A self-adaptive gradient projection algorithm for the nonadditive traffic equilibrium problem

Anthony Chen a,*, Zhong Zhou b, Xiangdong Xu a

a Department of Civil and Environmental Engineering, Utah State University, Logan, UT 84322-4110, USA
b Citilabs, 316 Williams Street, Tallahassee, FL 32303, USA

Abstract
Gradient projection (GP) algorithm has been shown as an efficient algorithm for solving the traditional traffic equilibrium problem with additive route costs. Recently, GP has been extended to solve the nonadditive traffic equilibrium problem (NaTEP), in which the cost incurred on each route is not just a simple sum of the link costs on that route. However, choosing an appropriate stepsize, which is not known a priori, is a critical issue in GP for solving the NaTEP. Inappropriate selection of the stepsize can significantly increase the computational burden, or even deteriorate the convergence. In this paper, a self-adaptive gradient projection (SAGP) algorithm is proposed. The self-adaptive scheme has the ability to automatically adjust the stepsize according to the information derived from previous iterations. Furthermore, the SAGP algorithm still retains the efficient flow update strategy that only requires a simple projection onto the nonnegative orthant. Numerical results are also provided to illustrate the efficiency and robustness of the proposed algorithm.

1. Introduction

A basic assumption of the traditional traffic equilibrium model is additivity (i.e., the route cost is simply the sum of the costs on the links that constitute that route). The biggest advantage of the additivity assumption is that it allows the route-flow variables to be removed from the objective function of the convex mathematical programming (MP) formulation (in the case of symmetric link travel time functions) or from the inequality of the variational inequality (VI) formulation (in the case of asymmetric link travel time functions). Thus, the corresponding traffic equilibrium problem can be solved without the need to store routes despite that the route-flow variables remain in the constraint set. This is a significant benefit when one needs to solve large-scale problems in transportation networks.

However, as pointed out by Gabriel and Bernstein [1], the additivity assumption is not appropriate in many real life situations, where the additive route cost structure is inadequate for addressing factors affecting a variety of transportation policies, such as nonlinear value of travel times, route-specific tolls, and emissions fees. A few studies have been performed by using different nonadditive route cost structures, such as the route-specific travel costs [2, 19], bi-criteria nonlinear route costs with elastic demand [3], length-based and congestion-based commonality factors used in the C-logit stochastic user equilibrium model [4], entry-exit based toll charges [5, 6], and feeder bus systems [7]. Recently, the nonadditive route costs have been applied to model risk-averse behavior in the route choice decision process [8–12]. Some formulations and properties of the nonadditive traffic equilibrium problems (NaTEP) were also explored, such as the nonlinear time/money relation [13], uniqueness and convexity of the bi-criteria traffic equilibrium problem [14], and the monotonicity of NaTEP formulated as a monotone mixed complementarity problem [15]. Furthermore, Altman and Wyn-ter [16] discussed the nonadditive cost structures in both transportation and telecommunication networks.

Under the nonadditivity assumption, the corresponding traffic equilibrium problem has to be formulated in the route-flow space, and thus cannot be solved using the traditional link-based algorithms, such as the Frank–Wolfe algorithm (see [17] for the details of the algorithm). Furthermore, as indicated by Bernstein and Gabriel [18], the diagonalization methods also do not work well on the nonadditive problem, since the diagonalized subproblems are poor approximations of the true problem.

Different formulations and solution approaches have been presented for solving various nonadditive traffic equilibrium problems. For example, Bernstein and Gabriel [18] presented a non-smooth equation/sequential quadratic programming (NE/SQP) method for solving the NaTEP with elastic demands. The method is based on first transforming the nonlinear...
complementarity problem (NCP) formulation of the NaTEP into a set of non-smooth equations and then finding the zero point of a non-smooth, non-convex optimization problem. Lo and Chen [19] used a new gap function proposed by Fischer [20] to convert the NCP formulation to an equivalent unconstrained optimization problem, which is smooth and convex. The unconstrained nature makes available a large number of already developed solution algorithms, such as the Newton method, Quasi-Newton method, Gradient method, etc. [21]. Lo and Chen [22] provided an alternate formulation via a smooth gap function using both route flows and O–D costs as the decision variables. Chen et al. [2] suggested a self-adaptive projection and contraction (PC) algorithm for solving a monotone VI formulation as a special case of the NaTEP with elastic demands. Han and Lo [23] also proposed a descent method for the co-coercive VI formulation to solve the NaTEP with elastic demand. The advantage of the last two algorithms is its simplicity in numerical implementations, while both only need some function evaluations of the mapping and a few trivial projection operations on the nonnegative orthant.

On the other hand, the gradient projection (GP) algorithm has been shown as a successful route-based algorithm for solving the traditional traffic equilibrium problem with additive route costs [24,25]. Under an ingenious approach that utilizes the special structure of the traffic equilibrium problem (see the implementation by Jayakrishnan et al. [24] and Chen et al. [25]), GP only needs to perform a simple projection on the nonnegative orthant in each iteration; therefore, the required computational effort is modest. Previous results reported by Jayakrishnan et al. [24] and Chen et al. [25] on the GP algorithm adopt the diagonal inverse Hessian approximation as a scaling matrix, assume a unity stepsize in each iteration, and use the "one-at-a-time" flow update strategy to equilibrate route flows one origin–destination (O–D) pair at a time. Though a near-optimal solution (e.g., 0.001 as the stopping criterion) can be achieved quickly, GP may have difficulty in obtaining the optimal solution (i.e., very accurate solution). This is partly due to the unity stepsize assumption purposely designed in GP to avoid expensive line searches. Attracted by the efficiency and simplicity of the GP algorithm, Scott and Bernstein [26] extended it for solving the NaTEP by using the 'all-at-once' flow update strategy and a modified scaling matrix to reflect the nonadditive route costs. However, the problem of choosing an appropriate stepsize is a critical issue in GP for solving the NaTEP due to the complex route cost structure. The results reported in [26] were mixed. For the Sioux Falls network, a "trial-and-error" approach of choosing a fixed stepsize was used to ensure convergence.

Thus, our study is motivated to develop a robust stepsize scheme in GP for solving the NaTEP. In this paper, a self-adaptive gradient projection (SAGP) algorithm is provided. The self-adaptive scheme has been successfully embedded in the original Goldstein–Levitin–Polyak (GLP) projection algorithm by Han and Sun [27] and demonstrated by Zhou and Chen [28] for solving the asymmetric traffic equilibrium problem. It has the ability to automatically adjust the stepsize according to the information derived from previous iterations. Thus, it is not necessary to use the "trial-and-error" approach as suggested by Scott and Bernstein [26] to select a suitable fixed stepsize. The self-adaptive scheme can significantly enhance the robustness and efficiency of the algorithm. Furthermore, the SAGP algorithm still retains the simple flow update strategy (i.e., simple projection on the nonnegative orthant), thus avoiding the need to solve convex quadratic programs in the original GLP projection algorithm.

