Interactive Evolutionary Multiobjective Optimization with Modular Physical User Interface

Atanu Mazumdar
University of Jyvaskyla, Faculty of Information Technology, P.O. Box 35 (Agora), FI-40014 University of Jyvaskyla, Finland
atanu.a.mazumdar@jyu.fi

Stefan Otayagich
University of Jyvaskyla, Faculty of Information Technology, P.O. Box 35 (Agora), FI-40014 University of Jyvaskyla, Finland
stefan.s.otayagich@jyu.fi

Kaisa Miettinen
University of Jyvaskyla, Faculty of Information Technology, P.O. Box 35 (Agora), FI-40014 University of Jyvaskyla, Finland
kaisa.miettinen@jyu.fi

ABSTRACT

Incorporating the preferences of a domain expert, a decision-maker (DM), in solving multiobjective optimization problems increased in popularity in recent years. The DM can choose to use different types of preferences depending on his/her comfort, requirements, or the problem being solved. Most papers, where preference-based and interactive algorithms have been proposed, do not pay attention to the user interfaces and input devices. If they do, they use character or graphics-based preference input methods. We propose the option of using a physical or tactile input device that gives the DM a better sense of control over providing his/her preferences. However, off the shelf hardware devices are not tailored to solve multiobjective optimization problems and provide many controls that may increase the cognitive load on the DM. In this paper, we propose a fully modular physical user interface to input preference information for solving multiobjective optimization problems. The modularity allows to arrange each input module in various ways depending on the algorithm, DM's requirements, or the problem being solved. The device can be used with any computer and uses web-based visualizations. We demonstrate the potential of the physical interface by solving a real-world problem with an interactive decomposition-based multiobjective evolutionary algorithm.

KEYWORDS

preference information, multicriteria decision making, decision support, decomposition-based MOEA, human machine interface, tactile interface

ACM Reference Format:
Atanu Mazumdar, Stefan Otayagich, and Kaisa Miettinen. 2022. Interactive Evolutionary Multiobjective Optimization with Modular Physical User Interface. In *Genetic and Evolutionary Computation Conference Companion (GECCO '22 Companion), July 9–13, 2022, Boston, MA, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3520304.3534008

GECCO '22 Companion, July 9–13, 2022, Boston, MA, USA

ACM ISBN 978-1-4503-9268-6/22/07…$15.00
https://doi.org/10.1145/3520304.3534008

1 INTRODUCTION

Multiobjective optimization problems (MOPs) consist of $K \geq 2$ conflicting objective functions that are required to be optimized simultaneously. An MOP can be defined as follows:

$$\begin{aligned} &\text{minimize } \{f_1(\mathbf{x}), \ldots, f_K(\mathbf{x})\}, \\ &\text{subject to } \mathbf{x} \in \Omega, \end{aligned} \tag{1}$$

where Ω is the feasible region of the decision space $\mathbb{R}^n$. The corresponding objective vector for a feasible decision vector $\mathbf{x}$ is $\mathbf{f}(\mathbf{x})$, and consists of the objective (function) values $(f_1(\mathbf{x}), \ldots, f_K(\mathbf{x}))$.

We have a plethora of algorithms to solve MOPs [15, 22], among them multiobjective evolutionary algorithms (MOEAs). MOEAs approximate a set of optimal solutions known as the Pareto front that represents the trade-offs between the objectives. In recent years, MOEAs have become quite popular since they are capable of handling MOPs with non-convex and disconnected Pareto fronts [25, 27]. The decision-maker (DM) who is the domain expert in the MOP being solved is generally interested in a subset of solutions (or a single solution) based on his/her expressed preferences [15]. There have been many developments in MOEAs to enable the DM to provide preferences in an a priori and interactive way to produce solutions in the region that is interesting to the DM [15, 20]. Solving MOPs with interactive algorithms is advantageous primarily because of two reasons. Firstly, it reduces the computation cost of solving expensive MOPs by evaluating fewer solutions. Secondly, it allows the DM to focus only on certain regions of the objective space, thereby reducing his/ her cognitive load [15].

There has been a significant amount of work on the development of various interactive algorithms, e.g. [12, 13, 26]. These approaches use different ways for the DM to express his/her preferences, i.e., reference point [8, 21], pairwise comparison [2], preferred/ non-preferred solutions [6, 10, 21], and preferred ranges [12]. Furthermore, certain interactive MOEAs also have preferences for uncertainty along with preferences for objectives [14]. Such a wide variety of preferences are available so that the DM can use an appropriate type that makes him/her comfortable while solving the MOP interactively. As concluded in the survey [1], previous works have not given enough attention to user interfaces, the way the DM provides the preferences, or which preference will be advantageous for the DM. To our knowledge, the past works on user interfaces were primarily focused on visualization techniques and not on input devices in the context of multiobjective optimization and decision making.

The previous interactive approaches used generic interfaces, i.e., keyboard, mouse, or touchscreen, to input the DM's preferences.

Though these user interfaces are versatile, they are not tailored to solve MOPs interactively. The advantage of using a tactile and tangible user interface for solving MOPs is not well covered to our knowledge. However, the works in [4, 24] have shown that users are more comfortable using a tactile or physical input device rather than a touchscreen or generic input devices like keyboard and mouse. These studies showed an improved concentration and speed in user input over generic interfaces. Thus, a physical user interface can give the DM a better feeling of control in providing preferences and directing the solution process. Feeling in control is a desirable property in interactive algorithms, as noted in a recent study [1]. A physical user interface also has applications in real-time interactive optimization (e.g., power station control, stock markets, etc.), where the concentration and speed of a DM in specifying his/her preferences are crucial.

