Parallel Algorithms for the Multiobjective Virtual Network Function Placement Problem *

Joseph Billingsley[1], Ke Li[1], Wang Miao[1], Geyong Min[1], and Nektarios Georgalas[2]

[1] Department of Computer Science, University of Exeter, UK
{jb931, k.li, wang.miao, g.min}@exeter.ac.uk
[2] Research and Innovation, British Telecom, nektarios.georgalas@bt.com

Abstract. Datacenters are critical to the commercial and social activities of modern society but are also major electricity consumers. To minimize their environmental impact, we must make datacentres more efficient whilst keeping the quality of service high. In this work we consider how a key datacenter component, Virtual Network Functions (VNFs), can be placed in the datacenter to optimise the conflicting objectives of minimizing service latency, packet loss and energy consumption. Multiobjective Evolutionary Algorithms (MOEAs) have been proposed to solve the Virtual Network Function Placement Problem (VNFPP), but state of the art algorithms are too slow to solve the large problems found in industry. Parallel Multiobjective Evolutionary Algorithms (PMOEAs) can reduce execution time by distributing the optimization process over many processes. However, this can hamper the search process as it is inefficient to share information between processes. This paper aims to determine whether PMOEAs can efficiently discover good solutions to the VNFPP. We found that PMOEAs can solve the VNFPP 5 - 10x faster than a sequential MOEA without harming solution quality. Additionally, we found that one parallel algorithm, PPLS/D, found *better* solutions than other MOEAs and PMOEAs in most test instances. These results demonstrate that PMOEAs can solve the VNFPP faster, and in some instances better, than sequential MOEAs.

Keywords: Network Function Virtualisation · Multi-Objective Optimisation · Parallel Multiobjective Evolutionary Algorithms

1 Introduction

Recent research indicate that datacenters will be responsible for between 3% and 5% of total energy consumption worldwide by 2030 [6]. With the disastrous impact climate change could have, there are environmental as well as business imperatives to improve the efficiency of datacenters. From 2010 onwards, datacenters became more energy efficient by reducing energy spent on 'overhead' [7]

* Supported by EPSRC Industrial CASE and British Telecom under grant 16000177.

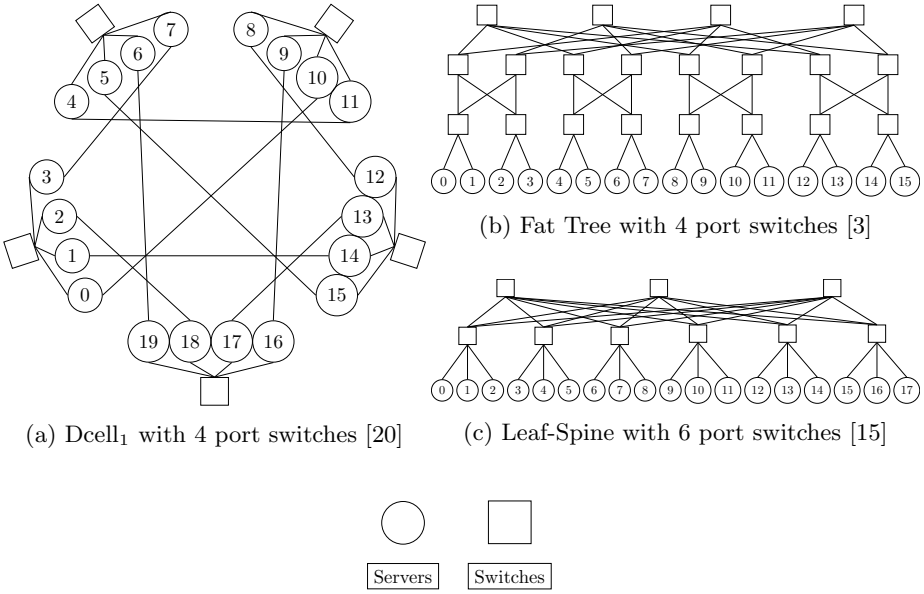
i.e. energy consumed by fans, pumps, transformers and other auxillary equipment. Despite these efforts the total energy consumed by datacenters doubled from 2010-2020 [17] and there are diminishing returns to reducing overhead further. Recent work has identified that the computing components of the datacenter, e.g. servers and switches, are now where the greatest efficiency improvements can be made [17].

