University of Pécs
Faculty of Sciences
Institute of Mathematics and Informatics

Global Optimization Libraries on The Top Of TensorFlow.js

Supervisor:
Dr. Kégl Tamás
Scientific Advisor

Author:
Kola-Olaleye Adeola Damilola
Computer Science BSc

Pécs, 2022

Chapter I

Abstract

The Global optimization libraries on top of TensorFlow.js –"TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications [1]". thesis improves the understanding of global optimization. The goal of the project is to construct a high-level library written in vanilla JavaScript to enable the development of web-based applications that use numerical computations methods. Presented with an algorithm, the corresponding method is implemented upon real functional. The methods are straightforward to implement, efficient, and tries to minimize memory usage whilst being well suited for problems with large data sets. An open-source hardware-accelerated JavaScript library specifically designed for numerical computations and machine learning would be the base of the thesis project. Being that its main perk over similar hardware-accelerated libraries, using any available GPU that is supported by WebGL. A web-based application will serve as a proof target for the Global optimization library. All computations will be performed on the client side as opposed to the server side. Some optimization methods used within the thesis would be L-BFGS, Adam, Powell, Jacobi eigenvalue algorithm. I will equally discuss the capabilities and limitations in further chapters within the thesis.

Chapter II

Introduction and research goals

1 Descriptions of topic

Over the past few decades, several fields of numerical computations of both scientific and research have gained ever-increasing importance. With this increase comes the demand of figuring out the optimal base set that can be used in checkpoints from a mathematical approach. Traditional optimization techniques within the mathematical realm are not capable to predict the global minimum of a function. Traditional optimization jurisdiction ends in finding the local minimum. Global optimization sets the bar by find the global optima of a provided object function within a design point array which may contain several local solutions. A globally optimal solution is one where there are no other feasible solutions with better objective function values for the objective function. A locally optimal solution is a solution whereby no other feasible solutions "within the neighbourhood" with better objective function values for the objective function. This can be imagined as points in hike. Respectively, the top of a mountain or at the ground level of a mountain depending on the objective function. There may exist a much higher peak at the mountain or steeper and deeper ground level at the mountain floor. The base of this thesis, will be to outplay the system. A library, that can be implemented within web applications to take full advantage over an existing GPU unit for numerical computations such as optimising methods[2]. A Global Optimization library can help developers build components for their web application, in order to utilize the GPU resources that support WebGL. Optimization problems are in various fields. The research field, educational field and scientifically field. Global optimization has so many aspects, even number analysis. Practical global optimization techniques has had steady growth over the years. Because of this growth, it has gained traction from practitioners, students and researchers from other fields across the spectrum of Mathematics.

In this thesis, I touched two types of global optimization methods upon. these are the Direct method and stochastic global optimization.

2 Core definitions

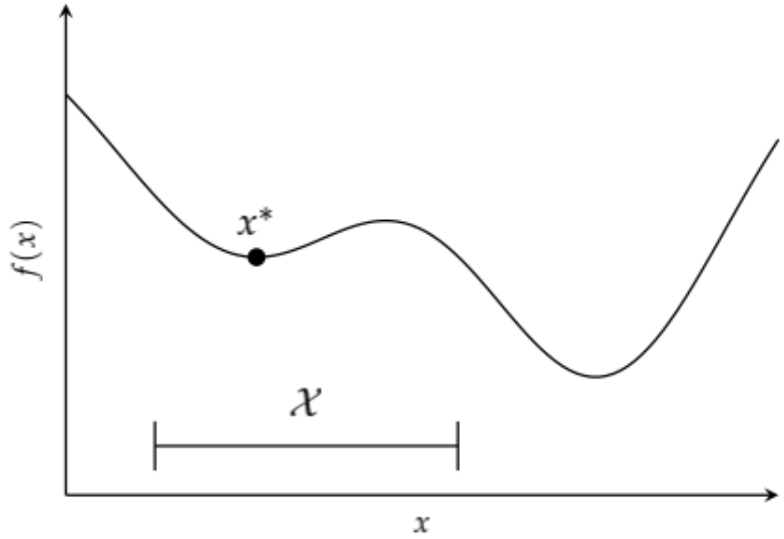
I can express a global optimization problem irrespective of its inherent real-world problem. A global optimization problem is made up of finding a point x where a function $f : X$, consisting of the minimise $f(x)$ and subject to $x \to X$. Here, x is part of an n-dimensional feasible region[3]. The object function

