# An Efficient Algorithm for Fast Service Edge Selection in Cloud-based Telco Networks

Javier Baliosian*, Luis M. Contreras†, Pedro Martínez-Julia‡, Joan Serrat§
*Universidad de la República, Uruguay; † Telefónica CTIO Madrid, Spain; ‡National Institute of Information and Communications Technology (NICT) Tokyo, Japan; §Universitat Politècnica de Catalunya, Spain.

***Abstract*—Telecommunication operators are increasingly integrating computational infrastructure into their networks at different location levels, including the network edge. This makes a highly distributed processing environment a reality, which is expected to enable next-generation services. This article proposes a novel and efficient algorithm to determine the best service execution locations through the "service edge", a concept that groups services in categories according to their requirements and benefits from the flexibility of distributed cloud resources. The paper is focused on describing the algorithm so that it can be integrated into the telco operators' management and orchestration systems. Simulation results underpin the practical feasibility of the proposed algorithm.**

## I. Introduction

Traditionally, telco networks (including both fixed and mobile networks) were designed following a hierarchical structure, with a precise function of traffic aggregation, having as a principal purpose the distribution of contents from the peering and transit points. This conventional design is being questioned due to the need to support different advanced services compelling a wide variety of performance requirements, traffic profiles, and connectivity types. Future telco networks will integrate cloud computing infrastructure [1] within several types of sites or Points of Presence (PoPs) of the operator's network, creating a topology that can satisfy Service Level Objectives (SLOs) such as throughput, delay, and processing capacity. In that context, the objective for the telco operator will be to facilitate an efficient distribution of the workload in the network, improving the overall network efficiency at the time of provisioning the services, accomplishing the expected user's service experience, and maximizing its profit. We understand profit here as the numerical expression of a given business policy. These PoPs will be distributed across different locations of the network [2], such as the operator's core network, central aggregation offices, base stations, or even on-premise or on-device. Figure 1 sketches the envisaged network. In the horizontal axis, we show the order of magnitude of the number of available cloud computing facilities quantified in three distinct domains. In the vertical axis, we show the order of magnitude of latency, expressed in milliseconds, that some services may require, hence being constrained to be deployed in the edge, regional, or centralized computing facilities.

Nevertheless, latency is not the only SLO that can impact the location of services at deployment time. We are considering 3GPP service scenarios [3][4], with the three types of 5G service categories, namely enhanced Mobile Broadband (eMBB), ultra Reliable Low Latency (uRLLC) and massive Machine Type Communications (mMTC). Services in these categories are characterized through different latency, jitter, or traffic density. All these performance parameters can be considered as constraints for deciding where to instantiate a given service. In addition, minimization of energy consumption, balancing network and cloud workloads, maximization of profit, and geographic or regulatory constraints, are examples of complementary decision criteria. The adoption of a specific subset of these criteria is up to the telco operator.

Without lack of generality, the work presented in this paper considers that services are characterized by the following four technical dimensions:

- **Throughput** (T), as the data rate at which the service generates or consumes data,
- **Latency** (L), characterized by the end-to-end delay of the data transmission,
- **Computation** (C), associated with the processing needed by the number of sessions to be established and
- **Storage** (S), identified by the memory size of the data to be stored.

The motivation to select these four dimensions is that they are key differentiating parameters of the three types of 5G service categories mentioned above and, at the same time, they are directly related to the capabilities of the hosting infrastructure (both, cloud and network infrastructures).

Moreover, in line with 5G, we assume that services are deployed on top of cloud-network slices. A cloud-network slice is constituted by slice parts, where each slice part is a bundle of computing and/or networking resources [5]. For illustrative purposes, a slice part could be represented utilizing the picture of Figure 1, i.e., a set of different types and locations of computing and networking resources, which are shared among several isolated slice parts. On the other hand, a 5G service is usually deployed as a bundle of microservices, on top of the computing resources of one or several slice parts, to satisfy specific computation and storage requirements, being connected through network-slice parts, guaranteeing throughput and latency bounds. It is worthy to say that different microservices of a given service can have different requirements, and therefore they may need to be placed on different service edges. In that framework, the problem faced by the telco operator that we deal with in the article can be sketched as follows: given a set of services or parts of services (i.e., microservices), each specified by a set of requirements in

terms of throughput, latency, computation, and storage needs, find the computing resources to host them, satisfying these requirements and maximizing the associated profit. In other words, our problem consists of finding the $service\ edge$, as described in our former work [6], for each one of these services or microservices. The grouping of microservices into deployment units, i.e., to be deployed jointly in the same computing facility, is service-specific and falls outside the scope of this paper.

Having a service orchestration tool to solve, fully automated, the above problem will become a must for telco operators aiming to rationalize the investment needed to support a high number of computing facilities on the network. Selecting the most appropriate execution environment for each service is, in essence, an exercise of efficiency and optimization. This paper describes an efficient mechanism for service deployment on computational infrastructures, taking into consideration parameters like throughput, latency, computation, and storage. With that aim, it extends and complements the work in [6] by modeling the problem as a multidimensional flow problem and proposing a new heuristic technique to solve it. This heuristic leverage realistic assumptions on the number of service and infrastructure types that can be encountered, addressing the typically intractable problem of optimally allocating multi-requirement services on hosting infrastructure. Its principal characteristic is that the time needed to solve the problem is short enough to be used in practical decision-aided systems in operational networks with results very close to the optimum. We envisage the proposed mechanism as part of the service management and orchestration system used by telco operators for the deployment of services, when transitioning towards a softwarized network approach.

The paper is organized as follows. The section after this introduction brings the necessary background to pave the way to describe our proposed heuristic; then, we present the first contribution of the paper, which consists of an extension of the conventional Successive Shortest Path algorithm (SSP) to a multidimensional space, which is called Vectorial Successive Shortest Path (VSSP) algorithm, setting the grounds of the following section where we show how the proposed VSSP algorithm can be used to decide the allocation of services on the existing computing environments. That is the second contribution of the paper. Later, we provide a simulation-based evaluation comparing the results of the proposed heuristic with the optimum solution given, when possible, by a Mixed Integer Linear Programming (MILP) solver. The final section concludes the paper and presents some future steps.

