Optimizing Network Economics Problem with Adaptive Algorithms for Variational Inequalities

Vladimir Semenov [1] and Serhii Denysov [,1,2]

[1] *Taras Shevchenko National University of Kyiv, 64/13 Volodymyrska Street, Kyiv, 01161, Ukraine*
[2] *Harbour.Space University, Carrer de Rosa Sensat 9-11, 08005, Barcelona, Spain*

Abstract

Effectiveness of adaptive extragradient algorithms for network economics problems is demonstrated with the modified model of blood supply chain network. Informational system for comparing behavior of algorithms for solving variational inequalities is described. Described algorithms include adaptive modifications of extrapolation from the past, forward-backward-forward and Tseng method. The blood supply chain model is a prominent example of more general perishable products delivery chain optimization problem. The provided software system enables users to edit model parameters, visualize networks, and solve path-based cost minimization problems using a selected subset of these algorithms.

Keywords [1]

Variational Inequality, Network Economics, Blood Supply Chain, adaptive algorithms, numerical experiments, optimization

1. Introduction

Variational inequality (VI) is a powerful approach to modelling diverse set of problems, including, but not limited to, optimization and equilibrium problems [1-4]. Between other applications, last years they show promising results in some machine learning areas – especially for GANs and other adversarial techniques [5,6]. And, finally, a family of network economics problems could be naturally formulated as a VI. Such formulation was obtained for different kind of network economics problems in the series of papers by Anna Nagurney [7,8].

Problem of special interest is modelling a delivery chain of perishable products [9], where it's unfeasible to store the product for a long time, and, as a result, product surplus in a demand point incurs serious discard charges. Moreover, every step of operation and transport of such kind of a product usually leads to some loss, also with corresponding discard charges. The similar model arises if we consider blood delivery supply chain, organized as sequential interaction of blood collection, processing and consuming facilities – from collection points to hospitals. Modelling of this problem for the structure of American Red Cross (ARC) was investigated in [10], with proposal of the VI formulation. Another VI model is used as part of blood supply chain optimization as bi-objective model in [11].

Intensive development of numerical algorithms for solving VI started from extragradient algorithm [12] and alternative algorithm [13] (now widely known as "extrapolation from the past", or EFP, in machine learning area). As operator calculation and projection to feasible set can be really computationally expensive operations for medium to large sized problems, efforts have been made to build algorithms that require minimum necessary amount of such operations.

Also, VI solving algorithms often require a priory knowledge of the Lipschitz constant for the operator to determine algorithm step size – and that becomes a serious problem for applying them to real world problems, as calculating it could be complicated or almost impossible. To tackle this difficulty, adaptive versions of VI algorithms were proposed, that allow to modify step size dynamically during algorithm execution while preserving theoretical guarantee of the convergency.

Dynamical System Modeling and Stability Investigation (DSMSI-2023), December 19-21, 2023, Kyiv, Ukraine
EMAIL: denisov.univ@gmail.com (S. Denysov); semenov.volodya@gmail.com (V. Semenov)
ORCID: 0000-0003-4397-4617 (S. Denysov); 0000-0002-3280-8245 (V. Semenov)

2. Mathematical model for optimization of blood supply chain

Let's us formulate the mathematical model of blood supply chain optimization, following [10].

2.1. Blood supply process and facilities

Structure of the facilities and corresponding logistic can be schematically represented as the following graph:

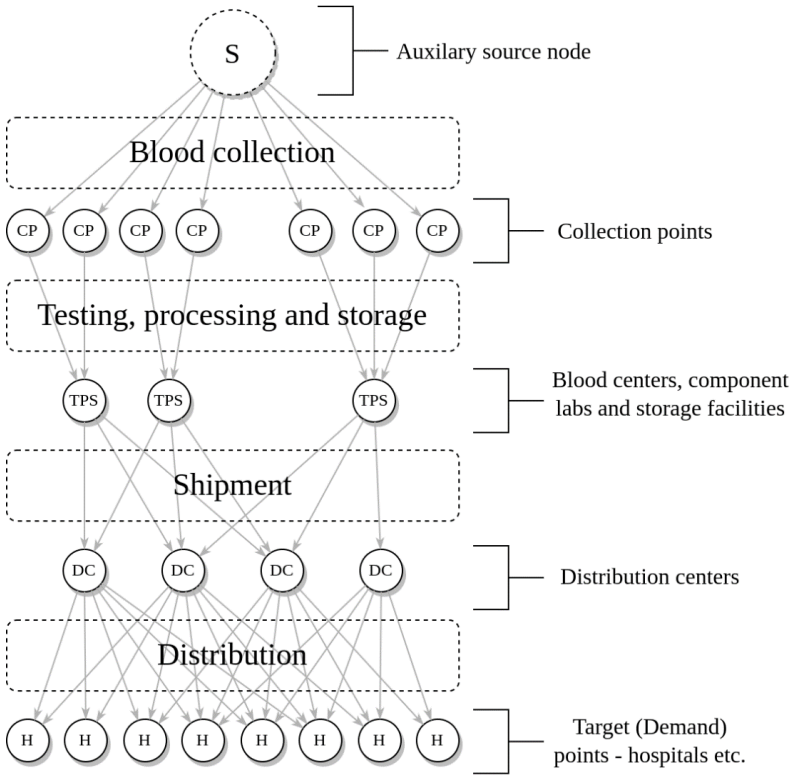

Figure 1: Blood supply network – testing, processing and storage layers merged

In the graph above, edges represent different operations in the blood supply process – and corresponding model parameters, such as collection risks or operational and waste discard costs, are bound to the edges. Nodes represent facilities taking part in the supply chain. The first node is added for the model to be concise, and has no specific meaning – but can be considered as a top-level organization structure unit, e.g., Red Cross regional division, as in [10].

The process starts with blood collection. Specifics of the collection layer is that edges have risk parameter and corresponding costs associated with it, as the process itself is risky – donors can miss appointments because of many reasons – for example, heavy rain greatly decreases visits number. The next stages are testing for infection and contamination, separation of components (plasma and red cells), and storing components in special facilities. In the graph above, blood centers, component labs, and storage facilities are shown as a single layer to make drawing and model description clearer. Actually, it's often the real case – and even if they are really separated, it does not incur significant model changes, as all edges have the same type of associated parameters.

Mathematical properties of the shipment stage edges also coincide with parameters from the testing, processing, and storage stage – but physical facilities are different, so we put them in a separate layer in the graph.

And the last, distribution, layer, corresponds to the last-mile delivery of the blood to demand points, which usually are hospitals. These nodes have a specific role in the model – a stochastic demand and corresponding surplus and shortage penalties are associated with them.

2.2. Mathematical model notation

Let's denote n_c, n_p, n_s, n_h – number of each type of nodes: collection, testing/processing/storage, shipment and demand (hospitals). And let's denote set of all edges as $\{e_j\}, j = 1..n_e$, where n_e is the total number of edges (links) in our supply chain network. And let's denote in-flow on the edge e_j as y_j (we will describe loss and out-flow later). Edge flows together form a vector $y = (y_1, \dots, y_{n_e})$.

The operations with the blood itself have costs, which also can be associated with the edges. Let's denote $c_j(y_j)$ – unit operational cost on edge e_j (it's dependent on the flow). Then the total operational cost on the edge is $y_j \cdot c_j(y_j)$.

Let's denote $P = \{p_i\}, i = 1..m$ – set of all simple paths from dummy source node to every demand point, where m is a total number of such paths. Initial flow on path $p_i = \{e_{i_1}, e_{i_2}, \dots, e_{i_{k_i}}\}$ is denoted as $x_i \geq 0$, and it is the initial flow of the first edge of the path, which is always edge between the source node and a blood collection node. Our goal is to find optimal path flows vector $x = (x_1, \dots, x_m)$. The first obvious property of the feasible set C is

$$x_i \geq 0, i = 1..m \quad (1)$$

Each stage of blood processing could incur some losses – for example, small part of blood is taken to test facility and discarded afterward. Let's associate loss multiplier $\alpha_j \in (0,1]$ with every edge $e_j, j = 1..n_e$. We interpret it the next way: if an edge incoming flow (in-flow) is y_j, then it's outcome (out-flow) is $\alpha_j y_j$. In this paper we assume, that multipliers are independent of link flow (thou it will be interesting to consider depending case in the future).

Losses during blood processing incur waste disposal cost. It can be significant for such kind products, as blood – so it needs to be included into the model. Waste amount for the edge e_j depends only on the edge flow and is equal to $(1 - \alpha_j)y_j$, so waste discard function is also flow dependent – let's denote unit discard cost as $w_j(y_j)$, and the total one will be $y_j \cdot w_j(y_j)$.

In terms of path flows, total loss on the path p_i can be calculated as $\beta_i = \prod_{j:e_j \in p_i} \alpha_j, i = 1..m$. As a result, if initial flow on the path p_i is x_i, corresponding final path flow is $\beta_i x_i, \forall\, i = 1..m$. And for each destination node $H_k, k = 1..n_h$, let's denote the supply amount as s_k. Obviously, we have

$$s_k = \sum_{p_i \in P_k} \beta_i x_i, \quad (2)$$

where P_k is a set of all paths from source node to demand node H_k. At the same time, real demand d_k is stochastic – we cannot know exact blood demand in a hospital for every moment in future. But we assume we know it's probability density function $F_k(t)$ and probability distribution $P_k(z) = p(d_k < z) = \int_0^z F_k(t)dt$.