$$f(x) : x \to X \tag{II.1}$$

is minimized by adjusting the individual nodes of the feasible region vector. Those are called solutions. Generality of formulation means an optimization problem can be rewritten from II.1 to II.2 with respect to equation II.1

$$maximize\, f(x)\, subject\, to\, x \to X \tag{II.2}$$

$$minimize\, -f(x)\, subject\, to\, x \to X \tag{II.3}$$



Source: https://algorithmsbook.com/optimization/files/optimization.pdf

Figure 1: A 1-dimensional optimization problem

This is mostly applied in applied number analysis or mathematical applications. As opposed to local optimization, Global Optimization does not find the minimum or maximum of the a functional for a given data set or data space. This is the scientific goal, but does not directly relate to the goal of the thesis project.
Creating an implementation for web developers, developer students alike to bundle a library to serve real world applications ranging from the prediction of Protein structure to the functional for the methanol molecule.

Definition 1. Any value of x from all the values within the feasible set X that minimizes the object function is called a *solution*. This point can be refereed to as

$$x^* \tag{II.4}$$

Figure 1 in relation to this shows an example of a 1-dimensional optimization problem.

Definition 2. A point x^- is refereed to as local minimum[2] of f if there is a $\delta > 0$ in such a way that

$$f(x^-) \leq f(x), \forall\, x\ with\ |x - x^-| < \delta. \tag{II.5}$$

A point which individually minimizes f within its own neighbourhood is called a strict local minimum.x^- can be referred to as a strict local minimum of f if there is a $\delta > 0$ such that

$$f(x^-) \leq f(x), \forall\, x^- =! x\ \&\ |x - x^-| < \delta. \tag{II.6}$$

For its corresponding opposite, non-strict local minimum is a local minimum that does not meet the requirements for a strict local minimum.

Definition 3. A feasible region or point has a greater tendency of becoming a local minimum if the local derivative is equals to zero and the 2nd derivative is on the positive axis.

$$\begin{aligned} f'(x^*) &= 0 \\ f''(x^*) &> 0 \end{aligned} \tag{II.7}$$

If the first local derivative is a zero derivative and the second is a non-negative derivative or zero of a point can also be at local minimum.

$$\begin{aligned} f'(x^*) &= 0 \\ f''(x^*) &\geq 0 \end{aligned} \tag{II.8}$$

Where f' is the first local derivative and f'' is the second derivative.

3 Global optimizations problems

There is no one specific way to group Global optimization problems. Grouping them in respective to the optimization model chosen, the objective function to be minimised are methods for grouping Global optimization problems. There many classifications of global optimization problems by authors, online repositories and publications. ***According to FrontlineSolvers publication of GO methods classification[4]***, global optimization methods can be broken into two factions:

- Multistart methods
- Genetic Algorithms[5]

Multistart methods will have a higher chance arriving at globally optimal solution(s). This may not hold true as the data set as Higher dimensional data sets for the objective function values. Increases over a certain threshold. Stochastic methods are better suited for higher dimensional objective function values. They are also better suited in collecting iterative information about the regions of attractions [6] of the best local minimum points. They mostly use "classical" smooth nonlinear solvers that by themselves can only find locally optimal solutions. They are a limited guarantee for these methods that it will arrive a global optimal solution at. ([7])
Genetic Algorithms. This methods are used in order to arrive at solutions for non-smooth optimization problems. They are better suited at finding more optimal solutions than classic smooth methods. It is worth noting that the computational effort and energy required by these methods increases not lineally but exponentially. Other similar examples include Tabu Search and Scatter Search. They could be further classified under Heuristic methods [8].
Tabu Search. Tabu Search is under the Handbook of Global Optimization[9] as being a meta-heuristic search method. Search methods may not move unto nodes that have already been visited. It uses local search methods for Global optimization in respects to mathematics. "neighborhood" nodes are searched and predict a solution to an objective function. This is performed again while trying to find an optimal solution. There is also a tendency for the solution to be non-optimal. Tabu search methodology has been primarily used to solve optimization problems that range for inventory, routing locations, but it can also be extended further in order to handle other Global optimization problems.[10]

Chapter III

Methodology

The various algorithms and implementations, tools, packaging tools and technologies are outlined in this chapter. This is an essential part of the thesis. The final project is divided into two core parts which are: the library main package and downloadable and a web application that implements the library core. As a base of proof for the thesis, a live demo implementation of the library should be practical.

