HERA: A Novel Heuristic Resource Allocator for Multi-SDM Multi-User Settings

Christos Antoniadis*, Kostas Katsalis†, Dimitrios Tyrovolas*‡, Sotiris A. Tegos*,
Sotiris Ioannidis‡, Panagiotis D. Diamantoulakis*, George K. Karagiannidis*§, Christos Liaskos¶‖
*Department of Electrical and Computer Engineering, Aristotle University of Thessaloniki, 54124 Thessaloniki, Greece
e-mail: {chrianto, tyrovolas, tegosoti, padiaman, geokarag}@auth.gr
†DOCOMO Euro-Labs, Munich, Germany
e-mail: katsalis@docomolab-euro.com
‡Dept. of Electrical and Computer Engineering, Technical University of Crete, Chania, Greece, e-mail: sotiris@ece.tuc.gr
§Artificial Intelligence & Cyber Systems Research Center, Lebanese American University (LAU), Lebanon
¶Computer Science Engineering Department, University of Ioannina, Ioannina, Greece, e-mail: cliaskos@ics.forth.gr
‖Institute of Computer Science, Foundation for Research and Technology Hellas, Greece

***Abstract*—The advent of Programmable Wireless Environments (PWEs) has transformed the wireless propagation phenomenon into a software-defined resource, leveraging Software-defined metasurfaces (SDMs). These new technologies have shown that wireless waves can be routed within a space, contrary to the regular, chaotic wireless propagation, yielding considerable benefits to communication efficiency, and even completely new applications. A new topic has risen, in the context of modeling these capabilities as network resources, and allocating them to users. In this context, the present paper contributes HERA, a heuristic resource allocator that receives a setup with users and SDMs as input, and produces the necessary configuration of the latter, for efficient shared performance. Simulation results demonstrate considerable efficiency in this task, while the heuristic nature of HERA sets the basis for a flexible definition of complex user objectives in the future.**

***Index Terms*—Software-Defined Metasurfaces, Programmable Wireless Environments, Resource Allocation, Heuristic Algorithms.**

I. Introduction

The evolution toward 6G networks introduces a groundbreaking shift in wireless communication, namely Programmable Wireless Environments (PWEs) that transform the wireless propagation medium into a flexible and programmable resource [1], [2]. This transformation allows for fine-grained control over signal paths, ensuring that electromagnetic waves can be manipulated precisely. Therefore, with this degree of control, PWEs provide the adaptability needed to address the growing complexity of next-generation communication systems, making them a strong candidate for the 6G vision.

At the core of this transformation, PWEs achieve their programmability through the integration of Software Defined Metasurfaces (SDMs) that coat physical objects in the environment [3], [4]. These programmable surfaces control electromagnetic waves by adjusting how signals are reflected, absorbed, or diffused, enabling real-time manipulation of RF transmissions. By introducing this new degree of freedom, PWEs can optimize signal paths and reduce interference, thereby transforming RF propagation into a resource that can be dynamically allocated. This shift marks a crucial advancement in wireless networks, where managing precisely the propagated signals is essential for meeting the stringent 6G requirements.

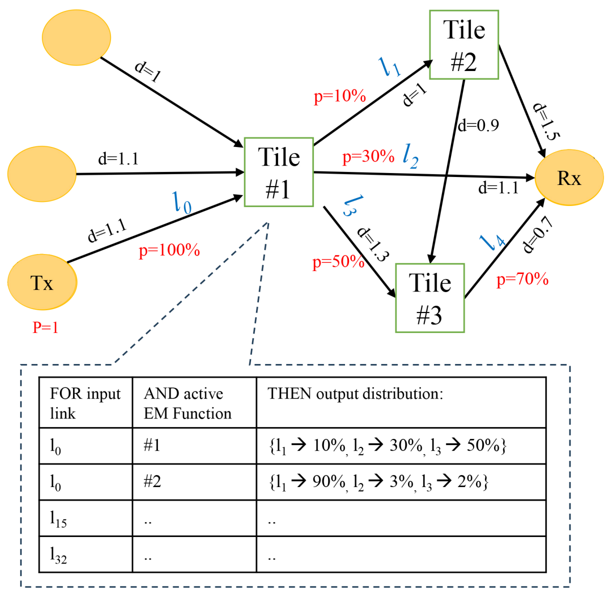
However, despite their remarkable capabilities, SDMs need to interface with the resource allocation mechanisms that govern modern communication systems [5]. Particularly, their is a need for high-level algorithms that can deduce the manner in which each SDMs needs to be configured to be shared efficiently among multiple users.