Network function virtualization is a technology that will allow datacenter components to be used more efficiently. A network function is a datacenter component that performs a specific task such as load balancing or packet inspection. Services, such as phone call handling or video streaming, are formed by directing traffic through network functions in a prescribed order. Traditionally, these functions were provided by 'middleboxes' implemented in purpose-built hardware. However, middleboxes cannot easily be reconfigured, added or removed from the datacenter and hence consume energy even when not required. Virtual network functions (VNFs) provide the same functionality as middleboxes but run on software that is executed on a virtual machine. New instances of VNFs can be created or destroyed in seconds [1] allowing the datacenter to spend energy on services only when necessary.

To use VNFs efficiently, the optimal number of instances of each VNF must be placed in the datacenter. In the VNF Placement Problem (VNFPP) there are multiple conflicting quality of service (QoS) metrics (e.g. expected latency, packet loss) for each service in the network, which must also be balanced against the energy consumption of the datacenter. A VNFPP instance defines a set of services to provide and a datacenter topology. A solution to the problem defines where VNFs for each service should be placed and how packets should traverse the datacenter to form services.

Many means of solving the VNFPP have been explored however no algorithm has had widespread acceptance. Whilst exact [8, 9, 25], heuristic [21, 29, 30] and metaheuristic [10, 32, 35] optimization methods have been considered, existing algorithms are limited by the speed they can evaluate solutions to large problems. Packet level models of a datacenter such as discrete event simulators, accurately measure the datacenter metrics [28] but are slow to converge. Surrogate models are used in many works [4, 21, 22, 29–31, 36] as indicative measures of the solution quality. However, it has never been shown that optimizing a surrogate is analogous to optimizing the datacenter metrics. The fastest, accurate models use queueing theory. Queueing theory models can closely approximate the datacenter metrics efficiently by modelling the behavior of queues in the datacenter. Despite this, most queueing theory models consider a simplified representation of the datacenter where queues are allowed to be infinite and hence packets are never dropped [2, 19, 27] whereas in practice, queues are finite and packet loss is significant. In a previous paper we proposed a fast and accurate queueing model that corrected this issue [12], however on large problems this model is slow to converge.

One means of remedying this issue is to utilise the capabilities of modern CPUs, in particular support for high degrees of multithreading. Evolutionary



(a) Dcell_1 with 4 port switches [20]

(b) Fat Tree with 4 port switches [3]

(c) Leaf-Spine with 6 port switches [15]

Fig. 1: Common datacenter topologies

algorithms are parallelisable and appropriate for the VNFPP as they can efficiently find approximate solutions to challenging optimisation problems. Parallel evolutionary algorithms can achieve sub or near linear [13, 18, 23, 24, 34, 37] and even super-linear [5, 26] reductions in execution time for increasing numbers of threads. However, increased parallelisation can also come at the cost of poorer solutions [13] or increased execution time [18].

In this work we determine whether PMOEAs can efficiently and effectively solve the VNFPP. First, in Section 2 we formally specify the VNFPP problem. Next, in Section 3 we describe the PMOEAs considered in this work. Section 4 presents the results of our evaluation. Finally, Section 5 summarizes the overall contributions.

2 Problem Formulation

In this section, we describe the VNFPP and its constraints in detail. In the VNFPP, services must be placed on virtual machines in a datacenter so as to maximize multiple QoS objectives whilst minimizing energy consumption. The datacenter is a collection of servers interconnected by a network topology (see Fig. 1). A service is a sequence of VNFs. Packets arrive at the first VNF in the service and must visit each VNF in sequence. A solution to the VNFPP places one or more instances of each VNF in the datacenter and specifies paths between VNFs to form services.

The VNFPP has three objectives: two QoS metrics, latency and packet loss, and a cost metric, energy consumption. As service latency, packet loss and en-

ergy consumption conflict [12] we formulate the VNFPP as a multi-objective optimization problem. Further, as services in a datacenter can number in the thousands it is not possible to treat each service quality metric as a separate objective. Instead, we aim to minimize the average service latency and the average service packet loss over all services, as well as the total energy consumption. Formally:

$$\text{minimize } \mathbf{F}(R_S) = (W, P, E)^T \tag{1}$$

where R_S is the set of routes for each service, W is the average service latency, P is the average service packet loss and E is the total electricity consumption of the datacenter. Several models exist to determine W, P and E for the VNFPP. This work uses the queueing theory model we proposed in [10] and extended in [11].

