Multi-Objective Network Synthesis for Dispersed Computing in Tactical Environments

Jared Coleman[a], Eugenio Grippo[a], Bhaskar Krishnamachari[a], and Gunjan Verma[b]

[a]University of Southern California, USA
[b]US Army's CCDC Army Research Laboratory, USA

ABSTRACT

Tactical operations like search and rescue or surveillance necessitate the rapid synthesis of physically dispersed assets and mobile compute nodes into a network capable of efficient and reliable information gathering, dissemination, and processing. We formalize this network synthesis problem as selecting one among a set of potentially deployable networks which optimally supports the distributed execution of complex applications. We present the NSDC (Network Synthesis for Dispersed Computing) Framework, a general framework for studying this type of problem and use it to provide a solution for one well-motivated variant. We discuss how the framework can be extended to support other objectives, parameters, and constraints as well as more scalable solution approaches.

Keywords: IoT, IoBT, Network Synthesis, Tactical Networks, Dispersed Computing, Optimization

1. INTRODUCTION

Network synthesis addresses the important problem of deploying scarce resources (communication nodes, sensors, compute devices, etc.) to best support a given task. Prior work in network synthesis has largely focused either purely on communication (k-connectivity,[1,2] fault-tolerance[3]) or sensor coverage[4] (and sometimes hybrids the two[5]). As the internet of things comes into the battlefield domain, though, the nature of tactical networks is shifting and operations like search and rescue or surveillance will rely on in-network distributed edge computing of sensor data. In this paper, we propose a novel network synthesis problem inspired by next-generation tactical network capabilities, prior work in communication-oriented network synthesis, and work on distributed application task scheduling over heterogeneous networks[6] (e.g. HEFT,[7] which we leverage in our example). In particular, we study the network synthesis problem of selecting one among a set of deployable networks which optimally supports a given tactical mission modeled as the distributed and coordinated execution of complex tasks/applications.

Task scheduling, whether done manually or automatically, has always been important in distributed computing. In this field, distributed tasks are typically modeled as directed acyclic graphs (DAGs) whose edges represent dependencies between tasks. Networks are typically modeled as graphs whose weighted edges reflect the communication strength between two nodes. While finding the best (with respect to a metric like makespan or throughput) task-to-processor allocation is generally NP-hard,[8] many heuristic-based scheduling algorithms have been proposed to solve this problem. Rather than find the best schedule for a fixed network and distributed application, our goal is to *synthesize* a network to best support a *family* of distributed applications. As such, we leverage existing scheduling algorithms to measure the quality of candidate networks during synthesis.

In addition to the problem formulation itself, other primary contributions of this work are the NSDC Framework, a general framework for studying this type of problem, and implementations targeting a well-motivated variant. We consider heterogeneous network elements wherein nodes may possess high- or low-speed satellite,

Further author information: (Send correspondence to J.C.)
J.C.: E-mail: jaredcol@usc.edu
B.K.: E-mail: bkrishna@usc.edu
E.G.: E-mail: egrippo@usc.edu
G.V.: E-mail: gunjan.verma.civ@army.mil

radio, and/or cellular links with other nodes in the network. Some nodes may have high computational power while others may be resource-constrained. Each link and node has a deployment cost and security risk factor associated with its speed, position, and links to other nodes. We also assume there exists a known set of possible missions to be supported by the network. Missions represent complex dispersed computing applications and are described as directed acyclic task graphs. We demonstrate the NSDC Framework through a multi-objective optimization problem which aims to find a subset of nodes and links to deploy, with three different objectives: (i) optimizing a performance metric such as minimizing the average execution time (makespan) for a given set of task graphs, (ii) minimizing deployment cost and (iii) minimizing risk.