One may choose to use off the shelf hardware, i.e. MIDI controllers, as a physical interface, and can be used with minimal development. However, these controllers are not tailored to provide the preferences we are dealing with while solving MOPs. Also, these controllers offer more (or less) controls to the DM that again is detrimental to the DM's needs, comfort and concentration. A physical interface that can provide only the necessary controls to the DM would be ideal. A possible way would be to develop physical interfaces for different interactive algorithms, types of preferences and number of objectives (of the MOP). However, such a physical interface will not be scalable or economically viable. A better way to tackle this challenge is to develop a physical interface that is highly modular. A modular design has already shown benefits in various research and design areas [9, 11]. A modular physical interface that can be arranged as 'Lego blocks' by the DM depending on the MOP being solved, type of preference information, his/her comfort and ergonomics would be a perfect solution. Such a design will make the DM more interested in solving the MOP and give him/her a sense of complete control over the preferences provided.

In this paper, we propose a modular physical interface that is highly flexible and can be arranged in a wide variety of combinations depending on the needs of the DM. The physical interface can be connected to any computer via USB and used to solve MOPs with a wide variety of preference-based MOEAs. We also propose a novel approach that can automatically detect the arrangement of the modules in the physical interface for visualization. This enables the DM to map the preferences to the physical arrangement easily. As a proof-of-concept, we demonstrate the potential of the physical interface with the open-source software framework DESDEO [17] by solving the river pollution problem [19] with an interactive version of RVEA [5, 12]. For simplicity, we use reference point type preference. However, the device can also be used with other multiobjective optimization algorithms and is capable of handling other types of preferences. It should be noted that the novel contribution of the paper is in decision making and multiobjective optimization and not in electronics or embedded systems.

The rest of the paper is structured as follows. In Section 2 we describe the design of the modules of the physical interface, the communication protocols used and the proposed automatic arrangement detection algorithm. In Section 3 we demonstrate the physical interface by solving an MOP interactively. Finally, in Section 4 we conclude our results and propose possible future research directions.

2 MODULAR PHYSICAL INTERFACE

We start by outlining requirements for physical interface development. For wider applicability, we develop the physical interface to handle different types of preferences, i.e. reference point, desired ranges, selecting preferred and non-preferred solutions. These preferences can be provided in the continuous or discrete scale. Also, it should be able to handle preferences for uncertainty if necessary. If the DM wants to change his/her preference type in the following interaction [12], the interface should be able to cope with that. As previously mentioned in the introduction, such a flexible physical interface can be achieved by constructing it with a modular design.

The modules are the building blocks of the physical interface, and it should not be difficult for the DM to arrange them according to the requirements. The designed modules should mesh together seamlessly, and the visualization should reflect the physical arrangement of the modules. All these requirements are aimed to make the DM's user experience more comfortable. Next, we discuss the steps taken in designing the modules and the automatic arrangement detection of the modules.

2.1 Design of the Modules

The physical interface consists of individual building blocks called modules. In other words, the overall arrangement of modules when they are connected with one another is called the physical interface. Each module has one or more *hardware peripherals* embedded in them that can be used for providing inputs (or outputs) to the DM. Next, we describe the design of the modules and the various hardware peripherals used.

Our primary criterion for the modular physical interface has been to have a Lego block type design to have a seamless connection between the module. We chose a rectangular design for each module so that they can be stacked together in a wide variety of arrangements. We used a five-pin connector on all four sides of the modules to connect them in an arrangement. The connection between the modules is for supplying power to the module, exchanging data and sensing directions of the other modules connected. Sensing directions is an important feature that is used to find the arrangement of the connected modules (explained in Section 2.3).

For enabling the DM to use an interactive algorithm, we needed tactile hardware that can be used to input preferences and embedded in the modules. We chose three types of hardware peripherals for preferences inputs. They were a) *rotary encoders*, b) *faders* or linear potentiometers and c) *push switches*. These hardware peripherals have different advantages over one another. The rotary encoders are 'endless', or they are not limited by the range of motion. Hence, they can be used for changing the preferences with any resolution (or the step size of change can be adjusted). The encoders are also suitable for integer type preference input. However, since the encoders do not have an absolute position and can rotate endlessly (360+ degrees), the DM is unable to get a good sense of the preferences by observing the position of the encoders. We resolved this problem by equipping the modules with faders. Faders have a fixed range of movement and provide visual (and touch) feedback to the DM

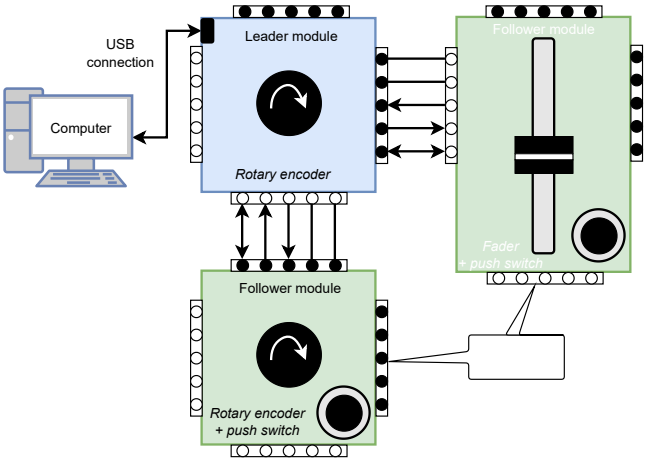

Figure 1: Three modules connected in a hypothetical arrangement (for demonstration purposes). The overall design of the modules with different hardware peripherals is also shown.

about the absolute value of the preferences based on their position. The DM can also realize the previous preferences he/she had set just by looking at the position of the faders. In addition to that, we selected push switches that can be used to start an iteration of the MOEA or stop the optimization process when the DM is satisfied with the solutions.