Finally, for a solution to the VNFPP to be feasible, it must satisfy two constraints. First, every VNF consumes some resources on a server. The total resources consumed by VNFs on a server cannot exceed the total resources of the server. Second, the solution must define one or more valid paths for each service. To be valid, a path must visit each VNF of the service in the order it appears in the service.

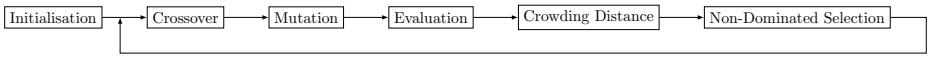
3 Parallel MOEAs for the VNFPP

In this section we outline the algorithms evaluated in this work and the genetic operators used to solve the VNFPP. First we describe the PMOEAs we evaluated and then we describe the genetic operators we have constructed to solve the VNFPP.

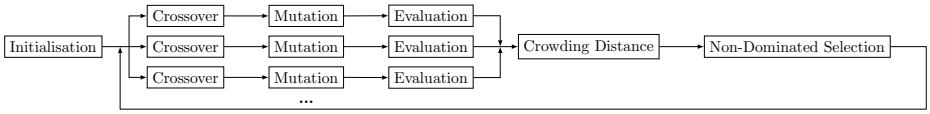
3.1 Parallel MOEA Frameworks

PMOEAs can be grouped into three categories: master/slave algorithms, multiple-deme and isolated-deme algorithms [14]. Each parallel algorithm distributes tasks over N_T threads. In this work we evaluated a PMOEA from each category against a serial MOEA that does not use multithreading. The remainder of this section describes these algorithms in detail.

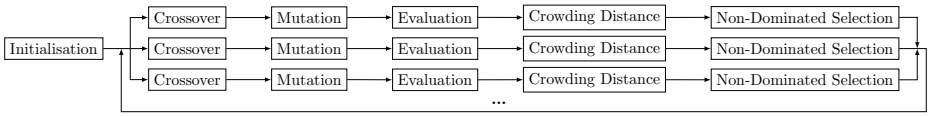
A serial MOEA is an evolutionary algorithm that only uses a single process. For this category, we selected the well known NSGA-II [16] algorithm, illustrated in Fig. 2a. NSGA-II maintains a population of N parent solutions. A population of N child solutions are generated via mutation and crossover of the parent population. Both populations are combined and then fast nondominated sorting and crowding distance procedures are used to determine which solutions survive to become the parent solutions of the next generation. This process repeats until the stopping condition is met. NSGA-II has been shown to be effective on a range of multiobjective optimization problems previously, including the VNFPP [12]. Additionally, several parallel variants of NSGA-II exist, allowing for a fair comparison between single and multithreaded algorithms.



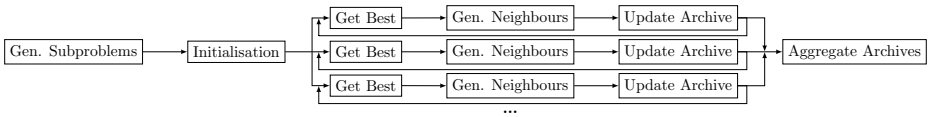
(a) NSGA-II [16]



(b) MS-NSGA-II



(c) MD-NSGA-II [33]



(d) PPLS/D [34]

Fig. 2: The four MOEAs we considered in this work. NSGA-II is a serial algorithm whilst the remaining parallelize some or all steps.

A master/slave algorithm has a single 'master' thread that allocates tasks to 'slave' threads. Once the slave threads are finished, the master thread aggregates the results. In this work we parallelise NSGA-II using this architecture, hereafter referred to as MS-NSGA-II (illustrated in Fig. 2b). In this implementation each thread performs crossover, mutation and evaluation to produce N/N_T solutions. Afterwards the solutions from each thread are aggregated to form the child population. As in NSGA-II, fast nondominated sorting and crowding distance procedures are used to select the parent population for the next generation. This process repeats until the stopping condition is met.