1 Tool Stack

This chapter with the development of interval based global optimization methods. The tool stack comprises a top overview bandage. The tool stack comprises the packaging tools, languages (programming), software frameworks and paradigms used within the framework of the Global Optimization library main and its corresponding web application implementation. I listed them in no order throughout the subsections.

1.1 Packaging Tools

To distribute and promote usage of the library, it would need to be published as a library. This will bundle and package source code into blobs that people can easily instal and use. Publishing a library to package can help in out-sourcing to0 more experienced members except for.

```
$ npm install global-opt
```

To install global-opt, the $--$ *global* flag will have to be set.

```
$ npm install --global global-opt
```

Afterwards, a successful installation of the library main, will allow users to import and use the library's functionality within their existing code-base.

```
import { adam } from "global-opt";
```

This may vary from JavaScript to Typescript. Although neither the example implementation nor the library main takes substantial advantage of Typescript. While there is no data, internal or externally, computational power consumption again.
The publication site would be npmjs[11] (Nightly Pocket Measurement). It's an open source repository for a substantial amount of both private and public JavaScript packages. When a package is published to npmjs, there are specific benefits that come with it. During development, npmjs can allow copying of packages into other CI/CD tools. It is a vast network for developers to get libraries. It also can manipulate merges and version control. At the root of the library file structure, the package. json which is a JSON(JavaScript Object Notation) file is located there. It is a unique method that web development environment used to identify specifics about a package. A typical package.json looks like this:

```
{
  "name": "global-opt",
  "version": "0.0.1",
  "description": "optimizing functions to aid in numerical web-dev
     applications",
  "author": "adeola",
  "repository": "adeola/thesis",
  "main": "main/index.js",
   "main": "main/index.js",
  "module": "main/index.es.js",
  "engines": {
    "node": ">=LTS",
    "npm": ">=5"
  },
   "dependencies": {},
  "scripts": {
    "build": "rollup -c",
    "start": "rollup -c -w",
    "prepare": "npm run build",
    "predeploy": "cd example && npm install && npm run build",
    "deploy": "gh-pages -d example/public",
    "release": "npm"
  }
}
```

- NPMJS: Nightly Pocket Measurement is an open source publishing repository for private and public repositories.

- JavaScript Modules: Having a modular library is key. JavaScript has native implementation, JavaScript modules. Global-opt uses the Commonjs[12] module approach. It is set within the package.json with the *module property* set to commonjs. This allows splitting up the large JavaScript program into individual modules. It gives an end user the flexibility to import only select needed parts of the library as modules.

1.2 Language, Programming Environment

The most vital goal and use cases when building up the program was that it could enable the development of web-based applications. Ideally this means that it had to run in JavaScript development and production environments excluding Node JS [The library is specifically made to be run on the browser, client side as opposed to the server side]. Global-opt is purely written in JavaScript, due to the nature of Typescript[13] being a syntactic sugar coated version of JavaScript, it may not rule compatibility out. Typescript has its own compiler which acts as a mediator between the two languages. The sole reason for building up the code-base was because of its constraints regarding run-time implementation. JavaScript is not the only technology that the web has been introduced to[**online11**]. Although it is definitely the programming language is hitting the web for users.

Tool	Implementation
Svelte[14]	Scripting
VS Code	Hybrid IDE/Code Editor
TensorFlow.JS	Library to expose global optimization methods
NPMJS	Library Repository Platform
Vercel	Integrated Web Hosting Platform
JSON	Non-Structured Data Format
GitHub	CI/CD Platform
Vanilla JavaScript	Core Programming language
Typescript	UI specific language
CommonJS	Module specific

Table 1: Tools used

Programming Environment refers to the concept that describes how the code-base behaves, runs, versioning, development methodologies all tie together to create the final program version:

- VS Code: This is the hybrid code-editor/IDE used. At its core, it's a shell that can be super-charged with various extensions.
- Common JS
- Vercel & Integrated Web Hosting Platform is an integrated web hosting service that connects directly to a CI/CD (Continuous Integration/Continuous Delivery) network, in this case GitHub and runs the build commands, and hosts a newer version anytime a push is made to the respective GitHub repository. It handles the building and self-hosting behind the scenes and allows developers to focus more on coding.
- JavaScript ECMAScript 2018 is a standardize versioning system used to describe new iterations of ECMA or JavaScript features.
- Yarn: Yarn is the code packaging system that can be used locally to sort out build issues, run test, production and dev builds and audit system vulnerabilities of a web application.