The present paper conrtibutes HERA, a heuristic resource allocator for PWEs utilizing SDMs, leveraging swarm optimization principles. We showcase applicability to multi-SDM settings via simulations, for a complex resource allocation that targets path equalization for multiple users at the same time.

The previous work is surveyed in Section II. HERA is detailed in Section III and evaluated in Section IV. Conclusion follows in Section V.

II. Previous Work and Systemic Context

There are two major directions in the field of programmable wireless propagation. The PWE approach, proposed in 2018 [1], made a case for even deterministic wireless propagation, leveraging high-efficiency SDMs. Later in 2019, the concept was simplified as smart radio environments [6], leveraging cheap reflectarrays with medium efficiency, with the objective of making the channel quality statistically better than the regular propagation case, and–primarily–constituting the overall system more financially compelling to early investors. In order to facilitate the classification of works, an OSI-compliant layer stack was proposed for programmable propagation systems [7]. In this stack, SDMs constitute physical-layer choices with different performance characteristics. The network layer for a system of cascaded SDMs units has been presented in [8], while indicative application layer instances are the statistical improvements of wireless channels for SDM [9], and latency-optimal extended reality systems for PWEs [10].



FOR input link	AND active EM Function	THEN output distribution:
l_0	#1	$\{l_1 \rightarrow 10\%, l_2 \rightarrow 30\%, l_3 \rightarrow 50\%\}$
l_0	#2	$\{l_1 \rightarrow 90\%, l_2 \rightarrow 3\%, l_3 \rightarrow 2\%\}$
l_{15}	..	..
l_{32}	..	..

Fig. 1. Overview of the employed graph model for SDMs.

The advances in PWE and SDM technology has also raised the issue of new resource modeling approaches and interfacing with existing resource allocation approaches. Recently, it was shown that the integration of SDMs with resource virtualization holds significant promise for revolutionizing the way telecommunications networks are designed and managed [5]. By leveraging these surfaces as virtualized assets, network operators can create dynamic, adaptive infrastructures that can be reconfigured in real-time to meet changing user demands. This enables a more agile and responsive approach to service delivery, allowing users to access customized services and applications with unprecedented flexibility and scalability. Furthermore, the abstraction layer provided by resource virtualization enables seamless integration of diverse hardware platforms and logical systems, paving the way for the development of heterogeneous networks that can support an increasingly complex array of user needs in the context of customizable wireless propagation.

The present work aligns with the resource modeling of SDMs, contributing an algorithm for efficiently sharing them among different users in a given setup.

III. The Proposed Heuristic Resource Allocator (HERA) for PWEs

The proposed HERA leverages the graph abstraction model for PWEs [8], as illustrated in Fig. 1 where each SDM and user is modeled as a node in a graph. Specifically, edges are established between any two nodes with a line of sight, and the length of these edges captures the effects of wave propagation latency and signal loss. In the literature, the SDM nodes are often referred to as tiles, with both terms being used interchangeably. Furthermore, each tile is capable of interacting with impinging waves through various electromagnetic (EM) functions, such as steering, splitting, and absorbing the waves. These EM functions are applied via software, allowing for dynamic control over how the tile interacts with incoming waves. Moreover, the behavior of each tile is governed by a special routing table, illustrated in the bottom of Fig. 1, which dictates how the wave is distributed across the tile based on its incoming direction, characteristics, and the selected EM function. Finally, when multiple EM functions are applied in succession, such as repeated wave steerings, they form continuous air routes or paths for the wave to propagate.

