K_{rr}	K_{sr}	K_{dr}	K_{bgp}	K_{obj}	$f(x)$
1	2	1	2	1	$\|x\|+2$

Table 2: Realizing framework parameters

	DATA	RSRCH	GRID	INT
DATA	-	*H-1*	×	✓
RSRCH	✓	-	✓	✓
GRID	×	*H-1-1*	-	×
INT	*D-1*	*H-1-1*	×	-

Table 3: Each cell (*row*, *column*) shows whether the policy group *column* can be reached by the policy group *row*. ✓/× means full/no reachability. *D-1*, *H-1* and *H-1-1* each denotes a subset of the subnets in *DATA* and *RSRCH*, which can be reached by the corresponding *row*. *H-1-1* is in turn a subset of *H-1*.

7.1.1 Inferring model parameters and framework inputs

We only need to calculating the model parameters for the Cisco IOS platform, as this platform is exclusively used by the campus network. Obtaining these parameters is straightforward, as we just need to run the heuristics proposed in [5] on corresponding configuration blocks that relate to each parameter, and count the number of referential links introduced. The results are shown in Table 2.

To infer the inputs as described in Sec. 3.2, we used a methodology that combines reverse-engineering the configuration files and discussions with operators. We were able to identify the inputs as follows. Table 3 shows the policy groups and the reachability policies among them. Fig. 11a shows the topology and what policy groups each routing instance contains. In particular the campus network has two routing instances denoted as EIGRP and OSPF. There are two policy groups in the network denoted as *DATA* and *RSRCH*. In addition, two external AS-es (denoted as GRID and INT) peer with this campus network. Each external AS can be viewed both as a single policy group and as a single routing instance. Finally, the $\mathbf{M_X}$ matrix, i.e., the set of routes exchanged between every pair of routing instances, is shown in Table 4.

7.1.2 Estimating intra-instance complexity

First, according to our framework, only the EIGRP instance will incur intra-instance route filters as it is the only instance that contains multiple policy groups.

Second, the EIGRP instance employs the typical star topology (Sec. 4.2.1), as shown in Fig. 11b. The border router R_1 also serves as the core router and connects the two policy groups: *DATA* and *RSRCH*. R_1 also directly connects to the other borders R_2, R_3 and R_4. Note that there is no direct link between the two policy groups, or between either policy group and R_2/R_3/R_4.

From the reachability matrix (Table 3), it is easy to see that intra-instance filtering is needed between the core router R_1 and the *DATA* policy group as only a subset of routes from R_1 can be sent to *DATA*. More specifically, the routes learned from *GRID* cannot be exposed to *DATA*. Using the model presented in Sec. 4, the route filter placement is determined and shown in Fig. 11b. Route filtering is not needed between R_1 and *RSRCH*, as *RSRCH* has full reachability to all other policy groups. The predicted complexity is shown by the diagonal cells in Table 5.

Comparing with the actual configuration: We measured the actual configuration complexity in the configuration files. The result is shown in the diagonal cells in Table 6. As predicted, only the EIGRP routing instance incurs intra-instance route filters, and the filter placement is exactly as predicted. Furthermore, the measured complexity numbers also match the estimated value well.

	EIGRP	OSPF	GRID	INT
EIGRP	-	all	*H-1-1*	*D-1, H-1-1*
OSPF	all	-	-	-
GRID	all	-	-	-
INT	all	-	-	-

Table 4: Each cell (*row*, *column*) shows the set of routes that routing instance *row* should advertise to routing instance *column*. "All" means that *row* should advertise all its routes (both internal and external ones) to *column*.

	EIGRP	OSPF	GRID	INT
EIGRP	7	1	6	30
OSPF	1	0	-	-
GRID	1	0	-	-
INT	2	-	-	-

Table 5: Estimated complexity for the original design. Each non-diagonal cell (*row*, *column*) shows the inter-instance complexity of advertising routes from *row* to *column*. The cells on the diagonal show the intra-instance complexity. "-" indicates that the two instances are not directly connected.

7.1.3 Estimating inter-instance complexity

Using the models presented in Sec. 5, we estimate the inter-instance complexity, and the result is shown in Table 5.

