Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 176 (2020) 281–290

Procedia Computer Science

www.elsevier.com/locate/procedia

24th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems

Data-driven Interactive Multiobjective Optimization: Challenges and a Generic Multi-agent Architecture

Bekir Afsar[a,*], Dmitry Podkopaev[b], Kaisa Miettinen[a]

[a] *University of Jyvaskyla, Faculty of Information Technology, P.O. Box 35 (Agora), FI-40014 University of Jyvaskyla, Finland*
[b] *Systems Research Institute, Polish Academy of Sciences, Newelska 6, 01-447 Warsaw, Poland*

Abstract

In many decision making problems, a decision maker needs computer support in finding a good compromise between multiple conflicting objectives that need to be optimized simultaneously. Interactive multiobjective optimization methods have a lot of potential for solving such problems. However, the growth of complexity in problem formulations and the abundance of data bring new challenges to be addressed by decision makers and method developers. On the other hand, advances in the field of artificial intelligence provide opportunities in this respect.

We identify challenges and propose directions of addressing them in interactive multiobjective optimization methods with the help of multiple intelligent agents. We describe a generic architecture of enhancing interactive methods with specialized agents to enable more efficient and reliable solution processes and better support for decision makers.

Keywords: Multiple criteria optimization; interactive methods; decision support; data-driven decision making; computational intelligence; agents; multi-agent systems.

1. Introduction

Many decision making problems arising in business, engineering and other fields of human activity (see, e.g., [12, 45, 49]) can be expressed as multiobjective optimization (MOO) problems and solved using mathematical methods and computer techniques (see, e.g., [18, 26, 39, 44]). A typical MOO problem formulation includes the following three elements: *a decision maker* (DM), a person or a stakeholder who needs to find a solution serving their interests in the best possible way; *a set of feasible solutions* defined by constraint functions; and *multiple (conflicting) objective functions* defined on the set of feasible solutions as characteristics the DM wants to optimize (maximize or minimize).

* Corresponding author. Tel.: +358 404 809 214
E-mail address: bekir.b.afsar@jyu.fi

1877-0509
10.1016/j.procs.2020.08.030

In an ideal case, the best solution for the DM would be optimal with respect to all the objectives simultaneously. Because of the conflict among the objectives, the existence of such a solution in practice is very rare. Instead, we can identify so-called *Pareto optimal solutions* [26, 39, 44]. A feasible solution is called Pareto optimal, if it is not *dominated* by any other feasible solution, i.e. none of the objectives can be improved without impairment in any others. It is justifiable to limit the choice to only such solutions, since they represent the variety of trade-offs or compromises between different objectives. In order to identify the specific Pareto optimal solution which is best for a DM, additional information is needed. This information is called *DM's preference information* and the chosen Pareto optimal solution is called *the most preferred solution*. By solving the MOO problem we mean finding the most preferred solution.

Solution methods can be divided in three classes based on when preference information is incorporated in the solution process [18, 26]. In *a priori* methods, the DM first expresses preference information and a solution reflecting it is sought for. *A posteriori* methods first derive a finite representation of Pareto optimal solutions and then the DM must choose the most preferred one on their own. In *interactive* methods, the DM expresses preferences iteratively and in that way directs the solution process and gets corresponding Pareto optimal solutions as feedback.

Interactive methods allow the DM to correct mistakes arising from incomplete knowledge, and learn about the trade-offs between objectives as well as the feasibility of one's preferences. Interactive methods involve less cognitive load on the DM and less computational load by focusing only on those solutions, which correspond to expressed preferences. Because the DM can learn during the interactive solution process about the problem and what is achievable, this contributes to the quality of solution [5].

Various interactive methods have been proposed (see, e.g., references in [27, 30]) and the environment where they are applied has drastically changed over time. The abundance of data and the progress in information technologies enables modeling complex systems at unprecedented scales and levels of detail. DMs dealing with data-intensive models can be easily overwhelmed if not supported, and addressing new challenges requires novel tools.

Tools of artificial intelligence provide opportunities that have not been explored much in addressing the challenges mentioned. Among them are *multi-agent systems* (MASs). An agent is a software or hardware module that receives perceptions from its environment and acts respectively. Software agents can perform autonomous actions in parallel and independently of other parts of the system, and give responses to changes in their environment [37]. A MAS is defined by a set of agents. Each agent in a MAS has a particular goal to achieve, can communicate with others directly or indirectly, and take actions according to the perceptions and interactions with the environment and other agents [14, 51].

The concept of a MAS has been applied in MOO in some cases but mainly limited to a posteriori methods like population-based heuristics. In some studies, agents represent individual solutions in a population and are managed in intelligent ways mimicking evolution (see, e.g. [13, 19, 20, 41]). On the other hand, agents can be used as optimizers within a MAS framework in a cooperative manner to improve the solution process. A MAS framework proposed in [10] includes optimization agents that guide a genetic algorithm for solving a multiobjective job-shop scheduling problem. A contract-net protocol is used to allocate tasks to a set of optimizers through agent negotiations that are managed by a negotiator agent. In [1], a multi-agent architecture that includes several multiobjective metaheuristic agents is proposed for solving multiobjective real-parameter optimization problems. A population of solutions is divided into subpopulations handled by individual optimization agents. This MAS architecture has been later improved by adding an assessment mechanism for evaluating the performance of each optimization agent [48]. In addition to the aforementioned studies in MOO, one can find a detailed literature review of frameworks for solving single- and MOO problems with a special focus on the use of a MAS in [40].