The remainder of the paper is organized as follows: a general VI formulation of the NaTEP is given in Section 2; Section 3 discusses the gradient projection algorithm; Section 4 presents the self-adaptive gradient projection (SAGP) algorithm as well as its convergence; numerical results are provided in Section 5 to illustrate the efficiency and robustness of the SAGP algorithm; finally, conclusions are summarized and some future researches are suggested in Section 6.

2. Formulation of the nonadditive traffic equilibrium problem

Throughout this study, we assume the origin–destination (O–D) travel demands are given and fixed. Consider a strongly connected network [N, A], where N and A denote the sets of nodes and links, respectively. Let R and S denote a subset of N for which travel demand \( q^g \) is generated from origin \( r \in R \) to destination \( s \in S \). The assumption of a strongly connected network guarantees that there exists at least one route from every O–D pair with positive travel demand. Let \( f^g_p \) denote the flow on route \( p \in P^s \), where \( P^s \) is a set of routes from origin \( r \) to destination \( s \). Let \( A = [a^g_{rs}] \) denote the route–link incidence matrix, where \( a^g_{rs} = 1 \) if route \( p \) from origin \( r \) to destination \( s \) uses link \( a \), and 0, otherwise. Then, we have the following relationships:

\[
q^g = \sum_{p \in P^s} f^g_p, \quad \forall r \in R, \ s \in S, \tag{1}
\]

\[
v_p = \sum_{r \in R} \sum_{s \in S} f^g_p a^g_{rs} q^g_r, \quad \forall a \in A, \tag{2}
\]

\[
f^g_p \geq 0, \quad \forall p \in P^s, \ r \in R, \ s \in S, \tag{3}
\]

where (1) is the travel demand conservation constraint; (2) is a definitional constraint that sums up all route flows that pass through a given link \( a \); and (3) is a non-negativity constraint on route flows.

Under the symmetric link cost and additive route cost assumptions, the traditional traffic equilibrium model can be formulated as a convex mathematical program and solved by a link-based traffic assignment algorithm (e.g., the Frank–Wolfe algorithm). However, under the nonadditive route cost structure (i.e., not only that the additivity assumption does not hold, the symmetry assumption is also not satisfied), it is therefore necessary to formulate the problem using route-flow variables and solve it with a route-based traffic assignment algorithm. A general nonadditive route cost function can be written as follows [1]:

\[
\eta^g_p = \sqrt{\rho_p} + \sum_{a \in A} \beta^g_{pa} \mu^g_a + g_p \left( \sum_{a \in A} \beta^g_{pa} \mu^g_a \right), \quad \forall p \in P^s, \ r \in R, \ s \in S, \tag{4}
\]

where \( \eta^g_p \) denotes the financial cost (such as toll) specific to route \( p \) between origin \( r \) and destination \( s \), \( \rho_p \) is the operating cost per unit travel time (e.g., fuel consumption, vehicle rental), and \( g_p \) is a function describing the value of time for route \( p \), which could be nonlinear. The second and third terms transfer travel times into an equivalent amount of money consistent with the first term. Typically, the general route cost function \( \eta^g_p \) and the link travel time function \( \tau_a \) are assumed to be positive and continuous; the operating cost factor \( \rho \) is positive; the route-specific cost \( \sqrt{\rho_p} \) and the valuation of time function \( g_p(\cdot) \) are continuous and nonnegative.

Let \( \mathbf{n} \) denote the route-cost vector \((\ldots, \eta^g_{w_1}, \ldots)^T\), \( \mathbf{\mu} \) denote the minimal cost between O–D pair \((r, s)\), and \( \mathbf{f} \) denote the route-flow vector \((\ldots, f^g_{w_1}, \ldots)^T\). The traffic equilibrium problem is to find the traffic-flow pattern by allocating the O–D demands to the network such that all used routes between each O–D pair have equal and minimum travel cost, and no unused route has a lower
travel cost, i.e., the following conditions hold \([17, 29]\):

$$
\eta_P^r(f) - \pi^r = \begin{cases} 
0 & \text{if } (f^P_r)^r > 0, \\
\geq 0 & \text{if } (f^P_r)^r = 0, \\
& \forall p \in P^r, \ r \in R, \ s \in S.
\end{cases}
$$

(5)

Such an equilibrium state is what results if each and every traveler simultaneously attempts to minimize his/her individual travel cost. Then, the NaTEP can be formulated as a variational inequality problem \([30, 31]\):

$$
\eta(f)^{\Omega}(f - g) \geq 0, \ \forall f, g \in \Omega,
$$

(6)

where \(\Omega\) represents the feasible route-flow set described by (1) and (3).

### 3. Projection algorithm and stepsize selection strategies

Many iterative methods have been developed for solving the variational inequality problem (e.g., the projection methods \([32, 33]\), the nonlinear Jacobian methods \([34]\), the successive overrelaxation methods \([35]\), the proximal point methods \([36]\) and the Newton-type methods \([37, 38]\). Among these iterative methods, the projection methods have received much attention due to its global convergence and simple implementation. Goldstein \([39]\) and Levitin and Polyak \([40]\) proposed a projection algorithm (known as GLP projection algorithm) as follows: given an initial point \(f_0 \in \Omega\), the algorithm generates a sequence \(\{f_k\}\) according to the following recursive equation:

$$
f_{k+1} = P_\Omega(f_k - z_k \eta(f_k)), \ k = 0, 1, \ldots,
$$

(7)

where \(P_\Omega[\tau]\) denotes a unique projection of a vector \(\tau \in R^m\) on \(\Omega\), \(|\tau|\) is the cardinality of the route set \(P = \{P^r\}, r \in R, s \in S\), and \(z_k \geq 0\) is a judiciously chosen positive stepsize. However, a large stepsize would lead to divergence, while a small stepsize would slow down the convergence. To guarantee convergence of the GLP projection algorithm, \(z_k\) must be chosen properly. That is, \(z_k\) needs to satisfy the following condition:

$$
0 < z_k \leq z_L \leq z_U < \frac{2\mu}{|\Omega|}.
$$

(8)

where \(z_L\) and \(z_U\) are the lower and upper bounds of the stepsize, respectively; \(L\) is the Lipschitz constant of the mapping \(\eta\) such that

$$
\eta(f) - \eta(f') \leq L|f - f'|, \ \forall f, f' \in \Omega.
$$

(9)

and \(\mu\) is the uniform modulus in the strongly monotone mapping of \(\eta\) such that

$$
(f - f')^T(\eta(f) - \eta(f')) \geq \mu|f - f'|^2, \ \forall f, f' \in \Omega.
$$

(10)

The efficiency of the GLP projection algorithm depends on two issues: (a) performing the projection operation onto the constraint set \(\Omega\), and (b) selecting a proper stepsize \(z_k\). In the GLP projection algorithm, the recursive Eq. (7) requires a solution of \(P_\Omega[\tau]\). If \(\Omega\) is a general polyhedron, performing the projection operation is typically equivalent to solving a quadratic programming problem; and if \(\Omega\) is a general convex set, it is even more complicated. Thus, in order to take advantage of the projection, \(\Omega\) should contain only simple constraints (e.g., non-negativity constraints) such that the projection operation can be easily performed. As for the selection of stepsize \(z_k\), it relies on the estimation of the strongly monotone modulus \(\mu\) and the Lipschitz constant \(L\). However, both \(L\) and \(\mu\) are problem-dependent, and it is generally difficult to estimate these constants in advance for realistic problems.

