Deployment of Linear Programming for Creating Cybersecurity Profiles

1st Michal Stejskal
Department of Telecommunications
Brno University of Technology
Brno, Czech Republic
231282@vut.cz

2nd Petr Dzurenda
Department of Telecommunications
Brno University of Technology
Brno, Czech Republic
0000-0002-4366-3950

***Abstract*—The cybersecurity job market is growing rapidly, with a great demand for cybersecurity professionals. Currently, there are many cybersecurity curricula, professional training, and certification schemes that have been created recently, however, they are independent of each other and without any synchronization. The recently released European Cybersecurity Skills Framework (ECSF) aims to bring order to this area. This article builds on the work of the European Union Agency for Cybersecurity (ENISA) and extends it with a practical tool that allows professionals to create their desired ENISA cybersecurity profiles. The core of our tool uses a Linear Programming (LP) algorithm to help professionals to find the most suitable combinations of professional training courses to follow in order to achieve their required ENISA profiles. Our experimental results show the efficiency of our implementation, in terms of the speed of finding a solution for different LP solver libraries.**

***Index Terms*—ENISA, ECSF, Cybersecurity Skills Framework, Cybersecurity, Cybersecurity Profile, Linear Programming, Simplex Algorithm**

I. Introduction

Because of the expanding demand for cybersecurity professionals on the European, but also the worldwide market, new cybersecurity terms, roles, and skills are being created and named. However, these names are often quite different. This can bring confusion and misunderstanding about what should be taught in professional courses and what is actually required by the job market. Therefore, ENISA undertook to solve this problem and created the ECSF that unifies the terms used in this specialization [1]. Taking this into account, the Cybersecurity Skills Alliance - A New Vision for Europe (REWIRE) project, created a connection between ECSF skills and knowledge and REWIRE cybersecurity competencies. These REWIRE competencies are then linked back to ENISA profiles. With this methodology, it is possible to map existing university curricula, professional training, and certification courses to ECSF. Furthermore, the REWIRE project designed a web application called Cybersecurity Profiler (CSP) which can serve people in this field, as it includes courses all over Europe and multiple tools, which can help with creating a university curriculum, professional training and certification courses, and understanding requirements for fulfilling job roles. We refer to [2] for more details.

A. Contribution and Paper Structure

This paper focus on our design, implementation, and experimental tests of the computing core of the CSP application allowing identifying which courses, training, and certification courses are recommended for a certain work role.

Section II describes the fundamental knowledge required for understanding the problem. Section III presents the target of the module, which is to assemble courses in such a combination, which fulfills parameters set by the user of the application and match with the chosen profile. Details of implemented libraries and code structure are presented as well. Section IV is dedicated to testing and experiments with different Linear Programming (LP) solver libraries, so that the best library can be selected by setting criteria relevant to this work.

II. PRELIMINARIES

In this section, we describe the ECSF to foreshadow its connection to created CSP application. Furthermore, linear programming and its branch, the simplex algorithm is introduced as the most prominent option for solving the problem of finding combinations from available courses.

A. European Cybersecurity Skills Framework (ECSF)

The ENISA has summarized knowledge, and skills in the field of cybersecurity into twelve profiles [1]. Classification as this gives options not only to individuals seeking proper education but also allows organizations to specify which professions they need and simultaneously training providers can use the same terms to offer their services. Unifying terms in this market will help this area to grow and hopefully provide better conditions for people to become professionals. More experts are required to protect critical infrastructure, privacy and similar vulnerabilities that arise.

Project REWIRE followed up on this work and included skill groups in the created profiles, that reflect which skills need to be mastered to fulfill the profile. Skill groups consist of more detailed skills and knowledge.

B. Integer Linear Programming and Simplex Algorithm

Linear programming is a mathematical discipline that targets finding the best outcome. The objective has to be finding the maximum or minimum of a function. The algorithm aims to find the maximum or minimum of the objective function while respecting the given constraints, see [3] for more details. The simplex [4] is a standard algorithm often used because it is simple and widespread. Simplex uses bounds to establish feasible region. The boundaries are then traveled along the edges until the optimum of function is reached. This method always presents global optimum if precautions against cycling are applied.

C. Solving packages for ILP

Solvers use different approaches or techniques to compute the solutions of a problem. As each solver usually allows to solve a vast number of different problems, choosing the proper solver can make a relevant difference in performance. It must be clear that solving a problem just by using the simplex method cannot fulfill all types of tasks, and Mixed Integer Programming (MIP) is proof of that [4]. Some problems include values that are required to be binary but the simplex method will produce a best global solution which might not be an integer value. Here the solver has to use its methods to find a solution with integer values. This can be achieved for example by Branch-and-Cut algorithm [5].