Besides a posteriori methods, applying a MAS in solving MOO problems has been considered only in a couple of papers. In [4], an agent-based support system is proposed for handling conflicting preferences of several DMs. As far as we know, the only approach of using agents with interactive methods is made in [35], where speeding up calculations in computationally expensive problems is mainly addressed. Surrogate agents transform a computationally demanding MOO problem into an inexpensive surrogate problem. The latter problem is used in the interactive solution process for generating approximate Pareto optimal solutions without waiting times. Preference agents learn from the DM's input and guide optimization agents and the DM towards Pareto optimal solutions of DM's interest.

In this paper, we discuss challenges that interactive methods should be able to address nowadays when one should utilize data in a versatile way to make good data-driven decisions. In this, we focus at the potential offered by modern artificial intelligence tools like MASs. To be more specific, we propose a general multi-agent architecture for interac-

tive methods. We elaborate how different agents can support in various stages of problem formulation and interactive solution processes.

The rest of this paper is organized as follows. In the next section, we present a general structure of interactive methods, describe modern challenges set on interactive methods, and provide a reasoning how introducing a MAS allows addressing those challenges. In Section 3, we describe our multi-agent architecture in detail, and outline functions of intelligent agents. Finally, we conclude in Section 4.

2. General structure and challenges of interactive methods

Many interactive MOO methods have the following general structure (see, e.g., references in [27, 30]).

Step 0. Initialization. Present to the DM some information about the problem.
Step 1. Ask the DM to specify preference information.
Step 2. Generate one or several *candidate solutions*, which are exact or approximate Pareto optimal solutions corresponding to DM's preferences e.g. by

- solving method-specific scalarized optimization problem(s) incorporating preferences;
- using a multiobjective population-based heuristic, where preference information is utilized, and selecting solution(s) from this set;
- solving scalarized optimization problem(s) formulated for a surrogate MOO problem, which is based on an approximate representation of the original Pareto optimal set.

Step 3. Show the candidate solutions to the DM. Ask the DM if one of them or previously generated candidate solutions is satisfactory as the final solution. If yes, stop; otherwise, start a new iteration and go to Step 1.

Observing DMs' behavior in real-life situations has led to often distinguishing two phases of interactive solution processes (e.g. [30]). In the *learning phase*, the DM aims at understanding the problem structure by exploring different Pareto optimal solutions to identify a region of interest. This phase can be characterized by the high variation of the preference information expressed by the DM. In the *decision phase*, the DM aims at identifying the most preferred solution by fine-tuning the search in the region of interest.

The rapid growth of scale and complexity of decision making problems tackled by MOO has brought about a higher dependence of the solution processes on data. Technological advancements allow operating larger and more complex systems, collecting and processing larger sets of data and solving more sophisticated optimization problems.

Early studies applying interactive MOO methods usually considered problems with 2-4 objectives, whereas nowadays problems with more objectives are rather common in some application domains (see e.g. [7, 24, 33]). For human cognition, there is a significant difference between considering 2D (and 3D) spaces that can be visualized easily, and spaces of higher dimensions, since the short-term memory capacity is limited (see, e.g., [11, 31, 34]).

In modern problems, objective functions can be defined by open-form expressions [36], derived from simulation data [47] or real-world data [46]. Even estimation of individual values of such objective functions may be time-consuming. Advances in heuristic methods and meta-modeling and other types of surrogates have enabled considering such complex problems in decision making contexts. Deriving candidate solutions in such problems may take a long time or even be practically impossible. With the advancement of data acquisition and processing techniques, it is possible to define parts of MOO problems through data rather than analytically. However, imperfections in data and/or statistical estimation errors may induce inaccuracies of the feasible solution set and/or objective function evaluations.

There are two changes in the real world, which can be considered as new opportunities for MOO. There is a potential of making interactive MOO methods available to DMs in the form of user-friendly decision support tools. A necessary condition for this is to make interactive methods understandable in explainable ways. In addition, the means of human-machine interaction with graphical user interfaces enable event-driven solution processes, where the DM is free to provide input by any available means at any time (thereby generating events) while the task of the machine is to react to these actions instead of computers asking questions. This allows to endow the DM with an active role in human-machine communication.