1.3 Libraries

TensorFlow.js being a vital part of the global-opt library. The global-opt library along with its modules and functions are built on top of TensorFlow.js. It exposes a finite set of optimizers along with their corresponding algorithms. Said algorithms can be used for optimizing numerical computations from a data set. Global-opt has some built-in implementations of optimising algorithms. However, each algorithm and variant is exposed within the API, giving full flexibility. Some of which are:

- *powellOptimizer*: "Powell's method can search in directions that are not orthogonal to each other." It is used for locating the local minimum of a function. Most optimising algorithms require derivatives. Here, Powell's method does not as the function doesn't need to be differentiable.

```
1  const powellOptimizer = function (functionCallback,
       vectorCallback) {
2      // Epsilons value for error checking
3      let eps = 1e-2;
4      let convergence = false;
5      // make copy of initialization
6      let vector = vectorCallback.slice();
7      // scaling factor, can be made smaller for minor improvement.
8      let alpha = 0.001;
9      let pfx = Math.exp(10);
10     let functional = functionCallback(vector);
11     while (!convergence) {
12         convergence = true;
13         // Perform update over all of the variables in random
               order
14         for (let i = 0; i < vector.length; i++) {
15             vector[i] += 1e-6;
16             let fxi = functionCallback(vector);
17             vector[i] -= 1e-6;
18             let dx = (fxi - functional) / 1e-6;
19             if (Math.abs(dx) > eps) {
20                 convergence = false;
21             }
22             vector[i] = vector[i] - alpha * dx;
23             functional = functionCallback(vector);
24         }
25         // A simple step size selection rule. Near x function
               acts linear (this is assumed at least) and thus very
               small values of alpha should lead to (small)
               improvement. Increasing alpha would yield better
               improvement up to certain alpha size.
26         alpha = pfx > functional ? alpha * 1.1 : alpha * 0.7;
27         pfx = functional;
28     }
29     let solution = {};
30     solution.vectorOptimized = vector;
31     solution.returnValue = functional;
32     return solution;
33 };
```

This is a more direct traditional approach without any form of wrapper It should not be confused with being a Direct method.

- *adamOptimizer[15]*: This is inferred from TensorFlow.js library. The Adam optimization algorithm otherwise known as Adaptive moment estimation is an extension to stochastic gradient descent. It uses learning rates to each parameter given. This implementation correctly fits in values for a quadratic function by learning the coefficients q, w, e.

```
1  const xs = tf.tensor1d([0, 1, 2, 3]);
2  const ys = tf.tensor1d([1.1, 5.9, 16.8, 33.9]);
3
4  const q = tf.scalar(Math.random()).variable();
5  const w = tf.scalar(Math.random()).variable();
6  const e = tf.scalar(Math.random()).variable();
7
8  const f = x => q.mul(x.square()).add(w.mul(x)).add(e);
9  const loss = (pred, label) => pred.sub(label).square().mean();
10 const learningRate = 0.01;
11 const optimizer = tf.train.adam(learningRate);
12 // Train the model.
13 for (let i = 0; i < 10; i++) {
14    optimizer.minimize(() => loss(f(xs), ys));
15 }
16 });
```

This is a First-Order method[16].

2 Usage of Various Algorithms

Algorithm 1 Adam Algorithm

Funct Post(WP, P)

Require α : *Stepsize*
Require $\beta_1, \beta_2 \subseteq [0, 1)$: *Exponential decay rates for estimate*
Require $f(\theta)$: *Stochastic objective function with parameters*
    $m_0 \leftarrow 0$
    $v_0 :\leftarrow 0$
    $t :\leftarrow 0$
while θ_t *is not converged* **do**
    $t \leftarrow t + 1$
    $g_t \leftarrow \triangle_\theta f_t(\theta_t - 1)$
    $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$
    $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t{}^(2)$
    $\overleftarrow{m_t} \leftarrow m_t \,/\, (1 - \beta_1{}^t)$
    $\overleftarrow{v_t} \leftarrow v_t \,/\, (1 - \beta_2{}^t)$
    $\theta \leftarrow \theta_{t-1} - \beta \cdot \overleftarrow{m_t} \,(\overleftarrow{v_t} + \in)$
end while
return θ_t (*Resulting parameters*)