2. PROBLEM DEFINITION

A task graph $G = (T, D)$ is a weighted directed acyclic graph with tasks $T = \{t_1, t_2, \ldots, t_n\}$ and edges D where $(t_1, t_2) \in D$ if task t_2 requires the output of task t_1 as input to execute. Each task $t \in T$ in G has its own computation cost $\texttt{compute-cost}(v)$ representing the normalized cost to execute the task (i.e. number of operations, etc.). Each edge $(t_1, t_2) \in D$ has a weight $\texttt{data-size}(t_1, t_2)$ representing the amount of data required to be passed from task t_1 to t_2 (see Figure 1).

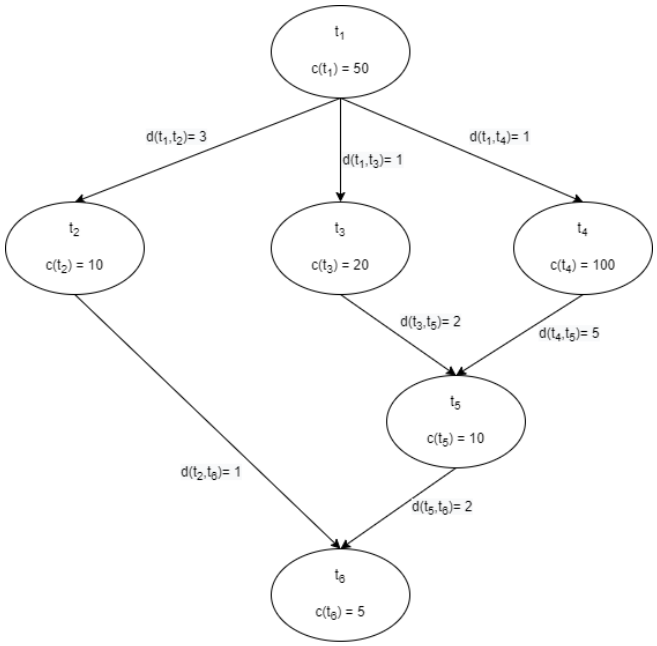

Figure 1. This example task graph is directed, acyclic, and has six tasks. Observe task t_1 has a compute cost of 50 while task t_2 has a compute cost of 10 indicating that the same processor would take five times as long to execute task t_1 as it would to execute task t_2. Also notice the data dependencies between the tasks. For example task t_5 requires 5 units of output from task t_4 as input to execute. If these tasks are executing on separate processors, the data must be sent over the network before execution.

A network $N = (V, E)$ represents a network of compute nodes V. Each node $v \in V$ has its own processing speed $\texttt{speed}(v)$ and thus can compute a task $t \in T$ in time $\texttt{compute-cost}(t)/\texttt{speed}(v)$. Every pair of nodes $v_1, v_2 \in V$ has a direct communication strength $\texttt{bandwidth}(v_1, v_2)$ (which may be zero, indicating the two nodes cannot communicate directly). In other words, it would take $D/\texttt{bandwidth}(v_1, v_2)$ time to send D units of data from v_1 to v_2.

Nodes and edges may have other attributes not directly related to scheduling such as risk or deployment cost which may still affect network synthesis. In the example problem studied in this paper, we assume each node and edge has both risk (denoted $\texttt{risk}(v)$ and $\texttt{risk}(v_1, v_2)$) and deployment cost (denoted $\texttt{deploy-cost}(v)$ and $\texttt{deploy-cost}(v_1, v_2)$). Then we frame the network synthesis problem as finding the subset of nodes $V' \subseteq V$ and

edges $E' \subseteq E$ which minimizes some *linear combination* of risk, deployment cost, and average makespan across a set of given task graphs $\mathcal{G}$:

$$\begin{aligned}\texttt{cost}(V', E', \mathcal{G}) = \; & R\left(\sum_{v \in V'} \texttt{risk}(v) + \sum_{v_1, v_2 \in V'} \texttt{risk}(v_1, v_2)\right) + \\ & D\left(\sum_{v \in V'} \texttt{deploy-cost}(v) + \sum_{v_1, v_2 \in V'} \texttt{deploy-cost}(v_1, v_2)\right) + \\ & M \cdot \frac{1}{|\mathcal{G}|} \sum_{G \in \mathcal{G}} \texttt{MAKESPAN}(V', \; E', \; G)\end{aligned} \tag{1}$$