A multiple-deme algorithm evolves multiple subpopulations in parallel and periodically exchanges solutions between them. In this work we use the algorithm proposed by Roberge et al. [33], hereafter referred to as MD-NSGA-II, which uses a multiple-deme architecture (illustrated in Fig. 2c). Roberge et al. first produce a set of initial solutions and group them into N_T equal sized subpopulations. The variation, evaluation and selection operators from NSGA-II are performed on each subpopulation in parallel. After a set number of generations, the subpopulations are aggregated into a new population and then randomly redistributed back into N_T subpopulations. This process repeats until the stopping condition is met.

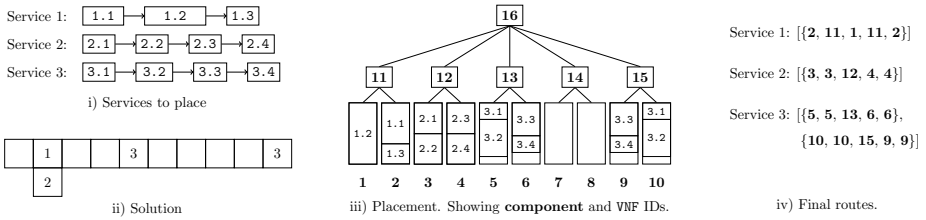


Fig. 3: A partial mapping from genotype to phenotype showing the assignment of services to servers.

An isolated-deme algorithm also evolves multiple subpopulations but does not exchange information between the subpopulations. In this work we use the isolated-deme algorithm PPLS/D [34] (illustrated in Fig. 2d). PPLS/D first decomposes the objective space into N_T scalar objective optimization subproblems. The algorithm maintains a population of unexplored solutions and an archive of non-dominated solutions for each subproblem. First, a solution is added to the unexplored population of each subproblem. Next, for each subproblem the algorithm selects the best solution from the unexplored population and generates a set of neighboring solutions. A neighboring solution is added to the unexplored population and the non-dominated archive if it is: 1) closer to the subproblem that generated it than any other subproblem, as measured by the angle between the solution and the subproblems and 2) is either a better solution to the subproblem than the current best solution or if it is not dominated by any solutions in the non-dominated set of the subproblem. Once the stopping condition is met, the algorithm aggregates the non-dominated archives from each subproblem and returns the overall non-dominated solutions.

PPLS/D was designed for unconstrained optimization problems, so required modification to fit the VNFPP. In our variant, an infeasible solution is accepted into the unexplored population only if there are no more feasible solutions in the population.

Additionally, to determine how PPLS/D is affected by the number of subproblems used, we generate N subproblems that are placed into N_T equally sized groups. Each subproblem in a group is executed in series and each group is executed in parallel. This technique produces solutions with the same solution quality as if greater numbers of threads were available whilst not unfairly benefitting PPLS/D.

3.2 Genetic Operators

As this work focusses on the benefits of parallelisation, we use existing genetic operators that were designed for the VNFPP. In particular we use operators we demonstrated to be effective in earlier work [12]. These operators use a genotype-phenotype representation where genetic operators consider a simplified solution representation (the genotype) which is converted into the 'true' solution representation (the phenotype) to be evaluated. In our implementation, the genotype

is an array of vectors where each vector represents a server in the datacenter (see Fig. 3 ii). Each vector contains zero or more service instances. The phenotype is the set of paths for each service (see Fig. 3 iv). The genotype-phenotype mapping has two parts: expansion and routing as shown in Fig. 3. The expansion step iterates over each server in the genotype. When a service instance is encountered, the algorithm places each VNF of the service on the closest server that can accommodate it. The routing step finds the set of shortest paths between each VNF and combines them to form a path.

MD-NSGA-II and MS-NSGA-II require initialization, mutation, crossover and evaluation operators. PPLS/D additionally requires a local search operator. Since all services are considered equally important, the initialization operator always generates genotypes where each service has the same number of service instances. The operator first determines the maximum number of service instances that can be placed in a datacenter whilst meeting this constraint. Then it generates a range of solutions between the maximum and minimum number of service instances that can be placed. The mutation operator has an equal probability of 1) swapping the contents of two servers, 2) adding a random service instance to a random server and 3) removing a random service instance from a server. The same operator is used to generate neighbors in PPLS/D. Crossover is performed using the N-Point crossover.