Before diving in into Adam, let me discuss, in a clear fashion the kind of objective functions that Adam specifically targets. Adam stems as a method for stochastic objective – Stochastic optimization is the application of randomness in an objective function. "Methods for stochastic optimization provide a means of coping with inherent system noise and coping with models or systems that are highly nonlinear, high dimensional".[17] functions. It requires a gradient being that it is a first order gradient-based optimization[18]. No one optimization method is perfect for every objective function. Being relatively fast to implement, low memory consumption and computational power consumed are some factors to consider when dissecting a global optimization method. A large array of problems in fields of science research, number analysis can benefit from Stochastic gradient-base optimization.

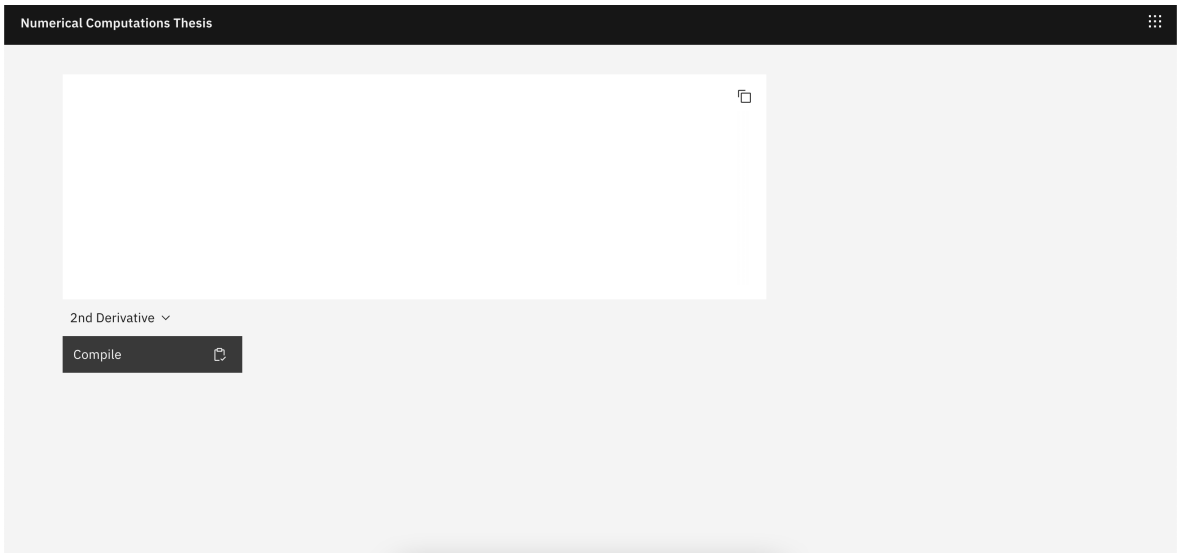
```
tf.train.Adam(
    learning_rate=0.001,
    beta_1=0.9,
    beta_2=0.999,
    epsilon=1e-07,
    amsgrad=False,
    name='Adam',
)
```

3 Implementation

A real world implementation to show the effectiveness of the global-op library. I created one such implementation as a base of thesis to display its practicality. Individual Global optimization algorithms can be picked and used depending on whether it corresponds to the given objective function. Here, there is a closed and controlled experiment, getting Long-range distances for the atoms being involved in the electrostatic part, Bond distances and and Ideal angles (theta) using an Energy functional as the reference point for the objective function.

$$\begin{aligned} Etotal = (vectorToOptimize) => bondFunction(letBonds)+ \\ angleFunction(letAnglesRad)+ \\ chargeFunction(longrangeC); \end{aligned} \tag{III.1}$$