## II. The Problem of Assigning Services to Hosting Infrastructure

As stated in the previous sections, our contribution is an algorithm that maximizes the profit of telco operators deploying services, or microservices, in their respective optimal service edges. As described in [6], the concept of service edge is an extension of the physical network edge. In fact, a service edge is characterized by a set of characteristics, which in this work are materialized as the throughput at which the service will be working, its end-to-end latency, and the computation and storage capabilities offered to that particular service. Services hosted in their optimal service edges will also fulfill their respective SLAs in terms of the above mentioned four characteristics.

The problem we are solving by means of the proposed algorithm is a particular case of the well-known Assignment Problem [7]. Although later in the article we will address more complex versions, we start the description of our proposal with a simple version of the problem; in it, services require only one type of resources (e.g., storage).

A common way to solve this kind of problem is through a modeling strategy that uses *flow networks*. Flow networks are used in [8] to minimize the cost of virtual switching induced by an NFV service chain deployment. This is a different problem, albeit related to ours, and showing the power of that modeling instrument.

A flow network is a graph constituted by a set of nodes and a set of edges; on this graph, a hypothetical flow emerges from one of the nodes (the source), flows through the edges passing through intermediate nodes, and disappears in the sink node. (see [9] for more details). Following a common way of using this abstraction, the Assignment Problem can be modeled as a flow network with four layers as in Figure 2; one layer composed of a single source node $s$; a layer of nodes representing each one a different type of services; another layer of nodes representing types of computing infrastructure, and finally, a layer corresponding to the sink node $t$. Nodes representing service types and infrastructure types are interconnected through a full mesh of graph edges. Please note that to avoid confusion with the edges of the telco network or even its communication links, we refer to the edges of the flow network graph as *graph edges*. Each graph edge is characterized with a weight and a capacity whose roles are described hereafter.

Essential to understand how a flow network works are the flows between service type nodes and resource type nodes. A flow between a given service type and resource type is an integer that accounts for the total amount of resources needed by services of this type. These flows are indeed determined by the algorithm adopted to solve the problem. The flow conservation property must hold at each node, meaning that the flow entering into a service type node must be equal to the sum of flows leaving that node. It is also worth mentioning that the flow leaving a given infrastructure type node towards the sink node, representing the total amount of resources needed of that type of infrastructure, can not exceed the resources available of that specific type of infrastructure. The weight of a graph edge linking a service type with an infrastructure type (i.e., from layer 2 to 3) is the one minus the normalized profit that the telco operator obtains deploying that service type in that infrastructure type. Graph edges where either the source or sink nodes are involved have no weight assigned. This way, looking for the minimum weight flow from source to target is equivalent to look for the flow that maximizes the profit.

A well-known algorithm to compute the minimum weight flow is the Successive Shortest Path algorithm (SSP) [10]. The SSP looks for the maximum flow with the minimum weight that can be injected from the source node to the sink node on

a flow network graph as represented in Figure 2.

This algorithm aims to iteratively look for the path of minimum aggregated weight between the source and the sink and then "send" as much flow as possible through that path. Observe that the weight of a graph edge can also be seen as a distance from one end to the other of that edge; that is the reason to talk about "shortest path" as synonymous with minimum weight path. This is exemplified by the red flow shown in the first step of Figure 3 that carries a maximum flow of two units through a path whose accumulated weight is $1 + 1 + 1 = 3$ units. This maximum flow reduces the capacity of the graph edges that constitute the path to support other flows. In the following iteration, to allocate the subsequent flow, we can realize that it would be better to send the initial flow through a higher weight path and avoid blocking the second flow, which is a stream that shares some links of the path mentioned above. To solve that, the idea is to allow the algorithm to "push backward" some already established flow to make space for a new one that would add more flow. Observe that allowing a higher flow means allocating more resources and, therefore, get a higher profit. That is represented in the second step of Figure 3, where the green flow of two units is pushing back the red flow to leave the solution as represented in the third step of Figure 3.

The advantage of this algorithm is in terms of its complexity from the point of view of the runtime, which has been demonstrated to be the lowest among the set of known approaches to solve the minimum weight maximum flow problem. A detailed description of the algorithm can be found in [10].

## III. Vectorial Successive Shortest Path

The one-dimensional (only one type of resources) assignment problem is not enough to model our 5G scenarios because they require to specify several simultaneous service requirements and the availability of different resources. Elaborating a similar idea, but for multidimensional flows (each flow is characterized by a vector of flows), it is not straightforward. Those different dimensions of the flow must be conveyed together. A key contribution of this paper is to extend the solution described for a one-dimensional flow network to a multidimensional one.

To highlight our approach, let us consider the main differences of the model with respect to the one-dimensional case. We depart, as before, from a graph constituted by a subset of nodes representing service types, another subset representing resource types, a source, and a sink. Flows are also from the source to the sink going through different links in the graph. One difference is that now, flows are vectors with as many components as dimensions we consider. Recall that such dimensions are related to throughput, latency, computation, and storage, as described above. In addition, another difference is that now, as the flows injected into the graph network are vectors, the graph edge's capacity is a vector with components in each of the dimensions.

Similar constraints to the uni-dimensional case can be formulated. In particular, flows have to be routed through the graph so that those passing through a given graph edge must carry an accumulated flow that is lower than or equal to the graph edge capacity in each dimension. In addition, the incoming flow at each node must be equal to the flow leaving it, except for the source and sink nodes.

The idea to extend the SSP algorithm to a multidimensional setup and thus the base of a novel algorithm called *Vectorial Successive Shortest Path* (VSSP) is that as in SSP, VSSP successively searches for the path of minimum weight between the source and the sink, and then "sends" as much flow as possible through that path, this is, as much flow as can pass through the thinner path's edge.

To illustrate how VSSP works, let us consider a simplifying example, with vectors of only two dimensions. Any of the graph edges will look like the one represented in Figure 4 with two different pipes, namely, the red and the green one. These two pipes have capacities as those shown in the cut highlighted in the top-right. The figure also shows two different types of services to be allocated, the *yellow* and *emerald* types; each type has *red* and *green* requirements and a number of service instances each. As in SSP, VSSP has to pass through this graph edge some flows corresponding to service instances selected so that the maximum number of instances is taken without exceeding the capacities of both of the two pipes. That is a well-known combinatorial optimization problem called the Multi-knapsack problem [11]. This combinatorial problem is known to exhibit non-polynomial (NP) time complexity; however, as the size of the problem is bounded by the number of dimensions of each infrastructure type, the problem can be solved in practice as a Mixed Integer Linear Programming [11] utilizing a solver like Matlab's *intlinprog*, among others.