We assume $\texttt{MAKESPAN}(V', E', G)$ for a subset of the network $N' = (V', E')$ and a task graph G is computed using a scheduler like the Heterogeneous Earliest Time Finish scheduler which uses a heuristic algorithm to produce a task-to-node allocation schedule to minimize the makespan (total execution time) of the DAG over the network (with no formal guarantees on its optimality). We call this problem which minimizes a linear combination of **r**isk, **d**eployment cost, and **m**akespan (RDM) the RDM Network-Synthesis problem:

Problem (RDM Network-Synthesis)**.** *Given a network $N = (V, E)$ and a set of task graphs $\mathcal{G}$, find a $V' \subseteq V$ and $E' \subseteq E$ which minimizes Equation* (1)*.*

3. THE NSDC FRAMEWORK

In this section we propose the NSDC Framework, a general framework for solving network synthesis problems like RDM Network-Synthesis presented in the previous section. RDM Network-Synthesis belongs to a family of network synthesis problems which involve a set or distribution of candidate networks, a set or distribution of distributed applications to support, a scheduler for evaluating the quality of networks for the set of distributed apps, and an optimizer which iteratively constructs/chooses candidate networks to evaluate, eventually arriving at or close to an optimum with respect to network attributes (i.e. risk, deployment cost, makespan, etc.).

The NSDC Framework provides common interfaces for optimizers, schedulers, networks, distributed applications, and objective functions. By creating different implementations which adhere to common interfaces, we can mix and match (see Figure 2) different optimizers, schedulers, etc. to solve problems all kinds of distributed computation oriented network synthesis problems like RDM Network-Synthesis. We have made an open-source Python implementation of these common interfaces available on GitHub*.

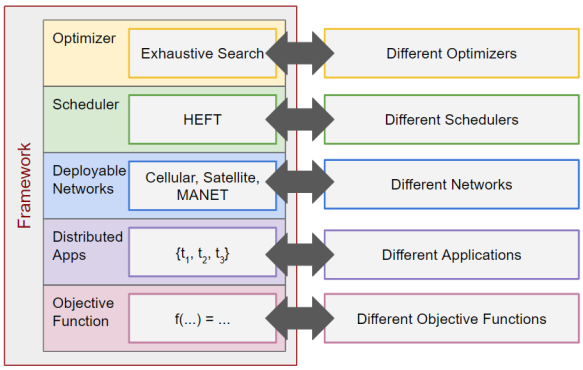

Figure 2. The NSDC Framework allows us to combine different optimizers, schedulers, networks, distributed applications, and cost functions to solve problems like RDM Network-Synthesis and others in the larger family of Network Synthesis problems it belongs to.

*https://github.com/ANRGUSC/NSDC

4. IMPLEMENTATION & RESULTS

To demonstrate the utility of the NSDC Framework, we consider an instance of RDM Network-Synthesis with the "mother network" (the set of all deployable nodes and communication links) depicted in Figure 3. We consider high/low-speed compute nodes, high/low speed connections between them and a set of five task graphs with high/low cost tasks and high/low data dependencies (Figure 3). We implemented a HEFT scheduler and an exhaustive search optimizer compatible with the NSDC Framework and ran it with the risk, deployment cost, and speeds for the different link/node types in the "mother network" specified in Table 1. We used the parameters $R \leftarrow 1, D \leftarrow 0.1, M \leftarrow 0.1$ for the overall cost metric to be minimized.