This serves as a base of the thesis. It covers the full scope in such a way that it feeds off the API that we created. Also described and testing out the theoretical aspects of performance issues of TensorFlow within the browser. It could be refereed to as a niche category, since TensorFlow has some limitations. Such limitations are further discussed in Chapter IV. It goes into detail the issues that plagued the thesis work. It also talked about the added benefits of the same Package. There is no one bullet kills all factors. It's a play with the pro's and con's, whilst trying to find an optimal solution. `https://global-optimization.vercel.app/` uses the exposed API's from the global-opt library to perform numerical optimizations on the client side.
Created using Svelte, a Typescript framework but interfaces smoothly with JavaScript code. This route was taken from the traditional JavaScript framework. Simply to touch more grounds in the web development spectrum. The library can be used within a web development environment that uses Typescript or JavaScript. Take a functional $f(x) = x^2$. Assuming fx was a vector. The derivation of an optimized model for such a function would be a good base example for the description of the topic.



Source: ttps://global-optimization.vercel.app/

Figure 2: Implementation of Global-Op library

Building the project

GitHub hosts the complete source code being publicly available. In order to re-build the whole project, the project repository will need to be installed on the host machine. Along with that, various build tools and external libraries.

```
git clone https://github.com/adeolaex/global-optimization.git
```

The repository has now been set up. Next thing to do is to get the dependencies installed or updated. They are all declared in the special package.json file.

```
cd global-opt
npm install
```

After we have installed the dependencies, running a dev server will be the next and final step. The dev flag is set because of the ability to hot reload pages immediately after changing them.

```
npm run dev
```

For a much for streamlined and faster build, use

```
npm run prod
```

Setting up TensoFlow.js would be the next functionality to implement once your multiplayer is active. TensofrFlow.JS.

Chapter IV

Result analysis

1 Discussions of Results

The general results from different aspects of the thesis were not short of the outcome. The goal was to create an extendable library that could be used for the development of web applications to utilize GPU power where available in order to compute numerical computations. In order to demonstrate solid results, an implementation example must be present and piratical. This was possible in relation to the goal of the topic. Some issues and limitations were noticed during the development which are dissected in latter sections. Non derivative algorithms like Powell were less GPU tasking and did not require additional edge cases tests. TensorFlow's API extends so many algorithms, although the implementation is limited. Testing it on a low-specked laptop with an integrated GPU, similar results with respects to the data set size were very similar.

2 Problems and Limitations

Before getting into the limitations and problems that arose, on the other end of shortcomings are the advantages that were noticed during the descriptive implementation for the base of thesis and optimization solutions. Both the limitations and advantages came by on a case-by-case schema. Some limitations were immediately noticed. Whilst others were not noticed until after being compared. The JavaScript counterpart of TensorFlow core is relatively fast and easy to set up as opposed to TensorFlow itself. This is due to run-time environment bottlenecks — will talk about that later on in this chapter. In accordance to scientific trends [19] and official publications TensorFlow.js had a nigh 1:1 implementation of some Global Optimization algorithms. They required an existing model, but nevertheless *worked*.

In order to incorporate GPU support in TensorFlow.js, additional software would have to be installed. Take an example, GPU cards with support for CUDA would need to incorporate the CUDA toolkit on a system level. As models increases, developers would have to cut down them down. TensorFlow.js can handle smaller models inside the browser. In practicality, outside of contained tests and experiments, most models will be large in terms of data sets. There is only so that it can do. There is a way to mitigate this "issue". Taking advantage of TensorFlow.js on Node.js. Basically running the computations on servers instead on the clients machine. However, this is not within the jurisdiction of this thesis, so it is negligible. Computations especially memory intensive and computational effort driven algorithms like the Genetic Global Optimization Algorithms might hold itself on a laptop, desktop machine. The same cannot be said about mobile devices. The speed of both variations of TensorFlow are on the same level working with small models, but when model sizes become larger, predicting approximations for the objective function reduces in speed. There are some area-specific limitations that where not in thesis work. TensorFlow.js has the advantage of utilizing both discrete and integrated graphical processing units. Crucial for web applications being released to an unknown demographic. Further limitations where the kind of objective function that was accepted in comparison of different methods. The number of iterations and memory complexity did not play well will most optimizers. Controlling the energy output in relation to the size of the data set of the objective function in question was another issue, in some case the time complexity was On. It could grow lineally as the objective function increased or exponentially on some more refined methods like Tabue Search. There were some limitations. This ranged from achieving an optimal time complexity on larger data sets. One of the major problems was backward compatibility with older versions of JavaScript and even sub-types of JavaScript like Typescript. The inability to use older versions of TensorFlow was an issue as it needed version strict dependencies. Other optimization algorithms were used. L-BFGS was implemented but later scrapped because of so many factors. Some numerical computations could not be applied to L-BFGS due the variable requirements needed.