As in SSP, we can realize that it would be globally better to send part of the previous flow through a higher weighted path and avoid blocking a more significant stream at each iteration. In this case, we proceed as the SSP does "pushing" backward some flow that is already passing through some graph edges and make space for a new one that would add more flow. The particularity in VSSP is that this flow retraction has to be performed for all dimensions in the proportions defined by the services represented by those flows. In the pictorial representation of Figure 4, the amount of *red* flow and *green* flow retracted from the graph edge must be equal to the amounts used by some combination of the *yellow* and *emerald* services crossing that graph edge.

Contrariwise SSP, we can not guarantee that VSSP ends up with the optimum solution. In addition, the complexity of the VSSP is higher than SSP, but as shown in the evaluation section, the time needed for the computation in practical scenarios is short enough to be considered a feasible solution.

## IV. Solving the Edge Assignment Problem with VSSP

5G infrastructures, such as those considered in this paper, might be potentially sized in some thousands of nodes, and the number of services to be allocated can be of a similar order. To reduce the size of the problem and make it computationally tractable, we assume that each infrastructure element, i.e., computation server, belongs to a type, like those

in Figure 1, with common characteristics in terms of storage capacity, computation power, and available throughput. Hence, we have a set of different infrastructure element types, each constituting what we call a $service\ edge$ [6]. Each of these infrastructure element types will be characterized by a three-dimensional vector of available resources corresponding to its maximum bandwidth, computation, and storage capacity (the delay dimension is accounted for differently, as explained later on). In addition, we also assume that each service, or service component, to be allocated belongs to a type with common requirements such as bandwidth, computation power, storage, and delay. Consequently, we have a set of different service types. We also assume that when a service of a particular type is deployed on a particular type of infrastructure element, it generates a given profit for the operator. Hence, we model the *service edge* selection as a vectorial, minimum-weight, maximum-flow problem, suitable to be solved employing the proposed VSSP heuristic.

Therefore, we build a graph organized into four layers as in Figure 2. Layers 1 and 4 contain just the source and sink nodes, $s$ and $t$. The flow generated by $s$ consists of a vector with integer components, each representing the total resources in each dimension. The sink node $t$ at layer 4 absorbs the vector flow generated at the source. Layer 2 has as many nodes as service types, whereas layer 3 has as many nodes as different infrastructure element types.

All the graph edges have the same format, namely, a scalar representing the weight of that edge and a three-dimensional vector with the flow capacities of that edge in each of the three dimensions. In the following paragraphs, we describe the values adopted for each graph edge.

Each graph edge between layers 1 and 2 is assigned a zero weight and a capacity vector that allows carrying the total aggregated flows of the service type that this graph edge connects. The reason to assign a zero weight to those graph edges is that the routing of a given amount of flow to a particular service type node is dictated by the number of services of that particular service type and therefore has not to be decided by the optimization algorithm.

The graph edges from layer 2 to layer 3, between a given service type and an infrastructure type, are weighted with a scalar that is the profit induced by deploying that service type on that infrastructure type with a negative sign. In that way, we play with weights derived from telco operator profits and leave the algorithm to select the graph edges that minimize weights (maximize profits). The capacity vector, in that case, is infinity in each one of the three dimensions, i.e., the capacity is not constrained here because the constraints posed by the infrastructure types in each dimension are better represented by the subsequent graph edges as explained hereafter.

Graph edges between layers 3 and 4 have assigned zero weight and a capacity vector representing the total capacity in each dimension, supported by the infrastructure type at which each graph edge is connected. Observe that, in this way, we include the resource capacity constraints in our model. On the other hand, the reason to assign zero weight to these graph edges is similar to assign zero weight to graph edges between layers 1 and 2.

Finally, to model the delay constraints, we remove all graph edges between services and those infrastructure element types not fulfilling the delay requirements. It is worthy to say that our approach would allow the modeling of more fine-grained delay specifications, but for that case, it would be necessary to create additional infrastructure element types quantifying the delay attribute.

### A. Service Allocation at Service Request Time

As the reader may have noted, the process described above works allocating a complete set of services at the same time (e.g., all services requested within a given time window are deployed together at a given time). Nevertheless, in a more realistic scenario, services have to be allocated as soon as requested by their customers. It would be service disruptive to re-allocate all the services every time a new request arrives; however, the method presented here permits assigning a new service request by just recomputing the flow of its particular service type. VSSP permits to do that very fast because it requires only a single iteration to compute the augmented flow. The same idea and complexity are valid for removing a service. The reader may notice that just recomputing a restricted number of flows may end up with results that may not be as good as if we were running the algorithm from scratch again. Nevertheless, the accuracy of that approximation is out of the scope of the paper and left for future work.

## V. Evaluation

In order to compare the performance of our approach, we selected a *ground-truth* reference provided by a well-established solver of Mixed Integer Linear Programming (MILP) problems. With *ground truth* we refer to an ideal value or *gold standard* that might not be found with the available resources of a real deployment. With this purpose, we use Matlab's *intlinprog* [12], which uses a sequence of strategies that have been accepted as very effective solving MILP problems [13]. Although this solver might find the optimum solution (when it does not, it can detect and inform this situation), it does not scale well with the problem's size. Therefore, the comparison we present in this section was made in the problem size range where *intlinprog* was able to find a solution in a reasonable time.

This evaluation aims to show that VSSP finds the service edges acceptably close to the actual optimum (given by our ground truth) in a period of time that is compatible with the management and orchestration processes of cloud-based telco networks. Thus, we present the results of two experiments: i) a comparison between the time spent by VSSP and MATLAB's *intlinprog* to solve problems of the same size, and ii) a comparison between the *profit* obtained through the solution of VSSP and this solver.