4 Experimental Results

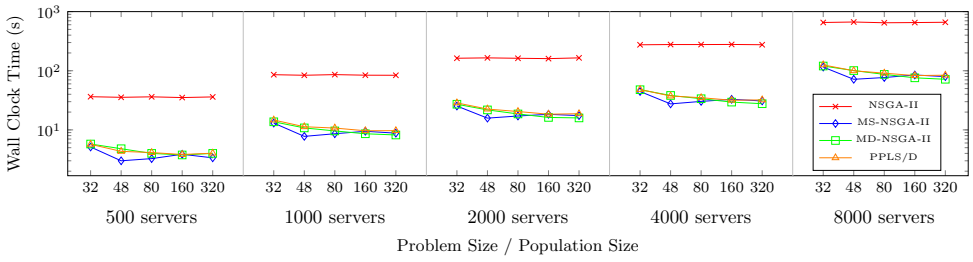
In this section, we first describe our test methodology and then present the results. Our experiments were configured as follows. We generated 30 VNFPP instances for each problem size and topology by varying the number and length of services and the size and speed of the VNFs. We assigned each algorithm a budget of 16000 evaluations to be used across all threads. All tests were run on a 8 core/16 thread CPU at 2.6GHz. All parallel algorithms were implemented using the parallelisation library Rayon.[1]

We used the parameter settings for NSGA-II proposed in [16] for NSGA-II, MD-NSGA-II and MS-NSGA-II. As PPLS/D is known to be sensitive to the population size [34], we also consider a range of population sizes for each test.

For analysis, we recorded the time each algorithm took to complete and the hypervolume of the final population. Since the three objectives typically have orders of magnitude different values it is necessary to first normalize the objectives. The utopian and nadir points were estimated by taking the best and worst objective values respectively from all tests. We used the reference point $1 + \epsilon$ for each objective where ϵ is a small number.

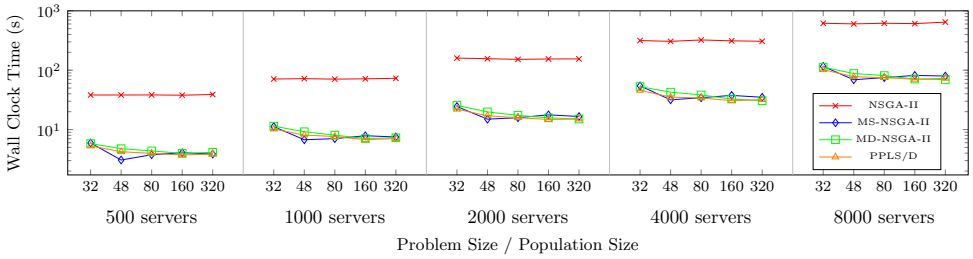
Fig. 4 and Fig. 5 demonstrate our two key findings. First, Fig. 4 shows that PMOEAs are typically 5-10x faster than the sequential NSGA-II. As the problem size increases, the time required to evaluate a solution also increases. However, this step can be parallelized reducing the overall execution time.

