User Modeling and User-Adapted Interaction (2022) 32:883–921
https://doi.org/10.1007/s11257-021-09318-3

EvoRecSys: Evolutionary framework for health and well-being recommender systems

Hugo Alcaraz-Herrera[1] · John Cartlidge[1] · Zoi Toumpakari[2] · Max Western[3] · Iván Palomares[4,5]

Received: 12 June 2020 / Accepted in revised form: 28 November 2021 / Published online: 31 January 2022

Abstract

In recent years, recommender systems have been employed in domains like e-commerce, tourism, and multimedia streaming, where personalising users' experience based on their interactions is a fundamental aspect to consider. Recent recommender system developments have also focused on well-being, yet existing solutions have been entirely designed considering one single well-being aspect in isolation, such as a healthy diet or an active lifestyle. This research introduces EvoRecSys, a novel recommendation framework that proposes evolutionary algorithms as the main recommendation engine, thereby modelling the problem of generating personalised well-being recommendations as a multi-objective optimisation problem. EvoRecSys captures the interrelation between multiple aspects of well-being by constructing configurable recommendations in the form of bundled items with dynamic properties.

✉ Hugo Alcaraz-Herrera
h.alcarazherrera@bristol.ac.uk

John Cartlidge
john.cartlidge@bristol.ac.uk

Zoi Toumpakari
z.toumpakari@bristol.ac.uk

Max Western
m.j.western@bath.ac.uk

Iván Palomares
ivanpc@ugr.es

1 Department of Computer Science, University of Bristol, Bristol, UK

2 School for Policy Studies, University of Bristol, Bristol, UK

3 Department for Health, University of Bath, Bath, UK

4 Department of Computer Science and Information Engineering, National Cheng Kung University, 70101 Tainan, Taiwan

5 Andalusian Research Institute in Data Science and Computational Intelligence (DaSCI), University of Granada, Granada, Spain

The preferences and a predefined well-being goal by the user are jointly considered. By instantiating the framework into an implemented model, we illustrate the use of a genetic algorithm as the recommendation engine. Finally, this implementation has been deployed as a Web application in order to conduct a users' study.

Keywords Recommender systems · Evolutionary computing · Genetic algorithms · Food recommendation · Physical activity recommendation · Well-being

1 Introduction

The Internet enables ubiquitous access to a vast array of online products and services. However, while this offers users the benefit of greater choice, finding a preferred product or service when presented with seemingly endless options requires significant exploration time. To attenuate this problem of *information overload*, recommender systems (RS) were introduced to supply a user with targeted results (i.e. *recommendations*) based on that user's individual preferences and the preferences of other users with similar characteristics (Aggarwal 2006).

While there is a considerable literature on recommender systems, across a variety of domains, the effort focused on health and well-being recommendation is comparatively scarce. The vast majority of these works focus on a single aspect of health, such as exercise (Berndsen et al. 2017; Pilloni et al. 2017; Reimer et al. 2016) or healthy food intake (Achananuparp and Weber 2016; Akkoyunlu et al. 2017; Schäfer 2016), in isolation. Another limitation of previous work is the lack of flexibility to recommend "tailored" items. In some domains, such as retail (Wu et al. 2019) or entertainment (Gómez-Uribe and Hunt 2015), this is not an issue because recommendations (i.e. products or movies) are intrinsically static, with unchanging properties. By contrast, recommendations in a personalised well-being domain—meal ingredients and serving sizes; duration and intensity of exercise—should be configurable.

We argue that the interrelationship between daily meals and exercise plays a fundamental role in general well-being and the adoption of healthy living habits. Therefore, it is crucial to consider this interrelationship in personalised well-being recommendation. Our present study constitutes—to the best of our knowledge—one of the first research efforts in this direction. We also produce recommendations that are dynamically tailored to users' preferences, such that we capture not only *what* to recommend, but also *how much*. To achieve this, we employ a genetic algorithm (GA). While GAs have been used in recommender systems before (e.g. Caldeira et al. 2018; Lv et al. 2015), this technique has not been used to realise the potential of producing highly tailored recommendations, which are necessary in domains such as well-being.

To overcome the aforesaid limitations in recommender systems for well-being, we propose a novel approach:

1. We firstly introduce *EvoRecSys*, a conceptual recommendation framework based on an evolutionary multi-objective optimisation problem, where constraints are modelled upon users' preferences, their physical condition, and their well-being goals. The underlying evolutionary algorithm explores the search space of all

possible combinations of recommendable items, which are in the form of meal-exercise *bundles*. These bundles capture the interrelationship between eating and exercising in the domain of personal well-being. Output solutions (i.e. items to recommend) balance what the user *likes* with what the user *needs* in order to achieve her/his well-being goals. For example, if a user wants to lose weight then the recommended food items will not only meet user preferences, but also keep within strict calorie limits; while recommended exercise will consider calories burned, so that weight loss is ensured. At the same time, general health guidelines, such as controlling the amount of saturated fats, sugars, etc., are observed.

2. To demonstrate the suitability of the EvoRecSys framework, we instantiate it as a model for general well-being, with four possible well-being goals as the personalised constraints for the user, a well-defined quantity of items to be recommended, and a specific evolutionary implementation.
3. Our instantiated model incorporates principles from collaborative filtering recommender systems. Using a similarity metric, our model identifies users that are similar to the target user in terms of preferences, physical condition, and well-being goal. We show that integrating principles from collaborative filtering helps deal with situations of incomplete recommendation user-related interaction information and enhances recommendation diversity, which is fundamental to motivating users during their well-being journey.

Our implemented model is evaluated and validated under two aspects: (1) we measure the algorithmic performance and optimality of the underlying evolutionary approach for different parameter settings and through benchmark against several baselines implementations, demonstrating the model's ability to produce efficient and optimal recommendations that are semantically meaningful; (2) we also conduct a user study with more than 200 participants and demonstrate that personalised recommendations are positively perceived under four criteria: health, diversity, serendipity, and attractiveness. An additional, A/B test shows user's tendency to prefer recommendations produced by the proposed EvoRecSys implementation against a CF-based implementation.

EvoRecSys advocates the use of genetic algorithms (GAs) as the core evolutionary technique to optimise recommendations. Although this paper illustrates a model instantiated upon EvoRecSys, different models emulating our conceptual framework can be seamlessly built by defining the necessary input data and objectives to optimise, thereby adapting it to the intended users and aims of the model in question. Furthermore, as opposed to recommending static immutable items, which conventional RS approaches usually deal with, EvoRecSys enables the generation of dynamic recommendations. In summary, this study provides the first effort in establishing a conceptual framework for producing configurable recommendations that incentivise users' well-being using evolutionary computing as the core of the recommendation engine.

This paper is structured as follows. Section 2 describes related research on recommender systems for food and exercise, along with studies that use GAs in the recommendation process. Section 3 explains the architecture and key elements that compose the general EvoRecSys framework. Section 4 presents a concrete implementation of EvoRecSys for health and well-being recommendations. Section 5 analyses

algorithmic performance. A user study is then presented in Sect. 6. Finally, Sect. 7 concludes.

2 Related work

There exist a number of recent research studies on RS for health and well-being. However, unlike our proposed approach, existing studies tend to consider food and physical activity recommendations in isolation, rather than as a combined bundle. Section 2.1 outlines relevant work focused on food recommendation. Section 2.2 describes research related to physical activity recommender systems. Section 2.3 shows studies that incorporate a genetic algorithm as a complementary technique or an extra step during the recommendation process. Finally, Sect. 2.4 presents studies on recommender systems whose output contains *bundles* of more than one recommendable item. Our contribution to the literature is summarised in Sect. 2.5.

2.1 Food recommender Systems

In the scope of personalised food recommendation, some approaches have focused on food substitution. For instance, Achananuparp and Weber (2016) hypothesised that food items consumed in the same context can be seamlessly replaced by each other—e.g. a *tuna* sandwich can be substituted for a *ham* sandwich if both are consumed with a salad—thereby allowing for greater diversity in daily meals. The "substitutability" between two food items was measured by the cosine similarity technique and two vector representations for those food items were explored: positive pointwise mutual information matrix (PPMI matrix) and singular value decomposition (SVD); with the latter obtaining best performance. This method produces top-10 food substitute candidates for each food item. The food data used in this work came from 9896 users of the web platform called *MyFitnessPal* (MFP) and their food consumption diaries.