Chapter V

Conclusions

The effectiveness of the method (Powell method) have been highlighted in the proof of concept by solving an optimization problem from the methanol molecule functional. The conclusion of problem is that on the given global optimisation problem Powell algorithm could predict and find accurate enough approximations within the order of magnitude from non-predicted values. The computation energy could remain within a sensible order of acceptance. This is crucial as data sets are expected to grow linearly. If the energy used grows exponentially, it creates a negative effect. The second algorithm using Adam, was based of Googles individual implementation of Adam. Although, it is a 1:1 version of the mathematical formulation[20]. The exporting of the library itself to an industry standard was also met. It can be downloaded, looked into and its API's used within an already existing code-base. In conclusion, the implementation still requires further work. One key objective that requires further work will detect the presence of a GPU. TensorFlow does not explicitly expose an API that allows such. When a numerical method is run through with an optimizer, or an optimizer is run with a demo data set, a GPU may not make a big difference. Because of the architecture design of TensorFlow, working with Tensors is far better with a Graphical Processing Unit. A flag could be invoked starting that the time complexity wouldn't be optimal.
The global-opt library is also able to keep track of basic errors. Specific errors cannot be caught when thrown. This results in a bad developer experience, as error handling is crucial for any piece of software. There may be certain edge cases, like a non-liner vector data structure being passed unto an optimizer, a non numerical function also being passed unto an optimizer. Time-out errors, connection errors are also not accounted for.

A better overview of which optimizer model to pick for which task and data structure could also be extended. Comment blocks give developers an insight on what a function, in this case, function within a module does or is expected to do. We also investigated the various GO methods, along with their theoretical aspect, shortcomings and computational tests. Future-proofing has become an important part of solidify a piece of software within the industry. The web does not change regularly, if at all, very little. Although, the formatting, data structures used within the project should be built to work and last, not just work for the time being. Adding an extra layer to interact with other object times. A practical example would be to read JSON file objects that contains huge data sets for a numerical computation. Making a side-by-side comparison as an afterthought in order to back up that scientific name.

Due to the limitations mentiond in Chapter IV, there are some edges cases that could led to sub-optimal solutions. The codebase could be extended with different versions which allows it to respond immediately an edge case is hit. The data sets that are accepted are within the jurisdiction of certain data structures. More exotic data sets cannot be used directly. Take a HashMap data structure. This will pose issues as it is not a directly used data structure implementation still requires further work. One key objective that requires further work will detect the presence of a GPU. TensorFlow does not explicitly expose an API that allows such. When a numerical method is run through with an optimizer, or an optimizer is run with a demo data set, a GPU may not make a big difference. Because of the architecture design of TensorFlow, working with Tensors is far better with a Graphical Processing Unit. A flag could be invoked starting that the time complexity wouldn't be optimal.

Chapter VI

References

[1] Google Brain Team. *TensorFlow.jsMachine Learning for JavaScript Developers*. 2015. URL: https://www.tensorflow.org/js (visited on 02/20/2022).

[2] M.J. Kochenderfer and T.A. Wheeler. *Algorithms for Optimization*. MIT Press, 2019. Chap. 5, p. 70. ISBN: 9780262039420. URL: https://books.google.hu/books?id=uBSMDwAAQBAJ.

[3] M.J. Kochenderfer and T.A. Wheeler. *Algorithms for Optimization*. MIT Press, 2019. Chap. 1, pp. 7–6. ISBN: 9780262039420. URL: https://books.google.hu/books?id=uBSMDwAAQBAJ.

[4] FrontlineSolvers. *GLOBAL OPTIMIZATION METHODS*. 2022. URL: https://www.solver.com/global-optimization (visited on 05/02/2022).

[5] M.J. Kochenderfer and T.A. Wheeler. *Algorithms for Optimization*. MIT Press, 2019. Chap. 9, pp. 147–151. ISBN: 9780262039420. URL: https://books.google.hu/books?id=uBSMDwAAQBAJ.