Next, we summarize challenges faced by interactive methods by the changing environment of the modern world:

1. ***Computational challenges.*** In the case of high computational complexity or data-dependence of the process of deriving candidate solutions in Step 2, the DM may encounter long or unpredictable waiting times during the solution process. On the other hand, the DM may have limited time to be spent with the method or a deadline to solve the problem. It is necessary to control delays between obtaining preference information and providing feedback to the DM.
2. ***Data uncertainty challenges.*** Data-driven problem formulations may include uncertainty caused by incomplete/missing data or an inaccurate problem model. This creates new challenges related to incorporation of uncertainty information in the solution process (see e.g. [29, 53]).
3. ***Cognitive challenges.***

   (a) In the case of many objective functions, with the accumulation of candidate solutions derived during the interactive solution process, the DM may be overwhelmed with information. The amount of communication with the DM should be adjusted in accordance with human cognitive capabilities. In addition, the MOO method has to be able to operate with incomplete preference information, when the DM is not able to fully address requests from the method.
   (b) When solving a problem with many objectives, the structure of the Pareto optimal set may be difficult to comprehend. The interactive method should be able to support the learning process, and also to handle contradictory preference information caused by incomplete knowledge of the problem by the DM.
   (c) Typically, in the literature, one interactive method is applied in each solution process. As mentioned, the needs of the DM are typically different in the learning and decision phases and it is desirable not to be limited to using a single method but to be able to switch the method and the type of preference information provided.

4. ***Accessibility challenges.*** Different DMs may be more comfortable with different preference expression styles depending on the circumstances [8, 16]. As mentioned, the form of preference information expected from the DM is specific to the interactive method used [27]. Inexperienced DMs may need assistance when selecting interactive methods and preference information types suitable for them. Even with such assistance, DMs should be able to change the interactive method and/or preference information type during the interactive solution process. This is related to the third cognitive challenge.

It is easy to see that endowing the DM with the central role in the solution process calls for an architecture, where the machine can perform actions in response to the DM's requests. In order to react swiftly in the presence of varying amounts of data, the machine should be able to do some work in the background anticipating future needs. Examples of such work could be constant improvement of a Pareto set approximation in a computationally complex problem, or updating the MOO problem model as soon as new data arrive.

It could also be useful to learn from the DM's actions in order to narrow down the focus of performed work. For example, predicting which areas in the Pareto optimal set are most interesting for the DM allows spending computational resources more efficiently by focusing on most important parts of the Pareto optimal set. This calls for employing computational intelligence. Finally, it is easy to see that different actions should be performed by autonomous parallel processes. As a result, the time spent by the DM on thinking about next actions can be efficiently utilized by the machine in processing data and preparing to react to DM's next requests.

The above arguments support the conclusion that MASs have potential in enhancing interactive methods. In the next section, we propose several types of agents working together. The ideas of some agents are borrowed from [35] and inspired by some other mentioned works on MASs in the context of MOO but the generic architecture is novel.

3. Generic multi-agent architecture for interactive methods

In this section, we propose a multi-agent architecture with various agents in different roles and describe how they address the challenges discussed in the previous section. As mentioned earlier, MASs contain several agents and each

agent has its own goal and actions. To perform these actions, the agents work in the shared environment and interact with each other in a cooperative manner. They exchange and share their findings to reach the overall goal of the system. Thanks to the cooperative nature of MASs, several interactive methods can be hybridized in one solution process. In the literature, many interactive methods have been proposed, and typically one method is selected and applied in the solution process. However, this is not necessarily ideal. In this study, the main aim of the hybridization is to use the appropriate interactive methods to solve MOO problems, as well as to support the DM by using best-suited methods for the needs of the learning and decision phases. Thus, the DM can gain more insight into the problem and find the most preferred solution in a shorter time.

In the proposed multi-agent architecture, each interactive method is operated by a dedicated agent, which makes the hybridization straightforward and adding new methods effortless. The cooperation between agents takes place through a shared environment in the MAS. It contains the MOO problem, surrogate models, preference information, candidate solutions, and the necessary data.

Everything starts with a MOO problem formulation. If we have data as a starting point, a problem model is created by fitting surrogate models to the data (this may need pre-processing the data first). Surrogate models are also needed if the problem involves computationally expensive functions. Otherwise, surrogates are not needed. If surrogates are used, they need to be updated if new data becomes available to increase their accuracy. The DM is supported with preference modeling and preference learning techniques while providing the preference information to direct the interactive solution process. Candidate solutions are accumulated in an archive and shared with relevant agents during the solution process in a cooperative way. The purpose of this cooperative structure is to use the strengths of each interactive method, reduce waiting times set on the DM, support them to decrease their cognitive load, and make them feel being in control in the solution process.

The proposed multi-agent architecture illustrated in Figure 1 has the following agent types to realize the purposes mentioned: a problem agent, a surrogate agent, a coordinator agent, a solution pool agent, a preference agent and an interactive method agent. Below we give more details on each agent.

Problem agent:

The main aim of the problem agent is to construct an optimization problem based on the given problem description. This agent is responsible for deriving decision variables, constraints, and objectives to be optimized. Given a complex and data-intensive real-life problem, constructing a MOO model may be a non-trivial task (and we do not here consider cases where deriving objectives from data requires additional tools). For some objective functions, we may only have data originating from the problem domain, rather than mathematical function formulations. Such objective functions are to be generated by constructing surrogate models based on the data available. On the other hand, some explicitly formulated objectives may be computationally expensive. Then computationally inexpensive surrogate models are needed in order to decrease the solution time. For the above listed reasons, the problem agent asks a surrogate agent to generate surrogate models for such objectives if they are present. Otherwise, the MOO problem is constructed simply on the objectives, decision variables, and constraints derived from the problem description.