In Figure 1, we show the design of a few modules connected in a hypothetical arrangement. The modules consist of rotary encoders and faders, and two of them are equipped with push switches. The figure also shows the five-pin connector, connecting the modules to one another. The modules also include a status light-emitting diode (LED) to show the connection status of the module. Every variant of the modules (with different types of hardware) is assigned a module type ID. This was done to distinguish between the various hardware peripherals available in a module.

The brain of each module was the ATmega32U4 microcontroller (in Arduino Pro Micro form factor) interfacing the faders, rotary encoders and switches. The major advantages of using the specific microcontroller are low cost and built-in USB peripheral (that will be utilized later for communication). We will not describe the complex electronics and interfacing in this paper. The schematics, printed circuit board design and codes are open source [1] , and the readers can refer for further details.

2.2 Communication Protocols Used

In Figure 1, we also show the connections between the modules and the computer. For connecting the modules to the computer, we used a USB 2.0 connection. In our design, all the modules are equipped with a USB port. The module to which we connect the USB (to the computer) becomes the *leader module* (also known as master) and is shown in blue. The rest of the modules are called *follower modules* (also known as slave) and is shown in green.

[1]GitHub repository for the physical interface can be found at https://github.com/industrial-optimization-group/DesdeoInterface

The leader module performs two tasks: a) collecting all the input data from all the modules and sending it to the computer over USB and b) managing the arrangement of the modules. The leader module is capable of sending data as serial transmission (UART) or by webUSB [3, 7, 23]. In the demonstration shown in Section 3.2 we have used webUSB to take advantage of the web browser-based visualizations during the interaction process.

We used the Padded Jittering Operative Network (PJON) [18], an open-source software-defined network protocol to establish the communication between the leader module and the follower modules. PJON supports various types of strategies for various media, i.e. Ethernet, serial, etc. We used a 'software bitbang' strategy as it does not require any additional hardware and relies on a simple wire connection between the modules. The current implementation can support a maximum of 256 modules.

2.3 Automatic Arrangement Detection of the Modules

When the modules are connected in a certain arrangement, the representation of the physical interface should be visible on display (monitor of the computer). This would allow the DM to map the different components on the modules to the inputs of the interactive algorithm being used.

The concept of finding the arrangement of the modules can be explained in simple words as 'finding people in the dark'. Suppose you want to find your friend in the dark. You flash your torch, and a few moments later, your friend flashes his torch back at you. If you do not receive a flash within a certain amount of time, you can conclude that your friend is lost. We implemented a similar strategy to find the arrangement of the modules by utilizing the microcontroller's digital input/output pins in all four directions of the module. In digital electronics, 'pulling a pin high' and 'pulling a pin low' in the microcontroller is analogous to flashing your torch and turning it off, respectively. Similarly, 'reading a pin' is analogous to your friend seeing the flash from your torch.

When follower modules are connected, the leader module assigns a unique ID to communicate with it. The leader instructs each connected module to pull a specific direction pin low at a certain point. This way, the leader module can keep track of the taken directions and determine where the modules are situated in the arrangement. Found modules IDs are saved in a stack along with the 'next direction' they should check. The next direction is the direction in which a specific module should look for other modules connected to itself after a scan is over. Before the arrangement detection process can start, the leader module sends a broadcast message informing all the modules to set their direction pins to input mode. In other words, they are reading the pin value instead of outputting a value.

Algorithm 1 shows the arrangement detection process done by the leader module. It starts with the leader module pulling its first direction pin low. If a connected module's direction pin gets pulled low and it has not been already found, it sends a packet containing its module type ID to the leader module and starts expecting a broadcast message containing an ID for itself. When the leader module receives the packet, it adds the detected follower module's ID to the stack and sends the broadcast packet to it. Note that

Algorithm 1 Arrangement detection algorithm (leader module)

Input: Modules connected in a physical arrangement
Output: Stack containing the directions how the modules are connected
Inform all modules that the search has started
Initialize the **stack** so that it only contains self
ID ← 1
while stack is not empty **do**
  current module, **next direction** ← **stack** top
  if next direction is valid **then**
    current module pull **next direction** low
  else
    stack pop
  end if
  Wait for **confirmation**
  if confirmation received **then**
    Determine **new module** position
    Assign **ID** to **new module** and increment **ID** by one
    Push **new module** with initial direction to **stack**
  end if
  Update **next direction**
end while

every other module will ignore the broadcast packet with the ID as they are not expecting it. The arrangement detection process then continues from the newly added module. The leader module keeps track of whether a follower module has checked all the four directions. When a follower module has checked all its directions, it is removed (popped out of the stack). The arrangement detection process continues from the topmost module of the stack until all the modules in the arrangement have checked all their directions (or until the stack is empty). After the arrangement detection is complete, the leader module passes the module types, IDs, and positions of the follower module to the computer.