[1] https://github.com/rayon-rs/rayon



(a) Fat Tree

(b) DCell



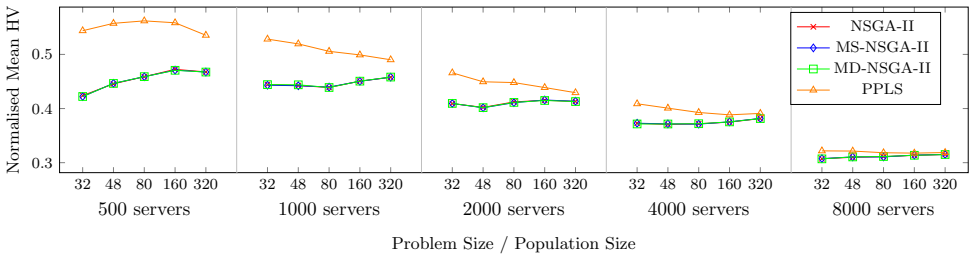
(c) Leaf-Spine

Fig. 4: Mean Execution Time vs Problem Size and Population Size

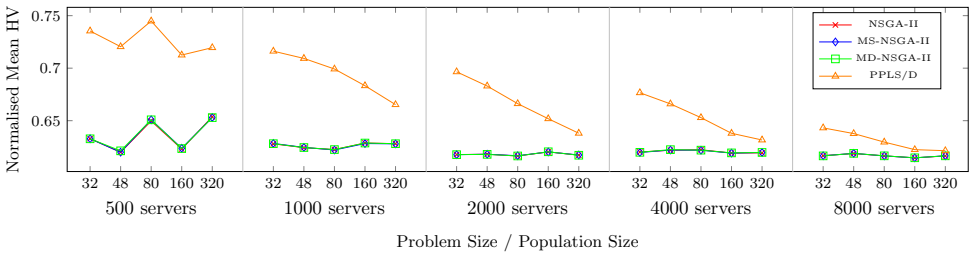
Second, Fig. 5 shows that the speed improvements from parallelisation did not come at the cost of solution quality. This demonstrates that PMOEAs solve VNFPPs faster and at no cost to solution quality.

Further, when the best parameter settings are considered for each algorithm, PPLS/D found *better* solutions on average than other algorithms on the majority of problems. These results must be considered with three caveats:

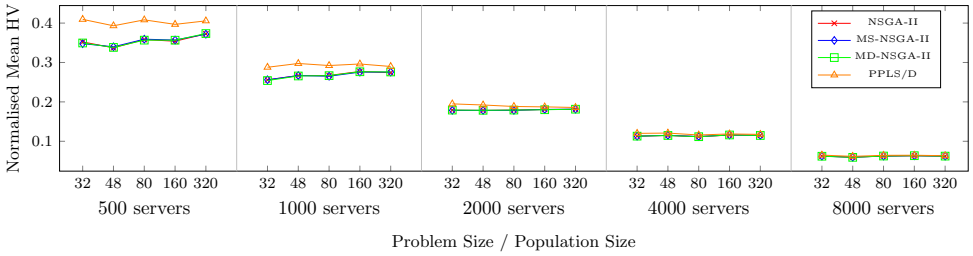
1. PPLS/D was less effective on larger problem instances. Whilst all algorithms experienced some decline when solving larger problem instances, this behavior is particularly apparent with PPLS/D. PPLS/D relies on local search operators to explore the solution space. On large problem instances, the local search area is very large. It is likely that on large problems PPLS/D only explores the neighborhood of the initial population. Alternative local search



(a) Fat Tree



(b) DCell



(c) Leaf-Spine

Fig. 5: Mean HV vs Problem Size and Population Size

operators may be able to explore the search space more effectively, however this remains for future research.

2. PPLS/D is sensitive to the population size whereas the other algorithms were not. In the majority of instances, PPLS/D produced better sets of solutions when a small population was used. Shi et al [34] proposed this was due to large population sizes restricting each subproblem to a small region of the solution space. In PPLS/D, a subproblem can only consider a solution if it is the closest subproblem to the solution (see Section 3). With a large population, the solutions generated by the local search are more likely to be near to a different subproblem, reducing the number of solutions considered, and slowing the search process. This appears particularly important on larger problem instances wherein the setting of the population size was the most important factor in whether PPLS/D outperformed the other algorithms.

3. PPLS/D maintains an archive population and hence stores more solutions on the Pareto front than the other algorithms used. PPLS/D typically produced 1-2 orders of magnitude more non-dominated solutions than the other algorithms. This in part explains the improved hypervolume for PPLS/D as more solutions can better approximate the Pareto front.

Despite these caveats, it is still notable that PPLS/D outperformed other algorithms using only local search operators. Some of the drawbacks of PPLS/D may be able to be resolved with alternative local search operators in future work.

5 Conclusion

In this work we determined whether PMOEAs could be used to solve the VNFPP more efficiently than sequential MOEAs without harming solution quality. We found that not only is this true, but that in certain instances PMOEAs produce better representations of the Pareto front than a comparable sequential algorithm. These results indicate that PMOEAs have potential to solve practical VNFPPs and are worthy of further research. Future work will explore alternative neighbor generation strategies to provide improved local search capabilities. Additional research could extend PMOEAs from single CPU, multithreaded systems to multi-cpu distributed systems in pursuit of larger speedups.