We are interested in minimizing expected difference between s_k and d_k – so, it's tempting to use something like $E\left(\sum_{k=1}^{n_h}(s_k - d_k)^2\right)$ as the loss function. But consequences of blood shortage and surplus are essentially different, and it makes sense to consider them separately in the model.

Let's denote blood shortage (undersupply) at the demand node H_k as

$$U_k = \begin{cases} d_k - s_k, d_k > s_k \\ 0, d_k \leq s_k \end{cases} = \max\{0, d_k - s_k\},$$

and blood surplus (oversupply) at the same node as

$$V_k = \begin{cases} s_k - d_k, d_k < s_k \\ 0, d_k \geq s_k \end{cases} = \max\{0, s_k - d_k\}.$$

Now corresponding part of the cost function for the model can be formulated the next way:

$$\sum_{k=1}^{n_h} \left(\gamma_{u_k} E(U_k) + \gamma_{v_k} E(V_k)\right)$$

where γ_{u_k}, γ_{v_k} are the penalties (costs), and $E(U_k)$, $E(V_k)$ – mathematical expectations of blood shortage and surplus correspondingly.

And, finally, it's preferrable for the model to count for the blood collection risks. These risks are associated with the first layer edges only, and let's assume they are also functions of the flow. Let's denote risks as $r_i(y_i), i = 1..n_c$. Risk minimization and cost minimization are different criteria, so in the cost function they can be weighted differently – let θ be the weight for risk part, and let cost part have weight of one.

It is worth making remarks about edge indices.

Remark 1: As the blood collection process has sequential stages, it's always possible to enumerate edges in such a way, that any edge from a later stage has an index greater than the index of any other edge from an earlier stage. E.g., an index of a distribution edge is always greater than an index of a collection edge. We assume such kind of numeration in all following reasoning to simplify the model formulation. As a result, edge indices in any path $p_i \in P$ make an increasing sequence.

Taking into account the remark above, each edge flow can be calculated by path flows:

$$y_j = \sum_{i:e_j \in p_i} x_i \alpha_{ij}, \tag{3}$$

where $\alpha_{ij} = \prod_{k:e_k \in p_i, k<j} \alpha_k$ – product of all loss multipliers of edges, which are preceding edge e_j in the path p_i.

Remark 2: We have exactly n_c collection edges on the first layer, so all edges $e_j, j = 1..n_c$ are collection edges (it gives us a simple way to write risk component of the cost function below).

2.3. Optimization problem and variational inequality

Now we can formulate the supply chain network optimization problem as minimization of the combined cost function

$$\Phi(\bar{y}) = \sum_{j=1}^{n_e} \left(y_j \left(c_j(y_j) + w_j(y_j) \right) \right) + \sum_{k=1}^{n_h} \left(\gamma_{u_k} E(U_k) + \gamma_{v_k} E(V_k) \right) + \theta \sum_{j=1}^{n_c} r_j(y_j) \tag{4}$$

with regards to (1) – (3). And using (3), we can reformulate (4) in terms of path flows:

$$\Phi(\bar{x}) = \sum_{i=1}^{m} x_i (C_i(x) + W_i(x) + \theta R_i(x)) + \sum_{k=1}^{n_h} \left(\gamma_{u_k} E(U_k) + \gamma_{v_k} E(V_k) \right) \tag{5}$$

where C_i and W_i are unit operation and waste discard unit cost functions for path p_i, and R_i is the risk cost function for the same path. These functions have the next form:

$$C_i(x) = \sum_{j:e_j \in p_i} c_j(y_j);\ W_i(x) = \sum_{j:e_j \in p_i} w_j(y_j);\ R_i(x) = \sum_{j:e_j \in p_i, j=1..n_c} r_j\left(y_j\right)$$

where y_j is expressed in terms of path flows with (3). Notation is a bit different from [10] to make algorithm implementation more straightforward, as a path edges sequence will be used in calculations.

The problem can be formulated as a classic variational inequality. We need to find $x^* \in C$ such, that

$$\left(x - x^*,\ \frac{d\Phi}{dx}(x^*) \right) \geq 0\ \forall \mathrm{x} \in C \tag{6}$$

where $C = R_+^m$ and

$$\frac{d\Phi}{dx_i} = \sum_{j:e_j \in p_i} \alpha_{ij} \left[c_j\left(y_j\right) + w_j\left(y_j\right) + y_j \left(c_j'\left(y_j\right) + w_j'\left(y_j\right) \right) \right] + \beta_i \left(\lambda_{u_{t_i}} \left(P_{t_i}(v_{t_i}) - 1 \right) + \lambda_{v_{t_i}} \left(P_{t_i}(v_{t_i}) \right) \right) + \theta \left(r_{s_i}\left(y_{s_i}\right) + r_{s_i}'\left(y_{s_i}\right) \cdot y_{s_i} \right)$$

where s_i and t_i are indices of the first and the last edge in the path p_i. Again, here y_j is expressed in terms of path flows, using equality (3).