The algorithm works as follows, the first step is creating bounds, if the desired outcome is an integer and a continuous number is produced instead, then bounds are created around the value. For example, if the produced number is 4.6 which is not an integer, then the next step is to create a lower bound that would be 4.0 and an upper bound of 5.0. This creates two new constraints which are included for the simplex. Two calculations are performed with bounds appended separately and if solutions consisting of integers are found, then the better of these solutions is considered optimal. Another case is when this does not produce an integer, then branching occurs and similar bounds are set again and calculation is performed on them until a feasible solution is found and considered optimal compared to other integer solutions generated by different branches.

III. CYBERSECURITY PROFILE DESIGNER

The objective for this section is to give a detail description of the module designed for web application. The first subsection describes how the module was designed into detail. Then in next subsection some terms and details about possibilities of external libraries and packages are explained.

A. CSP Search Problem Formulation

First, it is important to outline the problem that needs to be solved. There are 59 courses included in the set at the time. The user can choose a profile that interests him. Profiles and trainings consist of skill groups. Algorithm has to match the trainings to chosen profile so that all skill groups included in the profile are represented by trainings. Because single training cannot fulfill this, multiple trainings have to be put together to achieve possible combinations. The found solution has to consist of the best available courses that match the profile and the selected parameters set up by the user. It is also essential to get an optimal solution in the shortest possible time. The filtering applied by the user includes important numerical values as price, duration, and included skills of the courses. These values together with a number of courses demanded in the solution are subjected to optimization so that the best possible outcome is found. To achieve this, extensive analysis had to be conducted and multiple possibilities were discovered. Between the most promising had ranked LP that on its own offers multiple algorithms and Genetic algorithms (GA) [4].

Regarding GA there are multiple red flags, to begin with. Among the biggest ones ranks the uncertainty that an optimal solution will be found, i.e., the algorithm does not always finish at a globally optimal solution, as in our case, where the final demanded state is not outlined to the algorithm. Another issue worth mentioning is the deviation time from finding each solution.

On the contrary, LP does not have to deal with these fundamental problems, because solving a problem with them is basically following mathematical instructions that lead towards an optimal solution and thus the result is always the same as the same approach is repeated. Unfortunately, there is a problem associated with this. The solution produced by the mathematical approach cannot guarantee that the result will contain whole numbers which are relevant to this work. This problem is solved with the help of solver packages, which will be discussed in more detail in Section III.

B. Application Design

The application currently operates only in python. Data are drawn from JSON local files. After data are extracted from files, they are filtered based on criteria like the language in which are the lessons conducted or the country from which they originate. Another criterion is whether the course ends with certification and also whether it is conducted online, face to face, or both. All mentioned filter settings including mentioned price, duration and mainly chosen profile/s are set up by a user.

The process of optimization then comes into place. Firstly, the target of optimization is acquired, this is statically set for minimization as the objective is to minimize price, duration, and number of included courses. These are values that the solver calculates with. Then variables are created. In this case, variables are courses themselves and they are represented by either one or zero, depending on whether they will be included (1) or excluded (0). That is determined by calculations. Further, the objective function is defined together with constraints and bounds, including the filtering values.

The objective function is presented mathematically in Equation 1, where S_{min} is a minimization of the objective function. Here p_{r_i} stands for price and is added together with d_{u_i} which represents duration. Both of the values are multiplied by b_i which represents a binary variable deciding whether the course will be included or not.

$$S_{min} = \sum_{i=0}^{M}[(p_{r_i}b_i) + (d_{u_i}b_i)] + \sum_{i=0}^{M} b_i \tag{1}$$

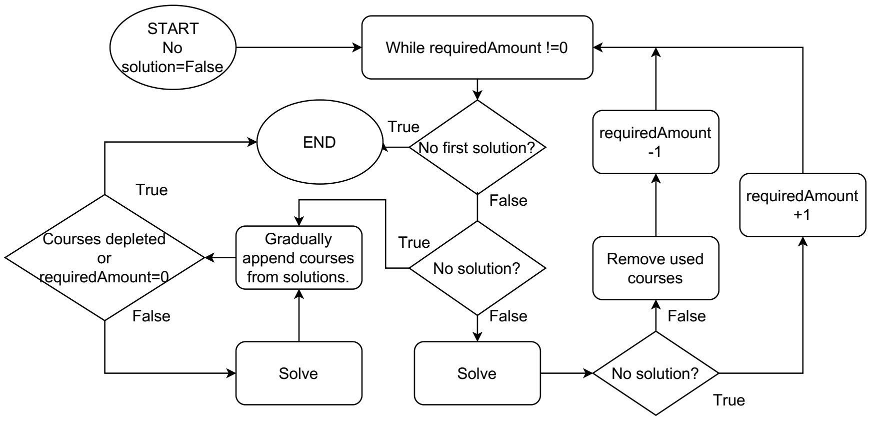

Fig. 1. Decision loop of application

The following equations describe the constraint functions that are applied to create the feasible region and by this fulfill the requirements. The $P_{\min}$ and $P_{\max}$ values defined in equation (2) refer to the price range in which the sum of the courses must move. In the equation (3), the duration limits of the selected courses are determined in the sum.

