. More precisely, we wish to construct a new flow $\bar{x}(t) = x(t) + z(t)$ with storage $\bar{y}(t)$, where $z(t)$ is defined by

$$z_{i,j}(t) = \begin{cases} \delta(\alpha), & \text{if } i = i_{i_k}, j = i_{k+1},\ t = \tau_{i_k}(\alpha) \text{ for some } \alpha \in [u, v],\ k = 1, \dots, q, \\ -\delta(\alpha), & \text{if } j = i_{i_k}, i = i_{k+1},\ t = \tau_{i_k+1}(\alpha) \text{ for some } \alpha \in [u, v],\ k = 1, \dots, q, \\ 0, & \text{otherwise.} \end{cases} \tag{26}$$

The function $\delta(\alpha)$ on $[u, v]$ will be chosen in such a way that the following conditions hold:

(i) $\bar{x}(t)$ remains feasible for CDNFP;

(ii) for all $\alpha \in (u, v)$, neither $W(\alpha)$ is a bi-augmenting arc-cycle under $\bar{x}$ nor $P(\alpha)$ is a bi-augmenting arc-path (under $\bar{x}$) between (i, α) and (j, β) with

$$0 < \bar{y}_{i_1}(\tau_1(\alpha)) < b_i(\tau_1(\alpha)),\ 0 < \bar{y}_{i_q}(\tau_q(\alpha)) < b_{i_q}(\tau_q(\alpha));$$

(iii) the objective function value improves in comparison to $x(t)$ (or remains the same).

Observe that although $\bar{x}(t)$ may not be an extreme point, it has fewer a bi-augmenting arc-cycles or arc-paths than $x(t)$ and one can repeat the process until an extreme point solution is generated.

For each $k = 2, \dots, q$ and $\alpha \in (u, v)$, we define

$$\delta^h_{i_{k-1},i_k}(\tau_{k-1}(\alpha)) = \begin{cases} a_{i_{k-1},i_k}(\tau_{k-1}(\alpha)) - x_{i_{k-1},i_k}(\tau_{k-1}(\alpha)), & \text{if } (i_{k-1}, i_k) \in A, \\ x_{i_k,i_{k-1}}(\tau_k(\alpha)), & \text{otherwise,} \end{cases} \tag{27}$$

$$\delta^l_{i_{k-1},i_k}(\tau_{k-1}(\alpha)) = \begin{cases} -x_{i_{k-1},i_k}(\tau_{k-1}(\alpha)), & \text{if } (i_{k-1}, i_k) \in A, \\ x_{i_k,i_{k-1}}(\tau_k(\alpha)) - a_{i_k,i_{k-1}}(\tau_k(\alpha)), & \text{otherwise.} \end{cases} \tag{28}$$

The terms $\delta^h_{i_{k-1},i_k}(\tau_{k-1}(\alpha))$ and $\delta^l_{i_{k-1},i_k}(\tau_{k-1}(\alpha))$ represent the maximum and minimum additional flow rate, respectively, that can be sent from node i_{k-1} to node i_k, at time $\tau_{k-1}(\alpha)$ by using either arc (i_{k-1}, i_k) or (i_k, i_{k-1}), depending on which one is in A.

Now we define

$$\delta^h(\alpha) = \min\{\delta^h_{i_{k-1},i_k}(\tau_{k-1}(\alpha)) : \ k = 2, \dots, q\},$$
$$\delta^l(\alpha) = \max\{\delta^l_{i_{k-1},i_k}(\tau_{k-1}(\alpha)) : \ k = 2, \dots, q\}.$$

Note that we have

$$\delta^l(\alpha) < 0 < \delta^h(\alpha), \quad \alpha \in (u, v).$$

It is not hard to see that the functions $\delta^h(\alpha)$ and $\delta^l(\alpha)$ give the highest and lowest flow rate, respectively, which can be augmented along $W(\alpha)$ or $P(\alpha)$ without violating the transit capacity constraints. In other words, we will have $0 \leq \bar{x}(t) \leq a(t)$ for every $t \in [0, T]$, if and only if

$$\delta^l(\alpha) \leq \delta(\alpha) \leq \delta^h(\alpha), \ \alpha \in [u, v]. \tag{29}$$

Moreover, if $\delta(\alpha) = \delta^l(\alpha)$ or $\delta(\alpha) = \delta^h(\alpha)$, then neither $W(\alpha)$ nor $P(\alpha)$ will be a bi-augmenting arc-cycle or arc-path, respectively, under the new solution $\bar{x}(t)$.