References

1. Abrita, S.I., Sarker, M., Abrar, F., Adnan, M.A.: Benchmarking VM startup time in the cloud. In: Bench'18: Bench Council International Symposium (2018)
2. Agarwal, S., Malandrino, F., Chiasserini, C., De, S.: Joint VNF placement and CPU allocation in 5g. In: INFOCOM'18: IEEE Conference on Computer Communications. pp. 1943–1951. IEEE (2018)
3. Al-Fares, M., Loukissas, A., Vahdat, A.: A scalable, commodity data center network architecture. In: SIGCOMM:'08 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications. pp. 63–74 (2008)
4. Alameddine, H.A., Qu, L., Assi, C.: Scheduling service function chains for ultra-low latency network services. In: CNSM'17: Conference on Network and Service Management. pp. 1–9. IEEE Computer Society (2017)
5. Alba, E.: Parallel evolutionary algorithms can achieve super-linear performance. Inf. Process. Lett. **82**(1), 7–13 (2002)
6. Andrae, A., Edler, T.: On global electricity usage of communication technology: trends to 2030. Challenges **6**(1), 117–157 (2015)
7. Avgerinou, M., Bertoldi, P., Castellazzi, L.: Trends in data centre energy consumption under the European code of conduct for data centre energy efficiency. Energies **10**(1470), 1–18 (2017)
8. Bari, M.F., Chowdhury, S.R., Ahmed, R., Boutaba, R.: On orchestrating virtual network functions. In: CNSM'11: Proc. of the 11th International Conference on Network and Service Management. pp. 50–56 (2015)

9. Baumgartner, A., Reddy, V.S., Bauschert, T.: Combined virtual mobile core network function placement and topology optimization with latency bounds. In: EWSDN'15: Proc. of the 4th European Workshop on Software Defined Networks. pp. 97–102 (2015)
10. Billingsley, J., Li, K., Miao, W., Min, G., Georgalas, N.: A formal model for multi-objective optimisation of network function virtualisation placement. In: EMO':19 Evolutionary Multi-Criterion Optimization - 10th International Conference. pp. 529–540 (2019)
11. Billingsley, J., Li, K., Miao, W., Min, G., Georgalas, N.: Multi-objective virtual network function placement: Formal model and effective algorithm (2020), unpublished
12. Billingsley, J., Li, K., Miao, W., Min, G., Georgalas, N.: Routing-led placement of vnfs in arbitrary networks. In: WCCI'20: World Congress on Computational Intelligence (2020)
13. Branke, J., Schmeck, H., Deb, K., Maheshwar, R.S.: Parallelizing multi-objective evolutionary algorithms: cone separation. In: Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2004, 19-23 June 2004, Portland, OR, USA. pp. 1952–1957. IEEE (2004)
14. Cantú-Paz, E., Goldberg, D.E.: On the scalability of parallel genetic algorithms. Evol. Comput. **7**(4), 429–449 (1999)
15. CISCO: Spine and leaf architecture: Design overview white paper. https://www.cisco.com/c/en/us/products/collateral/switches/nexus-7000-series-switches/white-paper-c11-737022.html, accessed: 18-09-2020
16. Deb, K., Agrawal, S., Pratap, A., Meyarivan, T.: A fast elitist non-dominated sorting genetic algorithm for multi-objective optimisation: NSGA-II. In: Schoenauer, M., Deb, K., Rudolph, G., Yao, X., Lutton, E., Guervós, J.J.M., Schwefel, H. (eds.) PPSN:'00 Parallel Problem Solving from Nature
17. Dodd, N., Alfieri, F., Caldas, M.N.D.O.G., Viegand, L.M.D.J., Flucker, S., Tozer, R., Whitehead, B., Brocklehurst, A.W.F.: Development of the eu green public procurement (gpp) criteria for data centres. Tech. rep., Server Rooms and Cloud Services, Publications Office of the European Union (2020)
18. El-Alfy, E.M., Alshammari, M.A.: Towards scalable rough set based attribute subset selection for intrusion detection using parallel genetic algorithm in mapreduce. Simul. Model. Pract. Theory **64**, 18–29 (2016)
19. Gouareb, R., Friderikos, V., Aghvami, A.H.: Delay sensitive virtual network function placement and routing. In: ICT'18: 25th International Conference on Telecommunications. pp. 394–398. IEEE (2018)
20. Guo, C., Wu, H., Tan, K., Shi, L., Zhang, Y., Lu, S.: Dcell: a scalable and fault-tolerant network structure for data centers. In: SIGCOMM'08: Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications. pp. 75–86 (2008)
21. Guo, H., Wang, Y., Li, Z., Qiu, X., An, H., Yu, P., Yuan, N.: Cost-aware placement and chaining of service function chain with VNF instance sharing. In: NOMS'20: IEEE/IFIP Network Operations and Management Symposium. pp. 1–8. IEEE (2020)
22. Kuo, T., Liou, B., Lin, K.C., Tsai, M.: Deploying chains of virtual network functions: On the relation between link and server usage. IEEE/ACM Trans. Netw. **26**(4), 1562–1576 (2018)
23. von Lücken, C., Barán, B., Sotelo, A.: Pump scheduling optimization using asynchronous parallel evolutionary algorithms. CLEI Electron. J. **7**(2) (2004)