$$\sum_{i=0}^{M}(p_{r_i}b_i) \geq P_{min} \qquad \sum_{i=0}^{M}(p_{r_i}b_i) \leq P_{max} \tag{2}$$

$$\sum_{i=0}^{M}(d_{u_i}b_i) \geq D_{min} \qquad \sum_{i=0}^{M}(d_{u_i}b_i) \leq D_{max} \tag{3}$$

The constraint given in the equation (4) on the left determines the maximum number of courses included in the solution. On the right side, s indicates the knowledge groups included in the selected courses. The groups in the selected $Prof$ profile must be included in the course group, and s can also contain redundant courses.

$$\sum_{i=0}^{M} b_i \leq C_{max} \qquad \sum_{i=0}^{M}(s\,b_i) \geq Prof \tag{4}$$

The decision loop is the last part of the application. As the user has the option to demand multiple solutions, the loop is required to fulfill settings if possible. The logic of the whole process is displayed in figure 1. In short, decisions are based on multiple factors as whether the first solution was acquired, if the answer is yes, then the cycle repeats. This time if no solution is produced secondary loop is triggered and already used courses, are extracted from the solution list and are appended gradually back to the set used by solving algorithm. This allows the algorithm to find different combinations and terminate itself after all courses from the solution set are depleted. In the opposite case, where the solution was found again, the first cycle repeats itself until it eventually falls into the second loop as well or it runs out of required solutions first.

C. Implementation Details

The application is implemented in Python 3.9.7 programming language.

The choice is due to the wide support of optimization libraries. Regarding different libraries on which whole mathematical functions are based, it offers three most popular packages: Pyomo [6], Scipy [7], and Pulp [8]. Each library uses different syntax to achieve similar functions and they also bring unique functionalities. On their own, these libraries only offer a bridge between the mathematical format and the actual solver itself. What it does, is that it processes given input and relays data in a specific format to the chosen solver.

Scipy [7] is a well-supported module by the community but not by many solving packages. Another disadvantage is its input format acceptability. Scipy cannot work with dictionaries and only calculates with two-dimensional matrix input.

Probably the most widespread is Pyomo [6] library which includes slightly more additional functions like non-linear optimization and can even perform calculations with object-oriented models. Unlike Scipy, Pyomo also allows to presentation of the input data directly from databases, dictionaries, or tables. The repertoire of solver packages is similar to that of Pulp and their syntax is also somewhat similar.

Pulp [8] supports a wide amount of solver packages and also allows binary optimization which is crucial for this article. Besides that mixed integer programming is supported. Pulp has of all libraries easiest problem implementation and with that also comes a very short booting time when the solver is initialized. Pulp is the chosen library for this project and is currently in use together with multiple solvers. The main solver in use is currently CBC (Coin-or branch and cut) [5], but tests were conducted as well with HiGHS [9] and GLPK [10]. We tested all these three solvers in our work. To do so, we use the pulp library as it provides linking and data transfer to individual packages. All discussed solvers are open-source. Multiple commercial solvers would probably outperform all three of those presented, but no tests were conducted on them since we are aiming to implement and release the CSP application under an open source software license.

IV. Experimental Results

In this section, the solver packages, which are potentially candidates for our use in the CSP application, were subjected to testing on the used data sets. The results are summarized in this section and are displayed in Figure 2.

Solver CBC is the default package for Pulp and can be used with Pyomo as well [5]. For MIP problems is applied branch-and-cut algorithm which was discussed at the end of section III. Pulp had previously used GLPK as the default solver, but it was replaced by CBC due to better performance. Another solving package is HiGHS which is still in early development but offers promising results [11]. In addition to MIP, it also allows for solving quadratic programming models and everything can be solved either in series or in parallel.