[6] R. Horst, P.M. Pardalos, and H.E. Romeijn. *Handbook of Global Optimization*. Handbook of Global Optimization v. 2. Springer, 2002. ISBN: 9781402006326. URL: https://books.google.hu/books?id=19W0Ast3pDIC.

[7] Tibor Csendes et al. *The GLOBAL Optimization Method Revisited*. 2027. URL: https://www.inf.u-szeged.hu/~csendes/ol.pdf (visited on 05/01/2022).

[8] R. Horst, P.M. Pardalos, and H.E. Romeijn. *Handbook of Global Optimization*. Handbook of Global Optimization v. 1. Springer, 2002. ISBN: 9781402006326. URL: https://books.google.hu/books?id=19W0Ast3pDIC.

[9] R. Horst, P.M. Pardalos, and H.E. Romeijn. *Handbook of Global Optimization*. Handbook of Global Optimization v. 2. Springer, 2002, pp. 387–390. ISBN: 9781402006326. URL: https://books.google.hu/books?id=19W0Ast3pDIC.

[10] F.W. Glover and M. Laguna. *Tabu Search*. Springer US, 1998. ISBN: 9780792381877. URL: https://books.google.hu/books?id=mFYt0C5cqtAC.

[11] NPMJS. *Network Powering Makers*. 2022. URL: https://www.npmjs.com/ (visited on 03/22/2022).

[12] Mozilla Dev Team. *JavaScript modules*. 2022. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Modules (visited on 03/23/2022).

[13] Mozilla Dev Team. *JavaScript modules*. 2022. URL: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Client-side_web_APIss (visited on 03/23/2022).

[14] Svelte Core Team. *Client-side component API*. 2022. URL: https://svelte.dev/ (visited on 02/08/2022).

[15] M.J. Kochenderfer and T.A. Wheeler. *Algorithms for Optimization*. MIT Press, 2019. Chap. 7, pp. 79–81. ISBN: 9780262039420. URL: https://books.google.hu/books?id=uBSMDwAAQBAJ.

[16] Google Brain Team. *TensorFlow Adam API*. 2022. URL: https://js.tensorflow.org/api/latest/#train.adam (visited on 03/23/2022).

[17] J.E. Gentle, W.K. Härdle, and Y. Mori. *Handbook of Computational Statistics: Concepts and Methods*. Springer Handbooks of Computational Statistics. Springer Berlin Heidelberg, 2012, pp. 171–201. ISBN: 9783642215513. URL: https://books.google.hu/books?id=aSv09LwmuRYC.

[18] M.J. Kochenderfer and T.A. Wheeler. *Algorithms for Optimization*. MIT Press, 2019. Chap. 5, pp. 70–76. ISBN: 9780262039420. URL: https://books.google.hu/books?id=uBSMDwAAQBAJ.

[19] Panos M.Pardalosa, H.EdwinRomeijna, and HoangTuyb. *Journal of Computational and Applied Mathematics – Recent developments and trends in global optimization*. 2022. URL: https://www.sciencedirect.com/science/article/pii/S0377042700004258 (visited on 05/01/2022).

[20] Diederik P. Kingma and Jimmy Lei Ba. *ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION*. 2015. URL: https://arxiv.org/pdf/1412.6980.pdf (visited on 04/21/2022).

List of Figures

List of Tables

UNIVERSITY OF PECS
Faculty of Sciences
Instituion ID FI58544

Declaration of Originality

Student's Name: Adeola Damilola Kola-Olaleye

Neptun code: BENP1Z

Title of Thesis: Global Optimization Libraries On The Top of TensorFlow.js

Hereby, I, a student of the Faculty of Sciences at the University of Pecs, in full knowledge of my liability, declare that this thesis is **my own original work.** I have clearly referenced all sources (both from printed and electronic sources) in the text in accordance with international requirements of copyright.

I acknowledge that it is considered plagiarism

- to cite verbatim without using quotation marks and crediting the author
- to paraphrase the source material without citing the source
- to indicate another author's published ideas as my own.

I declare that I have understood the definition of plagiarism and I accept that if my thesis violates copyrights, its assessment shall be fail (1), and the major director of the program shall initiate a disciplinary procedure in front of the Dean according to Article 59 (14) of the Code of Studies and Examinations of the University of Pecs.

Pecs, 30 day, April month, 2022 year

signed by student