Akkoyunlu et al. (2017) argued that it is possible to recommend healthy food substitutes that match user preferences within the same context; where context is defined as the set of other food items that are consumed with the target food. For example, in the meal {*tea*, *bread*, *juice*}, the context of *tea* is {*bread*, *juice*}. Using the French database, *INCA 2*, which contains food diaries of 2624 adults, the recommender model generates a graph, with nodes representing meals in the database. Under this design, substitutable nodes—those belonging to the same dietary context—are adjacent and form a fully connected sub-graph, or *clique*. Nodes are considered highly substitutable if consumed in similar contexts, and less substitutable if consumed together.

Caldeira et al. (2018) suggest that meal recipes can be recommended by considering their nutritional value, harmony of ingredients, and the availability of the ingredients. This research uses the Non-dominated Sorting Genetic Algorithm II (NSGA-II), introduced by Deb et al. (2002), which is an evolutionary algorithm with the following features: (i) elitism, to preserve the best solution of current population in the next generation; (ii) crowding distance techniques, to provide diversity in solutions; and (iii) non-dominated sorting techniques, to maintain a Pareto-optimal archive solutions.

Using this algorithm, a list of suggested meal recipes is found by considering the number of portions, quantity of ingredients, and tastiness. Their approach also makes it possible to specify a food style, such as *vegetarian* or *vegan*. The set of recipes used in this study was collected from Brazilian website *TudoGostoso*.[1]

Recently, Musto et al. (2020) proposed a knowledge-based strategy that incorporates "holistic" user profile information in a popularity-driven recipe recommender algorithm. Profiles include user data such as demographics, age, gender, and weight, as well as food requirements, physical activity level, and body mass index, in order to re-rank popularity-based recommendations so that user-related health factors are considered. This solution is different from state-of-the-art food RS in that it handles knowledge about users' physical health and behavioural characteristics rather than considering diet preferences alone. However, we note that while physical activity data are used as input to the model, only food recipes are recommended (unlike the approach that we propose in this paper, where we recommend a bundle containing interrelated food items for meals and physical activities).

2.2 Recommender systems for physical activity

In terms of physical activity recommender systems, there are several studies whose objective is to try to change the user behaviour towards a healthy physical lifestyle. For example, Reimer et al. (2016) advocated for users to change their habits in a tailored manner in order to reach exercise goals. The proposed framework of this research motivates a user through "nudges" (Thaler and Sunstein 2008). There are various types of nudges such as suggestions, praise and rewards. The accepted nudges by the user are used to create a personalised profile that will encourage the user to reach the goal. Furthermore, this framework utilises a collaborative filtering technique to generate recommendations focused on the goals. Users' socio-demographic data and their past behaviour are used in order to characterise the *feature vector* that represents each user. Similarly between two users is calculated by the cosine similarity.

Some research tackles the domain of sports. For instance, Pilloni et al. (2017) argue that it is possible to predict when a user is going to abandon an exercise routine based on their previous behaviour and thus prevent it. The proposed model uses a machine learning algorithm as the core of the recommendation process. The previous user behaviour is used as a training vector which has 34 features including covered distance, workout duration, and rest time. Once the algorithm is trained, it is able to predict if a user is going to abandon the routine. If so, a recommendation for encouraging the user to continue the routine is triggered. Otherwise, the system predicts the user will not abandon the routine. The study tested 4 classification algorithms: (i) random forest, (ii) AdaBoost, (iii) extra trees, and (iv) multi-layer perceptron; where random forest obtained the best performance. Data used for the analysis were taken from the *u4fit* platform.[2]

Following the same path, Berndsen et al. (2017) showed that amateur runners—those without advanced training, or access to a coach—can improve performance

[1] https://www.tudogostoso.com.br.

[2] https://www.u4fit.com.

using elite runners' behaviour as a target to follow. Two models—K-nearest neighbours (KNN) and extreme gradient boosting (XGB), which was shown to have higher performance—were trained to predict marathon times using users' finishing times at various distances. The predicted times are the basis of the recommendations, which is performed by collaborative filtering. For instance, if a runner finishes a 10 km race in 63 min, while an elite runner takes 46 min, the recommendation would be, *"you have to train a little bit more"*. The dataset employed in this work was taken from diverse websites where athletes are allowed to declare their race times, such as the website *RunnersWorld*.[3] Additionally, the work explored ways to best present recommendations to runners in order to *nudge* their training behaviours.

2.3 Evolutionary algorithms in recommender systems

Evolutionary computation techniques have been used in many domains, including, to name a few, Computer Science (Dutta et al. 2020; Gunasegaran and Cheah 2019), Geology (Rezaei and Asadizadeh 2020), Biology (Guo et al. 2020), and Chemistry (Buchely et al. 2020). In the area of RS, genetic algorithms (GAs) (one of the primary evolutionary computing techniques) have been scarcely applied to date.

One of the most typical applications of RS is in e-commerce. For example, Lv et al. (2015) proposed a framework to help the standard techniques of recommendation (collaborative filtering and content-based) to yield the quality of recommendations through a traditional GA and a class-based ontology, which is built by considering an item that the user is interested in. For instance, if the user is interested in a book, the class would be *Book* and some of its attributes would be *title* and *publication date*. The workflow of this framework consists of: (i) retrieving the items in the user's shopping cart from the log file of an e-commerce site; (ii) mapping each attribute as a class to build the ontology; (iii) using the GA to optimise the different feature weights in the set of items; (iv) using the coefficients calculated in the previous step to cluster items; and (v) recommending the nearest cluster to the product that the user was interested in during the last visit. To evaluate the framework, the *MovieLens* dataset[4] was used, showing a better performance than standard collaborative filtering and content-based techniques.

Hassan and Hamada (2018) have also used a GA as a optimisation step within a multi-criteria RS to compare well-known RS methods (collaborative filtering and content-based) and GA-based methods. In the study, three variations of GA were used: (i) a standard GA—a population of candidate solutions (called *individuals*) to an optimisation problem is evolved towards better solutions. Each candidate solution has a set of properties which can be mutated and altered by genetic operators with a fixed probability of occurrence (Whitley 1994); (ii) an adaptive GA—population information in each generation is used to adjust the probability of both mutation and crossover in order to maintain population diversity and sustain the convergence capacity (Srinivas and Patnaik 1994); and (iii) a multi-heuristic GA—the principal features of two or more heuristic approaches are combined to form a single algorithm for enhancing per-

[3] https://www.runnersworld.com.

[4] https://movielens.org.

formance and preventing premature convergence during the search process. Crossover and mutation rates are initially set high, and then reduced slowly over time. Results demonstrated that GA-based approaches could outperform collaborative filtering and content-based RS methods when tested using the *Yahoo! Movie* website.[5]

Karabadji et al. (2018) presented another multi-criteria based study and found that a GA can suggest a suitable set of neighbours in a collaborative filtering RS. The research demonstrated that a GA (i) alleviates settings problems related to selecting the N most similar neighbours and (ii) guarantees diversity by selecting groups of individuals that are different. In this way, both high similarity and high diversity are achieved. The model was tested using a dataset containing 239 ratings by 100 customers from 17 Algerian insurance companies and the *MovieLens* dataset[4].

Finally, Cui et al. (2017) argued that, while losing a certain degree of precision, it is possible to improve the metrics of diversity and novelty by adding a multi-objective GA during the recommendation process. The GA represents each individual as a 1-D integer array, where each loci in the genotype represents an item that can only appear once in the recommendation list. While the mutation operator has a standard functionality, the crossover operator is designed to preserve a user's habits such that if an item appears frequently in the user's recommendation lists, the probability of preserving it unchanged during the crossover process will increase. For evaluation, two objective functions are used: (i) accuracy and (ii) diversity. The authors validate the performance of their GA in combination with a set of traditional recommendation algorithms. Results demonstrate that the combination of their GA and the recommendation algorithms can achieve a good balance between precision and diversity.