Figure 2 shows that tests were conducted on two sets represented by 44 and 59 solvers on horizontal axis and time in seconds on vertical axis. The reason why a solution was found faster in a larger set is that fewer infeasible solutions were computed because there were more courses to choose from. Requested amount of combinations was three, but as it was more difficult to achieve this with less courses, actual number of computations was around nine. In contrast with a larger set it was only three, which was demanded. It is important to note that the data on the HiGHS package is not very accurate. There was a relatively frequent failure in the calculation, which was manifested in up to 36.84 % of cases in the first set and around 16.6 % in the second set. If we ignore this, HiGHS does have the best performance, but probably because it is still under development. It has frequent crashes that make it impossible to use this package normally [9]. Despite this, the HIGHS package proved to be the fastest, beating CBC by a hundred milliseconds at the most distinct point. Both solving packages far outperformed GLPK. This, and also its reliability is the reason why CBC was deployed as a solution package. Also GLPK allows only fairly short variable names, and along with HiGHS, have problems with special characters in them.

During tests, average memory usage was around 73,5 MB and GPU usage moved between 3,5 % – 6,7 % on Intel Core i5-6400 processor.

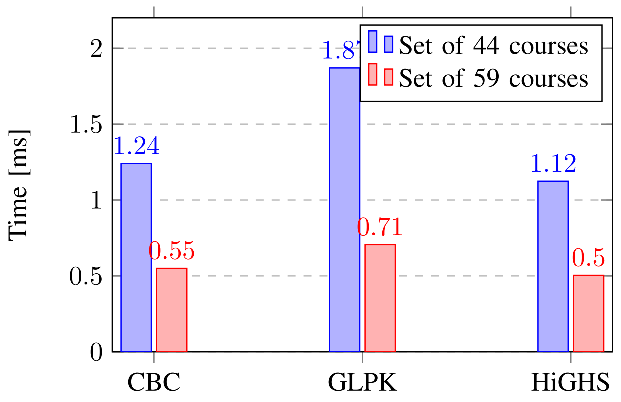

Fig. 2. Performance comparison of different ILP solver libraries

V. Conclusion

In this paper, we design, implement, and experimentally test the CSP application computing core. This core allows for identifying which professional training courses are recommended for a certain work role. Methods used for achieving desired results have proven to be efficient and remarkably quick.Linear programming with the simplex algorithm is largely used in different libraries allowing a comparison of their features. The Pulp library was fully compatible with the data format, and broad solver support enabled concurrent testing with promising results supporting the choice to deploy CBC in the final version of the application. The decision cycle seems to be sufficient but could provide more complex decisions in the future. Fundamental requirements for the module were met as it can match courses to profiles based on skill groups. There is room for improvement in the future in terms of more accurate matching. Scales could be implemented to properly display the value of each course and not to consider them all equal in terms of usefulness. More precise searching could be deployed as well because each course contains individual skills and knowledge which are not represented by skill groups in detail.

Acknowledgment

The following funding source is gratefully acknowledged: the ERASMUS+ programme of the European Union (grant 621701-EPP-1-2020-1-LT-EPPKA2-SSA-B 'REWIRE').

References

[1] European cybersecurity skills framework [online]. Accessed 2022-12-09. URL: https://www.enisa.europa.eu/topics/education/european-cybersecurity-skills-framework.

[2] Petr Dzurenda and Sara Ricci. Mapping the framework to existing courses and schemes [online]. Accessed 2022-12-01. URL: https://rewireproject.eu/wp-content/uploads/2022/11/REWIRE_R3.4.1_Deliverable-v7-Final.pdf.

[3] Kazuo Murota. Linear programming. In *Computer Vision*, pages 760–766. Springer, 2021.

[4] S. Skiena Steve. In *The Algorithm Design Manual*, volume 2, pages 266–414. London: Springer London, 2008.

[5] John Forrest. Introduction to cbc [online]. Accessed 2022-11-16. URL: https://coin-or.github.io/Cbc/intro.

[6] William Hart and Jean-Paul Watson. Pyomo model documentation [online]. Accessed 2022-11-25. URL: https://pyomo.readthedocs.io/en/stable.

[7] Eric Jones, Pearu Peterson, and Travis Oliphant. Scipy model documentation [online]. Accessed 2022-11-25. URL: https://docs.scipy.org/doc/scipy/dev/index.html#scipy-development.

[8] J.S. Roy, S.A. Mitchell, and F. Peschiera. Pulp model documentation [online]. Accessed 2022-11-13. URL: https://coin-or.github.io/pulp.

[9] Julian Hall, Ivet Galabova, and Michael Feldmeier. Highs package documentation [online]. Accessed 2022-11-24. URL: https://ergo-code.github.io/HiGHS/.

[10] Andrew O. Makhorin. Glpk - the gnu linear programming kit [online]. Accessed 2023-03-08. URL: https://www.gnu.org/software/glpk/glpk.html.

[11] Maximilian Parzen, Julian Hall, Jesse Jenkins, and Tom Brown. Optimization solvers: the missing link for a fully open- source energy system modelling ecosystem [online]. Accessed 2022-11-18. URL: https://zenodo.org/record/6534004.