Several different stepsize strategies have been suggested to overcome the difficulty. Nagurney and Zhang \([41]\) showed that the projection method is convergent for strictly monotone variational inequality problems if the stepsize \(z_k\) is chosen according to the following conditions:

$$
z_k > 0, \ \lim_{k \to \infty} z_k = 0, \ \text{and} \ \sum_{k=1}^{\infty} z_k = \infty.
$$

(11)

The method of successive average (MSA) \(\{z_k/\sum_{k=1}^{\infty} z_k\} k \in N\) is a special case of the above predetermined stepsize rule. Note that this rule does not make use of the information derived from previous iterations to determine the stepsize. Though convergence is ensured using the MSA stepsize, it suffers from the sublinear rate of convergence.

Bertsekas \([42]\) suggested a generalized Armijo’s rule to choose an appropriate stepsize, which could be regarded as the direct generalization of the well-known Armijo rule \([43]\) for the steepest descent method in unconstrained minimization problems. Given a non-stationary point \(f_0\), set

$$
z_k = \beta^{m_k} s_k
$$

(12)

where \(m_k\) is the first nonnegative integer \(m\) such that

$$
Z(f_k) - Z(f_k(\beta^{m_k} s_k)) \geq \sigma VZ(f_k)(f_k - f_k(\beta^{m_k} s_k)),
$$

(13)

where \(\sigma \in (0, 1), \beta \in (0, 1), \text{ and } s > 0\) are fixed scalars; \(Z\) is the objective function; and \(f_k(\beta^{m_k} s_k) = P_\Omega[f_k - \beta^{m_k} s_k V(f_k)]\).

It has been proved by Gafni and Bertsekas \([44]\) that without the requirement of Lipschitz condition, every limit point of \(\{f_k\}\) is a solution point. These rules can be considered as inexact line search strategies. However, they require the gradient information and typically need much more function evaluations in order to find a suitable \(z_k\). Thus, it is not suitable for solving the VI formulation of the NaTEP.

In order to achieve better performance, an appropriate scaled version of the gradient projection algorithm could be adopted by imposing a positive-definite symmetric scaling matrix \([32]\):

$$
f_{k+1} = P_\Omega[f_k - z_k T_k^{-1} \eta(f_k)], \ k = 0, 1, \ldots,
$$

(14)

where \(P_\Omega[\tau]\) denotes the unique projection of a vector \(\tau \in R^m\) on \(\Omega\) with respect to the matrix norm specified by \(T_k\), i.e.,

$$
|\tau|_{T_k} = (\tau^T T_k \tau)^{1/2}.
$$

In practice, one may take \(T_k(f_k) = \nabla \eta(f_k)\) which implies a Newton-type method, or simply take \(T_k(f_k)\) as a diagonal matrix, i.e., a diagonal approximation of the first derivatives of route costs.

The algorithm was demonstrated by using a hypothetical network for both “all-at-once” and “one-at-a-time” implementations, where a unity stepsize \(z_k=1\) is taken as an empirical choice. However, under the “all-at-a-time” strategy, the algorithm may encounter convergence difficulty, especially for larger networks, and recent results on large-scale traffic equilibrium problems reported by Jayakrishnan et al. \([24]\) and Chen et al. \([25]\) show that even with the “one-at-a-time” strategy, it may have difficulty in converging to the optimal solution (i.e., very accurate solution), though a near-optimal solution can be acquired quickly.

Recently, Scott and Bernstein \([26]\) extended the GP algorithm \([24]\) for solving the NaTEP. The key modification is that the scaling matrix \(T_k(f_k)\) is now taken as the first derivatives of the nonadditive route costs. However, choosing an appropriate stepsize \(z_k\) is still a critical issue and is nontrivial. The numerical results reported in \([26]\) showed that the stepsize \(z_k\) has a significant influence on the convergence of the algorithm and it is also dependent on network characteristics and travel demands. A hybrid approach (i.e., warm start by solving the additive traffic equilibrium problem to generate an initial working set of routes and complete it by solving the NaTEP) was proposed to alleviate the stepsize selection problem, but mixed results were reported. Thus, to resolve this difficulty, a self-adaptive gradient projection (SAGP) algorithm is introduced in the next section.
4. Self-adaptive gradient projection algorithm

The self-adaptive scheme was originally proposed by He et al. [45] for the Goldstein–Levitin–Polyak projection algorithm. Recently, it has been extended by Han and Sun [27] and successfully implemented by Zhou and Chen [28] for solving the asymmetric traffic equilibrium problem with additive route costs. The main idea is to determine the stepsize automatically by the information derived from previous iterations. The Lipschitz constant and strongly monotone modulus are not required in the self-adaptive scheme.

For completeness, the main steps of the self-adaptive GLP algorithm for solving the asymmetric traffic equilibrium problem with additive route costs are provided below [28]:

4.1. Self-adaptive Goldstein–Levitin–Polyak (SAGLP) projection algorithm

- **Step 0:** Initialization: set \( \delta \in (0, 1), u \in [0.5, 1], \varepsilon > 0, \alpha_{\text{max}} > 0, \alpha_0 > 0; \) set \( f_0 \in \Omega; \) and set \( \gamma_0 = \alpha_0, k = 0. \)
- **Step 1:** Self-adaptive scaling procedure: Given \( f_k \) is not a solution point, find the smallest nonnegative integer \( l_k \) such that
  
  \[
  \begin{aligned}
  & (2-\delta)z_{k+1}(f_k - f_{k-1})^T (\nu_k - \nu_{k-1} - \alpha_k z_k) - \alpha_k z_k (\nu_k - \nu_{k-1})^2 \\
  & 
  \geq \max \left\{ \frac{\alpha_k}{\alpha_{\text{max}}} \left( \| e(f_k, \nu_k) \|, 0 \right) \right\},
  \end{aligned}
  \]

  where \( e(f_k, \nu_k) = f_k - P_\Omega [f_k - z_k \nu(f_k)] \).
- **Step 2:** Selection of \( \gamma_{k+1} \):
  
  If \( 0.5z_{k+1}(f_k - f_{k-1})^T (\nu_k - \nu_{k-1} - \alpha_k z_k) - \alpha_k z_k (\nu_k - \nu_{k-1})^2 \\
  \geq \max \left\{ \frac{\alpha_k}{\alpha_{\text{max}}} \left( \| e(f_k, \nu_k) \|, 0 \right) \right\},
  \]

  then \( \gamma_{k+1} = \min (\gamma_{k+1}/u, \alpha_{\text{max}}); \) otherwise \( \gamma_{k+1} = 2z_{k+1}. \)
- **Step 3:** Termination criterion: If a predefined convergence criterion is met, stop with \( f_{k+1} \) as an approximate solution. Otherwise, set \( k = k+1 \) and go to Step 1.