2.4 Bundling in recommender systems

In recommender systems, an output recommendation may contain multiple items, which we call a *bundle*. For instance, Rapti et al. (2014) introduced an agent-based approach for generation of personalised product bundles for enterprise networks. The core process includes complementary associations between products and building bundles according to the customer preferences. Furthermore, the approach is able to adapt itself if the environment changes: customer profile modifications, product availability, and rule and constraint rule diversity. The authors provide an example under an e-Furniture context, employing an agent-based system in a network of enterprises that manufacture.

Bundling is also used in other recommender system domains such as the telecommunication industry. For example, Dragone et al. (2018) present a system whose outputs are combined services (mobile connectivity, broadband allocation, TV on demand, etc.) and electronic device plans (smartphones, tablets, TVs), selected according the customer necessities. The system considers the *constructive preference elicitation* framework, which allows to model the bundle offers as a defined set of variables and constraints. By using constraint optimisation, the system generates high-utility recommendations. Furthermore, an empirical validation study, where 134 participants were

[5] https://www.yahoo.com/entertainment/movies/.

involved, is presented. Results show that the outputs of the system were considered more satisfactory than those obtained with standard techniques used in the market.

Zanker et al. (2010) applied bundling to the tourism domain, with recommendations containing bundle collections of accommodation, activities, and restaurants. This system uses a constraint satisfaction problem (CSP) solver, which invokes numerous recommender systems to propose a ranked list of items for each product category based on the user model and available community knowledge. The authors evaluate their system using an example scenario consisting of 5 product classes with 30 different product properties and 23 representative constraints. The evaluation only focuses on computation time and does not consider the quality of the final recommendations. Results demonstrate the system is able to generate bundles within a time period that is acceptable for typical e-commerce situations.

2.5 Contribution

While bundle recommendations have been explored in a number of domains, we present the first research that considers recommendation bundling in the health and well-being domain. We focus on linking two aspects that have previously been studied separately: (i) nourishment (see Sect. 2.1) and (ii) physical activities (see Sect. 2.2). We combine physical activity and diet as this aligns with the weight loss/management goal of the user and is designed to illustrate how lifestyle bundles can be incorporated to provide more holistic advice. Elliot and Hamlin (2018) have presented evidence that people like to treat healthy lifestyle in a collective manner when making efforts to change their behaviour and improve health (also see Johns et al. 2014).

Also, while other recommender systems have included a GA, often the role of the GA is limited to a single step inside the recommendation process (see Sect. 2.3). By contrast, we introduce a novel approach for generating recommendations entirely based on a GA. Our approach also enables us to combine items with a high level of granularity, such that the attributes of each recommendable item can be optimised throughout the evolutionary process. This provides an opportunity to offer highly tailored recommendation items based on user preferences.

Furthermore, as previously stated in Sect. 2.3, GAs have been successfully used to improve traditional RS techniques such as collaborative filtering (Hassan and Hamada 2018; Karabadji et al. 2018) and matrix factorisation (Kilani et al. (2018)). Thus, the presented research represents an effort to continue the integration path of GAs in the RS domain.

3 EvoRecSys: general description

This section introduces EvoRecSys (Evolutionary Recommender System), a conceptual framework that reformulates the recommendation problem as a multi-objective optimisation problem in which solutions to the problem are modelled by configurable items or groups of items to recommend to the end user. Under this approach, it is plausible to build recommendations focused on (i) reaching a specific well-being goal

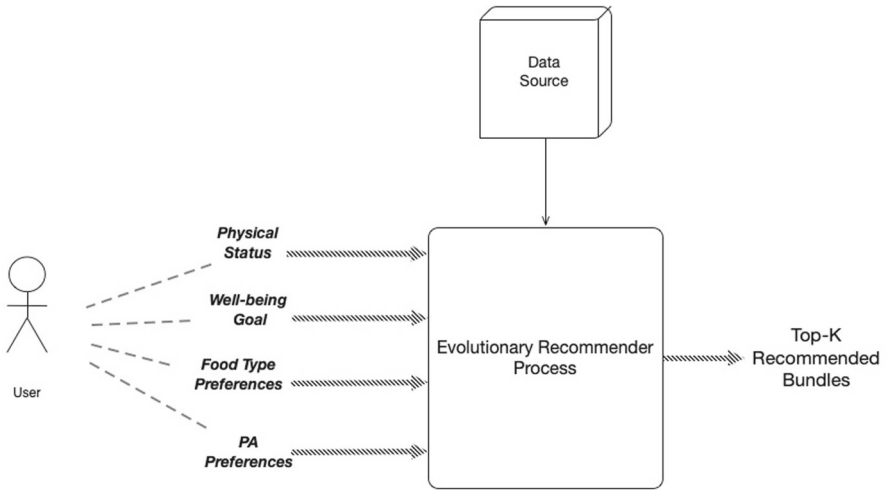

Fig. 1 Architecture of EvoRecSys for well-being

specified by the user and, at the same time, (ii) considering the user preferences. In other words, this framework creates recommendations balancing the user preferences and what types of food and physical activity should the user consider for reaching a goal.

The remainder of this section describes the main elements of our proposed approach for personalised well-being. Section 3.1 shows the general framework architecture, the workflow model and a general description of the data sources that could be used by the framework during the evolutionary-recommendation process. Section 3.2 defines essential concepts related to the framework. Finally, Sect. 3.3 describes the key features of the core element of this framework: a genetic algorithm.

3.1 Architecture and workflow

EvoRecSys receives the user input (physical characteristics, well-being goals, and food and exercise preferences) and recommends meal and physical activity (PA) bundles that are tailored to the user through an evolutionary optimisation process. The architecture and workflow are presented in Fig. 1.

The main inputs of EvoRecSys are divided into four elements of user-related information:

(i) *Physical status and exercising habits*. Data related to age, gender and body measurements of the user, as well as their frequency of exercising.
(ii) *Food category preferences*. How much the user likes certain ingredients, predetermined types of food, etc. In order to do this, a numerical scale can be used, for example the 5-point Likert scale. Due to the versatility of this element, it is possible to implement it in diverse ways. For instance, it might focus on a specific dietary requirements such as vegetarians, vegans or people with certain allergies.

(iii) *Preferences on types of physical activity*. Information that describes how much the user likes certain types of PA. As in the previous element, it can be implemented focusing on predetermined set of physical activities. For example, water activities, or sports where a ball is used. In order to measure the user preferences, a minimum–maximum-based numerical scale can be used, similar to the previous element.

(iv) *Well-being goal*. A goal chosen by the user from a set of predefined goals focused on a specific well-being aspect to be improved. The goals can be set for handling either general circumstances (losing weight, maintaining weight, etc.) or specific ones (control chronic diseases), depending on the specific aims of the model implemented upon this general framework. Without losing generality, our instantiated model in Sect. 4 focuses on general-purpose recommendations aligned with various well-being goals.

During the evolutionary recommendation process, the framework interacts with a data source that contains, at least, food data and PA data. Previous user preferences are also needed in models that gather users' interactions and feedback over time. Due to the decoupled architecture of the framework, the data source can be any readily accessible database (e.g. *myfitnesspal*, *u4fit*, or *Kaggle*)[6] as long as it contains the necessary data to perform the evolutionary recommendation. The framework outputs a list of K recommendations, which are tailored to the user preferences and the chosen well-being goal. Next, we describe the proposed evolutionary process and its core components.

3.2 Basic definitions

Genetic algorithms (GAs) are an optimisation technique inspired by natural selection. They operate by "evolving" a population of individuals, each representing a possible solution in the problem domain. Initially, the population is poor quality and widely dispersed across the search space. Over time, the population will gradually converge towards regions of space with better solutions that are closer to the user's preferences and their health needs.