A. Graph Model Initialization

The initialization algorithm for the graph is responsible for creating the nodes, edges, and EM functions, and subsequently using them to instantiate the routing tables mentioned earlier. To further enhance the flexibility of this module, we provide the transmitter, surfaces, and receiver objects, each defined by their coordinates within the designated space. These coordinates are then used to calculate the distances between the nodes, which, in turn, determine the lengths of the edges in the graph. By calculating these distances, Algorithm 1 can effectively model the wave propagation characteristics and set up the connections needed for accurate path creation.

Algorithm 1 InitializeNodes(surfaces,transmitters,receivers)
```
$nodeIdx \leftarrow 0$
**for each** $surface$ **in** $surfaces$ **do**
    **for** $i \leftarrow 0$ **to** $surface.numberOfTiles$ **do**
        $newNode \leftarrow$ **new** $Node(nodeId, surface.id, \text{“}Tile\text{”})$
        $newNode.SetPosition()$
        $graphNodes.Add(newNode)$
        $nodeId \leftarrow nodeId + 1$
    **end for**
**end for**
**for all** $transmitters$ **do**
    $newNode \leftarrow$ **new** $Node(nodeId, \text{“}Tx\text{”})$
    $graphNodes.Add(newNode)$
    $nodeId \leftarrow nodeId + 1$
**end for**
**for all** $receivers$ **do**
    $newNode \leftarrow$ **new** $Node(nodeId, \text{“}Rx\text{”})$
    $graphNodes.Add(newNode)$
    $nodeId \leftarrow nodeId + 1$
**end for**
```

In order to be able to retrieve nodes and edges from their respective lists in future applications, assigning unique identifiers is crucial. When instantiating edges representing the possible propagation paths within the system, all potential node pairs must be evaluated. However, for an edge to be considered valid, four essential conditions must be satisfied: (a) connections between nodes on the same surface are disallowed, (b) edges cannot be directed at a transmitter, (c) edges cannot originate from the receiver, and (d) line-of-sight criteria between the two nodes must be met.

The newly instantiated edges are subsequently integrated into the input and output link lists of the respective nodes. These lists are instrumental in the initialization of routing tables, as detailed below in the `InitializeRoutingTables()` method (Algorithm 3):

Algorithm 2 InitializeEdges()

```
$edgeId \leftarrow 0$
for all $possibleNodePairs$ do
    if $ConditionsAreMet()$ then
        $newEdge \leftarrow$ new $Edge(startNode.id, destNode.id)$
        $newEdge.length \leftarrow EuclidianDist(pair)$
        $newEdge.id \leftarrow edgeId$
        $graphEdges.Add(newEdge)$
        $startNode.outputEdges.Add(newEdge.id)$
        $endNode.inputEdges.Add(newEdge.id)$
        $edgeId \leftarrow edgeId + 1$
    end if
end for
```

For each element within the system, propagation routes are dynamically and randomly generated for all possible input directions, simulating the "leaky" nature of tile reflections. This process is iteratively executed based on the number of EM functions that require initialization, ensuring that each function is appropriately accounted for within the overall system behavior.

Algorithm 3 InitializeRoutingTables()

```
$graphTiles \leftarrow GetTiles(graphNodes)$
for each $tile$ in $graphTiles$ do
    $routingTable \leftarrow$ new $HashMap()$
    for each $edgeId$ in $tile.inputEdges$ do
        for $EMfuncId \leftarrow 0$ to $numEMFuncs$ do
            $powerDist \leftarrow ProduceDist(tile.outputEdges)$
            $key \leftarrow edgeId + EMfuncId$
            $routingTable.Put(key, powerDist)$
        end for
    end for
    $routingTables.Add(routingTable)$
end for
```

The `produceDist()` method is tasked with generating the power distribution, which is represented as a $2 \times N$ matrix. This matrix consists of pairs of power fractions and corresponding edge indices. Once generated, this matrix is stored in the routing table hashmap, keyed by the input edge ID and EM function number, ensuring that each unique combination is efficiently mapped to its respective power distribution data. When creating these distributions we consider the scenario that a fraction of the reflected power will be attenuated in the designated space thus inserting losses.