Even though the self-adaptive rule above is reminiscent to Bertsekas’s Armijo rule [42], it is more practical and robust. Note that the Armijo rule always starts from a fixed initial stepsize (i.e., \( s \) in Eq. (12)) per iteration. The acceptable stepizes of two consecutive iterations are normally very close, and they may be far away from the starting point \( s \) after first several iterations. Therefore, using the fixed initial stepsize may waste some computational efforts and reduce the algorithmic efficiency. By contrast, the self-adaptive scheme adjusts the initial point \( \gamma_{k+1} \) according to the previous stepsize \( z_k \), allocating the issue associated with the fixed initial stepsize in the Armijo scheme. Step 2, shown above, allows the stepsize sequence \( z_k \) to be non-monotone. This mechanism makes the algorithm more efficient and robust.

Furthermore, the global convergence can be shown under certain assumptions on the underlying mapping \( \eta \) (i.e., the route cost function) as those used in the GLP projection method [39,40] and the modified GLP projection method [45]. However, we need neither the Lipschitz constant \( L \) nor the strongly monotone modulus \( \mu \) to design the self-adaptive scheme. The scaling factor is self-adaptive in the sense that it iteratively adjusts itself to satisfy the Lipschitz condition and the strongly monotone assumption without a priori knowledge of the constants. The results reported in [28] showed that the self-adaptive scheme is significantly better than the strategies that use a fixed stepsize (may be empirically determined) or a predetermined stepsize sequence [41].

In the procedure described above, the projection is performed on the feasible route-flow set \( \Omega \), which is a polyhedron. Therefore, a convex quadratic program has to be solved to ensure flow feasibility. This requirement poses a huge computational effort when the network size is large. For example, updating route flows \( f_{k+1} \) via the projection Eq. (15) is equivalent to solving the following convex quadratic program:

\[
\min \quad \eta(f_k)^T (f_k - f_{k-1}) + \frac{1}{2} z_k (f_k - f_{k-1})^T (f_k - f_{k-1})
\]

s.t. \( f_k \in \Omega. \)

Under the special structure of the traffic equilibrium problem, if the convex quadratic program is decomposed by O–D pairs and the route-flow variables are rewritten in terms of the non-shortest route flows, a closed-form solution could be derived from the first-order optimality conditions. This approach has been adopted in Jayakrishnan et al. [24] and Chen et al. [25] by using Bertsekas and Gafni’s [32] GP algorithm (Eq. (14)) to solve the traffic equilibrium problem. Furthermore, this approach has also been applied in the modified GP algorithm [26] for solving the NaTEP.

Thus, motivated by the good features of the self-adaptive scheme [28] as well as the ingenious gradient projection approach [24,25], the self-adaptive gradient projection (SAGP) algorithm for solving the NaTEP can be presented as follows:

4.2. Self-adaptive gradient projection (SAGP) algorithm

- **Step 0:** Initialization: set \( \delta \in (0, 1), u \in [0.5, 1], \varepsilon > 0, \alpha_{\text{max}} > 0, \alpha_0 > 0, \text{ and } f_0 \in \Omega; \) set \( \gamma_0 = \alpha_0 \) and \( k = 0. \)
- **Step 1:** Self-adaptive scaling procedure: Given \( f_k \) is not a solution point, find the smallest nonnegative integer \( l_k \) such that \( z_{k+1} = u_k \gamma_k \) and update the non-shortest route flows

\[
j_{k+1}^{p,r} = \max \left\{ 0, j_{k+1}^{p,r} - x_{k+1}^p \right\}, \quad \forall p \in P^r, \quad p \neq p_{r,k}, \quad r \in R, \quad s \in S,
\]

satisfy

\[
(2-\delta)z_{k+1}(f_k - f_{k-1})^T (\nu_k - \nu_{k-1} - \alpha_k z_k) - \alpha_k z_k (\nu_k - \nu_{k-1})^2 \\
\geq \max \left\{ \frac{\alpha_k}{\alpha_{\text{max}}} \left( \| e(f_k, \nu_k) \|, 0 \right) \right\},
\]

where \( e(f_k, \nu_k) = f_k - P_\Omega [f_k - z_k \nu(f_k)] \).
- **Step 2:** Selection of \( \gamma_{k+1} \):
  
  If \( 0.5z_{k+1}(f_k - f_{k-1})^T (\nu_k - \nu_{k-1} - \alpha_k z_k) - \alpha_k z_k (\nu_k - \nu_{k-1})^2 \\
  \geq \max \left\{ \frac{\alpha_k}{\alpha_{\text{max}}} \left( \| e(f_k, \nu_k) \|, 0 \right) \right\},
  \]

  then \( \gamma_{k+1} = \min (\gamma_{k+1}/u, \alpha_{\text{max}}); \) otherwise \( \gamma_{k+1} = 2z_{k+1}. \)

Then, update the shortest route flows

\[
j_{k+1}^{p,r,k} = \min \left\{ j_{k+1}^{p,r,k} - z_{k+1} \gamma_k, \quad \forall r \in R, \quad s \in S,
\]

satisfy

\[
(2-\delta)z_{k+1}(f_k - f_{k-1})^T (\nu_k - \nu_{k-1} - \alpha_k z_k) - \alpha_k z_k (\nu_k - \nu_{k-1})^2 \\
\geq \max \left\{ \frac{\alpha_k}{\alpha_{\text{max}}} \left( \| e(f_k, \nu_k) \|, 0 \right) \right\},
\]

where \( p_{r,k} \) is the shortest route between origin \( r \) and destination \( s \) in the \( k \)th iteration; \( j_{k+1}^{p,r,k} = \eta_{p,r}^{k+1} - \eta_{p,r}^k - \bar{F}_{p,r}^k \), \( \bar{F}_p \) represents the vector \((\ldots, j_{k+1}^{p,r}, \ldots)^T\); \( F_k \) represents the vector \((\ldots, j_{k+1}^{p,r}, \ldots)^T\); \( e(f_k, \nu_k) = f_k - P_\Omega [f_k - z_k \nu(f_k)] \).

and set \( j_{k+1}^{p,r,k} = \min \{ j_{k+1}^{p,r,k}, \quad \forall r \in R, \quad s \in S, \quad p \neq p_{r,k} \}

and \( \eta_{p,r}^{k+1} \).
Theorem 1. The sequence \( \{ f_k \} \) generated by the SAGP algorithm satisfies the following inequality:

\[
|e(f_k, x_k)| \leq |e(f_{k+1}, x_{k+1})| + \Delta_k \left( f_{k+1} - f_k \right).
\]

Theorem 2. The SAGP algorithm is globally convergent.

Remark 1. The main advantages of the proposed SAGP algorithm are twofold:

- Instead of using a fixed or predetermined stepsize (or a monotone decreasing sequence), the self-adaptive stepsize strategy embedded in the SAGP algorithm is non-monotone. In other words, the stepsize can decrease as well as increase by making use of the information derived from previous iterations (i.e., route flows and route costs) during the solution process to determine a suitable stepsize. This self-adaptive feature ensures the algorithm decreases \( e(f_k, x_k) \) per iteration (see Theorem 1 and its proof in the Appendix). In addition, it makes the algorithm insensitive to the initial stepsize, which guarantees the robustness of the algorithm.