Assume an arrangement has four modules in a 2-by-2 grid as shown in Figure 2. If the starting direction is up and the search rotates clockwise, the modules are found in order 1, 3, 2. Moreover, the stack contains the following information just before the last module is found:

Module number	Next direction
Follower module 3	Left
Follower module 1	Left
Leader module	Down

After adding follower module 2 to the stack it is updated as follows:

Module number	Next direction
Follower module 2	Up
Follower module 3	Invalid
Follower module 1	Left
Leader module	Down

The modules are also 'plug and play'. Thus new modules can be connected or disconnected even after the algorithm has detected the arrangement. This is useful if we want to change a module and use a different type of hardware peripheral. Connecting new modules to the interface after the initial arrangement is straightforward, as every time a module powers up/connects, it informs the

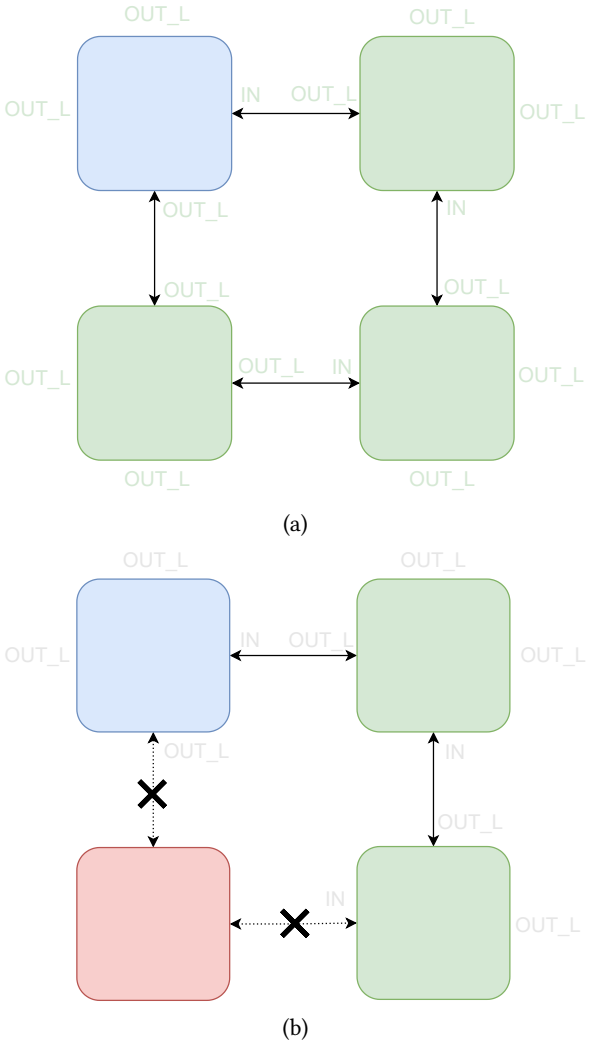

Figure 2: Four modules connected in an arrangement in the physical interface. The arrows represent the direction pins for detecting the arrangement. The direction pin for input mode is 'IN', and the output pins pulled low is 'OUT_L'. (a) The state of the physical interface after the initial arrangement. (b) The state after follower module 2 is disconnected, resulting in follower module 3's left direction pin to read high.

leader module about its connection. If the leader module receives information of a new connected module, it can simply rerun the arrangement detection algorithm and find the new module along with the previous ones. The problem with this approach is that the previous modules may be assigned new or different IDs. This would cause the computer to lose information. We solved by giving each module a module-based unique identifier that would always be the same or by reading the free direction pins to determine when and where a new module is connected.

After the initial arrangement is detected, the direction pins are responsible for monitoring for disconnected modules. Direction pins that resulted in finding a new module in the initial arrangement are set to input mode (IN), and the rest are pulled low (OUT_L). Now all the direction pins in the input mode should be reading low, and in case of a disconnection, a direction pin value will change to high, indicating a disconnection. When a module determines a disconnection, it informs about it to the leader module by sending its ID and the disconnection direction. This way, the leader module can easily identify which module was disconnected.

3 SOLVING MOPS INTERACTIVELY USING THE PHYSICAL USER INTERFACE

The proposed modular physical interface was designed primarily to solve MOPs with an interactive algorithm. Hence, we solved a river pollution problem [19] to demonstrate the potential of the proposed interface for solving MOPs interactively. The problem considers the water quality of a river being polluted by municipal waste from a city and industrial waste from a city. The water quality is measured in dissolved oxygen concentration (DO) and pounds of biochemical oxygen demanding material (BOD). The city and the fishery have one water treatment plant to remove BOD. The optimization problem aims to enhance overall water quality at both the city and state levels and simultaneously reduce costs.

The problem consists of four objectives and two decision variables. The objectives and the bounds of the decision variables are as follows:

$$\text{minimize } f_1(\mathbf{x}) = -(4.07 + 2.27x_1)$$

$$\text{minimize } f_2(\mathbf{x}) = -\left(2.60 + 0.03x_1 + 0.02x_2 + \frac{0.01}{1.39 - x_1^2} + \frac{0.30}{1.39 - x_2^2}\right)$$

$$\text{minimize } f_3(\mathbf{x}) = -\left(8.21 - \frac{0.71}{1.09 - x_1^2}\right)$$

$$\text{minimize } f_4(\mathbf{x}) = -0.96 + \frac{0.96}{1.09 - x_1^2}$$

$$\text{subject to } 0.3 \leq x_1, x_2 \leq 1.0.$$

The objectives are: a) f_1 is the negative DO level at the city, b) f_2 is the negative DO level at the state boundary, c) f_3 is the negative of the return on investment (ROI) at the fishery, and d) f_4 represents the decrease of the tax rate in the city due to treatment. The decision variables x_1 and x_2 represents proportionate amounts of BOD removed from water in the two treatment plants. Since the authors are not domain experts in this MOP, the preferences provided are only for the purpose of demonstration.

We used an interactive version of RVEA [5] as proposed in [12] to solve the MOP for demonstration purposes. It adapts the reference vectors in a manner to produce solutions according to the preferences of the DM. The spread parameter r was set to 0.1 for the purpose of demonstration and can be changed depending on DM's requirements.