The experiments made for this evaluation were performed on a synthetic network where only relevant aspects of topology and resources were simulated. The setup was configured to have always ten service types with 100 service instances each. Service types ranged from three to 30 normalized units of storage, throughput, and computing requirements in intervals of

three units. The number of infrastructure types was swept from 5 to 15. Infrastructure types range from 240 to 430 normalized available units of storage, throughput, and computation, in intervals of, at most, 25 units. The number of infrastructure elements inside each infrastructure type is set such that all their resources together can support 90% of the total service demand.

For this evaluation, we avoided lineal profit schemes to stress the edge selection method's time complexity; lineal schemes would make the problem easier to solve. Thus, the telco operator profit scheme is logarithmic in the number of resources required by a service type and inversely logarithmic in the size of the resources available in a particular infrastructure element.

The tests were run on an Intel® Core™ i7-8550U CPU, at 1.80GHz, with 16GB of RAM.

Figure 5 depicts in logarithmic scale the running time to achieve the solution for VSSP and Matlab's *intlinprog* solver as a function of the number of infrastructure types. The plots are the mean of 10 experiments, and the error bars represent the 95% confidence interval. The circles correspond to cases in which the *intlinprog* solver gives up timing out before reaching the optimum, which is normal behavior for this solver.

In this graph, it is possible to appreciate the significant run time difference between solving the problem using the MILP solver and VSSP. As mentioned, the 15 infrastructure types boundary was set to let the MILP solver finish in a reasonable time, but VSSP can deal with a much larger problem size. Although not included among the results for the sake of brevity, VSSP's scales well, as inferred from the trend of the figure's plot.

Figure 6 shows the total profit induced by the solutions computed by VSSP and Matlab's *intlinprog* solver. Again, the plots are the mean of 10 experiments, and the error bars are the 95% confidence interval. The number of service instances per service type is always 100, and the number of servers per infrastructure type changes to keep a ratio *demand/offer* of 0.9.

From this graph, we conclude that VSSP computes a sub-optimal solution close to the optimum computed by MATLAB's MILP solver; however, this is made in a time, which in some cases, is several orders of magnitude shorter, enabling its use in cloud-network orchestration systems.

## VI. Concluding Remarks

This paper proposes a technique for allocating services to computational resources spread over all network domains to fulfill heterogeneous service requirements in the context of 5G and beyond.

We formulate this service allocation problem and elaborate on how to solve it by extending the solution of a similar but simpler problem. This extension is called Vector Successive Shortest Path (VSSP), a heuristic derived from the well-known Successive Shortest Path (SSP) algorithm, which constitutes a contribution of the paper to state of the art. In addition, as a second contribution, we describe how to use VSSP to solve the allocation problem when services are characterized by four requirements: throughput, latency, computation, and memory storage.

Simulations have been conducted to compare the results obtained with VSSP against a conventional Mixed Integer Linear Programming (MILP) solver, which gives the optimum allocation. Results show relatively close telco operator profit for both but with orders of magnitude less computation time when obtained by VSSP. This shows that our solution approach is precise enough and scalable to be used in more complex scenarios. The impact of incrementally assigning/de-assigning services as soon as these services are requested or terminated is also addressed without disrupting those already deployed. However, its accuracy analysis is a significant challenge that is left for future work.

Solutions like our approach are needed to efficiently select the execution environment for services on distributed cloud facilities of different characteristics. This is a challenging problem that is going to be faced soon by telco operators. Even for delay-sensitive services, the service location is not necessarily the physical edge of the network but some other location yet convenient for respecting the committed SLOs, what is called the *service edge*. The performance of the VSSP algorithm reported in this paper makes it a firm candidate to find the *service edge* and therefore to be integrated on cloud-network orchestration systems (e.g., [14]).

## Acknowledgment

The European Commission has partly funded this work through the projects NECOS (Grant Agreement N$^o$ 777067) and 5G-DIVE (Grant Agreement N$^o$ 859881).

## References

[1] L. M. Contreras, V. Lopez, O. G. De Dios, A. Tovar, F. Munoz, A. Azanon, J. P. Fernandez-Palacios, and J. Folgueira, "Toward cloud-ready transport networks," *IEEE Communications Magazine*, vol. 50, no. 9, pp. 48–55, Sep 2012.

[2] Telefonica, "Telefónica Open Access and Edge Computing," no. February, 2019. [Online]. Available: https://www.telefonica.com/documents/737979/144981357/whitepaper-telefonica-opa-mec-feb-2019.pdf (Accessed on 2021-05-06).

[3] 3GPP, "TS 122 261 - V15.5.0 - 5G; Service requirements for next generation new services and markets (3GPP TS 22.261 version 15.5.0 Release 15)," Tech. Rep., 2018. [Online]. Available: https://portal.etsi.org/TB/ETSIDeliverableStatus.aspx (Accessed on 2021-05-06).

[4] 3GPP, "TS 123 501 - V15.2.0 - 5G; System Architecture for the 5G System (3GPP TS 23.501 version 15.2.0 Release 15)," Tech. Rep., 2018. [Online]. Available: https://portal.etsi.org/TB/ETSIDeliverableStatus.aspx (Accessed on 2021-05-06).

[5] "Novel Enablers for Cloud Slicing - NECOS Project." [Online]. Available: http://www.h2020-necos.eu (Accessed on 2021-05-06).

[6] L. M. Contreras, J. Baliosian, P. Martínez-Julia, and J. Serrat, "Computing at the edge: But, what edge?" in *NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium*, 2020.

[7] J. N. Hooker, "Chapter 15 - Operations Research Methods in Constraint Programming," in *Handbook of Constraint Programming*, ser. Foundations of Artificial Intelligence, F. Rossi, P. van Beek, and T. Walsh, Eds. Elsevier, 2006, vol. 2, pp. 527–570.

[8] M. C. Luizelli, D. Raz, and Y. Sa'ar, "Optimizing nfv chain deployment through minimizing the cost of virtual switching," in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, 2018, pp. 2150–2158.

[9] D. R. Ford and D. R. Fulkerson, *Flows in Networks*. Princeton, NJ, USA: Princeton University Press, 2010.

[10] R. Ahuja, T. Magnanti, and J. Orlin, *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, 1993.