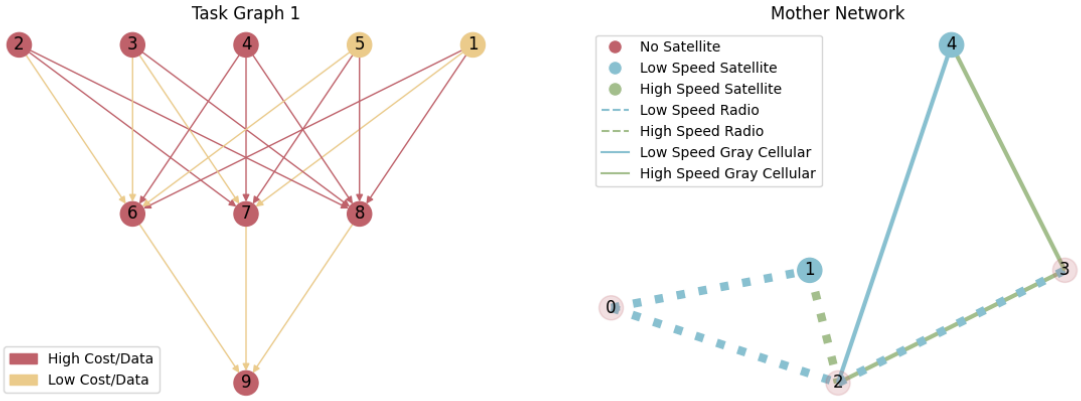

Figure 3. One of the five task graphs to be supported by the network (**left**) and the "mother network" which represents all deployable nodes and links (**right**). We consider five task graphs which have the same structure as Task Graph 1 depicted here but different cost/data requirements.

		Risk	Deploy Cost	Speed
Satellite Link	**Low**	0	5	0.2
	High	0	10	0.4
Radio Link	**Low**	0	5	0.2
	High	0	10	0.4
Gray Cellular Link	**Low**	1	1	0.8
	High	1	2	1.6
Compute Node	**Low**	0	1	0.1
	High	0	2	0.2

Table 1. Risk, deploy-cost, and speeds for the different deployable network elements.

The network which optimizes the specified cost function involves three of five possible nodes and four of nine possible communication links (Figure 4). Not surprisingly, the optimal network is not the fastest, the cheapest, nor the least risky (Figure 5). Rather, it represents the best balance of these three metrics subject to the provided cost function.

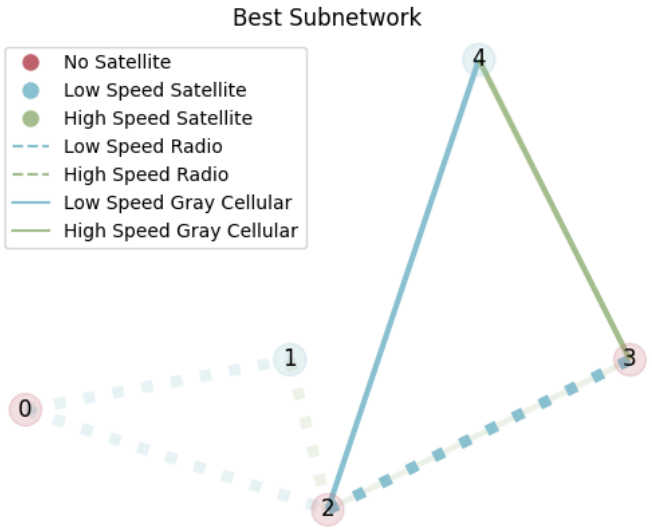

Figure 4. The network which minimized the specified linear combination of risk, deployment cost, and makespan was that which included the low-speed radio link between 2 and 3 and the risky gray cellular links between 2 and 4 (low-speed) and 3 and 4 (high-speed). Thus, nodes 2, 3, and 4 are deployed as compute nodes.