Before the actual solution process is started, the coordinator agent sends a request to the problem agent for the problem to be solved. Based on this request, the problem agent constructs the MOO problem as described above. Once the MOO problem is constructed, the problem agent sends it to the coordinator agent and the solution process is started. During the interactive solution process, after each iteration with the DM, the coordinator agent asks the problem agent for an updated version of the MOO problem if it is available. The problem agent creates an update with the help of the surrogate agent, whenever the latter can improve surrogate models by utilizing new information.

Surrogate agent:

The main goal of the surrogate agent is fitting metamodels to the available data and generating surrogate models such as (1) a model representing an objective function which has no closed form formulation, (2) a computationally inexpensive model that represents an objective which has a computationally expensive function formulation or (3) a surrogate representation of the whole Pareto optimal set, according to the type of request from the problem agent. To create a computationally efficient surrogate model, choosing the best suited surrogate model is needed first. The surrogate agent is responsible for selecting the best surrogate modeling technique and supervising the training of the selected surrogate model. Once the suitable model is selected, the surrogate agent trains this model by using the available data (offline data). Additionally, the surrogate agent updates these surrogate models according to the candidate solutions and/or an updates that take place if there is a change in the relevant data (online data).

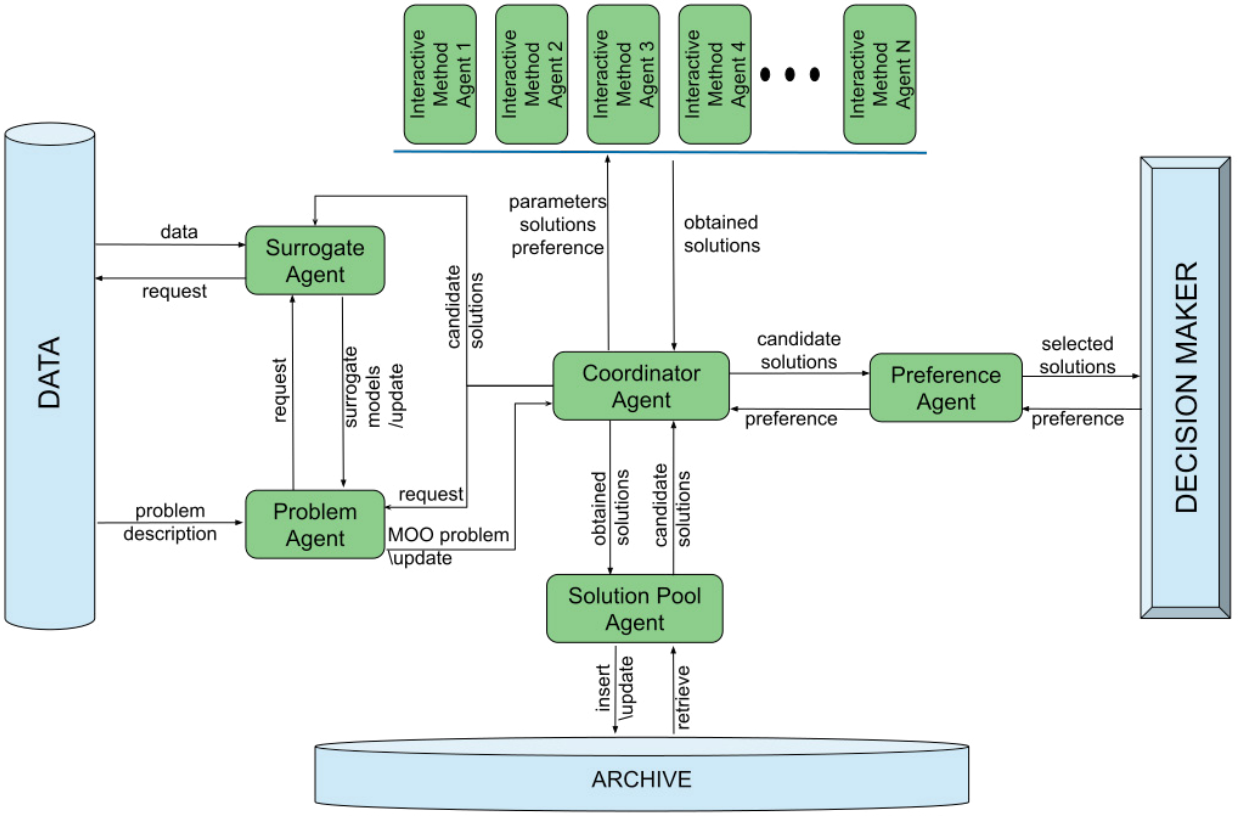

Fig. 1. A generic multi-agent architecture for interactive methods.