It is clear that augmenting flow along an arc-cycle does not affect the storage at nodes. Thus, if case (a) occurs, then augmenting flow by rate $\delta(\alpha) \in [\delta^l(\alpha), \delta^h(\alpha)]$ during $[u, v]$ yields a feasible solution. This feasible solution will be $\bar{x}(t) = x(t) + z(t)$ with $z(t)$ given by (26). Thus, condition (i) is satisfied. Moreover, if $\delta(\alpha) = \delta^l(\alpha)$ or $\delta(\alpha) = \delta^h(\alpha)$, then $W(\alpha)$ will not be a bi-augmenting arc-cycle under $\bar{x}(t)$ and as a consequence condition (ii) will be satisfied. In order to improve the objective function value, we need to compute the *cost of augmenting*, flow rate $\delta(\alpha)$ through $W(\alpha)$, that is, the difference in objective function values between $x(t)$ and $\bar{x}(t)$. We first define the *cost of arc-cycle* $W(\alpha)$ as follows:

$$\text{Cost}[W(\alpha)] = \sum_{k=2}^{q} \eta_k c_{i_{k-1},i_k}(\tau_{k-1}(\alpha)) - (1 - \eta_k) c_{i_k,i_{k-1}}(\tau_k(\alpha)),$$

where $\eta_k = 1$ if $(i_{k-1}, i_k) \in A$, and $\eta_k = 0$, otherwise. The cost of augmenting flow with rate $\delta(\alpha)$ during $[u, v]$ is now given by

$$\int_u^v \text{Cost}[W(\alpha)]\delta(\alpha)d\alpha.$$

We know that $c(t)$ is piecewise analytic on $[u, v]$, and so is $\text{Cost}[W(\alpha)]$. We can thus split up the interval $[u, v]$ into a finite number of subintervals in each of which $\text{Cost}[W(\alpha)]$ has the same sign. We can thus carry out the purification by setting $\delta(\alpha) = \delta^h(\alpha)$ throughout any subinterval where $\text{Cost}[W(\alpha)] \leq 0$ and $\delta(\alpha) = \delta^l(\alpha)$ throughout any subinterval where $\text{Cost}[W(\alpha)] > 0$. In either case we produce a feasible solution with improved objective function value (or at least no worse) such that $W(\alpha), \ \alpha \in (u, v)$, is not an arc-cycle under it.

Now we discuses how to augment flow on the arcs in $P(\alpha)$ during $[u, v]$ such that conditions (i)-(iii) are satisfied. This case requires a complicated argument because it affects the storage at the end nodes of the path. To simplify the analysis, we assume without loss of generality that (u, v) is chosen so that all problem data are analytic on (u, v).

Assume that $\delta(\alpha) \in [\delta^l(\alpha), \delta^h(\alpha)]$ is augmented along $P(\alpha)$ during $[u, v]$. This yields a flow $\bar{x}(t) = x(t) + z(t)$, where $z(t)$ is given by (26), with corresponding storage $\bar{y}(t)$. It is not difficult to see that if

$$\int_u^v \delta(s)ds = 0, \tag{30}$$

then $\bar{y}(t)$ remains unchanged except at the end nodes i_1 and i_q during $[u, v]$ and $[\tau_q(u), \tau_q(v)]$, respectively. More precisely, if (30) holds, then $\bar{y}(t)$ is given by

$$\bar{y}_i(t) = \begin{cases} y_{i_1}(t) - \int_u^\alpha \delta(s)ds, & \text{if } i = i_1 \text{ and } t = \tau_1(\alpha) \text{ for some } \alpha \in [u, v], \\ y_{i_q}(t) + \int_u^\alpha \delta(s)ds, & \text{if } i = i_q \text{ and } t = \tau_q(\alpha) \text{ for some } \alpha \in [u, v], \\ y_i(t), & \text{otherwise.} \end{cases}$$