Comparing with the actual configuration: We compare the predicted inter-instance complexity with the complexity measured in the configuration files. The differences are shown in Table 6. We see that the majority of the predicted numbers match the actual configuration well. There is a mismatch in the case of filtering routes between GRID and EIGRP. The measured value is greater than the prediction, which makes sense as the prediction is the minimum necessary complexity. The actual configuration may incur higher complexity, for example, due to redundant configurations or suboptimal configurations.

In particular, the outgoing routes from EIGRP to GRID are subject to filtering as only a subset of EIGRP routes can be sent to GRID. We note that the filtering may be configured either at the redistribution point (i.e. permitting only the subset of routes to enter BGP), or within the BGP session (i.e. permitting only the subset of routes to be advertised to GRID). However, in the actual configuration, the exact same filtering is implemented at *both* places. This is redundant configuration, and results in unnecessary increase in the complexity. Further, GRID can advertise all its routes to EIGRP, so there is no route filter needed in that direction. However, in the actual configuration, an unnecessary filter is configured, which simply allows all routes to pass. As a result, several referential links were created.

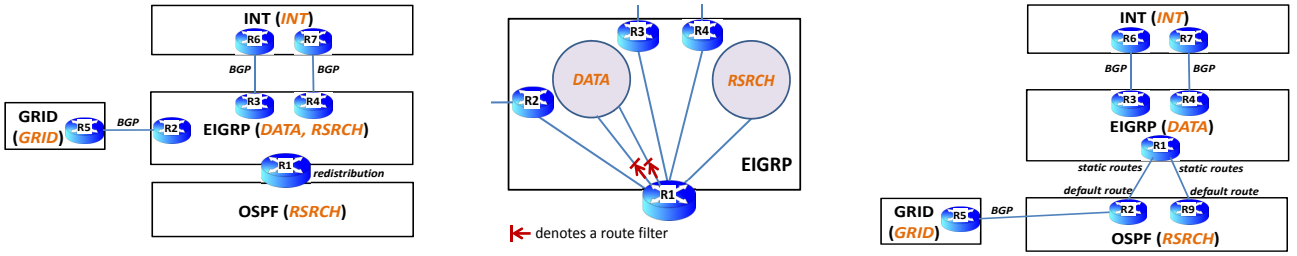
Overall, these results confirm that our framework can accurately estimate the complexity of a given routing design.

7.2 Case study of a routing design change

The campus network experienced a major design change recently. The change was primarily motivated by the need to increase the resiliency of the original design. Thus as the second part of the evaluation, we apply our framework to compare the new routing design with the original one. We first use our framework to analyze the change in complexity due to the redesign. We then consider whether alternative designs could have met the same resiliency objectives but with lower complexity.

7.2.1 Impact of redesign on complexity

Fig. 11c illustrates the new instance-level graph after the network redesign was completed. The primary purpose of the redesign was



(a) Instance-level topology of the original design.

(b) Detailed topology of EIGRP in the original design.

(c) Instance-level topology of the new design.

Figure 11: The original and new routing designs.

	EIGRP	OSPF	GRID	INT
EIGRP	$\epsilon = 0$	$\epsilon = 0$	$\epsilon = -6$	$\epsilon = 0$
OSPF	$\epsilon = 0$	$\epsilon = 0$	-	-
GRID	$\epsilon = -3$	$\epsilon = 0$	-	-
INT	$\epsilon = 0$	-	-	-

Table 6: Difference between complexity estimated using our models and the actual complexity measured from the configuration files for the original design.

	EIGRP	OSPF	GRID	INT
EIGRP	$\delta = -7$	$\delta = 7$	$\delta = -6$	$\delta = 0$
OSPF	$\delta = 29$	$\delta = 0$	$\delta = 6$	-
GRID	$\delta = -1$	$\delta = 1$	-	-
INT	$\delta = 0$	-	-	-

Table 7: Increase in the intra- and inter-instance complexity after the redesign.

to increase resiliency. In particular, the number of border routers connecting the OSPF instance to EIGRP was increased to two. In addition, two other changes were made: (i) the connecting primitive between EIGRP and OSPF was changed from route redistribution to static routes (configured on the EIGRP side) and default routes (configured on the OSPF side); and (ii) the subnets of the policy group *RSRCH* that were in the EIGRP instance were moved to OSPF. As a result, in the new design, EIGRP only contains subnets of the policy group *DATA*, while OSPF contains all subnets of the policy group *RSRCH*. Finally, we note that the policy groups and the reachability matrix were unchanged after the redesign.