- Note that the travel demand conservation constraints are removed in the above flow update Eq. (17). Thus, instead of solving a quadratic program for the projection operation over the feasible set (including the conservation constraints) as in the modified GLP projection method [27,28], the proposed SAGP algorithm needs only a modest computational effort to perform the simple projection operation on a nonnegative orthant.

Remark 2. In the aforementioned Lemma and Theorems, we assume the mapping is Lipschitz continuous with a positive constant \( L > 0 \), and also strongly monotone with a positive modulus \( \mu > 0 \). The self-adaptive scaling procedure (Step 1) will terminate in finite steps per iteration. Moreover, there exists a positive real number, denoted by \( \delta \geq 0 \), such that

\[
x_{k+1} = w_k > 0, \quad \forall k \geq 0.
\]

Theorem 1. Suppose that the mapping is Lipschitz continuous with a Lipschitz constant \( L > 0 \), and also strongly monotone with a positive modulus \( \mu > 0 \). The self-adaptive scaling procedure (Step 1) will terminate in finite steps per iteration. Moreover, there exists a positive real number, denoted by \( \delta \geq 0 \), such that

\[
x_{k+1} = w_k > 0, \quad \forall k \geq 0.
\]

In short, the self-adaptive scheme resolves the messy problem of finding an appropriate stepsize and the simple projection approach avoids the computational burden of solving convex quadratic programs. These two key features make the SAGP algorithm more robust and efficient.

Now, we give the global convergence results of SAGP algorithm. For the detailed proofs, interested readers may refer to the Appendix.

The main advantages of the proposed SAGP algorithm are).

5. Numerical experiments

In this section, we use two networks to show the efficiency and robustness of the proposed SAGP algorithm. First, the Nguyen–Dupuis network is used to compare the self-adaptive stepsize strategy with the commonly used fixed and predetermined stepsize strategies. Then, we use the Winnipeg network to show the robustness and the potential applicability of the SAGP algorithm on real networks. The algorithm is coded in Compaq Visual FORTRAN 6.6 and run on a workstation with Xeon 3.20 GHz processor and 2.00 GB of RAM.

5.1. Nguyen–Dupuis network

The Nguyen–Dupuis network consists of 13 nodes, 19 links, and 4 O–D pairs (see Fig. 1). Network characteristics, O–D demands, and link cost functions can be found in [46]. Since our focus here is to examine the stepsize strategies, we enumerate all the routes in this example (given in Table 1). However, any nonadditive shortest route-generation algorithm (e.g., [26]) can be incorporated to iteratively generate a route set.

The following nonadditive route cost function is adopted for the stepsize strategy comparison:

\[
\eta_p^2 = \eta^2_p + 2 \left( \frac{\sum_{a \in P} \delta^2 p_a}{15} \right) \quad \forall \ p \in P^S, \ r \in R, \ s \in S.
\]

The route-specific costs \( \eta_p^2 \) are added to the routes with dominant flows for each O–D pair found in the additive traffic equilibrium model. Specifically, we add a ten-unit cost to routes 1, 9, 15, and 20. We adopt the user equilibrium (UE) conditions as the stopping criterion.
criterion to monitor the convergence:

$$\max_{i,j} \sum_{p \in P^i} \frac{f_{p,k}^i}{Q^i} \left( \frac{p_{i,k}^j - p_{i,k}^j_{\text{max}}}{p_{i,k}^j_{\text{max}}} \right) \leq \epsilon.$$  (22)

where $\epsilon$ is the tolerance error set at $1e-10$. We define the left-hand-side term of Eq. (22) as the residual error. The maximum number of iterations allowed is 10,000. If the algorithm cannot satisfy the stopping criterion within the maximum number of iterations, it is considered failed. The initial parameters for the SAP2 algorithm are specified as $\delta = 0.2$, $u = 0.6$, $x_{\text{max}} = 99.999$. The initial route flows are specified by equally allocating the travel demand of each O–D pair to all the routes connecting that O–D pair (see Table 1).

In order to show the robustness and efficiency of the embedded self-adaptive scheme, we compare it with the fixed and predetermined stepsize strategies under different initial stepsize values. For the predetermined stepsize strategy, $\{a_k = a_0/k, k \in N\}$ is used as the stepsize sequence. The initial stepsizes are varied from $1e-4$ to $1e+4$ with an interval of one order of magnitude and the computational performance is reported in Table 2. As can be seen from this table, both fixed and predetermined stepsize strategies have difficulty in convergence for most of the initial stepsize values (i.e., cannot converge within 10,000 iterations). However, the self-adaptive stepsize strategy can converge with less than 250 iterations for all initial stepsize values. Thus, the SAP2 algorithm seems to be insensitive to the initial stepsize values in converging to a highly accurate

<table>
<thead>
<tr>
<th>Initial stepsize</th>
<th>Fixed</th>
<th>Predetermined</th>
<th>Self-adaptive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iter Proj</td>
<td>Iter  Proj</td>
<td>Iter Proj</td>
<td>Iter Proj</td>
</tr>
<tr>
<td>1.00e-04</td>
<td>/     /</td>
<td>/</td>
<td>200 266</td>
</tr>
<tr>
<td>1.00e-03</td>
<td>/     /</td>
<td>/</td>
<td>188 224</td>
</tr>
<tr>
<td>1.00e-02</td>
<td>/     /</td>
<td>/</td>
<td>183 224</td>
</tr>
<tr>
<td>1.00e-01</td>
<td>6979 6979</td>
<td>/</td>
<td>198 237</td>
</tr>
<tr>
<td>1.00e+00</td>
<td>704 704</td>
<td>/</td>
<td>247 255</td>
</tr>
<tr>
<td>1.00e+01</td>
<td>/     /</td>
<td>/</td>
<td>181 213</td>
</tr>
<tr>
<td>1.00e+02</td>
<td>/     /</td>
<td>/</td>
<td>225 260</td>
</tr>
<tr>
<td>1.00e+03</td>
<td>/     /</td>
<td>370 370</td>
<td>191 228</td>
</tr>
<tr>
<td>1.00e+04</td>
<td>/     /</td>
<td>1369 1369</td>
<td>239 265</td>
</tr>
</tbody>
</table>

Iter = no. of iterations, Proj = no. of projections, and / = cannot converge within 10,000 iterations.
solution with similar computational efforts. This experiment suggests that the self-adaptive scheme is quite effective in terms of both efficiency and robustness.

In order to further compare the above three stepsize strategies, we select four different cases whose numbers are in bold in Table 2, i.e., fixed stepsize with \( a_0 = 1 \), predetermined stepsize with \( a_0 = 1 \cdot e^{+3} \), self-adaptive stepsize with \( a_0 = 1 \), and self-adaptive stepsize with \( a_0 = 1 \cdot e^{+3} \). The evolution processes of the residual errors under the above four cases are shown in Fig. 2. The performance of the self-adaptive stepsize strategy with both \( a_0 = 1 \) and \( a_0 = 1 \cdot e^{+3} \) are similar and they outperform both the fixed stepsize strategy with \( a_0 = 1 \) and predetermined stepsize strategy with \( a_0 = 1 \cdot e^{+3} \). In this figure, the residual error does not decrease monotonically since it directly measures the maximum violation of the equilibrium conditions.