In the scope of this study, and based on existing coding schemes to represent individuals (Goldberg 1989), we encode individuals as meal-PA bundles, containing food attributes and PA features. A meal is defined as a set of N food items. Additionally, a PA item defines a physical exercise that could contain attributes such as the type of PA, duration, intensity level etc. This is illustrated in Fig. 2. Additionally, it is feasible to define semantic rules that guarantee coherent meal structures (an example will be shown in the concrete implementation presented in Sect. 4).

Depending on the design decisions made to build a model upon this general framework architecture, the output can have different structures. For instance, the typical output is the best evaluated individual after the evolutionary process with K bundles. Another possible output would consider the top N individuals after the evolutionary process under the assumption that each individual would have one bundle.

[6] https://www.myfitnesspal.com; https://www.kaggle.com.

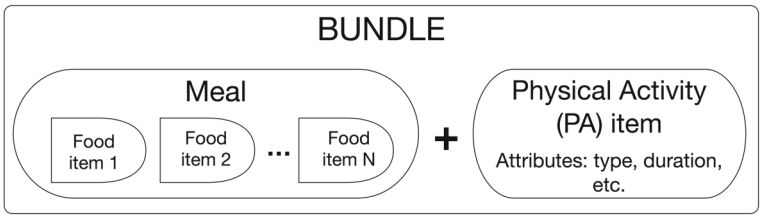

Fig. 2 Structure of a bundle or *recommendable item* in the EvoRecSys framework. An individual contains $K \geq 1$ bundles

3.3 The evolutionary process

Here, we describe the main steps of the genetic algorithm that drives evolution in the EvoRecSys architecture. First, a population of individual "recommendations" are initialised at random across the solution space. Each individual created must: (i) match user preferences; (ii) be within the intrinsic boundaries of the inputs defined; and (iii) have an interrelationship between food and PA items. To determine the performance of individual recommendations, a key element of a GA is the evaluation function or *fitness function*. It is inspired by the natural selection statement that says that the most adapted individuals in a certain environment have more opportunities to survive and hence, to transmit their genetic information to the next generation of individuals (Goldberg 1989). In order to provide suitable and consistent recommendations that meet (i) the user's preferences and (ii) her/his well-being goals, we define the fitness function upon a set of restrictions or objectives to optimise. Let $\mathcal{R} = \{\mu_1, \mu_2, \ldots, \mu_M\}$ represent the set of all possible restrictions to consider with $M \geq 1$. A fitness function FF_i associated with a user $u_i \in U$ with a goal $\mathcal{G}_i$ is defined based on $\mathcal{R}$ and her/his individual food-PA preferences Ψ_{u_i}. It assesses the matching degree to which a recommended bundle simultaneously meets $\mathcal{G}_i$ and the user preferences.

$$FF_i = FF(\Psi_{u_i}; \mathcal{G}_i) = \phi\left(\Psi_{u_i}; \Gamma_i(\mu_{i,1}), \Gamma_i(\mu_{i,2}), \ldots, \Gamma_i(\mu_{i,M})\right) \tag{1}$$

with $\mathcal{G}_i = \{\mu_{i,1}, \mu_{i,2}, \ldots, \mu_{i,M}\}$. $\Gamma_i(\mu_{i,j})$ is an *aptitude* function describing the degree to which restriction $\mu_{i,j}$ $(j = 1, \ldots, M)$ is satisfied by the individual; Ψ_{u_i} is a function that measures how much u_i preferences are met, and ϕ is a combination function, e.g. an averaging or aggregation operator (Beliakov et al. 2007). For example, Ψ_{u_i} could be a distance function between a representation of the user preferences and the properties of recommendable bundles in an individual. Figure 3 illustrates the definition and application of a fitness function FF_i.

The next evolutionary step is the selection of "parent" individuals to reproduce. This step focuses on choosing the *fittest* individuals (i.e. those with the *lowest* aptitude values). There are numerous selection methods, including: proportional selection, rank-based selection, tournament selection, disruptive selection, and elitism (for further reading, see Jong et al. 1997). Here, we use *tournament selection*, however EvoRecSys enables any selection method to be used.

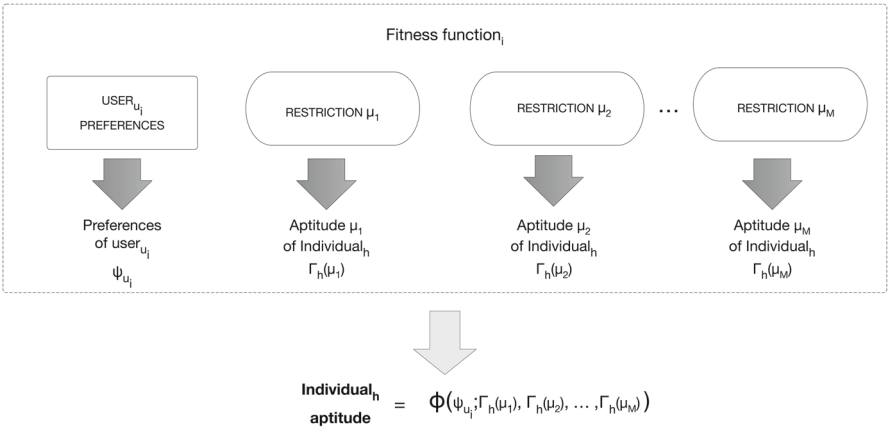

Fig. 3 Fitness (aptitude) evaluation for an individual consisting of K meal-PA bundles

We then produce the offspring population from the selected parents. To enable exploitation of good genetic combinations that have produced high fitness in parents, offspring should be similar to parents. At the same time, to explore solution space, we need to introduce some novelty in the offspring population. To achieve these two aims, we use the genetic operators *crossover* and *mutation*, respectively (e.g. see Goldberg 1989). The *crossover* operator randomly takes two individuals of the new population and it combines a part of each individual to randomly create two new ones, with the aim of further exploring a specific (and sometimes promising) part of the search space. Under the EvoRecSys framework approach, it is feasible implementing this genetic operator in different ways and granularity levels. Regarding meals, we suggest to recombine food items among individuals' bundles, rather than complete meals. Regarding physical activities, the suggestion is similar to meals: recombining them among individuals' bundles. The *mutation* operator acts on one individual, such that one element of the genotype (its representation in GA terms) is modified. This operator therefore explores the local region of search space. In the context of EvoRecSys, a variety of approaches to mutate exist. We suggest, nevertheless, to only mutate food items within meals and PA items within bundles of an individual. Due to the flexible architecture of bundles, both food items an PA items can be mutated regardless the stochastic process implemented for this genetic operator.

Finally, genetic algorithms have a number of other parameters to be considered, including population size (i.e. number of individuals), number of generations (or *evolutionary iterations*), crossover probability, and mutation probability. Holland (1975) states that the crossover operator is the most reliable operator in order to explore the search space, whereas the mutation operator is a complement of the crossover. Therefore, crossover should have a considerably high probability of taking place and mutation a comparatively small probability that it occurs. Regarding the number of generations and the population size, these parameters are directly proportional to the problem size (Jong et al. 1997). Said otherwise, the more elements individuals represent, the bigger the population size and the number of generations are for the sake of

Table 1 User inputs and their range values

Physical status	Input value range
Gender	0: Male, 1: Female
Age (years)	10, 11, …, 95
Height (m)	1.10, 1.11, …, 2.20
Weight (kg)	30.0, 30.1, …, 130.0
Activity level	0: sedentary, …, 3: very active
Physical activity	
Days per week	0, 1, …, 7
Minutes per session	0, 1, …, 120

exploring and converging into an optimal zone in the search space. Finally, considering the encoding used in this framework, it is plausible that other parameters might arise in order to control inherent processes related to the encoding itself. Similarly, other data sources or supplementary processes typical in recommender systems such as collaborative filtering could be flexibly incorporated (see Sect. 4.3). The abstract and conceptual nature of the EvoRecSys enables the flexible incorporation of additional parameters.