[11] H. Kellerer, U. Pferschy, and D. Pisinger, "Multidimensional Knapsack Problems," in *Knapsack Problems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 235–283.
[12] The MathWorks, "Optimization Toolbox™, User's Guide," The MathWorks, Natick, MA, Tech. Rep., 2019.
[13] H. D. Mittelmann, "Latest Benchmarks of Optimization Software," in *INFORMS Annual Meeting 2017*, Huston,TX, 2017.
[14] L. Contreras, D. Lachos, and C. Rothenberg, "Use of ALTO for Determining Service Edge," Internet Engineering Task Force, Internet-Draft draft-contreras-alto-service-edge-02, Nov. 2020. [Online]. Available: https://datatracker.ietf.org/doc/draft-contreras-alto-service-edge/ (Accessed on 2021-05-06).

## BIOGRAPHIES

**Javier Baliosian** (M.Sc. 1998, Ph.D. 2005) Associate Professor, at University of the Republic, Uruguay.

**Luis M. Contreras** (M.Sc. 1997, M.Sc. 2010) Technology Expert at Telefonica, Spain.

**Pedro Martinez-Julia** (M.Sc. 2010, Ph.D. 2015) Researcher at NICT, Japan.

**Joan Serrat** (M.Sc. 1977, Ph.D. 1983) Full Professor at UPC, Spain.

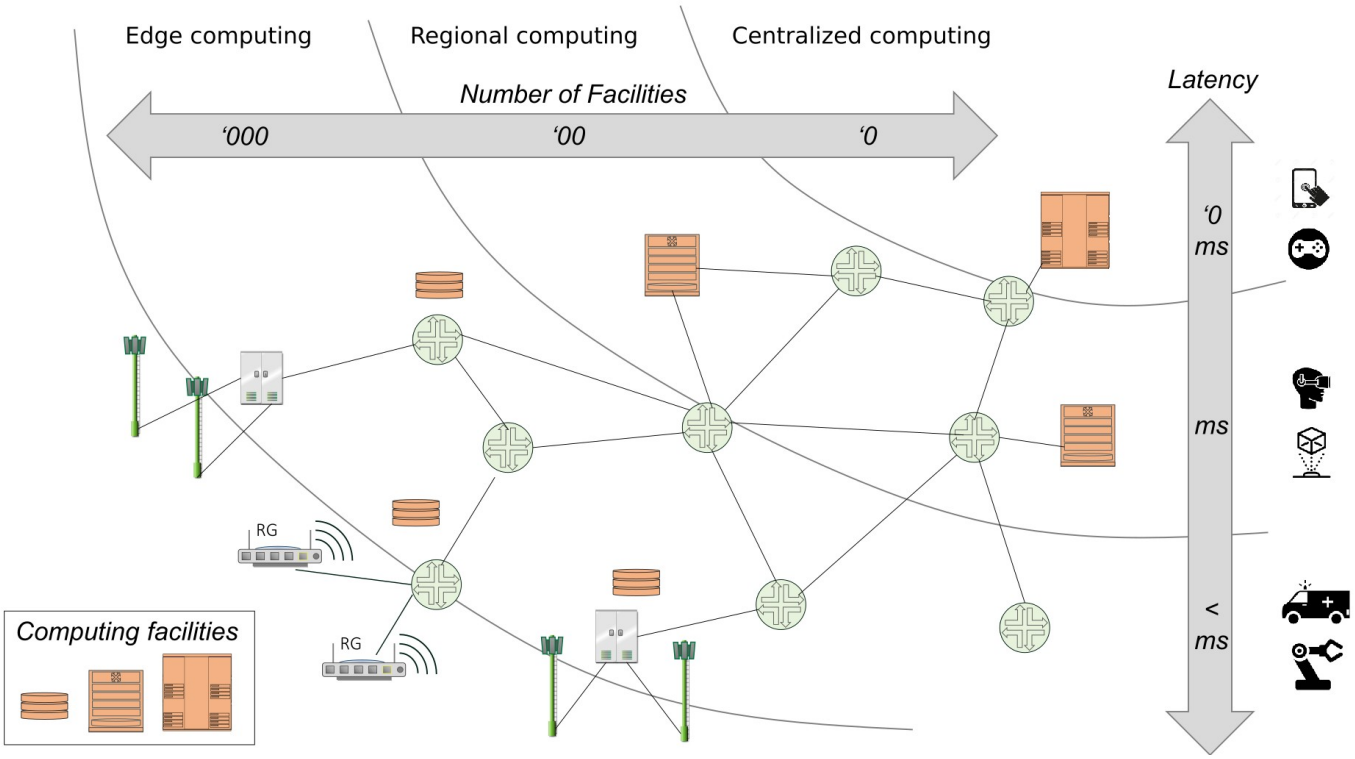


Fig. 1: Potential placement options depending on the type of service.