The surrogate agent updates surrogate models after obtaining new information. For the surrogates serving as approximations of computationally expensive objective functions or the Pareto optimal set, the information used is a set of new candidate solutions. For surrogate functions derived from data, the update utilizes new data once it becomes available. In both cases, the accuracy and the quality of the surrogate model is increased gradually during the solution process. For example, in [38], an automatic selection of a surrogate modeling technique for a given data set has been proposed. There, several surrogate models are trained by using the data generated from known optimization problems, and extracted features from this learning process are used to select the most appropriate surrogate modeling technique for a given new data set. As metamodels, there are deterministic models such as polynomials [25], neural networks [3], radial basis functions [9], support vector machines [42] and stochastic models such as Kriging [21] or Bayesian modeling [43] available in the literature.

As mentioned in the definition of the problem agent, generating and training surrogate models, which is described above, takes place before the actual solution process is started, based on a request by the problem agent. During the interactive solution process, surrogates are updated in the background if needed. Thanks to that, instead of computationally demanding problems, computationally less demanding surrogate problems are solved and the waiting time of the DM is eliminated in each iteration of the solution process.

Solution pool agent:

The main goal of the solution pool agent is to collect generated candidate solutions, organize them and give access to them for the coordinator agent and the DM on request through the preference agent. The shared archive that includes candidate solutions found so far is needed to hybridize interactive methods. After each iteration, the coordinator agent sends the obtained solutions to the solution pool agent. Some of these solutions might be dominated by the archived solutions which are accumulated during previous iterations. The solution pool agent is responsible for combining

received solutions in the current iteration with the previous solutions and eliminating dominated solutions from the combined set. In this way, interactive method agents cooperate with each other by storing their best solutions obtained so far in a common solution pool. The solution pool agent sends this archived set to the coordinator agent whenever the coordinator agent requests solutions.

The DM may need to access the archived solutions on two occasions. Firstly, the DM may consider selecting the final solution of the problem from previously derived solutions. Secondly, in some interactive methods, the DM expresses preferences with respect to a previously derived solution [28, 50]. In both cases, the preference agent assists the DM in exploring the solution pool and selecting a solution candidate through the solution pool agent. In the case of many objectives and many solution candidates accumulated, advanced techniques may be used in order to address the issue of cognitive overload (see e.g. [15]).

Coordinator agent:

Once the MOO problem is received from the problem agent, the coordinator agent selects one of the interactive methods and provides parameter settings, candidate solutions and preference information to the corresponding interactive method agent. The coordinator agent also creates an opportunity for the DM to switch to another method during the interactive solution process by suggesting to select another method, whenever it is appropriate.

In order to select an interactive method, the coordinator agent asks which preference expression mechanism is most comfortable and/or familiar for the DM. This naturally helps decreasing the cognitive load of the DM. Besides that, the coordinator agent monitors the solution process and tries to distinguish between the learning phase and the decision phase. This additional information can also help selecting the method which is most suitable in each moment. As a result, different interactive methods employed in a cooperative manner combine their strengths in order to improve the quality of the solution process.

Once the selected interactive method agent generates new solutions, the coordinator agent receives the obtained solutions and sends them back to the solution pool agent. The latter evaluates the performance of the selected interactive method agent based on the obtained solutions and the provided preference information in the previous iteration. For this purpose, it uses quality indicators that take into account the preference information to evaluate the quality of the solutions (e.g., [17, 23, 32, 52]). For the next iteration, the coordinator agent requests candidate solutions from the solution pool agent, checks for updates from the problem agent, gets new preference information from the preference agent, and sends all of these to the corresponding interactive method agent.

Preference agent:

The preference agent is the one that interacts with the DM, selects a subset of candidate solutions which is shown to the DM and takes the preference information based on the represented solutions. A preference model is created by using the given preference information. However, the provided preference information may be uncertain or inconsistent during the solution process since the DM is learning and gaining insight into the problem. Thus, the preference agent is observing the DM's behaviour (provided past preference information) and creates a preference model by using some machine learning techniques (e.g., [3, 9, 21, 25, 42, 43]) to learn the DM's preferences. Moreover, if the given new preference information has inconsistencies with the created preference model, the preference agent gives notice of this situation to the DM to make them aware of having provided contradictory preferences. One should notice that this does not necessarily mean that anything is wrong. It may be an indication of the DM having learnt more.

The preference agent lists the available forms for providing preference information based on the interactive methods included in the architecture, the DM selects one's preferred way, and the preference agent informs the coordinator agent about the type of the preference information which is to be used.

The preference agent can also be utilized in the following situations. In some real-world applications, the solution process may be repeated many times for similar problem instances. Providing preference information repeatedly can then be avoided if the preference agent utilizes the learned preference model to decrease the cognitive load of the DM for the next solution processes. When training the agent, previous candidate solutions presented to the DM are given as input and preference information provided by the DM as output. Once the preference agent is trained with this historical data, the created model can be used to produce preference information during the next solution processes. Some approaches have been proposed to predict DM's preferences for interactive methods (e.g., [2, 6, 22]).