To initiate the system's operation, an initial set of rays is generated, originating from the transmitters. These rays are selected from randomly chosen edges, each assigned a normalized power fraction of 1, forming the foundation for the subsequent analysis. (It is noted that in the context of the ensuing evaluation in Section IV, this process leverages simulations. In a real application setting, the system will follow an equivalent calibration phase via power measurements).

B. Wireless Propagation Modeling

Algorithm 4 simulates the propagation of rays within a network of interconnected nodes that was initialized in subsection III-A. It models how signals travel through different paths, reflecting off nodes, considering path loss, and determining the overall delay spread on the receiver.

Algorithm 4 Propagate() Method

```
$activeRays \leftarrow CopyList(inputRays)$
while $activeRays.isNotEmpty()$ do
    for each $ray$ in $activeRays$ do
        $ReflectRay(ray)$
    end for
    $activeRays.RemoveAll()$
    $activeRays.AddAll(newRays)$
    $newRays.RemoveAll()$
end while
$delaySpread \leftarrow CalculateDelaySpread(receiver.rays)$
$power \leftarrow CalculatePower(receiver.rays)$
return $new\ Solution(EMfuncsChosen, \ldots$
$\ldots delaySpread, power)$
```

The propagation is emulated by iteratively reflecting a list of active rays. Reflected rays will be discarded and the new ones created will be added to the list to repeat the process. To ensure that the computational needs of the algorithm remain within realistic boundaries, rays will be ignored after they have been attenuated beyond a certain threshold. After a ray reaches its destination node, the final edge traversed and the EM function activated are used as keys to access the node's routing table. This allows Algorithms 5 and 6 to determine the power fraction and the directions in which the new rays will be dispersed, guiding their subsequent paths.

Algorithm 5 ReflectRay(ray) Method

```
1: $routingTable \leftarrow GetRoutingTable(ray.nodeId)$
2: $node \leftarrow graphNodes[node.id]$
3: $ChooseEMFunction(node)$
4: $key \leftarrow node.activeEMFunctionId + ray.edgeId$
5: $powerDistribution \leftarrow routingTable[key]$
6: $CreateRays(ray, powerDistribution)$
```

Modeling the wireless ray propagation requires accurate path loss calculations, and two key models have been developed for this purpose [11]. The first model, known as the product of distances, breaks the signal path into segments, calculating the path loss for each segment individually before multiplying these losses together. This approach is particularly useful in far-field scenarios, where the transmitter, SDMs, and receiver are separated by significant distances, and signal reflections occur over larger areas. The product of distances model can be expressed mathematically as

Algorithm 6 CreateRays(ray,distribution) Method

```
for $i \leftarrow 0$ to $powerDistribution.length$ do
    $edgeId \leftarrow distribution[0]_i$
    $powerFraction \leftarrow distribution[1]_i$
    $newEdge \leftarrow allEdges[edgeId]$
    $pathLoss \leftarrow CalcPathLoss(ray.length, \ldots$
    $\ldots newEdge.length)$
    $totalLength \leftarrow ray.length + newEdge.length$
    $totalPower \leftarrow ray.power \times powerFraction \times$
    $pathLoss \times materialLoss$
    if $totalPower \leq threshold$ then
        continue
    end if
    if $newEdge.destinationNode.isReceiver()$ then
        continue
    end if
    $newRay \leftarrow$ new $Ray(totalPower, totalLength, \ldots$
    $\ldots newEdge)$
    $newRays.Add(newRay)$
end for
```

$$r = \frac{P_t G_t G_r \prod_{i=1}^{N_{RIS}} G_{RIS,i}}{\prod_{i=1}^{N_{RIS}+1} (4\pi f/c)^2 d_i^{\alpha}}, \tag{1}$$

where P_t is the transmitted power, G_t and G_r are the antenna gains of the transmitter and receiver, and d_i represents the distance for each segment of the signal path.