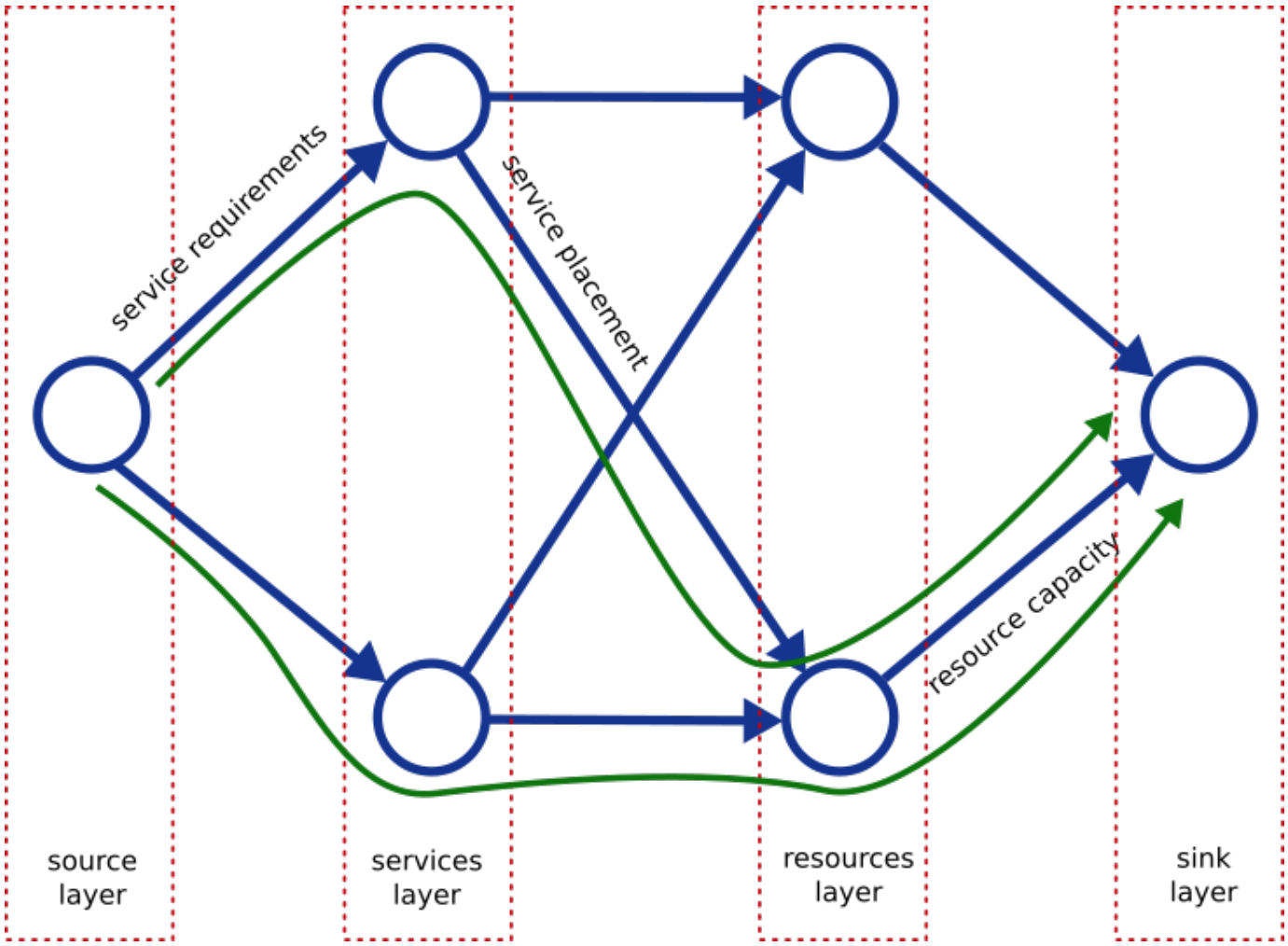


Fig. 2: The Assignment Problem modeled as a Network Flow Problem.

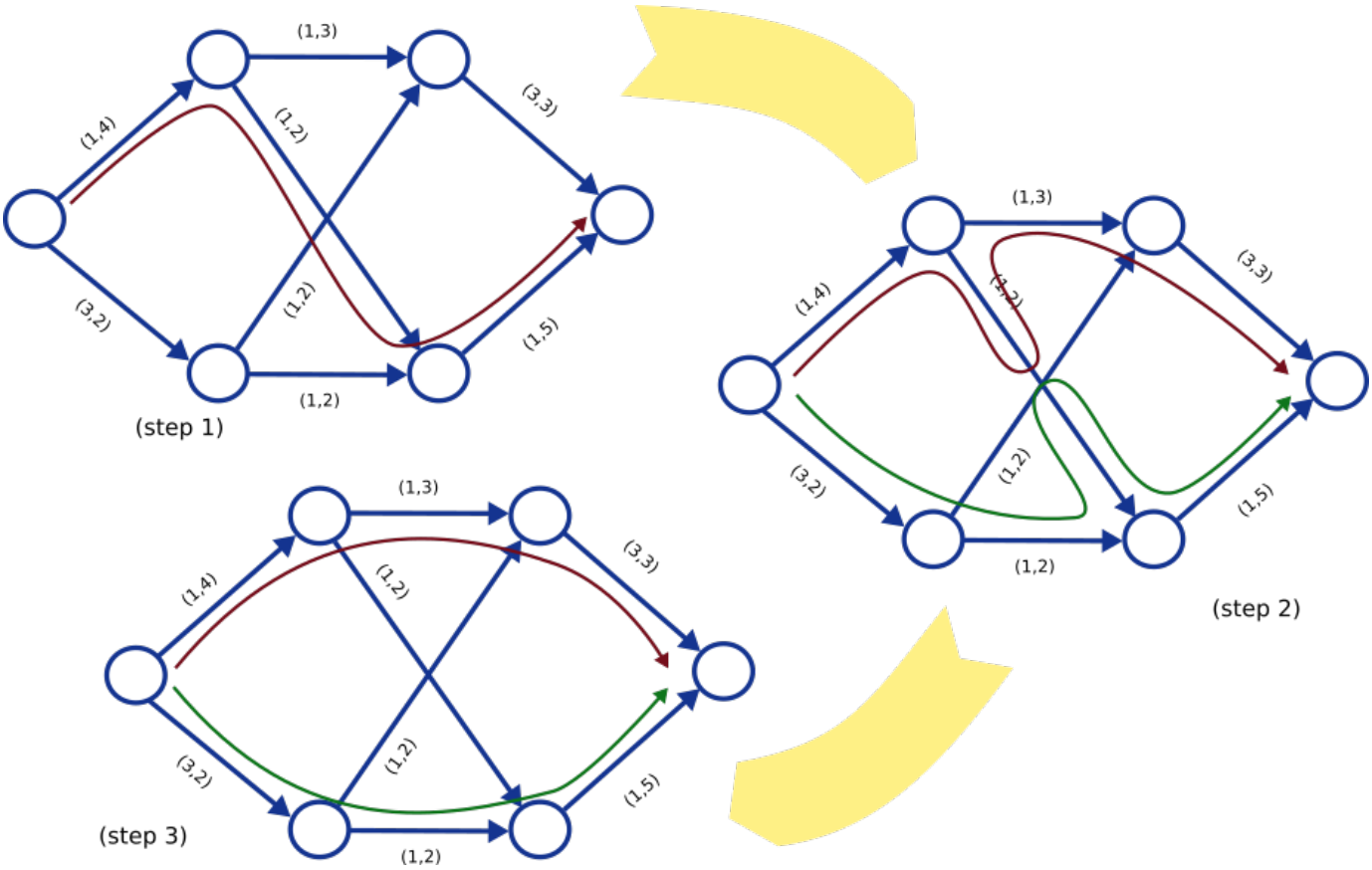


Fig. 3: Three consecutive steps solving the *assignment problem* utilizing SSP. Labels are pairs $(w, r)$ where $w$ is the weight of the graph edge and $r$ is its capacity.