4 EvoRecSys implementation for personalised well-being

We introduce a concrete proof-of-concept implementation of EvoRecSys in the domain of well-being and preventative health. Here, we do not consider the more difficult problem of accommodating clinical conditions, so the target user profiles exclude people with chronic diseases (diabetes, hypertension, allergies, etc.) and kinetic limitations (paralysed or amputated limbs). We also do not consider different traditions or cultural backgrounds. However, in future, the framework can be easily extended to encompass these more general cases. Section 4.1 presents the architectural considerations and the data used. Section 4.2 describes the specific design choices made for the GA. Finally, Sect. 4.3 describes the integration of a nearest neighbourhood-based mechanism, inspired by collaborative filtering, which is used in the mutation operator.

4.1 Architecture, inputs, and data source

We implement an evolutionary model following the description in Sect. 3.1. The inputs are: *physical status* and *exercising habits* (see Table 1), and *well-being goal*: (1) losing weight, (2) maintaining weight, (3) gaining weight, and (4) building muscle mass. Data on 166 food items allocated in 14 food types and 50 physical activity items (PAs) allocated in 8 types are taken from Health Canada (2008) and Arizona State University (2011), respectively (see Table 2), using the following preference categories (expressed using a 5-point numerical scale):

Table 2 Dataset structure

Food	Data type
Name	String
Main ingredient, vegetarian, vegan	Boolean
Serving size, kilocalories, protein, carbohydrate, sugar, fibre, fat, saturated fat, sodium	Float
Physical Activity (PA)	
Name	String
Intensity	Int. {0:low, 1:mod, 2:high}
MET (Metabolic Equivalent of Task), Duration (minutes)	Float

Food values (in grams) adopted from Health Canada (2008); Physical activity MET values taken from Arizona State University (2011)

Algorithm 1 Genetic Algorithm

```
1: procedure GA(popSize, maxGen)
2:     initialise pop[] of popSize individuals        ▷ See Section 4.2.1
3:     evaluate individuals                           ▷ See Section 4.2.2
4:     obtain best individual                         ▷ Individual with minimum aptitude value
5:     for gen 0 to maxGen do
6:         newPop[] ← execute tournament              ▷ See Section 4.2.3
7:         execute crossover over newPop[]            ▷ See Section 4.2.4
8:         execute mutation over newPop[]             ▷ See Section 4.2.4
9:         pop[] = newPop[]                           ▷ New population replaces old, for next gen
10:        evaluate individuals
11:        obtain best individual
12:    end for
13: end procedure
```

– *Food*: bread, cereal, dairy products, egg, fish, fruits, grains, legumes, meat, nuts, pasta, poultry, seafood, vegetables.
– *PA*: balance (yoga, tai-chi, etc.), bicycling, conditioning (cardio, gym workouts, etc.), dancing, running, sports, walking, water activities.

Previous users' data were also collected from an initial survey of 145 users, where each provided their physical status along with their food and activity preferences. This dataset is *only* used for finding users with similar preferences during the collaborative filtering stage (see Sect. 4.3). The output of this evolutionary model is the best evaluated individual, comprising K meal-PA bundle recommendations.

4.2 Evolutionary specifications of the implemented model

This subsection describes the specific design choices made for the GA components in the model implemented, based on the general framework guidelines introduced in Sects. 3.2 and 3.3. The principal element of EvoRecSys is a genetic algorithm (see Algorithm 1), which we describe below.

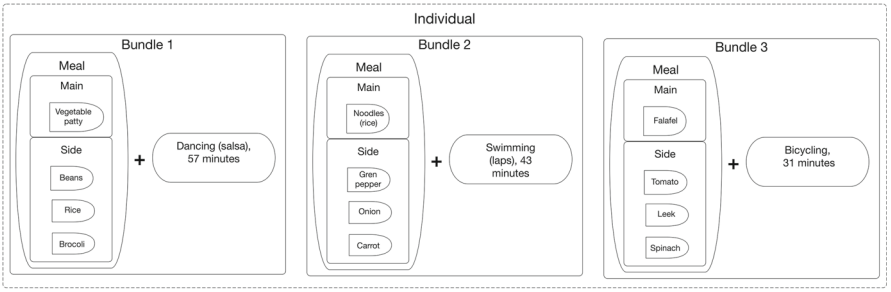

Fig. 4 Example of an individual in the implemented model, containing $K = 3$ bundles

Table 3 PAL values and their associated description

PAL	Description	Example job
1.2	Not very active	Office work
1.4	More or less active	Teacher
1.6	Active	Waiter/waitress
1.8	Very active	Builder

4.2.1 Creation of individuals

In this model, a meal contains four food items ($N = 4$). To ensure semantic consistency of portions, and to allow for finding diverse and non-repetitive recommendations, each meal contains two food types: (i) a single *main* food item and (ii) three *side* food items. Without loss of generality, we consider that each individual contains three bundles ($K = 3$), as shown in Fig. 4.

A relevant parameter that influences the interrelationship between the meal and the PA being jointly recommended is the number of intake calories that the user should consume per day. In order to calculate the target value, we use the Harris and Benedict equation due to the proven trustworthiness of its predictions in existing health-related literature (Lee and Kim 2012). Basal Metabolic Rate (BMR) is measured, as follows:

$$BMR_{male} = 655.1 + (9.563 \times weight) + (1.850 \times height)\check{}(4.676 \times age) \quad (2)$$

$$BMR_{female} = 66.5 + (13.75 \times weight) + (5.003 \times height)\check{}(6.755 \times age) \quad (3)$$

where weight is measured in kilograms, height is in centimetres, and age is in years. Total energy expenditure (TEE) is then obtained by multiplying BMR by the physical activity level (PAL) (Shetty et al. 1996):

$$TEE = BMR \times PAL \quad (4)$$

Table 3 shows the possible values for PAL according to the activity level which is asked to every user of our model:

Finally, the calculated TEE value is tailored according to the chosen goal. In this implementation, the set of available goals are: (i) losing weight; (ii) maintain-

ing weight; (iii) gaining weight; and (iv) gaining muscle mass. Under the evidenced assumption that a kilogram of body fat contains 7717.75 kilocalories (Wishnofsky 1958), it is feasible to estimate the number of required intake calories for some of the well-being goals previously defined. For instance, if a user reduces 551.26 intake kilocalories in the daily TEE, in 7 days the user would lose approximately 500 g of weight.

The resulting output represents the maximum number of kilocalories that a meal should have and it will be used to calculate the suggested time that should be spent on the PA, considering the chosen well-being goal. In other words, this value helps to determine the segments from the search space that fulfil the user PA preferences, her/his nourishment requirements and her/his goal during the stochastic process of creation of the population, excluding those that do not fulfil.

Remark 1 Although meal and exercise activities are both generated stochastically, they have a linking parameter in common, namely the tailored number of intake calories associated with the target user.

4.2.2 Evaluation of individuals

This model implements three specific restrictions that compose the fitness function to evaluate individuals (see Table 4). Let $\mathcal{R} = \{\mu_{hf}, \mu_{PA}, \mu_{cd}\}$ be the set of all restrictions to consider. A fitness function FF_i associated with a user $u_i \in U$ with a goal $\mathcal{G}_i$, is defined based on $\mathcal{R}$ and her/his individual food-exercising preferences Ψ_{u_i}. It assesses the degree that a recommended bundle simultaneously meets $\mathcal{G}_i$ and the user preferences.

$$FF_i = FF(\Psi_{u_i}; \mathcal{G}_i) = \phi\left(\Psi_{u_i}; \Gamma_i(\mu_{hf}), \Gamma_i(\mu_{PA}), \Gamma_i(\mu_{cd})\right) \tag{5}$$

where ϕ is the arithmetic mean averaging operator. The three restrictions are described as follows:

- The *Healthy Food Restriction* follows the England Government Dietary Recommendations (England 2016). Based on this, the restriction evaluates independently the amount of proteins, carbohydrates, sugar, fibre, fat, saturated fat, and salt in a meal.
- The *Exercising Restriction* evaluates the matching degree between the recommended time in the PA item and the average time that the user spends during the exercising time. This helps, for instance, to ensure that a PA for a given user is neither too mild, nor too ambitious or intense for her/him. We use the MET as the reference value to ensure that the meal-PA combination aligns with intended user's well-being goal.
- The *Consistency and Diversity Restriction* evaluates the food item diversity from two approaches: firstly, it evaluates the diversity in a single meal (among the food items that conforms the single meal) and secondly, it evaluates the diversity among meals within an individual. It also evaluates the diversity among exercising items within an individual. Moreover, this restriction evaluates, in terms of serving size,

Table 4 Set of restrictions used in EvoRecSys

Restriction name	Description	Notation
Healthy food	Evaluates the degree of healthiness of each nutrient that conforms a meal. It is internally defined upon the well-being goal	μ_{hf}
Tailored PA	Evaluates the matching degree between a PA item and her/his habits	μ_{PA}
Consistency & diversity	Evaluates the semantic consistency in a meal and the diversity of both meal and PA within the individual	μ_{cd}

how well-proportioned the food items are in each meal. This helps preventing too similar bundles within the same individual.