3. Extragradient algorithms for variational inequalities

Variational inequality (6) is a good example of a real-world problem, where effectiveness of algorithms could be tested, and the behavior could be compared. Let as provide necessary information about the algorithms used for the numerical experiments.

3.1. Preliminaries

At first, we need to introduce some notation. Let H be real Hilbert space with dot product $(,)$ and induced norm $||\cdot||$, C be a non-empty convex closed subset of H, and let's $A: H \to H$ be a mapping.

Definition 1: Mapping $A: H \to H$ is called monotone, if

$$(Ax - Ay, x - y) \geq 0 \ \forall \mathrm{x}, \mathrm{y} \in \mathrm{H}$$

Definition 2: The following problem is called variational inequality (VI):

$$\text{find } x \in C : (Ax, y - x) \geq 0 \ \forall \mathrm{y} \in C \tag{7}$$

Further algorithm formulations will be done for problem (7) with assumption that operator A is monotone and uniformly continuous on any bounded set, and solutions set of the VI (7) is not empty. Actually, for finite-dimensional space H it's enough for operator to be monotone and continuous. Also, formulations will use projection mapping with the next notation:

Definition 3: The mapping $P_C: H \to H$ is called metric projection to closed convex subset $C \subset H$, if $\forall x \in H$ $P_C x$ is the only element of C, for which

$$\|\mathrm{P_C x} - \mathrm{x}\| = \min_{\mathrm{z} \in \mathrm{C}} \|z - x\|.$$

The main idea behind the big family of projection methods for solving (7) is the result, that $x \in H$ is the solution of (7) if and only if x is a fixed point of $P_C(I - \gamma A)$. So, gradient projection method with the next computational procedure could be used for solving a VI:

$$x_{n+1} = P_C(x_n - \gamma A x_n), \text{ where step size } \gamma > 0.$$

But monotonicity of A is not enough for the procedure above to converge to the solution of the VI. It needs strong monotonicity or inversely strong monotonicity (co-coercivity). So, more advanced algorithms were proposed, which do not require extra assumptions on A – especially the extragradient kind of algorithms.

Historically the first extragradient algorithm, proposed in [12], has the next computational procedure:

$$\begin{cases} y_n = P_C(x_n - \gamma A x_n) \\ x_{n+1} = P_C\left(x_n - \gamma A y_n\right) \end{cases}, \tag{8}$$

It was proved, that for monotone and continuous $A: R^m \to R^m$ the sequence $\{x_n\}$, generated by procedure above, converges to the solution of (7), if step size $\gamma \in \left(0, \frac{1}{L}\right)$, where L is a Lipschitz constant of the mapping A. This algorithm requires two mapping calculations and two metric projections on every step, and also initially convergence was proven only for finite dimensional Euclidean space – but it provides strong baseline to compare with.

Later a lot of improvements and modifications were done by different authors – to avoid extra calculations, prove convergence under weaker assumptions, drop the requirement to know the Lipschitz constant in advance, or use Bregman divergence instead of Euclidean distance to speed up projection – see [14-20]. Part of such results were obtained by the research group from Taras Shevchenko National University of Kyiv, to which the current paper's authors belong. One more important direction is development of parallelized versions of VI algorithms, as in [21].

3.2. Adaptive modification of extragradient algorithms

Let us describe other selected algorithms, which are implemented as part of the system.

The first method was proposed in [13] in 1980, and gained popularity nowadays under the name extrapolation from the past. It has the next step calculation procedure:

$$\begin{cases} x_{n+1} = P_C(x_n - \gamma A y_n), \\ y_{n+1} = P_C(x_{n+1} - \gamma A y_n), \end{cases} \quad (9)$$

where convergence is proved with $\gamma \in \left(0, \frac{1}{3L}\right)$ for finite dimensional space. This procedure uses only one mapping calculation on the step, but with two projections. The weak convergence in infinite-dimensional space was proved in [20], where also modification with single projection and auxiliary hyperplane projection was proposed.

Another interesting method was proposed in [14] by P. Tseng in year 2000. The step formula has the next form:

$$\begin{cases} y_n = P_C(x_n - \gamma A x_n), \\ x_{n+1} = y_n - \gamma(A y_n - A x_n), \end{cases} \quad (10)$$

where weak convergency is proved for $\gamma \in \left(0, \frac{1}{L}\right)$. Here we have one projection and two mapping calculation on every step.

And, finally, algorithm with an elegant computational procedure was proposed by Malitsky and Tam in [15]. The step calculation is:

$$x_{n+1} = P_C(x_n - \gamma A(2x_n - x_{n-1}))$$

in case of linear mapping A, and

$$x_{n+1} = P_C(x_n - \gamma A x_n - \gamma(A x_n - A x_{n-1})) \quad (11)$$

in generic case. The algorithm converges with $\gamma \in \left(0, \frac{1}{2L}\right)$. Here we need only one mapping calculation and one projection on step.