Thus we will have $0 \leq \bar{y}(t) \leq b(t)$, for every $t \in [0,T]$, if and only if

$$\varphi^l(\alpha) \leq \int_u^\alpha \delta(s)ds \leq \varphi^h(\alpha),\ \alpha \in [u,v], \tag{31}$$

where

$$\varphi^h(\alpha) = \min\{y_{i_1}(\alpha), b_{i_q}(\tau_q(\alpha)) - y_{i_q}(\tau_q(\alpha))\},$$
$$\varphi^l(\alpha) = \max\{-b_{i_1}(\alpha) + y_{i_1}(\alpha), -y_{i_q}(\tau_q(\alpha))\}.$$

Summarizing, we obtain the following result.

Lemma 7. *Let $\delta(\alpha)$ be any bounded measurable function on $[u,v]$ satisfying (29), (30) and (31), and $z(t)$ be given by (26) accordingly. Then $\bar{x}(t) = x(t) + z(t)$ is feasible for CDNFP.*

The next step is to consider the cost of augmenting along $P(\alpha)$ given by the following lemma.

Lemma 8. *If $\delta(\alpha)$ satisfies (30), then*

$$V[CDNFP, \bar{x}(t), \bar{y}(t)] - V[CDNFP, x(t), y(t)] = \int_u^v Cost[P(\alpha)]\delta(\alpha)d\alpha,$$

where

$$Cost[P(\alpha)] = -\int_\alpha^v d_1(\tau_1(s))ds + \sum_{k=2}^{q} \{\eta_k c_{i_{k-1},i_k}(\tau_{k-1}(\alpha)) - (1-\eta_k)c_{i_k,i_{k-1}}(\tau_k(\alpha))\}$$
$$+ \int_\alpha^v d_q(\tau_q(s))ds.$$

Proof. The result follows easily by integrating by parts. □

Our aim is to improve the objective function value as much as possible. This leads to the following optimization problem:

$$\begin{aligned} PS:\ \min & \int_u^v \text{Cost}[P(\alpha)]\delta(\alpha)d\alpha \\ \text{s.t. } & \varphi^l(t) \leq \int_u^\alpha \delta(s)ds \leq \varphi^h(\alpha), \quad \alpha \in [u,v], \\ & \delta^l(\alpha) \leq \delta(\alpha) \leq \delta^h(t), \quad \alpha \in [u,v], \\ & \int_u^v \delta(\alpha)d\alpha = 0. \end{aligned}$$

This problem is an instance of SCLP with only one decision variable $\delta(\alpha)$ on $[u,v]$. The feasible region of PS is nonempty (as $\delta(\alpha) = 0, \alpha \in [u,v]$ is feasible) and bounded. Thus, by using a result from the theory of SCLP, it has a piecewise analytic optimum extreme point solution with a finite number of breakpoints (see Pullan [28]). Such an optimum solution has been derived by Anderson and Pullan [11]. In the sequel, by using the results in [11], we first compute an optimum solution for PS and then show that it is the desired solution for purification. To this end, we first need to define the following functions:

$$h(\alpha) = \int_u^\alpha \delta(s)ds, \quad \alpha \in [u,v],$$
$$\chi(\alpha) = \frac{d}{d\alpha}\text{Cost}[P(\alpha)], \quad \alpha \in [u,v].$$

Solving PS involves the computation of the function $h(\alpha)$, the integral of $\delta(\alpha)$, on $[u,v]$. It is convenient to compute the cost of augmenting in terms of $h(\alpha)$, that is,

$$V[\text{CDNFP}, \bar{x}(t), \bar{y}(t)] - V[\text{CDNFP}, x(t), y(t)] = -\int_u^v \chi(\alpha)h(\alpha)d\alpha.$$

The cost functions $c(t)$ and $d(t)$ are analytic on $[u,v]$, and hence is $\chi(\alpha)$. We can thus partition the interval $[u,v]$ into a finite number of subintervals in each of which $\chi(\alpha)$ has the same sign. We have two cases to consider, one where $\chi(\alpha)$ is positive on a subinterval and the other where $\chi(\alpha)$ is negative. The construction is essentially the same in the two cases. In fact one can be obtained from the other by replacing $P(\alpha)$ by its reverse, that is

$$(j,\beta) = (i_q, \tau_q(\alpha)), (i_{q-1}, \tau_{q-1}(\alpha)), \ldots, (i_1, \tau_1(\alpha)) = (i, \alpha).$$

Hence, without any loss of generality, we assume that $\chi(\alpha) \geq 0$ for all $\alpha \in [u,v]$, and so we wish to make $h(\alpha)$ as large as possible on $[u,v]$.