However, in situations where the SDM is placed closer to the communication nodes or where the surface of the SDM is large, a different model provides a more practical solution. The sum of distances model treats the entire signal path as a single segment, simplifying the path loss calculation by summing the distances between each point in the signal path. This model is particularly effective in near-field conditions, where the signal experiences minimal spreading, and can be captured in the formula

$$r = \frac{P_t G_t G_r \prod_{i=1}^{N_{RIS}} G_{RIS,i}}{(4\pi f/c)^2 \left(\sum_{i=1}^{N_{RIS}+1} d_i\right)^{\alpha}}. \tag{2}$$

Both models can be used, depending on the setting, providing a ray attenuation model. When the power carried by a ray drops beyond a threshold (defined by the system), the ray is considered as fully attenuated and is no longer tracked.

C. The HERA Resource Allocation Optimization process

The optimization problem aims to find the optimal EM function for each SDM, minimizing the delay spread at the receiver. This involves identifying a sequence of values, where the sequence length corresponds to the number of nodes in the graph, and each value represents a specific EM function assigned to a node. As described in the initialization of the routing tables in Algorithm 3, each EM function associated with a tile is assigned a unique integer index.

To solve this optimization problem, HERA implements an elitist swarm intelligence framework, where each member of the swarm is modeled as a *heuristic optimizing agent.* The algorithm begins by assigning a unique value to each EM function, that will be referred to as a *preference gradient.* As the simulation progresses and signals propagate, each time an SDM encounters a signal for the first time, an EM function is activated based on the distribution of preference gradients across all available options. Each agent then constructs a potential solution, evaluates its quality, and stores it in the *solutions* list. (Algorithm 7):

Algorithm 7 AllocateResources()

```
$InitializePreferenceGradients()$
while $termination\ condition\ is\ not\ met$ do
    $solutions \leftarrow SendAgents(numberOfAgents)$
    $solutions.Sort()$
    $bestSolutions \leftarrow solutions.Take(C)$
    for $i \leftarrow 0$ to $numberOfLocalSearches$ do
        Solution $sol \leftarrow bestSolutions[i]$
        $sol \leftarrow ApplyLocalSearch(sol)$
    end for
    $UpdateDistributions(bestSolutions, increment)$
    $IncreaseRoundSignificance()$
end while
```

After each of the N agents has independently constructed their respective solutions, the entire pool is sorted order based on solution quality. From this ordered list, only a subset C, of high-quality solutions will be updated by incrementing the corresponding EM function distributions. As the search progresses, each round becomes increasingly more significant, by adjusting this increment according to the formula β^n $(n = 0, 1, 2, \ldots)$ where n denotes the round of the HERA algorithm (a round consists of having sent N agents). The larger the values of β, the more quickly old solutions will be forgotten, effectively introducing a form of memory decay into the optimization process, as shown in Algorithm 8.

Algorithm 8 SendAgents(numberOfAgents)

```
for i $\leftarrow 0$ to $numberOfAgents$ do
    Solution $newSol \leftarrow Propagate()$
    $allSolutions.Add(newSol)$
    if $newSol.quality > globalBest.quality$ then
        $globalBestSolution \leftarrow newSol$
    end if
end for
```

As this process focuses on exploiting high-quality solutions, its exploratory capacity is enhanced by integrating a local search mechanism. This local search is applied in every round to refine a fraction of the best solutions found by the groups. The local search routine explores neighboring solutions by modifying only one EM function at a time. The function `createNodeVector(n)` randomly selects a subset of n nodes to be modified, controlling the complexity of each iter-