Finally, in this EvoRecSys instance, Ψ_{u_i} has been implemented as an additional restriction whose purpose is to evaluate how likeable are both the recommended meals and the recommended PA's, based on the user preferences. Thus, once all four restrictions are employed, the aptitude of the individuals is calculated as follows:

$$FF_{i,x} = (\mu_{hf_x} + \mu_{PA_x} + \mu_{cd_x} + \Psi_{u_i,x})/4 \tag{6}$$

Remark 2 The fitness function of the GA focuses on minimising error. The values that it yields are normalised to the range [0.0, 1.0]

4.2.3 Selection

We use *tournament selection* (Zhang and Kim 2000), which works as follows. First, a pair of individuals are randomly sampled from the population, with replacement. The aptitude values of the two individuals are compared, and the individual with the best aptitude (the *lower* value in this implementation) is selected and added (as an "offspring") to the new population. The process repeats N times (where N is the population size), until a new offspring population is formed with size equal to the parent population. Tournament size T directly controls selection pressure in the population. Note that, on average, since we are using a tournament of size $T = 2$, we expect: the best member of the parent population to have, on average, $T = 2$ offspring in the new population; the median member of the parent population to have, on average, $T/2 = 1$ offspring; and the lowest aptitude member is guaranteed to have no offspring. We also include *elitism*, such that we ensure that the best individual of each generation is reproduced (without modification) into the new offspring population. This ensures that good solutions are not lost during the reproduction process.

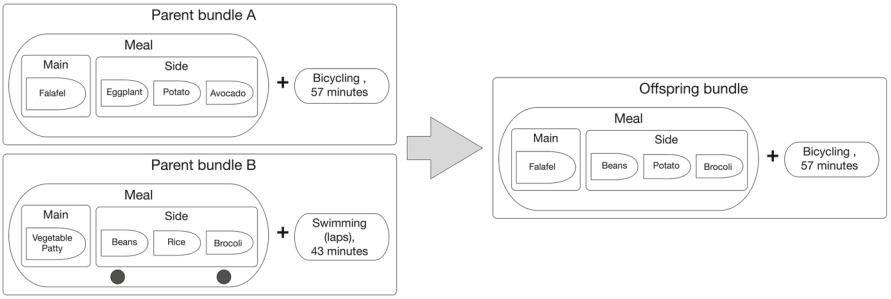

Fig. 5 Example of the crossover operator mechanism over a bundle

4.2.4 Genetic operators: crossover and mutation

Crossover and mutation follow a stochastic process and occur at the element level of each bundle. During crossover and mutation, each element is selected using the following method: (i) for each bundle, the bundle is selected with probability $p_0 = 0.9$; then, (ii) one element within the bundle $\{Main, Side, PA\}$ is selected with probability $\{0.2, 0.6, 0.2\}$; (iii) if *Side* is selected, each sub-element (each side meal) is selected with probability 0.5, enabling the possibility of multiple sides to be selected.

The *crossover* operator is used to recombine genetic code (the *items*) between two "parent" individuals, A and B (see example in Fig. 5). An offspring (i.e. a child) is created as a copy of parent A then, using the crossover process described above, for parent B, each element (or sub-element) that is selected will be inserted into the child. In the example shown in Fig. 5, the child is a copy of Parent A, with two side meals ("beans" and "broccoli") copied from parent B.

The *mutation* operator is directed by *collaborative filtering* (detailed fully in Sect. 4.3). First, "similar" neighbours are discovered using collaborative filtering over user preferences; then, when a mutation occurs, the element or sub-element selected is replaced by the corresponding element in the neighbouring user. Figure 6 shows an example of mutation in bundle number 2, for the element PA, which is replaced by the exercise activity "Yoga, 59 minutes", taken directly from a neighbour with similar preferences.

Remark 3 Using collaborative filtering within the mutation operator is non-standard. This novel contribution is designed to heuristically navigate through the population search space, guided by neighbours' preferences.

4.3 Directing evolution using nearest-neighbour collaborative filtering

In a collaborative filtering RS, items are typically recommended to a given user based on the preferences of similar users to her/him (Alhijawi et al. 2016; Karabadji et al. 2018). In essence, if u_a and u_b are similar users, and u_b has positively rated or liked an item x_j not seen by u_a yet, then x_j is likely to be recommended to u_a. Accordingly, our proposed model incorporates a strategy inspired by collaborative filtering in the core

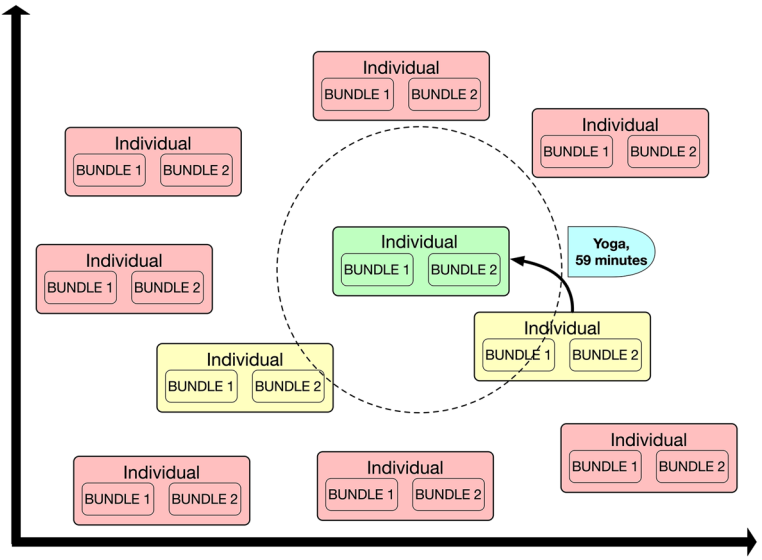

Fig. 6 Mutation operator example. K = 2 nearest neighbours are discovered using collaborative filtering (see Sect. 4.3). Then, one nearest neighbour is selected at random and the bundle element of the neighbour ("Yoga, 59 minutes") replaces the individual's element. (Color figure online)

GA that identifies similar users to the target user. Due to the multi-objective nature of our evolutionary approach, we consider a holistic notion of similarity among users that does not only consider their taste towards food and PA, but also their physical characteristics (weight, height, age, gender) and their selected well-being goal. For example, two users who have very similar food preferences but exhibit different physical characteristics and opposing goals, e.g. *losing weight* versus *gaining weight*, are unlikely to be considered similar.