In the following, we focus on the detailed evolution process of the stepsizes under the above three strategies. For the fixed stepsize strategy, we use the extended gradient projection (EGP) suggested by Scott and Bernstein [26], combining with a trial-and-error approach, to find the range of an appropriate stepsize for this problem. Fig. 3 shows the relationship between the performance of the EGP algorithm (in terms of the number of iterations required to converge) and the stepsize values between 0.2 and 20 with an interval of 0.2. Recall that in the original GP algorithm [24,25], an inverse matrix based on the first derivative of the route cost is used to scale the unit stepsize. However, in the SAGP algorithm, we do not need to calculate this inverse scaling matrix and thus the current stepsize without scaling can be larger than 1.0. From Fig. 3, it is clear that the range of stepsize values that can promise convergence within 10,000 iterations is approximately within 0.2–8.0. However, the performance of the EGP varies quite sporadically since it does not follow a smooth curve (i.e., not unimodal and certain points are not differentiable). This implies that the performance of the EGP algorithm is quite sensitive to the selection of an appropriate fixed stepsize. This phenomenon also occurs to the predetermined stepsize strategy.

Finally, without loss of generality, the self-adaptive stepsize strategy with \( a_0 = 1 \) is demonstrated in Fig. 4. We can observe that the stepsize can increase as well as decrease during the iteration process, which is attributed to Step 2 of the algorithm. According to the information derived from previous iterations, Step 2 automatically adjusts the stepsize for the next iteration, based on which Step 1 self-adaptively selects an appropriate stepsize to update the current solution. It is this feature that guarantees the robustness of the SAGP algorithm.

5.2. Winnipeg network

In this section, we use the Winnipeg network to examine the potential applicability of the proposed SAGP algorithm on a real transportation network. The Winnipeg network, shown in Fig. 5, consists of 154 zones, 2535 links, and 4345 O–D pairs. The network structure, O–D demands, and link performance parameters are from the Emme2 software [47]. A working route set from Bekhor et al. [48,49] is used, where the routes are generated by using a combination of the link elimination method [50] and the penalty method [51]. There are 174,491 routes in total with

![Fig. 4. Evolution process of the self-adaptive stepsize (a_0 = 1).](image-url)

![Fig. 5. Winnipeg network.](image-url)
an average of 40.16 routes per O–D pair. The maximum number of routes generated for any O–D pair is 50. The total travel demand of the network is 54,459 units. The nonadditive route cost function used in this network is as follows:

\[ Z_{rs}^p = 2.0 \left( \sum_{a \in A} t_{a}^{|r,s|}/10 \right)^2 + \left( \sum_{a \in A} t_{a}^{|r,s|}/10 \right) \quad \forall p \in P^s, \ r \in R, \ s \in S. \]  

(23)

We set the initial parameter values as follows: \( \delta = 0.15, \ u = 0.7, \ x_0 = 1.0, \ x_{\text{max}} = 99,999. \) The initial route-flow pattern is set by allocating the travel demand of each O–D pair to the free-flow-travel-time-based shortest route connecting that O–D pair. The residual error used is the same as that in the first network, and the tolerance error \( \epsilon \) is set at \( 1 \times 10^{-4}. \) In addition, we also calculate another commonly used convergence measure (i.e., average excess cost) corresponding to the convergent result based on Eq. (22)

\[ \text{average excess cost} = \frac{\sum_r \sum_s (\pi_{rs} - \pi_{rs}^f) f_{rs}^f}{\sum_r \sum_s f_{rs}^f}. \]  

(24)

(1) Convergence characteristics

The computational efforts under different tolerance errors from \( 1 \times 10^{-1} \) to \( 1 \times 10^{-4} \) with an interval of one order of magnitude are shown in Table 3. As can be seen, the stopping criterion (i.e., Eq. (22)) used in this paper is much stricter than

<table>
<thead>
<tr>
<th>Tolerance error</th>
<th>1e-1</th>
<th>1e-2</th>
<th>1e-3</th>
<th>1e-4</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of iterations</td>
<td>170</td>
<td>1993</td>
<td>4597</td>
<td>16,521</td>
</tr>
<tr>
<td>No. of projections</td>
<td>387</td>
<td>4639</td>
<td>10,715</td>
<td>38,539</td>
</tr>
<tr>
<td>CPU time (s)</td>
<td>33</td>
<td>365</td>
<td>837</td>
<td>2999</td>
</tr>
<tr>
<td>Average excess cost</td>
<td>1.50e-07</td>
<td>1.60e-08</td>
<td>5.01e-10</td>
<td>5.87e-13</td>
</tr>
</tbody>
</table>

Table 3: Computational performance under different tolerance errors.
the average excess cost criterion in Eq. (24). It is worth noting that Boyce et al. [52] suggested that “a relative gap of 0.01% is required to ensure that the traffic assignments are sufficiently converged to achieve link-flow stability”. According to their findings, the relative gap of 1e−4 approximately corresponds to an average excess cost of 1.35e−5 in their example. Thus, the tolerance error of 1e−4 used for the Winnipeg network is sufficiently accurate for traffic assignment applications. In the following, we focus on the tolerance error of 1e−4 for further analysis.

First, we examine the convergence characteristics of the SAGP algorithm. The evolution of the residual error with respect to iteration number and CPU time are shown in Fig. 6. As can been seen, the algorithm has a linear convergence rate, which is to be expected. The evolution process of the stepsize is also shown in Fig. 7. We can observe that the stepsize automatically adjusts itself to determine a suitable stepsize to ensure convergence, which is the one of the main features in the SAGP algorithm.

(2) **Effect of different initial stepsizes**

Second, we investigate the impact of different initial stepsizes \( \alpha_0 \) on the computational performance of the algorithm. For illustration purpose, we compare the computational efforts under seven different initial stepsizes as well as their mean and standard deviation (SD) in Table 4. As can been seen from this table, the number of iterations required to reach convergence, number of projection operations, and CPU time are all insensitive to the values of the initial stepsize. The algorithm is thus capable of reaching the convergent solution with a wide range of initial stepsizes under similar computational efforts.

(3) **Effect of demand levels**

Finally, we investigate the impact of the demand levels on the computational performance of the proposed SAGP algorithm. The demand level is varied from 0.6 to 1.4 with an interval of 0.2. As can be seen from Table 5, the number of iterations (and also number of projections) and CPU time increase with the demand level at a linear rate. In addition, for higher demand levels, more projection operations are needed in some iterations, which is shown by the difference between the number of projections and number of iterations. Moreover, with the increase of the demand level, the range of the stepsize generally decreases (i.e., taking smaller steps due to higher congestion levels).