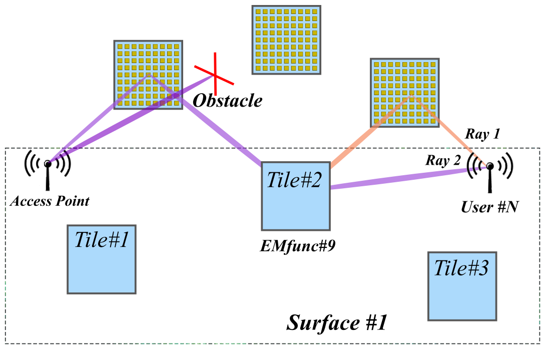


Fig. 2. The basic studied setup in terms of users and tiles.

ation. This approach balances exploration and computational cost, allowing for efficient and adaptable optimization.

Algorithm 9 ApplyLocalSearch(Solution sol)

```
$CreateNodeVector(n)$
$ListOfSolutions \leftarrow$ **new** $List()$
**for each** $nodeId$ **in** $nodeVector$ **do**
    **for** $EMfuncId \leftarrow 0$ **to** $numEMfuncs$ **do**
        **if** $EMfuncId.Equals(currentEMfunc)$ **then**
            **continue**
        **end if**
        $solCopy \leftarrow createDeepCopy(sol)$
        $solCopy.EMfuncsChosen[nodeId] \leftarrow EMfuncId$
        $solCopy \leftarrow Propagate(solCopy.EMfuncChosen)$
        $listOfSolutions.Add(solCopy)$
    **end for**
**end for**
$newSol \leftarrow listOfSolutions.getBest()$
**if** $solution\ is\ improved$ **then**
    $CheckIfIsNewGlobalOptimum()$
    **return** $newSol$
**end if**
**return** $sol$
```

IV. Evaluation

HERA is evaluated via a simulator implemented using two programming languages: Java and C#. The system initialization and graph construction are handled in Java, and was chosen due to its efficiency in managing complex data structures like graphs, which are integral to the system's architecture. The second part of the project, where the core simulation and optimization algorithms are executed, is implemented in C#, which offers better computational performance, making it suitable for the simulation phase of the project.

In this simulation, we model an enclosed space measuring $15m \times 8m \times 5m$, which contains a single user, an access point and tiles that are arranged across the walls and ceiling. Multiple realizations of this model are generated, each representing randomized positions and configurations of all system elements, including transmitters and receivers. The communication between the user and access point exclusively relies on interactions with the surrounding tiles. To simulate real-world conditions, we vary routing losses, line-of-sight probabilities, and receiver multi-path effects. This approach rigorously tests the efficiency and performance of HERA under diverse environmental scenarios.

In the subsequent simulations, we uniformly distribute 8, 16, 32, and 64 tiles across five surfaces (walls and ceiling). The transmitter emits power isotropically at $P_t = 0dB$, and the receiver's minimum detectable signal is set to $MDS = -120dB$, with the path loss exponent defined as $\alpha = 3$. For each tile, 20 distinct EM functions are randomly initialized. HERA is configured to deploy 30 agents per search, with each group returning 6 optimal solutions and the memory decay parameter set to $\beta = 2$. Initially, all preference gradients are set to 1, and the optimization process terminates after a maximum of $R_p = 18000$ propagation scenarios have been tested, each with a unique set of random EM functions.

After 100 iterations, we begin applying a single local search to refine the highest quality solutions, incrementing by one at the $250, 500, 3000, 6000$, and 9000 iteration marks, until reaching the maximum capacity of the best solutions list. The depth of the search is defined as 0.25, meaning that the EM functions of one in four tiles are altered each time the local search is invoked.