We need to introduce several other definitions. Let

$$\phi^h(t) = \int_u^t \delta^h(t)ds,$$
$$\phi^l(t) = \int_t^v \delta^l(t)ds,$$

for $t \in [u,v]$. We now define a function $f(\alpha)$ on $[u,v]$ by

$$f(\alpha) = \min\{-\phi^l(\alpha), -\varphi^l(\alpha), \varphi^h(\alpha), \phi^h(\alpha)\}.$$

It is clear that $f(\alpha) > 0$ for any $\alpha \in (u,v)$, and $f(u) = f(v) = 0$. Moreover, $f(\alpha)$ is piecewise analytic and continuous on $[u,v]$ (see Lemma 3.2 in [11]).

Following Anderson and Pullan [11], we let

$$h(\alpha) = \min_{t\in[u,v]} g(t,\alpha), \tag{32}$$

where

$$g(t,\alpha) = \begin{cases} f(t) + \int_t^\alpha \delta^h(s)ds & t \leq \alpha \ , \\ f(t) - \int_\alpha^t \delta^l(s)ds & t > \alpha \ . \end{cases}$$

The function $h(\alpha)$, given by (32), satisfies the following properties:

1. $h(\alpha)$ is absolutely continuous on $[u,v]$ and $\dot{h}(\alpha) \in [\delta^l(\alpha), \delta^h(\alpha)]$, for all $\alpha \in [u,v]$ (see Lemma 3.4 in [11]).
2. $0 < h(\alpha) \leq \varphi^h(\alpha)$, for all $\alpha \in [u,v]$.
3. $\varphi(u) = \varphi(v) = 0$.

By the above observations, a feasible solution for PS may be defined by

$$\delta(\alpha) = \dot{h}(\alpha), \quad \alpha \in [u,v]. \tag{33}$$

Now we summarize the above discussion in the following theorem.

Theorem 4.7. *The function $\delta(\alpha)$ defined by (32) and (33) yields a solution $\bar{x}(t)$ which satisfies conditions (i)-(iii).*

Proof. Assume that $\delta(\alpha)$ is given by (32) and (33). If $z(t)$ is given by (26) by using $\delta(\alpha)$, then, by the above observations, $\bar{x}(t) = x(t) + z(t)$ is a feasible solution and has an improved objective function value in comparison with $x(t)$ (or at least no worse if $\chi(\alpha) = 0$ on (u,v)). Thus, conditions (i) and (iii) are satisfied. Moreover, we can show that $\delta(\alpha)$ is the unique (up to equality a.e.) optimum solution for PS (see Corollary 4.1.1 in [11] for details). On the other hand, PS is an instance of SCLP, and its feasible region is bounded and nonempty. So, by a result of Pullan [28], PS has an optimum extreme point solution. This means that $\delta(\alpha)$ is an extreme point of the feasible region for PS. The extreme points of PS must be as $\delta(\alpha) = \delta^l(\alpha), \delta(\alpha) = \delta^h(\alpha), \int_0^\alpha \delta(s)ds = \phi^l(\alpha)$ or $\int_0^\alpha \delta(s)ds = \phi^h(\alpha)$ (see [3]). The last two terms are equivalent to $h(\alpha) = \phi^l(\alpha)$ or $h(\alpha) = \phi^h(\alpha)$, respectively. The $h(\alpha)$ given by (32) is positive on (u,v). Hence, $\delta(\alpha)$ must be equal to $\delta^l(\alpha)$, $\delta^h(\alpha)$ or $\phi^h(\alpha)$ on (u,v). This means that $P(\alpha)$ is not an arc-path under $\bar{x}(t)$ with $0 < \bar{y}_{i_1}(\tau_1(\alpha)) < b_i(\tau_1(\alpha))$, and $0 < \bar{y}_{i_q}(\tau_q(\alpha)) < b_{i_q}(\tau_q(\alpha))$, and the proof is complete. $\square$

If Assumption 1 is supposed to hold, then it becomes relatively easier to compute an optimum extreme point solution for PS.