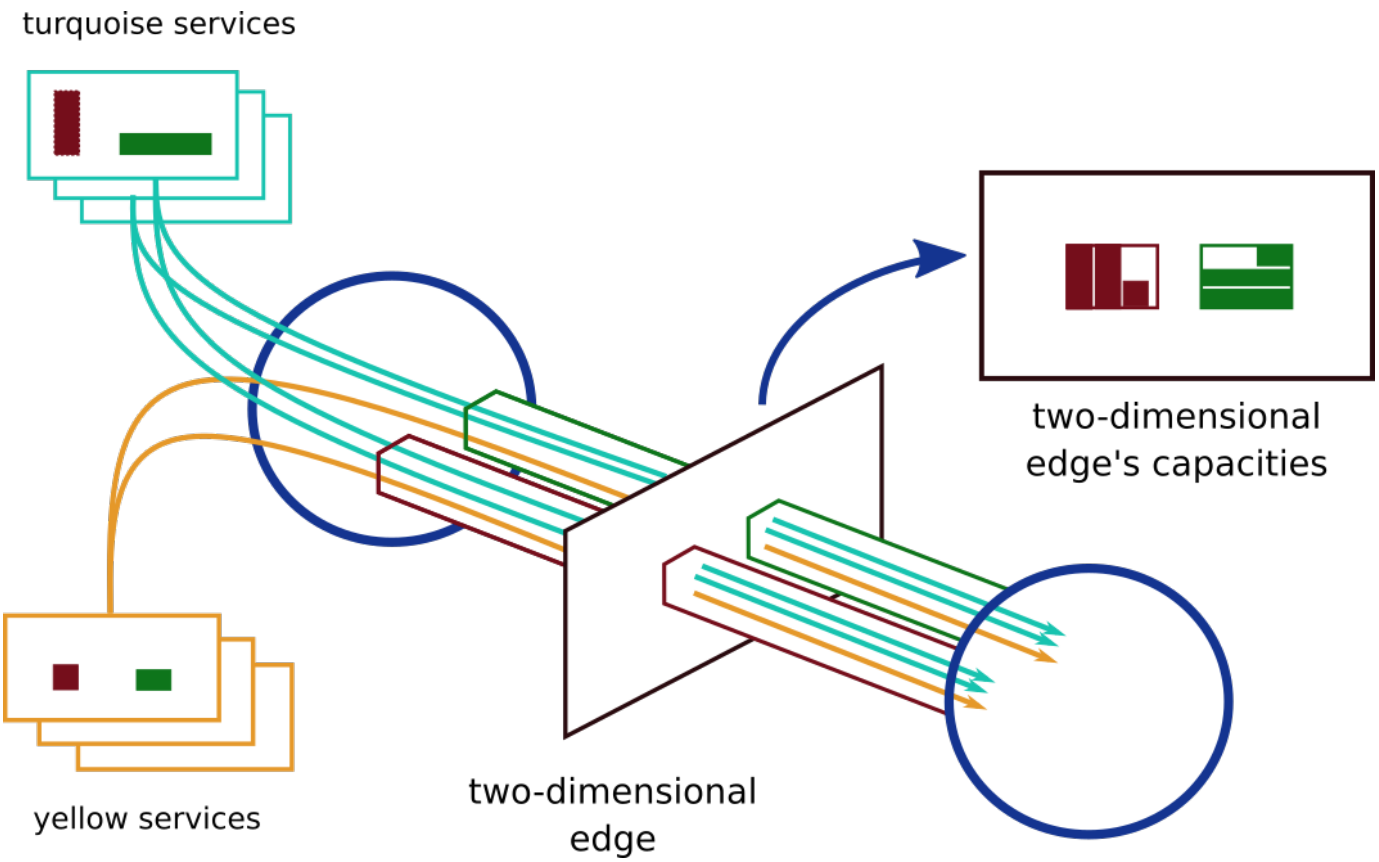


Fig. 4: At each graph edge, accommodating different services to optimize the use of the different resources is equivalent to solve a multi-knapsack problem.