Interactive method agent:

The main function of the interactive method agent is to control the interactive method to provide generated candidate solutions to the coordinator agent. In the proposed architecture, there can be several interactive method agents

and each of them covers a different interactive method. This means that the interactive method agent provides an interface to manage the connection between the coordinator agent and the interactive method. It runs one iteration of the interactive method with received parameter values, candidate solutions, and the preference information on request from the coordinator agent. As mentioned, many interactive methods convert the original MOO problem into single objective optimization problems by incorporating the given preference information and solving these subproblems by applying appropriate single objective optimization methods to get candidate solutions reflecting the preferences of the DM [26]. Once the solutions are obtained for that iteration, the interactive method agent sends them to the coordinator agent.

The functioning of the proposed multi-agent architecture is demonstrated below. We present an example sequence of agent interactions in the architecture, where the surrogate modeling is used in the MOO problem:

1. The problem agent takes the problem description and requests surrogate models from the surrogate agent according to the problem description.
2. The surrogate agent generates surrogate models based on the request and sends these models to the problem agent.
3. The problem agent constructs the MOO problem and sends it to the coordinator agent.
4. The coordinator agent selects the appropriate interactive method agent, sends its parameter values, candidate solutions and the preference information to the selected interactive method agent.
5. The interactive method agent runs its interactive method with the provided parameter values, solutions and preference information. After finishing, it sends the obtained solution candidates to the coordinator agent.
6. The coordinator agent evaluates the quality of the obtained solutions and sends them to the solution pool agent.
7. The solution pool agent inserts obtained solutions to the archive, updates the solutions in the archive and sends the candidate solutions back to the coordinator agent.
8. The coordinator agent sends the candidate solutions to the preference agent. The preference agent selects a subset of candidate solutions, represents them to the DM and receives preference information from the DM.

   (a) If the DM wants to stop, one selects a solution as a most preferred solution and the solution process is terminated.
   (b) If the DM wants to continue, the coordinator agent sends the candidate solutions to the surrogate agent, asks if there is an update to the problem agent and receives the updated surrogate problem. The solution process continues with step 4.
   (The surrogate agent updates surrogate models based on the best obtained solutions so far, or if there is a change in the data. The problem agent receives the updated surrogate models and updates the optimization problem.)

Finally, we summarize how challenges facing interactive MOO identified in Section 2 are addressed with the multi-agent architecture described. Using the surrogate agent allows eliminating waiting times in computationally complex problems, as well as organizing a decision making process when the problem is (partly) derived from data. The preference agent allows addressing the cognitive challenges by utilizing learned preferences in order to reduce the cognitive load, and assisting the DM. The coordinator agent, in cooperation with the preference agent and interactive method agents, allows addressing the accessibility issue by enabling interactive method selection according to DM's needs. This also contributes to addressing the cognitive challenge by providing more suitable ways of expressing preferences. The connection between data and problem formulation maintained by the problem agent using the surrogate agent opens possibilities to incorporate mechanisms of dealing with data uncertainty into the solution process.

4. Conclusions

We have discussed various challenges set on formulating and solving MOO problems to be able to utilize data available in making data-driven decisions. We have focused at interactive methods because of their advantages in supporting the DM in learning about the problem and about the feasibility of one's preferences.

In order to address challenges discussed and support making decisions with interactive methods, we find artificial intelligence tools like agents of high potential. In this, we have proposed a generic multi-agent architecture for interactive methods and described various agents with distinctive roles in the solution process. The architecture was presented in a general manner and further detailed implementations are naturally needed. However, it allows to outline how various challenges can be addressed.

Our motivation has been to show that combining a MAS with interactive MOO can be fruitful. Realizing this potential requires multidisciplinary efforts combining e.g., artificial intelligence, software architecture and various optimization tools. As a future research direction, we shall utilize the architecture in the open source software development of the DESDEO framework (desdeo.it.jyu.fi) for interactive multiobjective optimization methods.

Acknowledgements

This research was partly funded by the Academy of Finland (grants 311877 and 322221). The research is related to the thematic research area DEMO (Decision Analytics utilizing Causal Models and Multiobjective Optimization, jyu.fi/demo) of the University of Jyvaskyla.