Theorem 4.8. *If Assumption 1 holds, then $h(\alpha)$, as given by (32), is equal to $f(\alpha)$ on $[u, v]$.*

Proof. If $d(t)$, $r(t)$ and $a(t)$ are piecewise constant, and $c(t)$ and $b(t)$ are piecewise linear, then CDNFP has an optimum solution which is piecewise constant by a result of Pullan [31]. Thus we can consider only piecewise constant solutions for CDNFP and assume that the interval (u, v) is chosen such that $\delta^h_{i_{k-1},i_k}(\tau_{k-1}(\alpha))$, $\delta^l_{i_{k-1},i_k}(\tau_{k-1}(\alpha))$, $k = 2, \dots, q$ are constant, and $y_{i_1}(\alpha)$, $b_{i_1}(\alpha)$, $y_{i_q}(\tau_q(\alpha))$, $b_{i_q}(\tau_q(\alpha))$ are linear on (u, v). Thus, $\phi^l(\alpha)$, $\varphi^l(\alpha)$, $\varphi^h(\alpha)$, $\phi^h(\alpha)$ are piecewise linear, continuous and concave on (u, v) since they are the minimum of a finite set of lines, and so is $f(\alpha)$. Now by a similar argument as the proof of Theorem 5.1 in [3], we can now show that $h(t) = f(t)$ for $t \in [u, v]$. □

As mentioned earlier, the new solution $\bar{x}(t)$ may not be an extreme point, but the above procedure can be repeated until an extreme point is produced. The convergence properties of the algorithm are obviously of great interest. For all examples with $r(t)$ and $a(t)$ piecewise constant, and $b(t)$ piecewise linear that we have examined the algorithm produces an extreme point after a finite number of iterations. Whether or not this is true for each CDNFP and every initial feasible solution is an open question.

Example 4.1. Here we consider Example 3.1 again to show that the proposed purification algorithm may be incorporated into the solution algorithms for CDNFP.

We perform the purification algorithm for the solution obtained at the end of the first iteration. This solution is

$$x_{1,2}(t) = \begin{cases} 0.6, & t \in [0,4), \\ 0.4, & t \in [4,6), \\ 0, & t \in [6,10], \end{cases} \qquad x_{1,3}(t) = \begin{cases} 0.8, & t \in [0,6), \\ 0, & t \in [6,10], \end{cases}$$

$$x_{2,3}(t) = \begin{cases} 0, & t \in [0,2), \\ 0.6, & t \in [2,6), \\ 0, & t \in [6,10], \end{cases} \qquad x_{2,4}(t) = \begin{cases} 0, & t \in [0,6), \\ 0.4, & t \in [6,8), \\ 0, & t \in [8,10], \end{cases}$$

$$x_{3,4}(t) = \begin{cases} 0, & t \in [0,2), \\ 0.4, & t \in [2,4), \\ 1.6, & t \in [4,8), \\ 0, & t \in [8,10], \end{cases}$$

with corresponding storage $y(t)$ derived as

$$y_1(t) = \begin{cases} 8 - 1.4t, & t \in [0,4), \\ 7.2 - 1.2t, & t \in [4,6), \\ 0, & t \in [6,10], \end{cases} \qquad y_2(t) = 0, \quad t \in [0,10],$$

$$y_3(t) = \begin{cases} 0, & t \in [0,2), \\ -0.8 + 0.4t, & t \in [2,4), \\ 1.6 - 0.2t, & t \in [4,8), \\ 0, & t \in [8,10], \end{cases} \qquad y_4(t) = \begin{cases} 0, & t \in [0,4), \\ -1.6 + 0.4t, & t \in [4,6), \\ -8.6 + 1.6t, & t \in [6,8), \\ 20 - 2t, & t \in [8,10]. \end{cases}$$