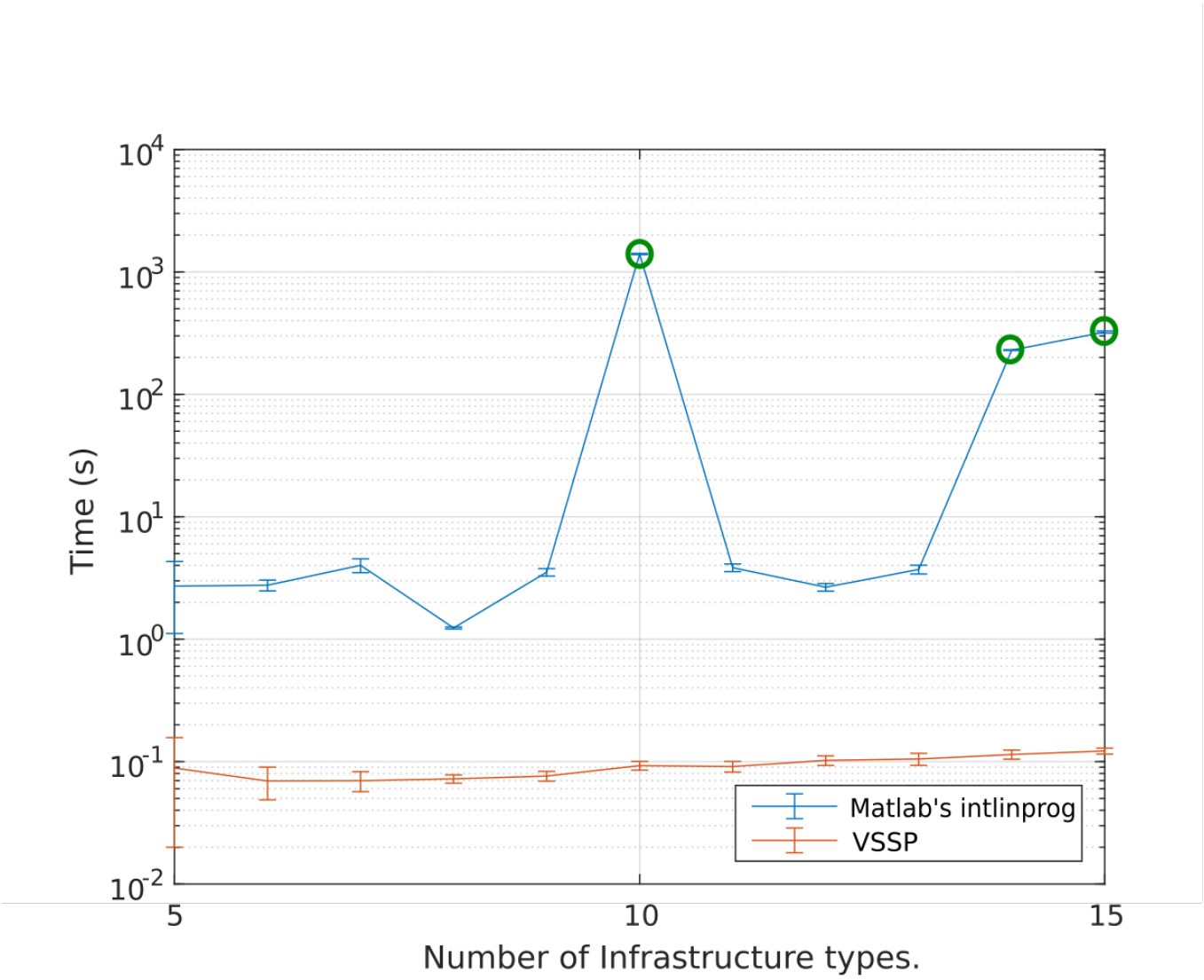


Fig. 5: Running times for VSSP and Matlab's *intlinprog* solver in a logarithmic scale. Green circles correspond to cases in which the *intlinprog* solver gave up per time before reaching an optimum.

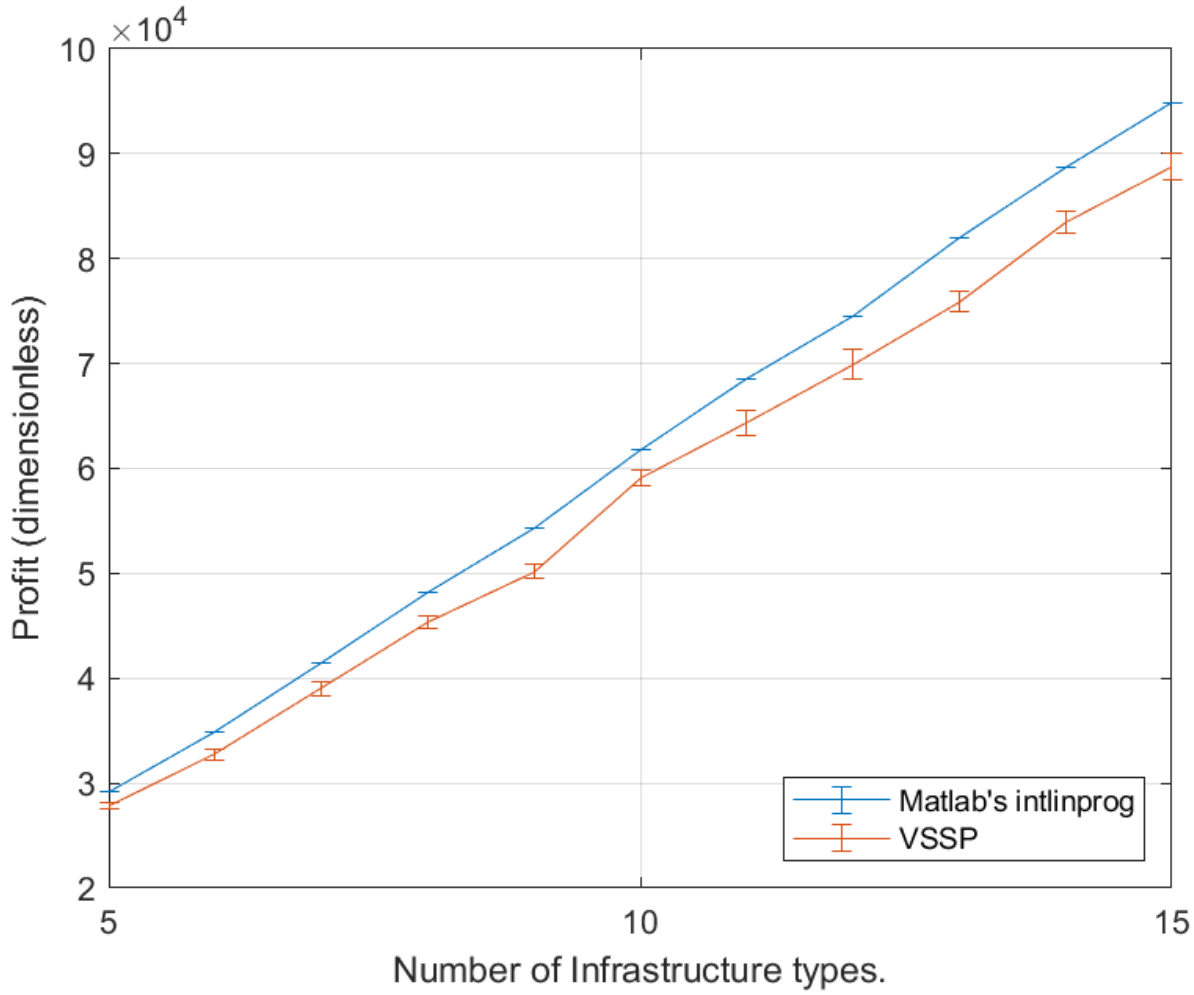


Fig. 6: Total profit induced by VSSP and Matlab's *intlinprog* solver.