### Table 5

<table>
<thead>
<tr>
<th>Demand level</th>
<th>0.6</th>
<th>0.8</th>
<th>1</th>
<th>1.2</th>
<th>1.4</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of iterations</td>
<td>8828</td>
<td>11,215</td>
<td>16,521</td>
<td>28,236</td>
<td>34,145</td>
</tr>
<tr>
<td>No. of proj - No. of iter</td>
<td>17,661</td>
<td>26,171</td>
<td>38,539</td>
<td>74,427</td>
<td>89,951</td>
</tr>
<tr>
<td>CPU time (s)</td>
<td>1580</td>
<td>2047</td>
<td>2999</td>
<td>5356</td>
<td>6515</td>
</tr>
<tr>
<td>Maximum stepsize</td>
<td>0.0100</td>
<td>0.0343</td>
<td>0.0343</td>
<td>0.0058</td>
<td>0.0058</td>
</tr>
<tr>
<td>Minimum stepsize</td>
<td>0.0013</td>
<td>0.0033</td>
<td>0.0011</td>
<td>0.0001</td>
<td>0.0001</td>
</tr>
<tr>
<td>Range of stepsize</td>
<td>0.062</td>
<td>0.0310</td>
<td>0.0332</td>
<td>0.0057</td>
<td>0.0057</td>
</tr>
</tbody>
</table>

6. **Concluding remarks**

Gradient projection algorithm has recently been extended to solve the nonadditive traffic equilibrium problem. However, it is a nontrivial issue to choose an appropriate stepsize, which could significantly affect the convergence performance of the algorithm. Several different strategies for choosing the stepsize are discussed and a self-adaptive gradient projection (SAGP) algorithm is proposed. The advantage of the proposed algorithm is its simplicity and robustness. The embedded self-adaptive scheme can automatically adjust itself to find an appropriate stepsize according to the information derived from previous iterations to ensure convergence. Thus, the expensive and inefficient line search based on the “trial-and-error” approach could be avoided. Furthermore, by redefining the feasible route-flow set in terms of the non-shortest route flows, the SAGP algorithm avoids the computational burden of solving convex quadratic programs per projection. In contrast, only some simple projection operations on a nonnegative orthant are needed. Numerical results indicated that the proposed SAGP algorithm is efficient in achieving a very accurate solution and is also robust with respect to different initial stepsize values.

For future research, the performance of the SAGP algorithm needs to be further examined by using various networks with different topological characteristics. In addition, in this study, we assume the route set is known *a priori* which can be obtained using a choice set generation method [53,54]. For large-sized problems, it is more suitable to use a route-generation procedure (e.g., [2,19]) that embeds a nonadditive shortest path problem (e.g., [55–57]) to generate routes on a need basis in each iteration. We will explore these issues in our future research.

### Table 4

<table>
<thead>
<tr>
<th>Initial stepsize</th>
<th>0.3</th>
<th>0.5</th>
<th>0.8</th>
<th>1.0</th>
<th>3.0</th>
<th>5.0</th>
<th>8.0</th>
<th>Mean</th>
<th>SD</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of iterations</td>
<td>16,606</td>
<td>16,692</td>
<td>16,480</td>
<td>16,521</td>
<td>16,734</td>
<td>16,331</td>
<td>16,323</td>
<td>16,527</td>
<td>163</td>
</tr>
<tr>
<td>No. of projections</td>
<td>38,741</td>
<td>38,945</td>
<td>38,454</td>
<td>38,539</td>
<td>39,050</td>
<td>38,098</td>
<td>38,041</td>
<td>38,553</td>
<td>391</td>
</tr>
<tr>
<td>CPU time (s)</td>
<td>3038</td>
<td>3049</td>
<td>3027</td>
<td>2999</td>
<td>3075</td>
<td>3015</td>
<td>2994</td>
<td>3028</td>
<td>29</td>
</tr>
</tbody>
</table>

Fig. 7. Evolution process of the stepsize.
Acknowledgments

The authors would like to thank the editor and an anonymous referee for their constructive comments for improving the paper. This research was supported by the NSF CAREER Grant CMS-0134161.

Appendix

Before presenting the global convergence of the SAGP algorithm, we use the basic idea of the gradient projection algorithm to decompose the route-flow vector in the original VI problem into two parts: the shortest and non-shortest routes, as shown in Eq. (25). This manipulation will facilitate performing the projection operation in the proposed solution algorithm.

\[ \eta(t) \rangle_0, \forall t \in \Omega \] 

\[ \Rightarrow \sum_{i=1}^n \sum_{j=1}^n \eta_{ij}(t) \left( f_i - f_j \right) \geq 0 \]

\[ \Rightarrow \sum_{i=1}^n \sum_{j=1}^n \eta_{ij}(t) \left( f_i - f_j \right) + \sum_{i=1}^n \sum_{j=1}^n \eta_{ij}(t) \left( f_j - f_i \right) \geq 0 \]

\[ \Rightarrow \sum_{i=1}^n \sum_{j=1}^n \eta_{ij}(t) \left( f_i - f_j \right) - \sum_{i=1}^n \sum_{j=1}^n \eta_{ij}(t) \left( f_j - f_i \right) \geq 0 \]

\[ \Rightarrow \sum_{i=1}^n \sum_{j=1}^n \eta_{ij}(t) \left( f_i - f_j \right) \geq 0. \] (25)

We further use \( \bar{t} \) to denote the vector of non-shortest route flows. Note that the shortest route flow can be obtained from the conservation constraint for each O-D pair. Then, Eq. (25) can be rewritten in the following compact form:

\[ \bar{f} \left( \eta(t) \right) \geq 0, \forall t \in \Omega = R_{\alpha}^{1} \left( R_{\beta} \left( S \right) \right) \]

where \( \bar{f}(\eta) \left( \eta_{ij}(t) \right) = \eta_{ij}(t) \left( f_i - f_j \right), \forall t \in t_t, p \neq p_t, r \in R, s \in S \) is the mapping of the VI problem. After reformulation, the feasible route-set of the new VI problem just consists of the nonnegative orthant \( R_{\alpha}^{1} \left( R_{\beta} \left( S \right) \right) \). Note that similar reformulation scheme has also been proposed in the MP formulation (see [58]).

Proof of Lemma 1. Based on Step 2, we have \( \eta_k \leq \eta_{max}, \forall k \geq 0 \). Recall that the mapping is assumed to be Lipschitz continuous with a Lipschitz constant \( L > 0 \). In addition, it is strongly monotone with a positive modulus \( \mu > 0 \). Thus, we have the following two inequalities:

\[ \| f_k - f_{k+1} \| \leq L \| \bar{f}_k - \bar{f}_{k+1} \| \] (27)

\[ \| \bar{f}_k - \bar{f}_{k+1} \| \leq \mu \| f_k - f_{k+1} \| \] (28)

Combining Eqs. (27) and (28), we have

\[ (2-\delta)z_{k+1} \left[ f_k - f_{k+1} \right] \leq \| f_k - f_{k+1} \| \leq \mu \| f_k - f_{k+1} \| \]

\[ \Rightarrow (2-\delta)z_{k+1} \leq \left( \delta - \mu \right) \left( f_k - f_{k+1} \right)^{2} \]