24. Luo, J., Fujimura, S., Baz, D.E., Plazolles, B.: GPU based parallel genetic algorithm for solving an energy efficient dynamic flexible flow shop scheduling problem. J. Parallel Distributed Comput. **133**, 244–257 (2019)
25. Miotto, G., Luizelli, M.C., da Costa Cordeiro, W.L., Gaspary, L.P.: Adaptive placement & chaining of virtual network functions with NFV-PEAR. J. Internet Services and Applications **10**(1), 3:1–3:19 (2019)
26. Mühlenbein, H., Schomisch, M., Born, J.: The parallel genetic algorithm as function optimizer. Parallel Comput. **17**(6-7), 619–632 (1991)
27. Oljira, D.B., Grinnemo, K., Taheri, J., Brunström, A.: A model for qos-aware VNF placement and provisioning. In: NFV-SDN'17: IEEE Conference on Network Function Virtualization and Software Defined Networks. pp. 1–7. IEEE (2017)
28. Pongor, G.: Omnet: Objective modular network testbed. In: MASCOTS'93: Proceedings of the International Workshop on Modeling, Analysis, and Simulation On Computer and Telecommunication Systems. pp. 323–326 (1993)
29. Qi, D., Shen, S., Wang, G.: Towards an efficient VNF placement in network function virtualization. Comput. Commun. **138**, 81–89 (2019)
30. Qu, L., Assi, C., Shaban, K.B., Khabbaz, M.J.: A reliability-aware network service chain provisioning with delay guarantees in nfv-enabled enterprise datacenter networks. IEEE Trans. Network and Service Management **14**(3), 554–568 (2017)
31. Rankothge, W., Le, F., Russo, A., Lobo, J.: Optimizing resource allocation for virtualized network functions in a cloud center using genetic algorithms. IEEE Trans. Network and Service Management **14**(2), 343–356 (2017)
32. Rankothge, W., Ma, J., Le, F., Russo, A., Lobo, J.: Towards making network function virtualization a cloud computing service. In: IM'15: Proc. of the 2015 IFIP/IEEE International Symposium on Integrated Network Management. pp. 89–97 (2015)
33. Roberge, V., Tarbouchi, M., Labonté, G.: Comparison of parallel genetic algorithm and particle swarm optimization for real-time UAV path planning. IEEE Trans. Ind. Informatics **9**(1), 132–141 (2013)
34. Shi, J., Zhang, Q., Sun, J.: PPLS/D: parallel pareto local search based on decomposition. IEEE Trans. Cybern. **50**(3), 1060–1071 (2020)
35. Soualah, O., Mechtri, M., Ghribi, C., Zeghlache, D.: Energy efficient algorithm for VNF placement and chaining. In: CCGRID'17:International Symposium on Cluster, Cloud and Grid Computing. pp. 579–588. IEEE Computer Society / ACM (2017)
36. Vizarreta, P., Condoluci, M., Machuca, C.M., Mahmoodi, T., Kellerer, W.: Qos-driven function placement reducing expenditures in NFV deployments. In: ICC'17: IEEE International Conference on Communications. pp. 1–7. IEEE (2017)
37. Zhang, Y., Yu, W., Chen, X., Jiang, J.: Parallel genetic algorithm to extend the lifespan of internet of things in 5g networks. IEEE Access **8**, 149630–149642 (2020)

Acknowledgements K. Li was supported by UKRI Future Leaders Fellowship (Grant No. MR/S017062/1) and Royal Society (Grant No. IEC/NSFC/170243). J. Billingsley was supported by EPSRC Industrial CASE and British Telecom (Grant No. 16000177). The authors would also like to acknowledge the use of the University of Exeter High-Performance Computing facility in carrying out this work and thank the developers and maintainers of Rust and the Rayon library.