Table 7 presents the change in complexity estimated by our framework. Overall, the total complexity in the new design increased. This is in part due to the fact that the resilience of the new design also increased, i.e., it used two border routers for the OSPF routing instance, compared to one in the old design. We note that the new design eliminated the intra-instance complexity in the EIGRP routing instance, as now EIGRP only contained a single policy group. On the other hand, the inter-instance complexity between EIGRP and OSPF increased in the new design, caused by the need to implement the different reachability requirements for the two policy groups *RSRCH* and *DATA*.

7.2.2 Could alternative designs lower complexity?

In the previous section, we noted that while the primary goal of the redesign was to improve resiliency, operators made two additional changes that were not strictly necessary to achieve this goal: (i) changing the connecting primitive between OSPF and EIGRP from route redistribution to static/default routes; and (ii) moving all *RSRCH* subnets to OSPF. We hypothesized these changes may have been made to lower complexity. To isolate the impact of each

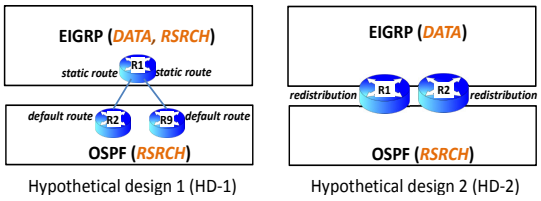


Figure 12: The two hypothetical designs.

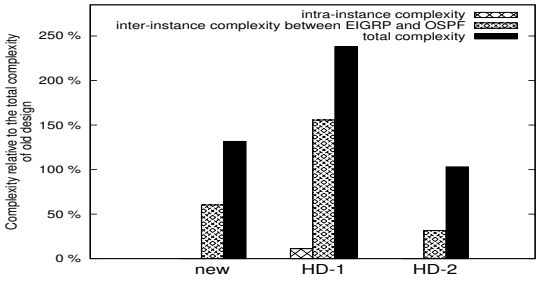


Figure 13: Comparison of complexity of different designs.

of these changes, we considered two hypothetical designs termed *HD-1* and *HD-2*, as shown in Fig. 12. Both designs use two border routers for OSPF, to achieve the same resiliency requirement as the new design. *HD-1* uses static and default routes to connect EIGRP and OSPF, and represents a design where only the first of the two additional changes above were made. *HD-2* involves a rearrangement of policy groups and represents a design where only the second of the two additional changes above were made. Route redistribution is used to connect the instances.

We apply our framework to estimate the complexity for both hypothetical designs. The results are shown in Fig 13. For ease of comparison, we normalized all bars to the total complexity of the original campus design. We see that while *HD-1* is a worse alternative design as its total complexity (third bar) increases compared to the actual new design, *HD-2* is a better alternative as its total complexity decreases compared to the actual new design.

We next seek to better understand why *HD-1* has higher complexity than the actual new design. The main difference between the two designs is whether the policy group *RSRCH* is placed entirely in the OSPF routing instance (actual new design), or split across both OSPF and EIGRP (*HD-1*). We observe that by placing *RSRCH* entirely in OSPF, the address space of OSPF is more unified, which allows better aggregation of its routes. This results in a reduction of the size of $\mathbf{M_X}(OSPF, EIGRP)$ from 9 to 3, which translates to fewer static routes needed, and thus results in less

	static route	redistribution	BGP
default route	38	20	25
redistribution	38	20	25
BGP	43	25	26

Table 8: Complexity associated with different choices of connecting primitive between EIGRP and OSPF. Each cell (*row*, *column*) shows the complexity of the design that uses the *row* (*column*) connecting primitive on the OSPF (EIGRP) side.

inter-instance complexity (second bar in Fig. 13). In addition, *HD-1* incurs significant intra-EIGRP complexity (first bar), while the actual new design eliminates that complexity.

Next, we compare the actual new design and *HD-2*. The main difference between the two is the connecting primitive used to connect OSPF and EIGRP. We found that using route redistribution (*HD-2*) lowers the complexity compared to using static routes (actual new design). This indicates that by changing the connecting primitive from redistribution to static/default routes during the redesign process, the operators introduced unnecessary design complexity.