References

[1] Acan, A., Lotfi, N., 2017. A multiagent, dynamic rank-driven multi-deme architecture for real-valued multiobjective optimization. Artificial Intelligence Review 48, 1–29.
[2] Battiti, R., Passerini, A., 2010. Brain–computer evolutionary multiobjective optimization: a genetic algorithm adapting to the decision maker. IEEE Transactions on Evolutionary Computation 14, 671–687.
[3] Beale, H.D., Demuth, H.B., Hagan, M., 1996. Neural network design. Pws Publishing, Boston .
[4] Bechikh, S., Said, L.B., Ghédira, K., 2011. Negotiating decision makers' reference points for group preference-based evolutionary multi-objective optimization, in: Proceedings of the 11th International Conference on Hybrid Intelligent Systems, IEEE. pp. 377–382.
[5] Belton, V., Branke, J., Eskelinen, P., Greco, S., Molina, J., Ruiz, F., Słowiński, R., 2008. Interactive multiobjective optimization from a learning perspective, in: Branke, J., Deb, K., Miettinen, K., Słowiński, R. (Eds.), Multiobjective Optimization: Interactive and Evolutionary Approaches. Springer-Verlag, Berlin, Heidelberg, pp. 405–433.
[6] Branke, J., Greco, S., Słowiński, R., Zielniewicz, P., 2014. Learning value functions in interactive evolutionary multiobjective optimization. IEEE Transactions on Evolutionary Computation 19, 88–102.
[7] Breedveld, S., Craft, D., van Haveren, R., Heijmen, B., 2019. Multi-criteria optimization and decision-making in radiotherapy. European Journal of Operational Research 277, 1–19.
[8] Buchanan, J., Gardiner, L., 2003. A comparison of two reference point methods in multiple objective mathematical programming. European Journal of Operational Research 149, 17–34.
[9] Buhmann, M.D., 2003. Radial Basis Functions: Theory and Implementations. Cambridge University Press.
[10] Cardon, A., Galinho, T., Vacher, J.P., 2000. Genetic algorithms using multi-objectives in a multi-agent system. Robotics and Autonomous Systems 33, 179–190.
[11] Cowan, N., 2001. The magical number 4 in short-term memory: A reconsideration of mental storage capacity. Behavioral and Brain Sciences 24, 87–114.
[12] Cui, Y., Geng, Z., Zhu, Q., Han, Y., 2017. Review: Multi-objective optimization methods and application in energy saving. Energy 125, 681–704.
[13] Drezewski, R., Siwik, L., 2006. Co-evolutionary multi-agent system with sexual selection mechanism for multi-objective optimization, in: Proceedings of the IEEE International Conference on Evolutionary Computation, IEEE. pp. 769–776.
[14] Ferber, J., Weiss, G., 1999. Multi-agent Systems: An Introduction to Distributed Artificial Intelligence. Addison-Wesley Reading.
[15] Filatovas, E., Podkopaev, D., Kurasova, O., 2015. A visualization technique for accessing solution pool in interactive methods of multiobjective optimization. International Journal of Computers Communications & Control 10, 508–519.
[16] Gettinger, J., Kiesling, E., Stummer, C., Vetschera, R., 2013. A comparison of representations for discrete multi-criteria decision problems. Decision Support Systems 54, 976–985.
[17] Hou, Z., Yang, S., Zou, J., Zheng, J., Yu, G., Ruan, G., 2018. A performance indicator for reference-point-based multiobjective evolutionary optimization, in: Proceedings of the IEEE Symposium Series on Computational Intelligence, IEEE. pp. 1571–1578.
[18] Hwang, C.L., Masud, A., 1979. Multiple Objective Decision Making – Methods and Applications: A State-of-the-Art Survey. Springer, Berlin.
[19] Jiang, S., Zhang, J., Ong, Y.S., 2012. A multiagent evolutionary framework based on trust for multiobjective optimization, in: Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 1, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC. p. 299–306.
[20] Kisiel-Dorohinicki, M., Socha, K., 2001. Crowding factor in evolutionary multiagent system for multiobjective optimization, in: Proceedings of the International Conference on Artificial Intelligence, CSREA Press. pp. 695–700.
[21] Kleijnen, J.P., 2009. Kriging metamodeling in simulation: A review. European Journal of Operational Research 192, 707–716.