Though the algorithm is capable of handling different types of preferences, we decided to use reference points as our preference type for the purpose of demonstration. The reference point is widely used in interactive MOEAs and represented on a continuous scale in the physical interface inputs. However, other preference types in discrete scales (e.g., preferred and non-preferred solutions) can also be provided through the rotary encoders. The modules of the physical user interface should be arranged depending on the type of preference for better representation and comfort of the DM.

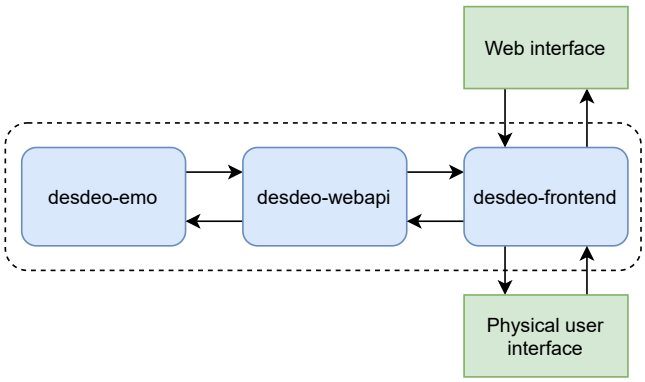

Figure 3: Structure of the physical interface with the DESDEO framework.

The MOEA was run for 100 generations at each iteration. The parameter settings for the number of reference vectors, crossover and mutation were set as suggested in [5]. Next, we discuss how the interactive algorithm was integrated with the proposed physical interface.

3.1 Interfacing with DESDEO

The physical interface can be used with any algorithm, framework, software etc., that can access the USB (interfaced as serial port) of the computer. For our tests and demonstration, we used the open-source framework DESDEO [17] that contains various decomposition-based MOEAs, visualization tools and benchmark problems. The interactive version of RVEA was already available in the DESDEO framework and hence was a suitable choice. However, the most attractive feature provided in DESDEO is the web-based interface for visualization and solving MOPs interactively. This was useful since we could utilize the experimental webUSB [3, 7, 23] technology supported by browsers such as Chrome, Edge and Opera. The webUSB technology allows the browser to access the microcontroller on the physical interface over USB.

In Figure 3 we show the interfacing between the various components of the DESDEO [17] framework and the proposed physical interface. The DESDEO framework can be broadly separated into frontend and backend. The frontend is the presentation layer that includes the visualization tools and the physical user interface. The backend consists of running the MOEA and other background tasks. The interactive RVEA [5, 12] classes are in the desdeo-emo package. The desdeo-emo package is the backend of our application and runs the optimization algorithm. Access to the desdeo-emo packages is provided by the desdeo-webapi package, a web API (application programming interface) based on Flask. The desdeo-frontend package provides the web-based graphical user interface (GUI) and provides the DM with a problem definition tool, visualization tools

and interaction. The desdeo-webapi block acts as the bridge between the frontend and backend, as shown in Figure 3. The dotted box represents the full stack (including the frontend and backend packages) of the DESDEO framework that we will be utilizing in this paper. As the proposed physical interface is a frontend device, interfacing it via USB was implemented in desdeo-frontend.

Our implementation in desdeo-frontend [2] performs two tasks. Firstly, it interfaces the physical interface and creates a visual layout of the physical arrangement of the modules. This allows the DM to easily map between the physical inputs and the preferences he/she wants to assign to each of the modules. Secondly, it sends the updated data when the DM actually provides his/her preferences or other inputs to the relevant fields. This can be better understood from the demonstration of using the physical interface to solve an MOP interactively.

3.2 Demonstration

The first step of using the physical interface is choosing the module type for providing the components of the reference point for each objective. As discussed in Section 2, we have two types of modules primarily consisting of a fader and a rotary encoder. We also have a momentary switch (push to on and release to off) in addition to the fader and rotary encoder that can be used for starting an iteration or stopping the optimization process. As the river pollution problem consists of four objectives, and the preference type we are demonstrating is a reference point, we need four modules. We chose four variants of the modules and connected them in an arrangement to construct the physical interface as shown in Figure 5. Then the physical interface is connected to the USB port of the computer.

The user then starts the desdeo-webapi and desdeo-frontend server and opens Chrome (or other supported) browser on the computer. After going through the login process and selecting the river pollution problem and interactive RVEA as the optimization algorithm, the user is presented with the visualization shown in Figure 4. Next, the user presses the 'Connect' button on the visualization and chooses the connected USB device from a list. After choosing the physical interface device, the automatic arrangement detection process is started. The exact arrangement of the modules in the physical interface setup is then displayed in the bottom part of the visualization (as shown in Figure 4). The visual representation of the arrangement enabled the user to feel more connected to the actual physical layout of the modules. It should be noted that the leader and follower modules 1–3 are referred to as Master and Node 1–3, respectively, in the visualization.

The user then assigns each of the inputs of the modules (selected from the drop-down list within each module) to the fields (objectives, iterate, stop etc.) available. For the purpose of demonstration, we assigned the objectives 'WQ City' and 'WQ Border' to the rotary encoders and 'ROI City' and 'Tax City' to the faders. Any changes made in the physical interface is now reflected on the relevant fields of objective values shown on the left side of Figure 4. The minimum and maximum values of the reference point that the DM can provide are also displayed. The switch (circle without an arrow) on the first module (Master) is assigned to start the iteration. The switch one on the last module (Node 3) is assigned to stop the optimization process. Now the preparations are ready by the user, and the DM can start solving the problem by providing preferences using the physical interface.