This solution is not an extreme point of the feasible region since

$$P_1(\alpha) : (1, \alpha), (2, \alpha + 2), (4, \alpha + 4),$$

is an arc-path under $x(t)$ with $0 < y_1(\alpha) < b_1(\alpha)$ and $0 < y_4(\alpha + 4) < b_4(\alpha + 4)$, for all $\alpha \in (4, 6)$. For this arc-path, we have

$$\chi(\alpha) = \tfrac{d}{d\alpha}\text{Cost}[P_2(\alpha)] = 1.6, \quad \alpha \in [4, 6],$$

Now we proceed with the purification by augmenting flow along $P(\alpha)$ during $[4, 6]$. For each $\alpha \in [4, 6]$, we have

$$\delta^h_{1,2}(\alpha) = 0.2, \qquad \delta^h_{2,4}(\alpha + 2) = 1.8,$$

$$\delta^l_{1,2}(\alpha) = -0.4, \qquad \delta^l_{2,4}(\alpha + 2) = -0.4,$$

and so

$$\delta^l(\alpha) = -0.4, \qquad \delta^h(\alpha) = 0.2.$$

Moreover,

$$\varphi^l(\alpha) = -20 + 2\alpha, \qquad \varphi^h(\alpha) = 7.2 - 1.2\alpha,$$
$$\phi^l(\alpha) = -2.4 + 0.4\alpha, \qquad \phi^h(\alpha) = -0.8 + 0.2\alpha.$$

Now $f(\alpha)$ is obtained as

$$\begin{aligned} f(\alpha) &= \min\{2.4 - 0.4\alpha, 20 - 2\alpha, 7.2 - 1.2\alpha, -0.8 + 0.2\alpha\} \\ &= \begin{cases} -0.8 + 0.2\alpha, & t \in [4, \frac{16}{3}), \\ 2.4 - 0.4\alpha, & t \in [\frac{16}{3}, 6], \end{cases} \end{aligned}$$

and hence we have

$$\delta(\alpha) = \dot{f}(\alpha) = \begin{cases} 0.2, & t \in [4, \frac{16}{3}), \\ -0.4, & t \in [\frac{16}{3}, 6]. \end{cases}$$

The cost of augmenting flow $\delta(\alpha)$ along $P_1(\alpha)$ during $\alpha \in (0, 4)$ is $-\frac{64}{75} = -0.426\bar{6}$.

In addition to $P_1(\alpha)$, there is another arc-path

$$P_2(\alpha) : (3, \alpha), (4, \alpha + 2),$$

under $x(t)$ with $0 < y_3(\alpha) < b_3(\alpha)$ and $0 < y_4(\alpha + 4) < b_4(\alpha + 4)$, for all $\alpha \in (2, 4)$. Here, we have

$$\chi(\alpha) = \tfrac{d}{d\alpha}\text{Cost}[P_2(\alpha)] = -1.2, \quad \alpha \in [2, 4],$$

so we consider the reverse arc-path

$$\overleftarrow{P}_2(\alpha) : (4, \alpha), (3, \alpha - 2), \quad \alpha \in [4, 6]$$

Now we proceed with the purification by augmenting flow along path $\overleftarrow{P}_2(\alpha)$ during $[4, 6]$. For each $\alpha \in [4, 6]$, we have

$$\delta^l(\alpha) = -1.2, \qquad \delta^h(\alpha) = 0.4,$$
$$\varphi^l(\alpha) = 1.6 - 0.4\alpha, \qquad \varphi^h(\alpha) = -1.6 + 0.4\alpha,$$
$$\phi^l(\alpha) = -7.2 + 1.2\alpha, \qquad \phi^h(\alpha) = -1.6 + 0.4\alpha.$$

Now $f(\alpha)$ is determined by

$$f(\alpha) = \min\{-1.6 + 0.4\alpha, 7.2 - 1.2\alpha\} = \begin{cases} -1.6 + 0.4\alpha, & t \in [4, 5.5), \\ 7.2 - 1.2\alpha, & t \in [5.5, 6]. \end{cases}$$

and therefore

$$\delta(\alpha) = \dot{f}(\alpha) = \begin{cases} 0.4, & t \in [4, 5.5), \\ -1.2, & t \in [5.5, 6]. \end{cases}$$

The cost of augmenting flow $\delta(\alpha)$ along path $\overleftarrow{P}_2(\alpha)$ during $[4, 6]$ is -0.12.