[22] Li, K., Chen, R., Savić, D., Yao, X., 2018. Interactive decomposition multiobjective optimization via progressively learned value functions. IEEE Transactions on Fuzzy Systems 27, 849–860.
[23] Li, K., Deb, K., Yao, X., 2017. R-metric: Evaluating the performance of preference-based evolutionary multiobjective optimization using reference points. IEEE Transactions on Evolutionary Computation 22, 821–835.
[24] Lygoe, R.J., Cary, M., Fleming, P.J., 2013. A real-world application of a many-objective optimisation complexity reduction process, in: Purshouse, R.C., Fleming, P.J., Fonseca, C.M., Greco, S., Shaw, J. (Eds.), Proceedings of the 7th International Conference on Evolutionary Multi-Criterion Optimization, Springer, Berlin, Heidelberg. pp. 641–655.
[25] Madsen, J.I., Shyy, W., Haftka, R.T., 2000. Response surface techniques for diffuser shape optimization. AIAA Journal 38, 1512–1518.
[26] Miettinen, K., 1999. Nonlinear Multiobjective Optimization. Kluwer Academic Publishers, Boston.
[27] Miettinen, K., Hakanen, J., Podkopaev, D., 2016. Interactive nonlinear multiobjective optimization methods, in: Greco, S., Ehrgott, M., Figueira, J.R. (Eds.), Multiple Criteria Decision Analysis: State of the Art Surveys. 2nd edition. Springer New York, pp. 927–976.
[28] Miettinen, K., Mäkelä, M.M., 2006. Synchronous approach in interactive multiobjective optimization. European Journal of Operational Research 170, 909–922.
[29] Miettinen, K., Mustajoki, J., Stewart, T., 2013. Interactive multiobjective optimization with NIMBUS for decision making under uncertainty. OR Spectrum , 1–18.
[30] Miettinen, K., Ruiz, F., Wierzbicki, A.P., 2008. Introduction to multiobjective optimization: Interactive approaches, in: Branke, J., Deb, K., Miettinen, K., Słowiński, R. (Eds.), Multiobjective Optimization: Interactive and Evolutionary Approaches, Springer. pp. 27–57.
[31] Miller, G.A., 1956. The magical number seven, plus or minus two: Some limits on our capacity for processing information. The Psychological Review 63, 81–97.
[32] Mohammadi, A., Omidvar, M.N., Li, X., 2013. A new performance metric for user-preference based multi-objective evolutionary algorithms, in: Proceedings of the IEEE Congress on Evolutionary Computation, IEEE. pp. 2825–2832.
[33] Mönkkönen, M., Juutinen, A., Mazziotta, A., Miettinen, K., Podkopaev, D., Reunanen, P., Salminen, H., Tikkanen, O.P., 2014. Spatially dynamic forest management to sustain biodiversity and economic returns. Journal of Environmental Management 134, 80–89.
[34] Murdock Jr, B.B., 1962. The serial position effect of free recall. Journal of Experimental Psychology 64, 482–488.
[35] Ojalehto, V., Podkopaev, D., Miettinen, K., 2015. Agent assisted interactive algorithm for computationally demanding multiobjective optimization problems. Computers & Chemical Engineering 77, 105–115.
[36] Peitz, S., Ober-Blöbaum, S., Dellnitz, M., 2019. Multiobjective optimal control methods for the navier-stokes equations using reduced order modeling. Acta Applicandae Mathematicae 161, 171–199.
[37] Russell, S.J., Norvig, P., 1995. Artificial Intelligence: A Modern Approach. Prentice-Hall.
[38] Saini, B.S., Lopez-Ibanez, M., Miettinen, K., 2019. Automatic surrogate modelling technique selection based on features of optimization problems, in: Proceedings of the Genetic and Evolutionary Computation Conference Companion, Association for Computing Machinery. p. 1765–1772.
[39] Sawaragi, Y., Nakayama, H., Tanino, T., 1985. Theory of Multiobjective Optimization. Academic Press., Orlando.
[40] Silva, M.A.L., de Souza, S.R., Souza, M.J.F., de Franca Filho, M.F., 2018. Hybrid metaheuristics and multi-agent systems for solving optimization problems: A review of frameworks and a comparative analysis. Applied Soft Computing 71, 433–459.
[41] Siwik, L., Natanek, S., 2008. Solving constrained multi-criteria optimization tasks using elitist evolutionary multi-agent system, in: Proceedings of the IEEE Congress on Evolutionary Computation, IEEE. pp. 3358–3365.
[42] Smola, A.J., Schölkopf, B., 2004. A tutorial on support vector regression. Statistics and computing 14, 199–222.
[43] Snoek, J., Larochelle, H., Adams, R.P., 2012. Practical bayesian optimization of machine learning algorithms, in: Pereira, F., Burges, C.J.C., Bottou, L., Weinberger, K.Q. (Eds.), Advances in Neural Information Processing Systems 25. Curran Associates., pp. 2951–2959.
[44] Steuer, R.E., 1986. Multiple Criteria Optimization: Theory, Computation, and Application. John Wiley and Sons.
[45] Stewart, T., Bandte, O., Braun, H., Chakraborti, N., Ehrgott, M., Göbelt, M., Jin, Y., Nakayama, H., Poles, S., Di Stefano, D., 2008. Real-world applications of multiobjective optimization, in: Branke, J., Deb, K., Miettinen, K., Słowiński, R. (Eds.), Multiobjective Optimization: Interactive and Evolutionary Approaches. Springer Berlin Heidelberg, pp. 285–327.
[46] Sun, Z., Dawande, M., Janakiraman, G., Mookerjee, V., 2019. Data-driven decisions for problems with an unspecified objective function. INFORMS Journal on Computing 31, 2–20.
[47] Tabatabaei, M., Hakanen, J., Hartikainen, M., Miettinen, K., Sindhya, K., 2015. A survey on handling computationally expensive multiobjective optimization problems using surrogates: non-nature inspired methods. Structural and Multidisciplinary Optimization 52, 1–25.
[48] Tamouk, J., Acan, A., 2018. A multimetric and multideme multiagent system for multiobjective optimization. Computational Intelligence 34, 1122–1154.
[49] Vallerio, M., Hufkens, J., Impe, J.V., Logist, F., 2015. An interactive decision-support system for multi-objective optimization of nonlinear dynamic processes with uncertainty. Expert Systems with Applications 42, 7710–7731.
[50] Vassilev, V.S., Narula, S.C., Gouljashki, V.G., 2001. An interactive reference direction algorithm for solving multi-objective convex nonlinear integer programming problems. International Transactions in Operational Research 8, 367–380.
[51] Wooldridge, M., 2009. An Introduction to Multiagent Systems. John Wiley & Sons.
[52] Yu, G., Zheng, J., Li, X., 2015. An improved performance metric for multiobjective evolutionary algorithms with user preferences, in: Proceedings of the IEEE Congress on Evolutionary Computation, IEEE. pp. 908–915.
[53] Zhou-Kangas, Y., Miettinen, K., Sindhya, K., 2019. Solving multiobjective optimization problems with decision uncertainty: an interactive approach. Journal of Business Economics 89, 25–51.