The performance of HERA can be assessed by comparing its results to those obtained from a brute force algorithm. In this evaluation, R_p random propagation scenarios are tested, and the average minimum delay spread achieved is calculated across K graphs. This can be mathematically expressed as:

$$\frac{1}{K}\sum_{j=0}^{K-1} \min_{i=0,1,2,\ldots,Rp} (\text{DS}(V_i)) \tag{3}$$

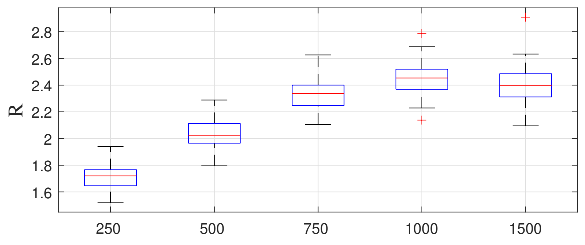
where each element V_i is a random combination of EM functions for all tiles in the graph and the delay spread is written as DS. Additionally, the average delay spread for the case where HERA is utilized for the selection of the applied EM functions for the same amount of repetitions R_p is given as

$$\frac{1}{K}\sum_{j=0}^{K-1} \text{DS}(V_{HERA}) \tag{4}$$

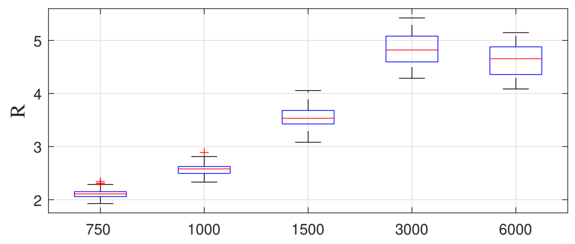
Finally, to compare the performance of HERA with the random search algorithm, we can use the following metric

$$R = \frac{\frac{1}{K}\sum_{j=0}^{K-1} \min_{i=0,1,2,\ldots,Rp} (\text{DS}(V_i))}{\frac{1}{K}\sum_{j=0}^{K-1} \text{DS}(V_{HERA})} \tag{5}$$

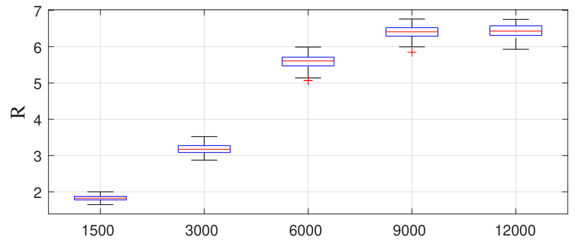
The results are shown in Fig. 3. The evaluation test has been repeated 100 times over $K = 100$ graphs for all four scenarios. In every case, HERA significantly outperforms the brute force search, achieving on average of 2 up to 6 times smaller delay spread. In scenarios with fewer tiles (e.g., 8, 16, and 32 tiles), HERA demonstrates notable enhancements over the baseline, particularly after 1000, 3000 and 9000 repetitions, respectively. This indicates that even in smaller problem spaces, HERA



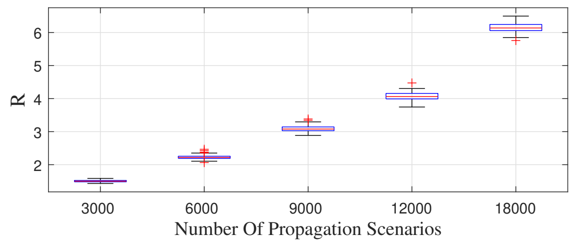
(a) 8 tiles.



(b) 16 tiles.



(c) 32 tiles.



(d) 64 tiles.

Fig. 3. Comparative performance between the proposed HERA and random search in terms of achieved delay spread, for various randomized setups.

converges quickly to better solutions. For larger configurations (e.g., 64 tiles), HERA's performance continues to improve with additional propagation scenarios, reflecting its scalability and efficiency in exploring vast solution spaces.

HERA showcases the prospects of heuristics in the SDM resource allocation task. In the future, we plan to exploit the flexible nature of heuristics for offering user service level agreement templates with high expressiveness, promoting their application to realistic setups.