The new solution is

$$\bar{x}_{1,2}(t) = \begin{cases} 0.6, & t \in [0, \frac{16}{3}), \\ 0, & t \in [\frac{16}{3}, 10], \end{cases} \qquad \bar{x}_{1,3}(t) = \begin{cases} 0.8, & t \in [0, 6), \\ 0, & t \in [6, 10], \end{cases}$$

$$\bar{x}_{2,3}(t) = \begin{cases} 0, & t \in [0, 2), \\ 0.6, & t \in [2, 6), \\ 0, & t \in [6, 10], \end{cases} \qquad \bar{x}_{2,4}(t) = \begin{cases} 0, & t \in [0, 6), \\ 0.6, & t \in [6, \frac{22}{3}), \\ 0, & t \in [\frac{22}{3}, 10], \end{cases}$$

$$\bar{x}_{3,4}(t) = \begin{cases} 0, & t \in [0, 3.5), \\ 1.6, & t \in [3.5, 8), \\ 0, & t \in [8, 10], \end{cases}$$

with corresponding storage $y(t)$ derived as

$$\bar{y}_1(t) = \begin{cases} 8 - 1.4t, & t \in [0, \frac{16}{3}), \\ 4.8 - 0.8t, & t \in [\frac{16}{3}, 6), \\ 0, & t \in [6, 10], \end{cases} \qquad \bar{y}_2(t) = 0, \quad t \in [0, 10],$$

$$\bar{y}_3(t) = \begin{cases} 0, & t \in [0, 2), \\ -1.6 + 0.8t, & t \in [2, 3.5), \\ 4 - 0.8t, & t \in [3.5, 4), \\ 1.6 - 0.2t, & t \in [4, 8), \\ 0, & t \in [8, 10], \end{cases} \qquad \bar{y}_4(t) = \begin{cases} 0, & t \in [0, 5.5), \\ -8.8 + 1.6t, & t \in [5.5, 8), \\ 15.2 - 1.8t, & t \in [8, \frac{28}{3}), \\ 24 - 2.4t, & t \in [\frac{28}{3}, 10]. \end{cases}$$

The objective function value of $\bar{x}(t)$ is $123.76 - 0.426\bar{6} - 0.12 = 123.63\bar{3}$. We can check that there is no bi-augmenting cycle under $\bar{x}(t)$ and hence, by Theorem 4.6, it is an extreme point solution. This solution is also optimum since there is no augmenting cycle under $\bar{x}(t)$ with negative cycle (see Theorem 6.7 in [21]). We have thus obtained not only an extreme point solution, but also an optimum solution.

References

[1] R. K. Ahuja, T. L. Magnanti and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*, Prentice-Hall, Englewood Cliffs, NJ, 1993.

[2] E. J. Anderson. *A Continuous Model for Job-Shop Scheduling*, PhD. Thesis, University of Cambridge, Cambridge, U.K., 1978.

[3] E. J. Anderson, P. Nash and A. F. Perold, *Some properties of a class of continuous linear programs*, SIAM J. Control and Optimization 21 (1983), pp. 258–265.

[4] E. J. Anderson and P. Nash, *Linear Programming in Infinite-Dimensional Spaces: Theory and Applications*, John Wiley & Sons, 1987.

[5] E. J. Anderson, P. Nash and A. B. Philpott, *A class of continuous network flow problems*, Mathematics of Operations Research 7 (1982), pp. 501–514.

[6] E. J. Anderson and A. S. Lewis, *An extension of the simplex algorithm for semi-infinite linear programming*, Math Programming 44 (1989), pp. 247–269.

[7] E. J. Anderson and A. B. Philpott, *A continuous-time network simplex algorithm,* Networks 19 (1989), pp. 395–425.

[8] E. J. Anderson and A. B. Philpott, *On the solutions of a class of continuous linear programs*, SIAM J. Control and Optimization 32 (1994), pp. 1289–1296.

[9] E. J. Anderson, *Extreme-points for continuous network programs with arc delays*, J. Inform. Optim. Sci. 10 (1989), pp. 45–52.

[10] E. J. Anderson and A. B. Philpott, *Optimisation of flows in networks over time*, in F. P. Kelly (eds.): *Probability, Statistics and Optimisation*, John Wiley & Sons, Chichester, UK, 1994, pp. 369–382.