To reflect this holistic view, we quantify similarity $sim(u_a, u_b)$ between users $u_a, u_b \in U$, using: (i) food preferences; (ii) PA preferences; (iii) physical status; and (iv) well-being goal. Let $FT = \{ft_1, ft_2, \ldots\}$ be a non-empty finite set of food types and let $\mathbf{p}_a^{ft} = [p_a^{ft_1} \; p_a^{ft_2} \ldots p_a^{ft_{|FT|}}]$ be a vector describing u_a's preferences towards food types. Then, the food-based similarity between u_a, u_b is computed using the following formula:

$$sim^{ft}(u_a, u_b) = 1 - d(\mathbf{p}_a^{ft}, \mathbf{p}_b^{ft}) \tag{7}$$

with $d(\cdot, \cdot)$ a normalised distance metric between two vectors, e.g. Euclidean distance. Let $AT = \{at_1, at_2, \ldots\}$ be a non-empty finite set of PA types. Accordingly, let $\mathbf{p}_a^{at} = [p_a^{at_1} \; p_a^{at_2} \ldots p_a^{at_{|AT|}}]$ be a vector describing u_a's preferences towards such PA types. The PA-based similarity between u_a, u_b is computed as follows:

$$sim^{at}(u_a, u_b) = \cdots 1 - d(\mathbf{p}_a^{at}, \mathbf{p}_b^{at}) \tag{8}$$

The user's physical status is modelled after the attributes employed to calculate the standardised calorie expenditure function: height, weight, age, and gender. Formally, we have $Status_a = [weight(kg),\ height(cm),\ age(yr),\ gender(m/f)]$. The rationale is that two users with similar physical status and activity levels will have a similar calorie expenditure rate, and therefore, the interrelationship between their food intake and exercise requirements in the recommended bundle should is similar. Based on their TEE value [Eq. (4)], we use the following formula to calculate the similarity between two users' physical status:

$$sim^{st}(u_a, u_b) = 1 - \frac{|TEE_a - TEE_b|}{TEE_{max} - TEE_{min}} \tag{9}$$

Finally, an aggregation function Φ_W is used to combine the three similarities on food preferences, PA preferences, and physical status, into one:

$$sim(u_a, u_b) = \Phi_W(sim^{ft}(u_a, u_b), sim^{at}(u_a, u_b), sim^{st}(u_a, u_b)) \tag{10}$$

with W a weighting vector for adjusting the relative importance of food preference, PA preference, and physical status. Finally, the selected *well-being goal*, is used to apply a "rewarding effect" on the aggregated similarity if the two users share the same well-being goal, thereby making users with a common goal more likely to be nearest neighbours of each other:

$$sim'(u_a, u_b) = \begin{cases} \sqrt{sim(u_a, u_b)} & \text{if } u_a, u_b \text{ have the same well-being goal,} \\ sim(u_a, u_b) & \text{otherwise.} \end{cases} \tag{11}$$

Intuitively, since $0 \leq sim(u_a, u_b) \leq 1$, we have $\sqrt{sim(u_a, u_b)} \geq sim(u_a, u_b)$. A simple k-nearest neighbour strategy is then applied to identify the k most similar users to u_a based on $sim'(u_a, u_b)$. Information about the preferences and needs of these neighbours is used to direct the mutation operator of our evolutionary process, leading to more diverse and meaningful personalised recommendations by further exploring the search space.

5 GA Performance analysis

Here, we analyse, optimise, and benchmark the performance of the genetic algorithm used in EvoRecSys.

5.1 Finding suitable aptitude and semantic coherency of recommendations

An essential aspect in the proposed EvoRecSys implementation is to have a comprehensive understanding of the fitness value, which indicates the quality and semantic coherency of the recommendations. In order to demonstrate how to interpret a fitness value, we illustrate using an example based on a vegetarian user with 1925 calories intake per day, which yields 642 calories per meal. This example user spends 43 min

Table 5 Aptitude values of example bundles for a vegetarian user with 1925 calories intake per day (642/meal), 43 min per exercise session, and goal of "losing weight"

Apt.	Bundle content	Observations
0.6406	F: noodles (46 g), cucumber (1325 g), cucumber (1624 g), bread (28 g). PA: Pilates for 85 min	Repeated side items exceeding 1 kg. Main item portion is poor. PA duration is nearly twice target value
0.3462	F: soft tofu (601 g), leek (698 g), walnuts (6 g), bread (20 g). PA: bicycling for 46 min	Meal serving size exceeds 1 kg. One side food exceeds 0.5k g; whereas another food item is almost 0 g
0.2481	F: noodles (250 g), bread (69 g), mixed nuts (5 g), quinoa (20 g). PA: swimming for 45 min	No repeated items. Serving sizes are more adequate. PA suggested time is close to the target
0.2473	F: rice noodles (258 g), quinoa (82 g), Italian bread (67 g), lentils (86 g). PA: running for 42 min	Serving sizes are coherent. PA suggested time is almost the same as the target time

Table 6 GA parameter settings with best performance

Parameter	Label	Value
Number of individuals	*popSize*	250
Number of generations	*maxGen*	150
Crossover probability	*probCross*	0.6
Mutation probability	*probMut*	0.1

per exercise session and has "losing weight" as well-being goal. Table 5 shows examples of fitness values (from worst to best aptitude) for bundles tailored to this example user. Using Table 5, we consider **0.2480** as an acceptable fitness threshold for assuring semantic coherency in the output recommendations.

Remark 4 Table 5 shows that small variations in fitness values may yield considerable changes in the quality of the output recommendations. This signals that the fitness function is sensitive and nonlinear.

5.2 Finding optimal parameters for the genetic algorithm

Since this implementation of EvoRecSys has been deployed as a Web application hosted in a domestic-use machine (see Sect. 6), the evolutionary process must execute in real time. We therefore conducted experiments to find optimal GA parameter values that can consistently reach the required aptitude threshold. We ran 50 repeated evolutionary trials over the following parameter space: $popSize = \{10, 20, \ldots, 50, 100, 150, \ldots, 300\}$; $maxGen = \{100, 150, \ldots, 300\}$; $probCross = \{0.1, 0.2, \ldots, 1.0\}$; $probMut = \{0.1, 0.2, \ldots, 1.0\}$. The best parameter values, which we use from now on, are shown in Table 6.

Figure 7 presents the mean performance of the best individuals across 50 trials (i.e. the mean system performance; with shaded region showing 95% confidence interval).

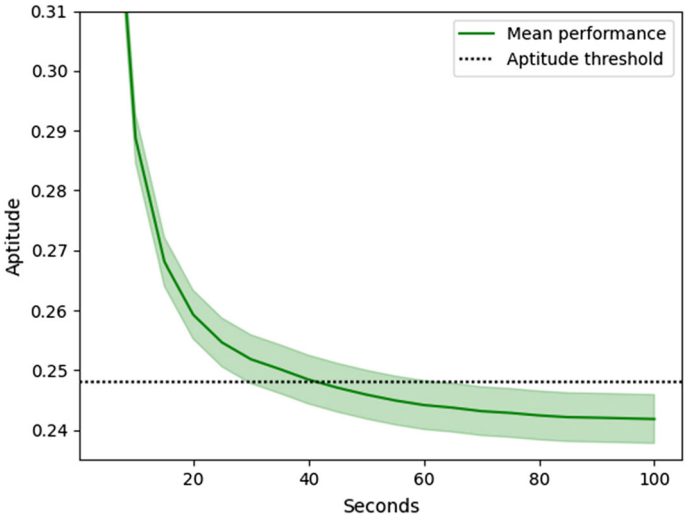

Fig. 7 System performance across 50 trials. Green line indicates the mean performance and the shaded area presents the confidence interval (95%). The aptitude threshold (0.2480) is shown as dotted line. (Color figure online)

The horizontal dotted line represents the fitness threshold we require for semantic coherency (see Sect. 5.1). Although we can be confident that EvoRecSys will reach the desired aptitude threshold after 60 s, the system continues to improve and does not equilibrate until 80 s. Therefore, we consider 60 s as the minimum computational time required to build coherent recommendations (on the given hardware); but to ensure the best possible recommendations, we use a run time of 80 s when building recommendations during the user study (Sect. 6). We believe the performance improvement is worth the additional 20 s that each user must wait, and also since more powerful hardware would reduce the run times, we focus on producing the best quality recommendations rather than minimising wait time.

Remark 5 All GA trials were conducted using a standard domestic-use machine. Deploying the model on high-performance hardware would significantly reduce run times.