Given these insights, we next want to find out whether mutual route redistribution is the best connecting primitive to use to connect EIGRP and OSPF, and if alternative primitives could further lower complexity. For this purpose, we enumerate all possible connecting primitives, and apply our framework to estimate the complexity associated with each alternative design choice. The results are shown in Table 8. Note that it is not feasible to use static routes (default routes) on the OSPF (EIGRP) side, so the corresponding column and row are omitted. The table shows that mutual route redistribution indeed achieves the minimum complexity. A similar complexity could have also been obtained through a design that uses a combination of default routes and route redistribution. We also see that different choices of connecting primitive may lead to significant difference in resulting complexity.

In summary, these results show that (i) the design change of moving subnets of the policy group *RSRCH* from EIGRP to OSPF greatly reduced both intra- and inter-instance complexity; and (ii) the change of connecting primitive actually made the network more complex and thus should have been avoided; and (iii) different design choices may result in significantly different complexity. Overall, this case study highlights the power of our framework in systematically comparing multiple design alternatives and in guiding operators towards approaches that lower complexity while meeting the same design objectives.

7.3 Operator interview

We discussed the above results with the operators of the campus network, and they were able to confirm many of our observations. In particular, they confirmed that moving the *RSRCH* subnets from EIGRP to OSPF significantly reduced the management complexity. In fact, the motivation of that change was to make the *RSRCH* network more unified and simplify the network design. In addition, the operators also acknowledged that our hypothetical design 2 (*HD-2* in Fig. 12) that uses route redistribution instead of static routes could indeed be a less complex design. The primary reason they decided to use static routes in the new design was because this particular operator team consisted of people with varying expertise and skill levels (including senior operators, part-time student workers, and new hires), all of whom could potentially alter configuration files. While configuring static routes did not require extensive prior knowledge, configuring route redistribution required greater knowledge and expertise, particularly given the potential for routing loops. The operators indicated however that they would prefer route redistribution if only a small number of senior operators managed the network. Overall, these results confirm that our framework provides useful guidance to operators. An open question for future work is whether current complexity metrics must be refined to take operator skill levels into account.

8 Discussion and Open Issues

Incorporating other design objectives and constraints: In putting together a routing design, operators must reconcile a variety of objectives and constraints such as performance, complexity, hardware constraints etc. This paper focuses on the design complexity, given that it is very important, is difficult to quantify, and has received limited attention from the community. In future, it would be interesting to also factor in other important requirements. For example, hardware constraint may restrict the number of route filters that a router can support. Such restriction may in turn impact both intra- and inter-instance route filter placements. We believe our framework can be easily enhanced to systematically determine the best filter placements, so that the hardware constraint is honored, while the total design complexity is minimized. In addition, it may be interesting to consider other design objectives such as performance (e.g., measured as average hop counts between any two subnets), and costs (restricting the number and hardware capacity of devices that can be used). While some of these objectives and constraints may not be critical in a typical over-provisioned enterprise environment, they are nevertheless worthwhile to consider.

Joint optimization of multiple design tasks: This work builds upon a "divide and conquer" network design strategy that is commonly practiced by the operational community [24]. In particular, such a design process consists of four distinct stages: (i) wiring and physical topology design; (ii) VLAN design and IP address allocation; (iii) routing design; and (iv) deployment of services such as VoIP and IPsec. We further break down the task of routing design into two sequential steps: (1) creating routing instances and determining the set of routes to be exchanged between each pair of these instances, and then (2) configuring policy groups and the necessary glue logic. Step (1) is relatively straightforward, typically influenced by factors such as the proximity of routers (e.g., in the same building, city, etc.), administrative boundaries (e.g., different network segments are managed by different operators), and equipment considerations (e.g., EIGRP is available only on Cisco routers). Therefore, this work focuses on the second step while assuming that the first step has been accomplished. In future, it should be beneficial to consider multiple design stages and steps in one framework and explore ways to improve routing design further through joint optimization of all pertinent design choices.