All methods above in their original form require knowledge of Lipschitz constant of the mapping A. Let's describe the adaptive modifications, where step size γ will be modified during algorithm run to achieve convergency without using a priori known L, which can be hard to calculate.

Here is the adaptive version of extrapolation from the past algorithm [17]. At first, we select $x_0, y_0 \in C$, $\tau \in \left(0, \frac{1}{3}\right)$, $\gamma_0 > 0$. And starting from $n = 0$ we use the next iteration procedure:

$$\begin{cases} x_{n+1} = P_C(x_n - \gamma_n A y_n), \\ y_{n+1} = P_C(x_{n+1} - \gamma_n A y_n), \\ \gamma_{n+1} = \begin{cases} min\left\{\gamma_n, \tau \dfrac{\|y_{n+1} - y_n\|}{\|A y_{n+1} - A y_n\|}\right\}, & if\ A y_{n+1} \neq A y_n \\ \gamma_n, & otherwise \end{cases} \end{cases}$$

with a stop condition $x_{n+1} = x_n = y_n$.

Tseng algorithm can also be modified for adaptive step size calculation [18]. Let $x_1 \in C$, $\tau \in (0,1)$, $\gamma_1 > 0$. And starting from $n = 1$ we have:

$$\begin{cases} y_n = P_C(x_n - \gamma_n A x_n), \\ x_{n+1} = y_n - \gamma_n(A y_n - A x_n), \\ \gamma_{n+1} = \begin{cases} min\left\{\gamma_n, \tau \dfrac{\|x_n - y_n\|}{\|A x_n - A y_n\|}\right\}, & if\ A x_n \neq A y_n, \\ \gamma_n, & otherwise \end{cases} \end{cases}$$

with the stop condition $x_n = y_n$.

And here is an adaptive modification of Malitsky and Tam algorithm [17]. Let $x_0, x_1 \in H$, $\gamma_0, \gamma_1 > 0$, $\tau \in \left(0, \frac{1}{2}\right)$. Step is the next (again, starting from $n = 1$):

$$\begin{cases} x_{n+1} = P_C\left(x_n - \gamma_n A x_n - \gamma_{n-1}(Ax_n - Ax_{n-1})\right), \\ \gamma_{n+1} = \begin{cases} min\left\{\gamma_n, \tau\dfrac{\|x_{n+1} - x_n\|}{\|Ax_{n+1} - Ax_n\|}\right\}, if\ Ax_{n+1} \neq Ax_n, \\ \gamma_n, \qquad otherwise. \end{cases} \end{cases}$$

Remark 3: For all adaptive versions above, γ update procedure does not incur extra mapping calculation – all values will be cached and reused in algorithms implementation, so number of mapping calculations and projections is the same for both adaptive and stationary variants.

4. Software for solving blood supply chain optimization problem

To allow solving blood supply chain optimization problem (5), formulated as VI (6), it was added to our numerical experiments software suite for VI algorithms. The system allows to plug subset of implemented algorithms to solve one of implemented problems and analyze algorithms behavior.

4.1. Brief software description

Basically, all VI based problems have common parts, which are used by algorithms – and the same is true for extragradient algorithms family. A problem should have constraints with projecting operation, and provide the mapping calculation routine. On the other side, big part of an algorithms' parameters is common for all algorithms – e.g., starting point or initial step size. And some parameters are more specific – like the second starting point or the adaptive step size calculation multiplier. That is reflected inside the system with two class hierarchies – for algorithms and problems correspondingly.

The simplified schema is shown on the Figure 2. Test suite is responsible for running selected set of algorithms for the problem and saving run history (time and all problem and algorithm parameters on every iteration). It also interacts with graphing component (now shown on the figure), to draw convergency rate graphs according to different metrics.