The DM inputs the preferences by moving the rotary encoders (on Master and Node 1) and faders (on Node 2 and 3) to adjust components of the reference point for the objectives. After pressing the iterate switch (on the Master module), the DM is presented with a set of solutions visualized on the parallel coordinate plot (as shown in Figure 4). The solutions are indicated in green, whereas the reference point is shown in red. The DM repeats the interaction process by providing the component of the reference point individually for each objective with the physical interface. He/she repeats this step as many times as needed to find a set of solutions that he/she is satisfied with. In the end, the DM selects a solution as the final decision from the set and the optimization process is stopped by pressing the stop switch (on Node 3).

From the perspective of the DM, the physical user interface proves to be an improved way of providing the preferences. The modular nature of the interface keeps unnecessary clutter away for the DM while providing preferences compared to generic input devices like a keyboard or mouse. This enabled the DM to concentrate more on providing preferences. The visualization showed the physical arrangement of the modules and can be assigned for different preference inputs is so desired.

It has been observed that DMs often have different phases during the interactive solution process [16]. In the so-called learning phase, the DM learns about different trade-offs among the objectives to find a region of interest in the Pareto front. This may involve changing the preferences even drastically. The physical interface enables the DM to change the preferences rapidly to a new location in the objective space. This increases the speed of interaction compared to using generic input devices. In the so-called decision phase, the DM fine-tunes the solutions in the region of interest to find the most preferred solution. Fine adjustments are convenient to do using the rotary encoders to reach a final, most preferred solution.

4 CONCLUSIONS

In this paper, we have proposed a novel way for the DM to input preferences by using a modular physical user interface when solving MOPs with interactive multiobjective optimization algorithms. The user interface is capable of handling different types of preference inputs and can be used with different types of interactive multiobjective optimization algorithms. We developed modules that can be connected to each other in arrangements that the DM is comfortable using. We also proposed a novel algorithm that can automatically detect the physical arrangement of the modules. This is used to create a visual representation of the physical interface and further improve the DM's user experience.

As a proof-of-concept, we demonstrated that the physical user interface is beneficial in the context of solving MOPs with preferences. We showed the potential of the physical interface by solving the river pollution problem with interactive RVEA utilizing the open DESDEO framework. Overall, the physical interface opened up a new avenue in decision support by providing the DM with an alternative way of providing preferences. The modularity of

[2] GitHub repository for the physical interface implemented in desdeo-frontend can be found at https://github.com/industrial-optimization-group/physical-interface-v1.1