This means that

\[ (2-\delta)z_{k+1} \left[ f_k - f_{k+1} \right] \leq \| f_k - f_{k+1} \| \leq \left( \delta - \mu \right) \left( f_k - f_{k+1} \right)^{2} \]

\[ \Rightarrow \min \left\{ \left( \delta - \mu \right) \left( f_k - f_{k+1} \right)^{2}, 0 \right\} \]

\[ \leq \left( \delta - \mu \right) \left( f_k - f_{k+1} \right)^{2} \]

will be automatically satisfied if \( z_k \leq \min \{(2-\delta)\mu/L^{2}, \eta_k \} \). Based on the facts that \( \gamma_k \leq \gamma_{max} \) and \( \lim_{k \to \infty} u_k = 0 (u \in [0, 1]) \), the self-adaptive scaling procedure (i.e., Step 1) will terminate within finite steps in the inner loop. \( \bar{u} \) is the smallest nonnegative integer that satisfies Eq. (30). Thus, we have

\[ z_{k+1} \geq z_{min} : = \min \left\{ \left( \frac{2-\mu u}{L^{2}} \right), \eta_k \right\} > 0. \] (31)

Proof of Theorem 1. According to the definition of \( e(f_{k+1}, z_{k+1}) \) and the next iterative solution \( f_{k+1}, \) we have

\[ \| e(f_{k+1}, z_{k+1}) \|^2 = \| f_k - f_{k+1} - 2z_{k+1}(f_k - f_{k+1}) \|^2 \]

\[ = \| f_k - f_{k+1} - 2z_{k+1}(f_k - f_{k+1}) \|^2 \]

\[ \leq \| f_k - f_{k+1} - 2z_{k+1}(f_k - f_{k+1}) \|^2 \]

\[ + \left( z_{k+1} \right)^2 \| f_k - f_{k+1} \|^2 \]

\[ \leq \left( \delta - \mu \right)\left( f_k - f_{k+1} \right)^{2} \]

(32)

where the first inequality comes from the non-expansive property of the projection operator. From Eq. (30), we have the following two cases:

Case 1: If \( z_{k+1} \leq z_{k} \), then

\[ (2-\delta)z_{k+1} \left[ f_k - f_{k+1} \right] \leq \| f_k - f_{k+1} \| \leq \mu \| f_k - f_{k+1} \| \]

\[ \Rightarrow (2-\delta)z_{k+1} \left[ f_k - f_{k+1} \right] \leq \| f_k - f_{k+1} \| \]

\[ \leq \left( \delta - \mu \right) \left( f_k - f_{k+1} \right)^{2} \]

(33)

Substituting Eq. (33) into Eq. (32), we obtain

\[ \| e(f_{k+1}, z_{k+1}) \|^2 \leq \| f_k - f_{k+1} \|^2 - \delta \left( z_{k+1} \right)^2 \left( f_k - f_{k+1} \right)^{2} \]

\[ \leq \left( \delta - \mu \right) \left( f_k - f_{k+1} \right)^{2} \]

(34)

where the equality uses the definitions of both \( e(f_{k+1}, z_{k+1}) \) and the next iterative solution \( f_{k+1}, \) and the last inequality adopts the basic inequality of \( e(f_{k}, z_{k}) \).

Case 2: If \( z_{k+1} > z_{k} \), then

\[ (2-\delta)z_{k+1} \left[ f_k - f_{k+1} \right] \leq \| f_k - f_{k+1} \| \leq \mu \| f_k - f_{k+1} \| \]

\[ \Rightarrow (2-\delta)z_{k+1} \left[ f_k - f_{k+1} \right] \leq \| f_k - f_{k+1} \| \]

\[ \leq \left( \delta - \mu \right) \left( f_k - f_{k+1} \right)^{2} \]

(35)

Substituting Eq. (35) into Eq. (32), we obtain

\[ \| e(f_{k+1}, z_{k+1}) \|^2 \leq \| f_k - f_{k+1} \|^2 - \delta \left( z_{k+1} \right)^2 \left( f_k - f_{k+1} \right)^{2} \]

\[ = \left( \delta - \mu \right) \left( f_k - f_{k+1} \right)^{2} \]

(36)

where the last inequality comes from the basic inequality of \( e(f_{k}, z_{k}) \). Thus, in both cases, inequality (20) is satisfied. □

Proof of Theorem 2. From inequality (20) and the strongly monotone condition (28), we have

\[ \| e(f_{k+1}, z_{k+1}) \|^2 \leq \| e(f_{k}, z_{k}) \|^2 - \delta \mu z_{k+1} \left( f_k - f_{k+1} \right)^{2} \]
\[ ||e(f_k, x_k) ||^2 = ||e(f_k, x_k) -\delta x_k z_{k+1} ||^2 = ||e(f_k, x_k) -\delta x_k z_{k+1} - (f_k-x_k) f_k ||^2 \]

(37)

where the definitions of \( f_{k+1} \) and \( e(f_k, x_k) \) are used in the first and second equalities, respectively. Since the non-negativity of the term \( \delta x_k z_{k+1} \) and \( ||e(f_k, x_k)||^2 \), the above inequality means that

\[ ||e(f_{k+1}, x_{k+1}) ||^2 \leq ||e(f_k, x_k) ||^2 \leq \cdots \leq ||e(f_0, x_0) ||^2 . \]

(38)

Thus, the sequence \( ||e(f_k, x_k)|| \) is bounded. Then, according to Lemma 2.2 in [27], there exists a positive constant \( c \), such that

\[ ||f_k - f_s|| \leq c ||e(f_k, x_k)|| . \]

(39)

This means that the solution sequence \( f_k \) is also bounded. Hence, summation of inequality (20) can result in

\[ \sum_{k=0}^{\infty} \lambda_k ||e(f_k, x_k)|| < +\infty . \]

(40)

Since \( \lambda_k \geq \lambda_{min} > 0 \), \( \forall k \geq 0 \) (Lemma 1), it follows from the basic inequality of \( e(f_k, x_k) \) that

\[ \lambda_k e(f_k, x_k) \leq \lambda_k e(f_k, x_k) . \]

(41)

Considering the strongly monotone condition, it follows from Eqs. (40) and (41) that

\[ \sum_{k=0}^{\infty} \lambda_k ||e(f_k, x_k)|| \leq \sum_{k=0}^{\infty} \lambda_k + \lambda_k ||e(f_k, x_k)|| < +\infty . \]

\[ \Rightarrow \sum_{k=0}^{\infty} ||e(f_k, x_k)|| < +\infty . \]

(42)

Since \( f_k \) is bounded, it has at least one cluster point. Let \( f^c \) denote a cluster point of the sequence \( f_k \), and \( f_k, f_k \) denote the corresponding sub-sequence that converges to the cluster point \( f^c \). According to the continuity of the sequence \( ||e(f_k, x_k)|| \) and Eq. (42), we have

\[ ||e(f^c, x_{min})|| = \lim_{k \rightarrow +\infty} ||e(f_k, x_{min})|| = 0 . \]

(43)

This means that \( f^c \) is a solution to the VI problem. Due to the strong monotonicity of the mapping, there is only one solution to the VI problem. Therefore, the sequence \( f_k \) converges to the unique solution \( f^c \). □

References