Complexity-aware top-down design: The complexity models presented in this paper pave the way for complexity-aware top-down routing design. Such top-down design takes as input the high-level design objectives and constraints, and seeks to minimize design complexity while meeting other design requirements. In doing so, our complexity models can be used to guide the search of the design space to systematically determine (i) how policy groups should be grouped into routing instances; (ii) optimum placement of route filters; and (iii) what primitives should be used to connect each pair of routing instances. We defer the development of such a top-down design framework to future work.

Emerging architectures and configuration languages: In recent years, researchers have started investigating new network architectures based on logically centralized controllers (e.g., software defined networking [2]), and declarative configuration languages (e.g., Frenetic [11]). These approaches have the potential to simplify network management by shifting complexity away from the

configuration of individual devices to programming of the centralized controllers. While these approaches have much potential, hard problems remain such as the need to update network devices in a consistent fashion [22], and building appropriate coordination mechanisms across multiple controllers. Further exploration of the opportunities and challenges of utilizing these new architectures to simplify network design complexity is an important area of future work.

9 Related Work

In recent years, there has been much interest in both industry [1], and academia [5] in developing formal metrics to capture network configuration complexity. We have discussed in detail how our work differs from [5] in Sec. 1. Similarly, our work also differs from other research [7, 15] that measures the configuration complexity in longitudinal configuration data-sets in a bottom-up fashion. There is a considerable amount of prior work on modeling individual routing protocols, particularly BGP [3, 8, 12, 14], and also OSPF [21], to ensure correct, safe, and efficient behaviors from these protocols. There is also recent progress on safe migration of IGP protocols [25] and on modeling the interaction between multiple routing algorithms deployed in the same network [4]. In contrast, our work analyzes how specific routing protocols and primitives should be combined to meet a given set of design objectives, and the focus is on minimizing the complexity of the resulting design. Our notion of policy groups is similar to policy units introduced in [6], but has some differences in that (i) we require subnets within the same policy group to be full reachable to each other; and (ii) we restrict our definition to reachability restrictions on the routing plane since our focus is on routing design, (i.e., we do not consider data-plane mechanisms like packet filters, firewalls, etc.). Algorithms to extract policy units from low-level configuation files were introduced in [6]. In contrast, our focus is on estimating the number of route filters and filter rules, and consequently the resulting configuration complexity, when multiple policy groups are present in a routing instance.

10 Conclusion and Future Work

In this paper, we present a top-down approach to characterizing the complexity of enterprise routing design given only key high-level design parameters, and in the absence of actual configuration files. Our overall modeling approach is to (i) formally abstract the routing specific operational objectives which can help reason about whether and how a combination of design primitives will meet the objectives; and (ii) decompose routing design into its constituent primitives, and quantify the configuration complexity of individual design primitives using bottom-up complexity metrics [5]. We have validated and demonstrated the utility of our approach using longitudinal configuration data of a large-scale campus network. Estimates produced by our model accurately match empirically measured configuration complexity metrics. Discrepancies when present were mainly due to redundant configuration lines introduced by network operators. Our models enable what-if analysis to help evaluate if alternate routing design choices could lower complexity while achieving the same objectives. Analysis of a major routing design change made by the operators indicates that while some of their design changes were useful in lowering complexity, others in fact were counter-productive and increased complexity. Further, our models helped point out alternate designs that could further lower complexity.

Overall, we have taken an important first step towards enabling systematic top-down routing design with minimizing design complexity being an explicit objective. Future work includes modeling a wider range of routing design objectives and primitives (such as selection logic), developing algorithms for automatically producing complexity-optimized routing designs in a top-down fashion, and using similar models to capturing complexity of other enteprise design tasks.

11 Acknowledgments

This material is based upon work supported by the National Science Foundation (NSF) Career Award No. 0953622, NSF Grant CNS-0721574, and Cisco. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of NSF or Cisco. We thank Brad Devine for his insights on Purdue's network design. We thank Michael Behringer, Alexander Clemm, Ralph Droms and our shepherd Vyas Sekar for feedback that greatly helped improve the presentation of this paper.