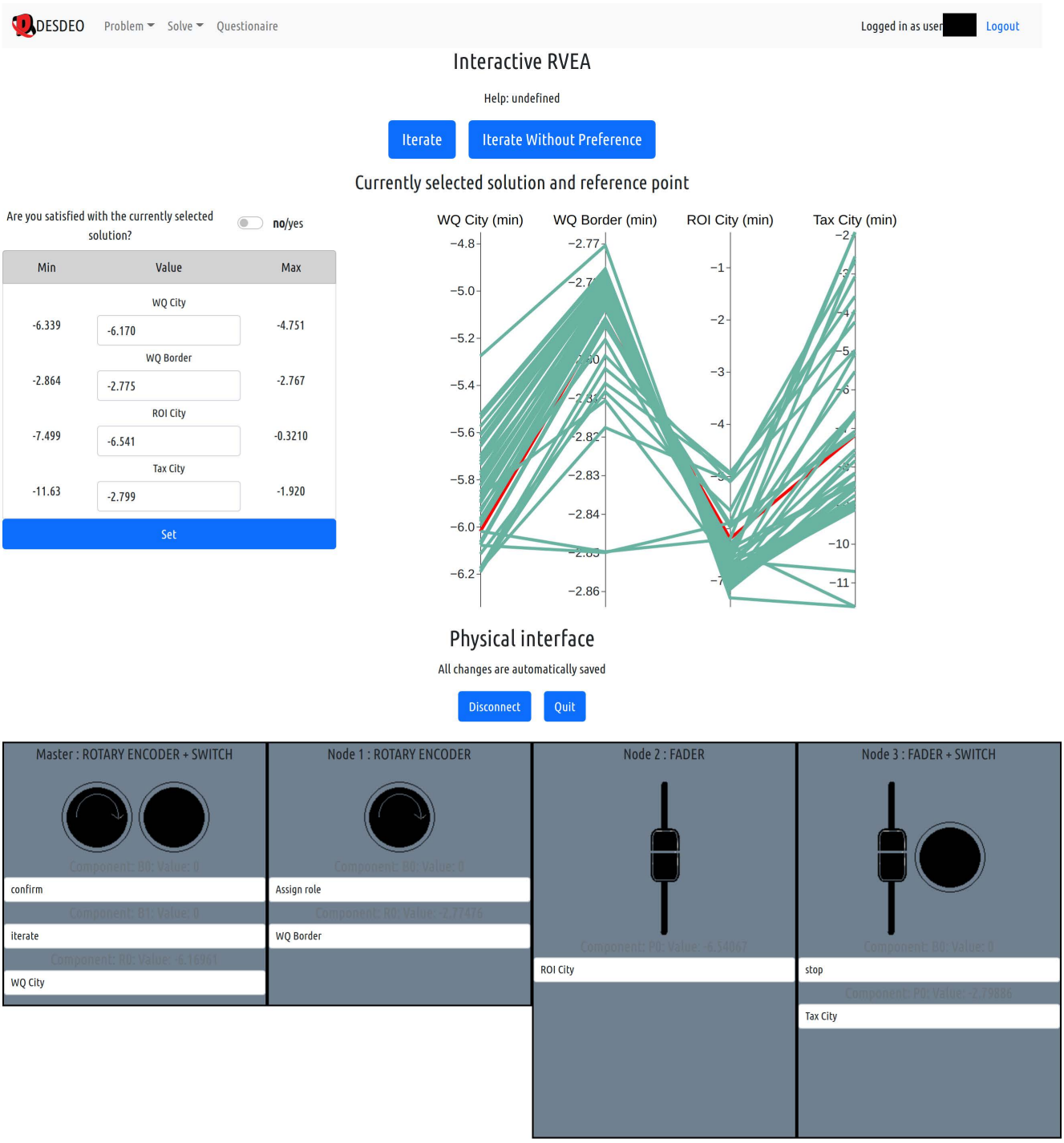

Figure 4: Solving a problem with interactive RVEA using the modular physical interface. The figure shows the values of the preferences in terms of components of a reference point, parallel coordinate plot showing the solutions found (and reference point in red), detected arrangement of the modules in the physical interface.

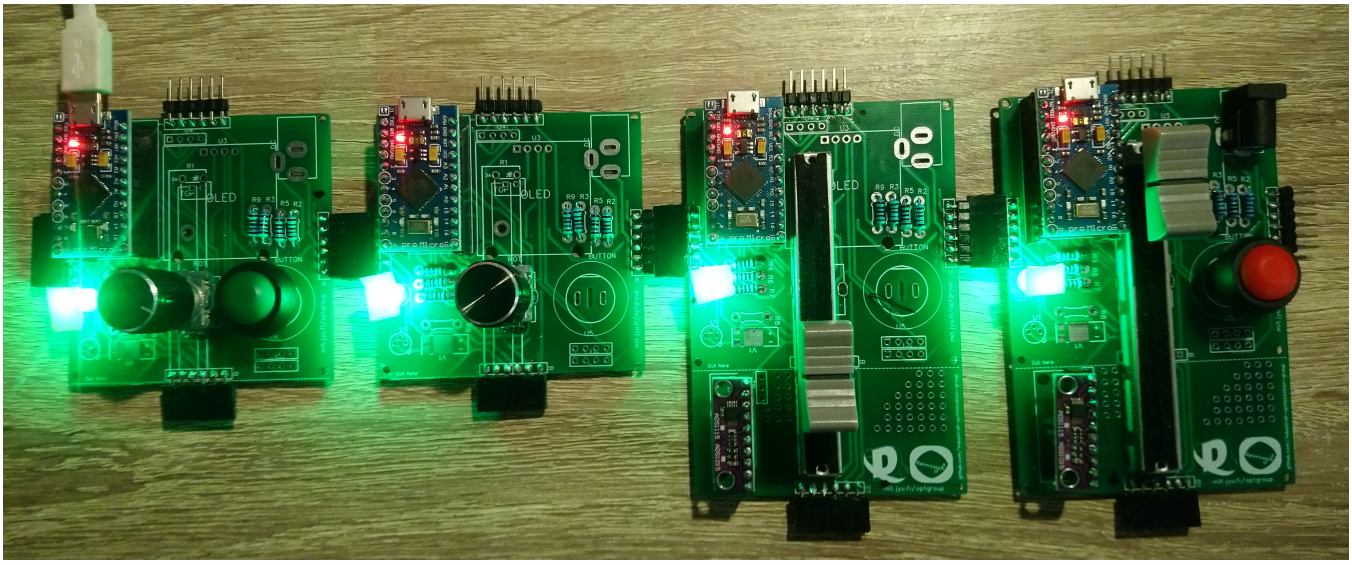

Figure 5: Real arrangement of four different modules connected together to setup the physical interface.

the physical interface and the respective visualizations enabled a greater amount of flexibility and a more comfortable user experience. The proposed modular physical interface achieved its goal of solving an MOP by providing preference information in a way that gives a feeling of control over the typical user interfaces.

The development of the modular physical interface created the foundation for further research on improving the decision making process. In our future research, we plan to conduct more experiments comparing different types of user interfaces. We shall test the ergonomics, input speed, time consumed to reach the most preferred solution, and overall user experience. Tests are also to be conducted with various other types of preferences, interactive optimization methods and solving complex MOPs with a higher number of objectives. The design of the modules can be further improved with magnetic pogo pin connectors, displays, and haptic feedback to further enhance the DM's experience. In future, we also plan to perform tests on the robustness of the network of modules and utilize other networking protocols, e.g., USB, Bluetooth, MODBUS, etc.

ACKNOWLEDGMENTS

This research is related to the thematic research area DEMO (Decision Analytics utilizing Causal Models and Multiobjective Optimization, jyu.fi/demo) of the University of Jyväskylä.

REFERENCES

[1] B. Afsar, K. Miettinen, and F. Ruiz. 2021. Assessing the Performance of Interactive Multiobjective Optimization Methods: A Survey. *ACM Computing Surveys* 54 (2021), article 85.

[2] R. Battiti and A. Passerini. 2010. Brain–Computer Evolutionary Multiobjective Optimization: A Genetic Algorithm Adapting to the Decision Maker. *IEEE Transactions on Evolutionary Computation* 14 (2010), 671–687.

[3] F. Beaufort. 2016. Access USB Devices on the Web. (2016). "https://gweb.dev/usb/" Updated Feb 23, 2021.

[4] L. Besançon, P. Issartel, M. Ammi, and T. Isenberg. 2017. Mouse, Tactile, and Tangible Input for 3D Manipulation. *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (2017), 4727–4740.

[5] R. Cheng, Y. Jin, M. Olhofer, and B. Sendhoff. 2016. A Reference Vector Guided Evolutionary Algorithm for Many-Objective Optimization. *IEEE Transactions on Evolutionary Computation* 20 (2016), 773–791.