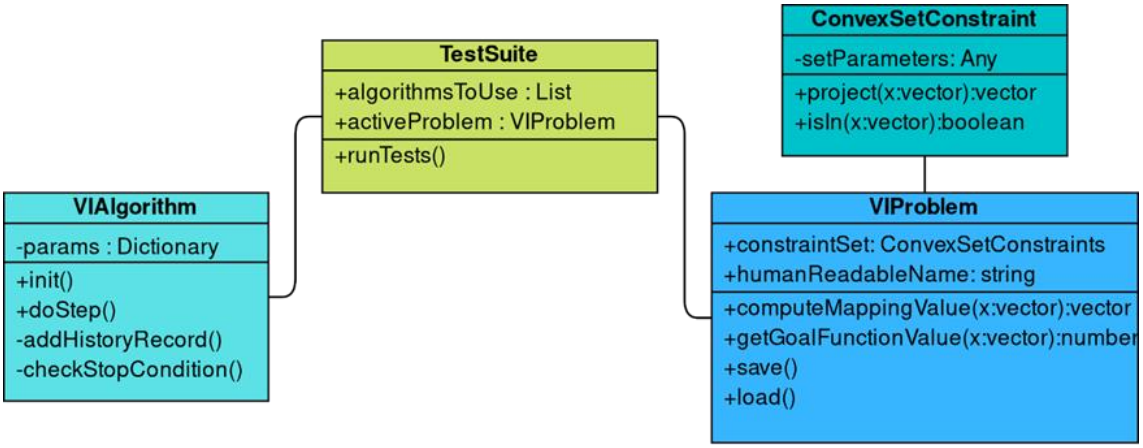

Figure 2: simplified class diagram

The system[2] contains Python 3 implementation of algorithms mentioned above, both adaptive and static modifications. Also, more than ten test problems implemented withing the architecture above, from simple test problems to more real problems, such as PageRank calculation or traffic network equilibrium search.

The functionality can be used from code and a command line (selecting problem, algorithms set and run parameters), which is convenient enough for researchers with python programming skills. For blood supply chain problem, we added visual interface for problem editing, running the algorithms and obtaining the results. The problems selector and editor UI (Figure 3 below) is implemented as a web application using Plotly Dash framework [22].