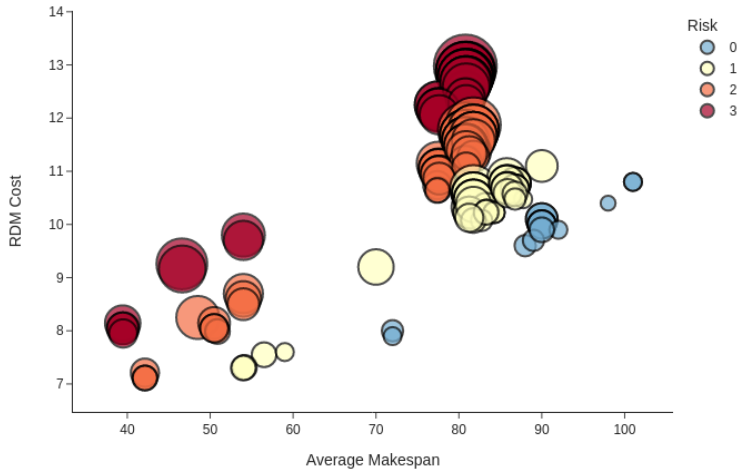

Figure 5. The results from exhaustive search over the example described demonstrates how the optimal network may not simultaneously minimize risk, deployment cost, and makespan. Rather, the optimal network is that for which these attributes are most balanced (as a linear combination according to given parameters). In the figure, each marker represents a valid network and larger markers indicate a higher deployment cost for the network.

5. CONCLUSION

In this paper, we formalized a novel network synthesis problem inspired by next-generation IoBT tactical operations involving dispersed computing. We presented the NSDC Framework, a general framework for solving these kinds of problems and demonstrated its utility on a well-motivated example. The NCDC Framework presented in this paper opens many avenues for future research. First, exhaustive search is inherently not scalable and an exploration into other optimizers (i.e. simulated annealing) for the outer loop would be valuable. Second, the HEFT scheduling algorithm used in the example is also not scalable. Other scheduling algorithms must be

implemented. A particularly interesting and recent area of research is in training GCNs to mimick algorithms like HEFT for more rapid scheduling.[9] Finally, more complex variations of the proposed network synthesis problem (i.e. networks with more complex connectivity, larger and more realistic distributed applications, mobile compute nodes, etc.) can now be studied more easily leveraging the NSDC Framework.

ACKNOWLEDGMENTS

This work was supported in part by Army Research Laboratory under Cooperative Agreement W911NF-17-2-0196.

REFERENCES

[1] Steiglitz, K., Weiner, P., and Kleitman, D., "The design of minimum-cost survivable networks," *IEEE Transactions on Circuit Theory* **16**(4), 455–460 (1969).

[2] Kamalesh, V. and Srivatsa, S., "On the design of minimum cost survivable network topologies," in [*Nat. Conf. on communication at IIT, Guwahati, India*], 394–397, Citeseer (2009).

[3] Beckett, R., Mahajan, R., Millstein, T., Padhye, J., and Walker, D., "Network configuration synthesis with abstract topologies," in [*Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation*], 437–451 (2017).

[4] Younis, M. and Akkaya, K., "Strategies and techniques for node placement in wireless sensor networks: A survey," *Ad Hoc Networks* **6**(4), 621–655 (2008).

[5] Ghosh, P., Bunton, J., Pylorof, D., Vieira, M., Chan, K., Govindan, R., Sukhatme, G., Tabuada, P., and Verma, G., "Rapid top-down synthesis of large-scale iot networks," in [*2020 29th International Conference on Computer Communications and Networks (ICCCN)*], 1–9, IEEE (2020).

[6] Singh, K., Alam, M., and Sharma, S. K., "A survey of static scheduling algorithm for distributed computing system," *International Journal of Computer Applications* **129**(2), 25–30 (2015).

[7] Topcuoglu, H., Hariri, S., and Wu, M.-Y., "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE transactions on parallel and distributed systems* **13**(3), 260–274 (2002).

[8] Ullman, J. D., "Np-complete scheduling problems," *Journal of Computer and System sciences* **10**(3), 384–393 (1975).

[9] Kiamari, M. and Krishnamachari, B., "Gcnscheduler: Scheduling distributed computing applications using graph convolutional networks," *arXiv preprint arXiv:2110.11552* (2021).