[6] T. Chugh, K. Sindhya, J. Hakanen, and K. Miettinen. 2015. An Interactive Simple Indicator-Based Evolutionary Algorithm (I-SIBEA) for Multiobjective Optimization Problems. In *Evolutionary Multi-Criterion Optimization, Proceedings*, A. Gaspar-Cunha, C. Henggeler Antunes, and C. C. Coello (Eds.). Springer, 277–291.

[7] Many contributors. 2016. WebUSB Arduino. (2016). "https://github.com/webusb/arduino" Accessed February 1, 2022.

[8] K. Deb and S. Chaudhuri. 2005. I-EMO: An Interactive Evolutionary Multi-Objective Optimization Tool *(PReMI'05)*. Springer, 690–695.

[9] W. Ferdous, Y. Bai, T. D. Ngo, A. Manalo, and P. Mendis. 2019. New Advancements, Challenges and Opportunities of Multi-Storey Modular Buildings – A State-of-the-Art Review. *Engineering Structures* 183 (2019), 883–893.

[10] M. Gong, F. Liu, W. Zhang, L. Jiao, and Q. Zhang. 2011. Interactive MOEA/D for Multi-Objective Decision Making *(GECCO '11)*. Association for Computing Machinery, 721–728.

[11] A. Gwiazda, W. Banas, A. Sekala, K. Foit, P. Hryniewicz, and G. Kost. 2015. Modular industrial robots as the tool of process automation in robotized manufacturing cells. *IOP Conference Series: Materials Science and Engineering* 95 (2015), 012104.

[12] J. Hakanen, T. Chugh, K. Sindhya, Y. Jin, and K. Miettinen. 2016. Connections of Reference Vectors and Different Types of Preference Information in Interactive Multiobjective Evolutionary Algorithms. In *Proceedings of the 2016 IEEE Symposium Series on Computational Intelligence (SSCI)*. 1–8.

[13] K. Li, R. Chen, G. Min, and X. Yao. 2018. Integration of Preferences in Decomposition Multiobjective Optimization. *IEEE Transactions on Cybernetics* 48 (2018), 3359–3370.

[14] A. Mazumdar, T. Chugh, J. Hakanen, and K. Miettinen. 2020. An Interactive Framework for Offline Data-Driven Multiobjective Optimization. In *Bioinspired Optimization Methods and Their Applications, Proceedings*, B. Filipič, E. Minisci, and M. Vasile (Eds.). Springer, 97–109.

[15] K. Miettinen. 1999. *Nonlinear Multiobjective Optimization*. Kluwer Academic Publishers.

[16] K. Miettinen, F. Ruiz, and A.P. Wierzbicki. 2008. Introduction to Multiobjective Optimization: Interactive Approaches. In *Multiobjective Optimization: Interactive and Evolutionary Approaches*, J. Branke, K. Deb, K. Miettinen, and R. Slowinski (Eds.). Springer, 27–57.

[17] G. Misitano, B. S. Saini, B. Afsar, B. Shavazipour, and K. Miettinen. 2021. DESDEO: The Modular and Open Source Framework for Interactive Multiobjective Optimization. *IEEE Access* 9 (2021), 148277–148295.

[18] G. B. Mitolo. 2020. Padded Jittering Operative Network. (2020). "https://github.com/gioblu/PJON" Accessed February 1, 2022.

[19] S.C. Narula and H.R. Weistroffer. 1989. A Flexible Method for Nonlinear Multicriteria Decision-Making Problems. *IEEE Transactions on Systems, Man, and Cybernetics* 19 (1989), 883–887.

[20] R. C. Purshouse, K. Deb, M. M. Mansor, S. Mostaghim, and R. Wang. 2014. A Review of Hybrid Evolutionary Multiple Criteria Decision Making Methods. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*. 1147–1154.

[21] A. B. Ruiz, M. Luque, K. Miettinen, and R. Saborido. 2015. An Interactive Evolutionary Multiobjective Optimization Method: Interactive WASF-GA. In *Evolutionary Multi-Criterion Optimization, Proceedings*, A. Gaspar-Cunha, C. Henggeler Antunes, and C. C. Coello (Eds.). Springer, 249–263.

[22] N. Saini and S. Saha. 2021. Multi-Objective Optimization Techniques: a Survey of the State-of-the-Art and Applications. *The European Physical Journal Special Topics* 230 (2021), 2319–2335.

[23] A. K. Talukder. 2020. The Next Generation Web: Technologies and Services. In *Big Data Analytics*, L. Bellatreche, V. Goyal, H. Fujita, A. Mondal, and P. K. Reddy (Eds.). Springer, 209–229.

[24] P. Tuddenham, D. Kirk, and S. Izadi. 2010. Graspables Revisited: Multi-Touch vs. Tangible Input for Tabletop Displays in Acquisition and Manipulation Tasks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '10)*. ACM, 2223–2232.

[25] Q. Xu, Z. Xu, and T. Ma. 2020. A Survey of Multiobjective Evolutionary Algorithms Based on Decomposition: Variants, Challenges and Future Directions. *IEEE Access* 8 (2020), 41588–41614.

[26] J. Zheng, G. Yu, Q. Zhu, X. Li, and J. Zou. 2017. On Decomposition Methods in Interactive User-preference based Optimization. *Applied Soft Computing* 52 (2017), 952–973.

[27] A. Zhou, B.-Y. Qu, H. Li, S.-Z. Zhao, P. N. Suganthan, and Q. Zhang. 2011. Multiobjective Evolutionary Algorithms: A Survey of the State of the Art. *Swarm and Evolutionary Computation* 1 (2011), 32–49.