[2] Source code repository: https://github.com/compmath-sdeni/vi-alg-suite

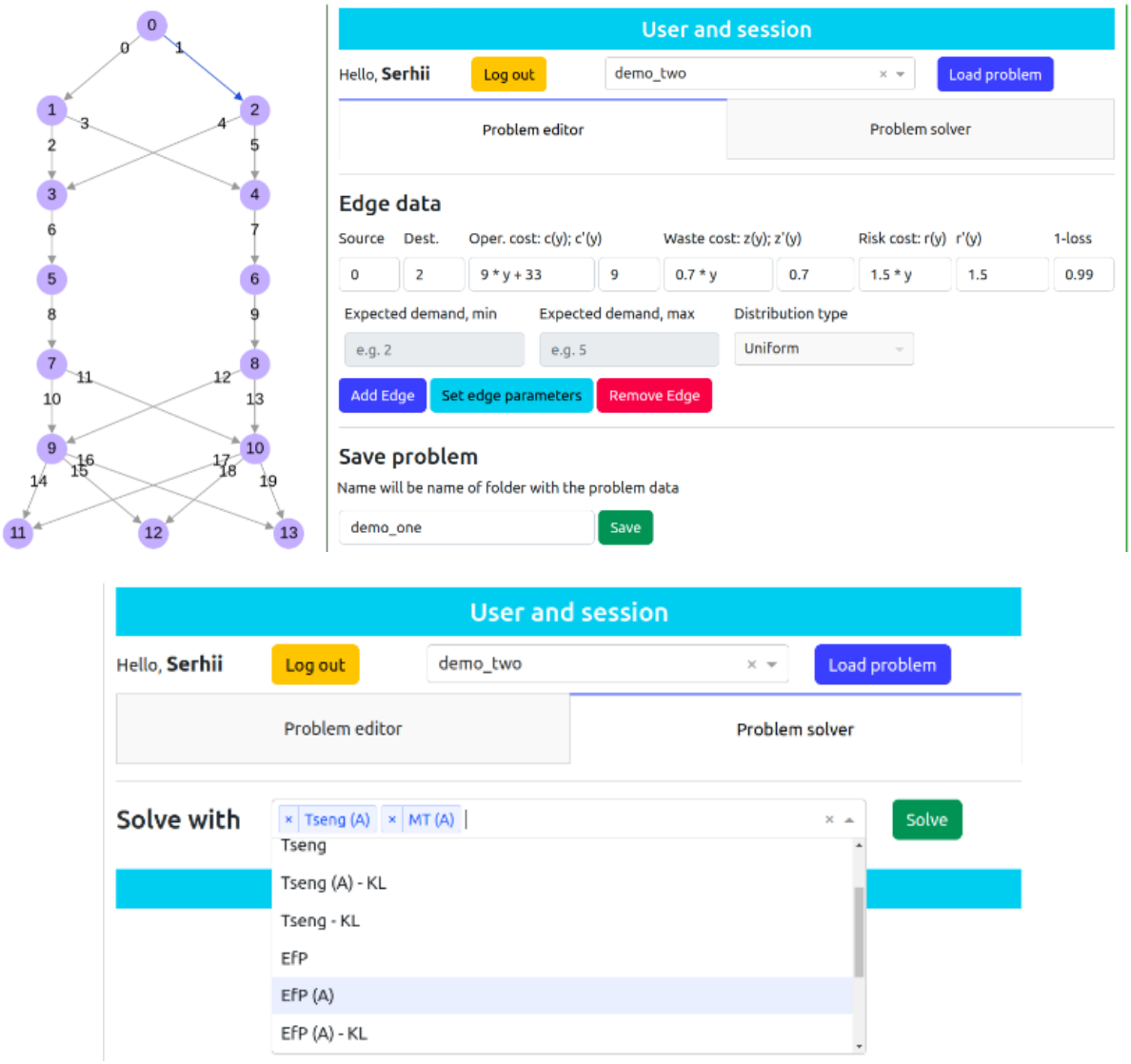

Figure 3: Screenshots of the problem editor and solver selection

4.2. Numerical experiments

Figure 4 shows, how calculation results are presented in web version of UI. For now, output includes convergence rate graph (iteration number vs. distance between approximate solutions) and textual information with final results, timing and run parameters. For problems with known solution graph can be switched to iteration vs. distance to real solution, and horizontal axis can be switched to calculation time instead of iteration number. Within textual information we also have specific characteristic of the problem, which are defined withing the problem class – for example, here we have supply amounts for each hospital. Under the hood, the system also saves detailed run history for every algorithm (in tabular format), so a researcher can monitor state and behavior on every iteration.

It can be seen, that in terms of precision after 1000 iterations adaptive algorithm of Tseng slightly outperforms EFP and Malitsky-Tam (MT) on this test problem. At the same time, from numbers we see, that time for 1000 iterations of Tseng is two times bigger (0.64 sec. vs 0.32 sec.) compared to both EFP and MT – which is expected, as in this problem projection is very cheap ($C = R^+$), but the mapping computation is very expensive (because of complicated derivative in (6)). Also, it worth noting, that despite much smaller distance between step iterations (0.001 for Tseng vs. 0.002 and 0.003 for MT and EFP), goal function value is not so different (80493 vs. 80499).

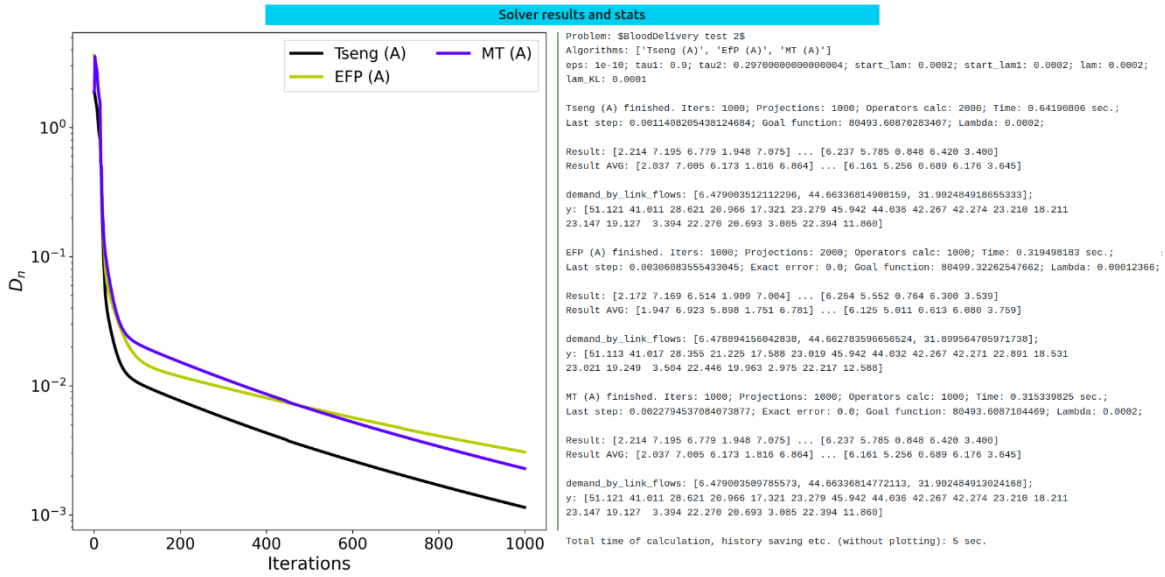

Figure 4: Screenshots of the solver results UI – graph and brief text data

It should be noted, that timing from the single run is not a stable measure of performance – but similar relative timings were obtained when running many experiments repeatedly and averaging the results. Calculations were done on Ubuntu Linux machine with Intel Core i7-1065G7 1.3 – 3.9GHz and 16GB RAM with python 3.10, numpy 1.24.3.

5. Conclusion

Considered adaptive versions of extragradient algorithms allow effectively solving blood supply chain optimization problem, without need of complicated handcrafting of step size and starting parameters. Implementation-friendly model formulation allows achieving reasonable performance and can be used for other problems of network economics family. And developed software system for applying VI algorithms to a different kind of problems considerably decreases time needed to both to get a solution for suitable problem and to conduct numerical experiments and compare behavior of algorithms from extragradient family. As further development directions, it is worth adding other types of algorithms, which are suitable for some subset of problems, and add a possibility for researchers to more easily plug in own algorithms and problems to the system.