[11] E. J. Anderson and M. C. Pullan, *Purification for separated continuous linear programs*, Mathematical Methods of Operations Research 43 (1996), pp. 9–33.

[12] J. B. Conway, *A Course in Functional Analysis*, 2nd Edition, New York, Springer-Verlag, 1990.

[13] L. Fleischer and J. Sethuraman, *Efficient algorithms for separated continuous linear programs: The multi-commodity fow problem with holding costs and extensions*, Mathematics of Operations Research 30 (2005), pp. 916–938.

[14] L. R. Ford and D. R. Fulkerson, *Constructing maximal dynamic flows from static flows*, Operations Research 6 (1958), pp. 419–433.

[15] L. R. Ford and D. R. Fulkerson, *Flows in Networks*, Princeton University Press, New Jersey, 1962.

[16] B. Klinz and G. J. Woeginger, *Minimum-cost dynamic flows: The series-parallel case*, Networks 43 (2004), pp. 153–162.

[17] T. Leon and E. Vercher, *A purification algorithm for semi-infinite programming*, European Journal of Operational Research 57 (1992), pp. 412–420.

[18] A. S. Lewis, *Extreme points and purification algorithms in general linear programming* in E. J. Anderson and A. B. Philpott (eds.): *Infinite Programming: Proceedings of an International Symposium on Infinite Dimensional Linear Programming*, Springer, Berlin (1985), pp. 123–135.

[19] X. D. Luo, *Continuous Linear Programming: Theory, Algorithms and Applications*, PhD thesis, MIT Operations Research Center, Cambridge MA, 1995.

[20] X. Luo and D. Bertsimas,*A new algorithm for state-constrained separated continuous linear programs*, SIAM J. Control Optim. 37 (1998), pp. 177–210.

[21] E. Nasrabadi, *Dynamic Flows in Time-varying Networks*, Ph.D. thesis, Technische Universität Berlin, Berlin, 2009.

[22] A. F. Perold, *Extreme points and basic feasible solutions in continuous time linear programming*, SIAM J. Control Optim. 19 (1981), pp. 52–63.

[23] A. B. Philpott, *Algorithms For Continuous Network Flow Problems*, Ph.D. thesis, University of Cambridge, UK, 1982.

[24] A. B. Philpott, *Network programming in continuous time with node storage*, in E. J. Anderson and A. B. Philpott (eds.): *Infinite Programming: Proceedings of an International Symposium on Infinite Dimensional Linear Programming* Springer, Berlin, 1985, pp. 136–153.

[25] A. B. Philpott, *Continuous-time flows in networks, Mathematics of Operations Research*, 15 (1990), pp. 640–661.

[26] A. B. Philpott and M. Craddock, *An adaptive discretization algorithm for a class of continuous network programs*, Networks 26 (1995), pp. 1–11.

[27] M. C. Pullan, *An algorithm for a class of continuous linear programs*, SIAM J. Control Optim. 31 (1993), pp. 1558–1577.

[28] M. C. Pullan, *Forms of optimal solutions for separated continuous linear programs*, SIAM J. Control Optim. 33 (1995), pp. 1952–1977.

[29] M. C. Pullan, *A duality theory for separated continuous linear programs*, SIAM J. Control Optim. 34 (1996), pp. 931–965.

[30] M. C. Pullan, *Existence and duality theory for separated cnotinuous linear programs* Math. Modelling Systems 3 (1997), 219–245.

[31] M. C. Pullan, *A study of general dynamic network programs with arc time-delays*, SIAM J. Optim. 7 (1997), pp. 889–912.

[32] M. C. Pullan, *Convergence of a general class of algorithms for separated continuous linear programs*, SIAM J. Optim. 10 (2000), pp. 722–731.

[33] M. C. Pullan, *An extended algorithm for separated continuous linear programs*, Mathematical Programming 93 (2002), pp. 415–451.

[34] M. Skutella, *An introduction to network flows over time*, in W. Cook, L. Lovász and J. Vygen (eds.): *Research Trends in Combinatorial Optimization*, Springer, Berlin, 2009, pp. 451–482.

[35] G. Weiss, *A simplex based algorithm to solve separated continuous linear programs*, Mathematical Programming, 115 (2008), pp. 151–198.