12 References

[1] IRTF Network Complexity Research Group. `http://irtf.org/ncrg`.
[2] Open Networking Foundation. `http://www.opennetworking.org`.
[3] C. Alaettinoglu, C. Villamizar, E. Gerich, D. Kessensand, D. Meyer, T. Bates, D. Karrenberg, and M. Terpstra. *Routing Policy Specification Language (RPSL)*. Internet Engineering Task Force, 1999. RFC 2622.
[4] M. A. Alim and T. G. Griffin. On the interaction of multiple routing algorithms. In *Proc. ACM CoNEXT*, 2011.
[5] T. Benson, A. Akella, and D. Maltz. Unraveling the complexity of network management. In *Proc. of USENIX NSDI*, 2009.
[6] T. Benson, A. Akella, and D. A. Maltz. Mining policies from enterprise network configuration. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, pages 136–142, 2009.
[7] T. Benson, A. Akella, and A. Shaikh. Demystifying configuration challenges and trade-offs in network-based isp services. In *Proc. of ACM SIGCOMM*, 2011.
[8] H. Boehm, A. Feldmann, O. Maennel, C. Reiser, and R. Volk. Network-wide inter-domain routing policies: Design and realization. Apr. 2005. Draft.
[9] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker. Ethane: Take control of the enterprise. In *Proc. ACM SIGCOMM*, 2007.
[10] Cisco Systems Inc. Reliable static routing backup using object tracking. `http://www.cisco.com/en/US/docs/ios/12_3/12_3x/12_3xe/feature/guide/dbackupx.html`.
[11] N. Foster, R. Harrison, M. J. Freedman, C. Monsanto, J. Rexford, A. Story, and D. Walker. Frenetic: a network programming language. In *Proceedings of the 16th ACM SIGPLAN international conference on Functional programming*, pages 279–291, 2011.
[12] J. Gottlieb, A. Greenberg, J. Rexford, and J. Wang. Automated provisioning of BGP customers. In *IEEE Network Magazine*, Dec. 2003.
[13] A. Greenberg, G. Hjalmtysson, D. A. Maltz, A. Myers, J. Rexford, G. Xie, H. Yan, J. Zhan, and H. Zhang. A clean slate 4D approach to network control and management. *ACM Computer Communication Review*, October 2005.
[14] T. G. Griffin and J. L. Sobrinho. Metarouting. In *Proc. ACM SIGCOMM*, 2005.
[15] H. Kim, T. Benson, A. Akella, and N. Feamster. The evolution of network configuration: A tale of two campuses. In *Proc. of ACM IMC*, 2011.
[16] F. Le, G. G. Xie, D. Pei, J. Wang, and H. Zhang. Shedding light on the glue logic of the Internet routing architecture. In *Proc. ACM SIGCOMM*, 2008.
[17] F. Le, G. G. Xie, and H. Zhang. Understanding route redistribution. In *Proc. International Conference on Network Protocols*, 2007.
[18] F. Le, G. G. Xie, and H. Zhang. Instability free routing: Beyond one protocol instance. In *Proc. ACM CoNEXT*, 2008.
[19] F. Le, G. G. Xie, and H. Zhang. Theory and new primitives for safely connecting routing instances. In *Proc. ACM SIGCOMM*, 2010.
[20] D. Maltz, G. Xie, J. Zhan, H. Zhang, G. Hjalmtysson, and A. Greenberg. Routing design in operational networks: A look from the inside. In *Proc. ACM SIGCOMM*, 2004.
[21] R. Rastogi, Y. Breitbart, M. Garofalakis, and A. Kumar. Optimal configuration of ospf aggregates. *IEEE/ACM Transaction on Networking*, 2003.
[22] M. Reitblatt, N. Foster, J. Rexford, C. Schlesinger, and D. Walker. Abstractions for network update. In *Proceedings of the ACM SIGCOMM*, 2012.
[23] X. Sun, S. Rao, and G. Xie. Modeling complexity of enterprise routing design. Technical Report TR-ECE-12-10, School of ECE, Purdue University, 2012.
[24] E. Sung, X. Sun, S. Rao, G. G. Xie, and D. Maltz. Towards systematic design of enterprise networks. *IEEE/ACM Trans. Networking*, 19(3):695–708, June 2011.
[25] L. Vanbever, S. Vissicchio, C. Pelsser, P. Francois, and O. Bonaventure. Seamless network-wide IGP migrations. In *Proc. ACM SIGCOMM*, 2011.