6. Acknowledgements

This work was supported by the Ministry of Education and Science of Ukraine (project "Computational algorithms and optimization for artificial intelligence, medicine and defense", 0122U002026) and Vodafone Business machine learning and data science internship program.

7. References

[1] F. Facchinei, J. S. Pang, Finite-Dimensional Variational Inequalities and Complementarity Problems, Springer Series in Operations Research, vol. II, Springer, New York, 2003.
[1] D. Kinderlehrer, G. Stampacchia, An Introduction to Variational Inequalities and Their Applications, Society for Industrial and Applied Mathematics, Philadelphia, 2000.
[2] G. Kassay, V. Radulescu, Equilibrium Problems and Applications. London: Academic Press, 2019.
[3] R. Polyak, Finding Nonlinear Production-Consumption Equilibrium, arXiv preprint arXiv:2204.04496, 2022.

[4] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, Sh. Ozair, A. Courville, Y. Bengio, Generative Adversarial Networks, Advances in Neural Information Processing Systems (2014) 2672–2680.
[5] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, A. Vladu, Towards deep learning models resistant to adversarial attacks, arXiv preprint, arXiv:1706.06083, 2017.
[6] Anna Nagurney, Network economics: A variational inequality approach, Kluwer Academic Publishers, Dordrecht, 1999.
[7] A. Nagurney, Supply Chain Network Economics: Dynamics of Prices, Flows, and Profits, Edward Elgar Publishing, 2008. doi:10.1111/j.1467-9787.2008.00567_4.x.
[8] A. Nagurney, M. Yu, A. H. Masoumi, L. Nagurney, Networks Against Time: Supply Chain Analytics for Perishable Products, 2013. doi:10.1007/978-1-4614-6277-4.
[9] A. Nagurney, A. H. Masoumi, M. Yu, Supply Chain Network Operations Management of a Blood Banking System with Cost and Risk Minimization, Comput Manag Sci 9 (2012) 205–231. doi:10.1007/s10287-011-0133-z.
[10] M. Attari, S. Pasandideh, S. Niaki, A hybrid robust stochastic programming for a bi-objective blood collection facilities problem (Case study: Iranian blood transfusion network), Journal of Industrial and Production Engineering 36 (2019) 154-167. doi:10.1080/21681015.2019.1645747.
[11] G. M. Korpelevich, An extragradient method for finding saddle points and for other problems, Matecon. 12 (1976) 747–756.
[12] L. D. Popov, A modification of the Arrow-Hurwicz method for search of saddle points, Mathematical notes of the Academy of Sciences of the USSR 28 (1980) 845–848.
[13] P. Tseng, A modified forward-backward splitting method for maximal monotone mappings, SIAM Journal on Control and Optimization 38 (2000) 431–446. doi:10.1137/S0363012998338806.
[14] Y. Malitsky, M. K. Tam, A Forward-Backward Splitting Method for Monotone Inclusions Without Cocoercivity, SIAM Journal on Optimization 30 (2020) 1451–1472. doi:10.1137/18M1207260.
[15] A. Beck, First-Order Methods in Optimization. Philadelphia: Society for Industrial and Applied Mathematics, 2017.
[16] S. Denysov, V. Semenov, Adaptive Variants of the Extrapolation from the Past Method and the Operator Extrapolation Method, In: Shkarlet, S., et al. Mathematical Modeling and Simulation of Systems, MODS-2022, Lecture Notes in Networks and Systems 667, Springer, Cham, 2023, pp. 49–60. doi:10.1007/978-3-031-30251-0_4.
[17] S. V. Denisov, V. V. Semenov, P. I. Stetsyuk, Bregman Extragradient Method with Monotone Rule of Step Adjustment, Cybernetics and Systems Analysis 55 (2019) 377–383. doi:10.1007/s10559-019-00144-5.
[18] L. Chabak, V. Semenov, Y. Vedel, A New Non-Euclidean Proximal Method for Equilibrium Problems, in: O. Chertov et al. (eds.), Recent Developments in Data Science and Intelligent Analysis of Information (2019), Advances in Intelligent Systems and Computing 836, Springer, Cham, 2019, pp. 50–58. doi:10.1007/978-3-319-97885-7_6.
[19] Y. V. Malitsky, V. V. Semenov, An extragradient algorithm for monotone variational inequalities, Cybernetics and Systems Analysis, 50 (2014) 271–277. doi:10.1007/s10559-014-9614-8.
[20] M. Liu, W. Zhang, Y. Mroueh, X. Cui, J. Ross, T. Yang, P. Das, A decentralized parallel algorithm for training generative adversarial nets, arXiv preprint, arXiv:1910.12999, 2019. doi:10.48550/arXiv.1910.12999.
[21] Plotly Technologies Inc. Collaborative data science. Montréal, QC, 2015. https://plot.ly
.