V. Conclusion

This work contributed HERA, a novel resource allocator for programmable wireless environments leveraging swarm optimization heuristics. HERA performs a tunable, agent-based exploration of the solution space created by the combined capabilities of the SDMs units in a setup, yielding a shared configuration that can effectively serve the needs of multiple users. Simulations showcased the HERA effectiveness in a task involving delay spread optimization via a challenging channel and ray equalization.

Acknowledgment

This work has been funded by the EU H2020 research and innovation programme 6G TransContinental Edge Learning (6GXCEL, GA 101139194), and via H2020 CYBERSEC-DOME (GA 101120779).

References

[1] C. Liaskos, S. Nie, A. Tsioliaridou, A. Pitsillides, S. Ioannidis, and I. Akyildiz, "A new wireless communication paradigm through software-controlled metasurfaces," *IEEE communications magazine*, vol. 56, no. 9, pp. 162–169, 2018.

[2] J. Chen, Y.-C. Liang, Y. Pei, and H. Guo, "Intelligent reflecting surface: A programmable wireless environment for physical layer security," *IEEE Access*, vol. 7, pp. 82 599–82 612, 2019.

[3] L. Dai, B. Wang, M. Wang, X. Yang, J. Tan, S. Bi, S. Xu, F. Yang, Z. Chen, M. D. Renzo, C.-B. Chae, and L. Hanzo, "Reconfigurable intelligent surface-based wireless communications: Antenna design, prototyping, and experimental results," *IEEE Access*, vol. 8, pp. 45 913–45 923, 2020.

[4] E. Martini and S. Maci, "Theory, analysis, and design of metasurfaces for smart radio environments," *Proceedings of the IEEE*, vol. 110, no. 9, pp. 1227–1243, 2022.

[5] C. Liaskos, K. Katsalis, J. Triay, and S. Schmid, "Resource management for programmable metasurfaces: Concept, prospects and challenges," *IEEE Communications Magazine*, 2023.

[6] M. D. Renzo, M. Debbah, D.-T. Phan-Huy, A. Zappone, M.-S. Alouini, C. Yuen, V. Sciancalepore, G. C. Alexandropoulos, J. Hoydis, H. Gacanin *et al.*, "Smart radio environments empowered by reconfigurable ai meta-surfaces: An idea whose time has come," *EURASIP Journal on Wireless Communications and Networking*, vol. 2019, no. 1, pp. 1–20, 2019.

[7] C. Liaskos, L. Mamatas, A. Pourdamghani, A. Tsioliaridou, S. Ioannidis, A. Pitsillides, S. Schmid, and I. F. Akyildiz, "Software-defined reconfigurable intelligent surfaces: From theory to end-to-end implementation," *Proceedings of the IEEE*, vol. 110, no. 9, pp. 1466–1493, 2022.

[8] C. Liaskos, A. Tsioliaridou, S. Nie, A. Pitsillides, S. Ioannidis, and I. F. Akyildiz, "On the network-layer modeling and configuration of programmable wireless environments," *IEEE/ACM transactions on networking*, vol. 27, no. 4, pp. 1696–1713, 2019.

[9] S. Gong, X. Lu, D. T. Hoang, D. Niyato, L. Shu, D. I. Kim, and Y.-C. Liang, "Toward smart wireless communications via intelligent reflecting surfaces: A contemporary survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 4, pp. 2283–2314, 2020.

[10] C. Liaskos, A. Tsioliaridou, K. Georgopoulos, I. Morianos, S. Ioannidis, I. Salem, D. Manessis, S. Schmid, D. Tyrovolas, S. A. Tegos *et al.*, "Xr-rf imaging enabled by software-defined metasurfaces and machine learning: Foundational vision, technologies and challenges," *IEEE Access*, vol. 10, pp. 119 841–119 862, 2022.

[11] M. Segata, P. Casari, M. Lestas, A. Papadopoulos, D. Tyrovolas, T. Saeed, G. Karagiannidis, and C. Liaskos, "Cooperis: A framework for the simulation of reconfigurable intelligent surfaces in cooperative driving environments," *Computer Networks*, vol. 248, p. 110443, 2024.