5.3 Benchmarking

Here, we benchmark the performance of our proposed algorithm. In particular, since our use of collaborative filtering in the mutation operator is novel, we are interested in quantifying the benefit that this process brings. To achieve this, we compare four approaches, each containing different components of the model. These are:

1. *EvoRecSys-naïve*: Rather than calculate the number of intake calories (see Sect. 4.2.1), individuals are created by choosing a random number of intake calories within the range [4, 5000]. Both genetic operators are enabled, however the

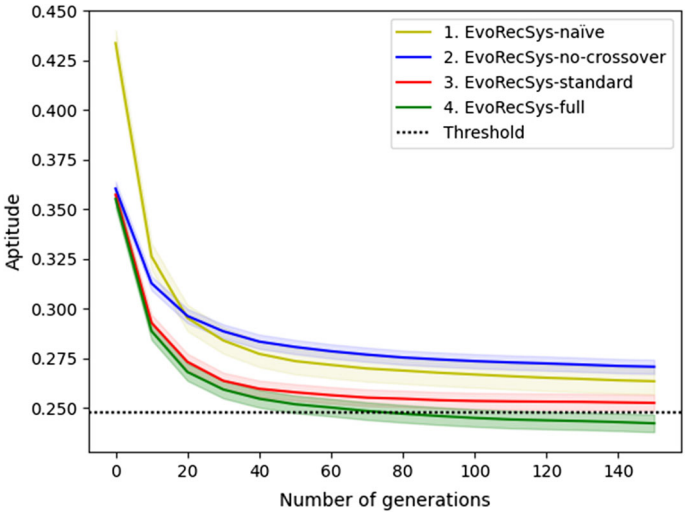

Fig. 8 Benchmark of the four EvoRecSys modified instances. Mean performance ± 95% confidence interval (shaded region). (Color figure online)

mutation operator works under the "standard" approach such that a randomly selected item is replaced by another item of the same class (i.e. a side replaces a side; a main replaces a main) that is randomly selected from the database (i.e. collaborative filtering is *not* used in the mutation operator).

2. *EvoRecSys-no-crossover*: In this baseline approach, the crossover operator is disabled. Thus, the evolutionary task relies exclusively on the mutation operator, namely guided by the nearest-neighbour collaborative filtering strategy described in Sect. 4.3 and illustrated in Fig. 6. In addition, individuals are built considering the value calculated by the process described in Section 4.2.1.
3. *EvoRecSys-standard*: Both genetic operators are enabled. However, the mutation operator works under the standard approach, as described in approach (1). Furthermore, individuals are created by the procedure described in Sect. 4.2.1.
4. *EvoRecSys-full*: The proposed implementation in this paper. This approach has no modifications; it includes crossover and the collaborative-filtering-based mutation operator, as described in previous sections.

We conducted 50 trials on each approach. Figure 8 presents the mean performance of the best individual (the lowest aptitude value) across all trials, with 95% confidence interval presented as shading. While there are relatively small differences in best aptitude between each approach, these differences translate into significant differences in coherency of recommendation (see Remark 4). We see that the "full" system (4), which includes both crossover and mutation directed by collaborative filtering, produces the lowest aptitude values (see Remark 2), which indicates that it performs best. In particular, (4) significantly outperforms the "standard" approach (3) (paired t-test, $p < 0.0001$), indicating that directing mutation using collaborative filtering is beneficial. Approach (3) also significantly outperforms approaches (1) and (2) (paired t-test, $p < 0.001$). The "naïve" approach (1) appears to tend towards better performance

values than the "no crossover" approach (2); however, this difference is not significant (paired t-test; $p > 0.05$). Approach (1) starts poorly because of the randomised configuration of the initial population, with mean aptitude of 0.4337 at generation 0. However, the addition of the crossover operator enables approach (1) to quickly catch and then slightly overtake the performance of approach (2). In summary, these results show that both crossover and mutation with CF are *necessary* components for the system to perform best and are the *only* configuration that consistently reaches the desired aptitude threshold.

6 User study

Based on the previous experimental evaluation to determine an optimal configuration of our EvoRecSys Web implementation,[7] we deployed it to conduct a cohort study with users who volunteered to interact with the system. The study provides additional insight about the system performance from the subjective perspective of end users, analysing their response towards recommendations.

It is important to note that the Web front-end used for both user studies was designed to provide the GA with a default value in cases where the user, whether deliberately or accidentally, skipped a question related to food or physical activity preferences. We set a default value of 3, which corresponds to the neutral preference in a 5-point numerical scale. Thus, recommendations are generated even if the user skips all preference-related questions.

6.1 Subjective analysis of EvoRecSys recommendations

Volunteers were invited to conduct a series of interactions with the system throughout three steps, for approximately 10 min:

i. Providing explicit rating information about food/PA preferences, physical status, exercising habits, and well-being goal (see Fig. 9a, b).
ii. Receiving a list of three bundle recommendations and evaluating their satisfaction with each one (see Fig. 9c).
iii. Assessing overall perception of recommendations received based on four criteria: diversity, serendipity, appeal, and healthiness.

A total of 205 users completed the three tasks. A geographical distribution of the country from where these users participated is shown in Table 7, and Table 8 summarises their demographic and physical characteristics along with their exercising habits and well-being goal.

For each of the three meal-PA recommendations received, users were asked to rate the suggested meal and exercise (see Fig. 9c) using a 5-point Likert scale, and to optionally mark one of more of the recommended combinations as favourite. Figure 10 shows, on the left, the average user satisfaction with individual meals and PAs suggested alongside their standard deviation. The last two bars show the average results

[7] Open-source code is available, https://github.com/oguh1-61803/evorecsys.

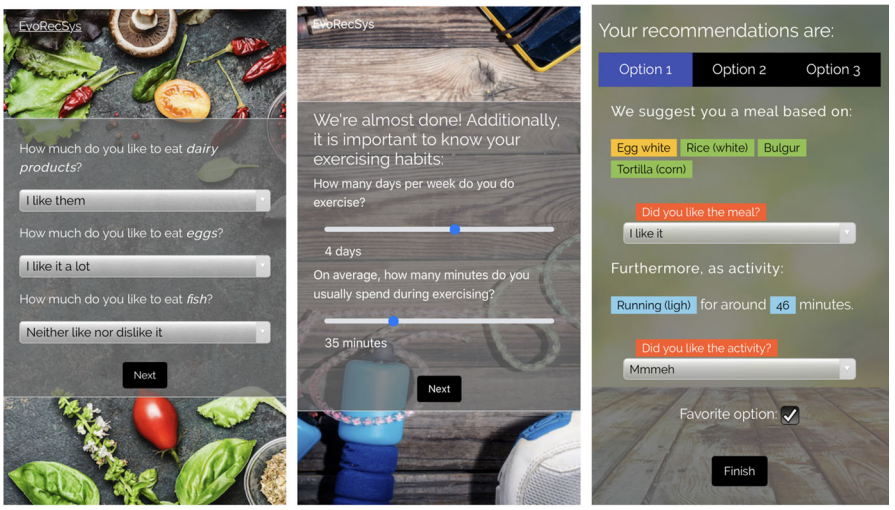

Fig. 9 EvoRecSys interface for the user study: **a** eliciting food preferences, **b** eliciting exercising habits, **c** providing bundle recommendations for their evaluation by the user

Table 7 Geographical distribution of 205 volunteer users for the study

Country	Number of participants
Brazil	4
China	9
Mexico	80
Netherlands	4
Others	11
Spain	46
UK	7
US	44

Table 8 Demographics, physical status, PA habits, and well-being goal of participants

Gender	Age	Height (cm)	Weight (kg)	PA (days/wk)	PA (min/day)	Goal
F: 123	14–17: *6*	<150: *6*	<50: *10*	0: *17*	<15: *16*	WL: *101*
M: 82	18–29: *94*	150–159: *63*	50–70: *103*	1: *24*	15–30: *17*	WM: *48*
	30–39: *65*	160–169: *67*	70–90: *65*	2: *38*	30–45: *72*	WG: *5*
	40–49: *29*	170–179: *58*	90–110: *26*	3: *55*	45–60: *28*	MB: *51*
	≥50: *11*	> 180: *11*	>110: *6*	4: *24*	60–75: *49*	
				5+: *47*	>75: *22*	

WL weight loss, *WM* weight maintenance, *WG* weight gain, *MB* muscle build

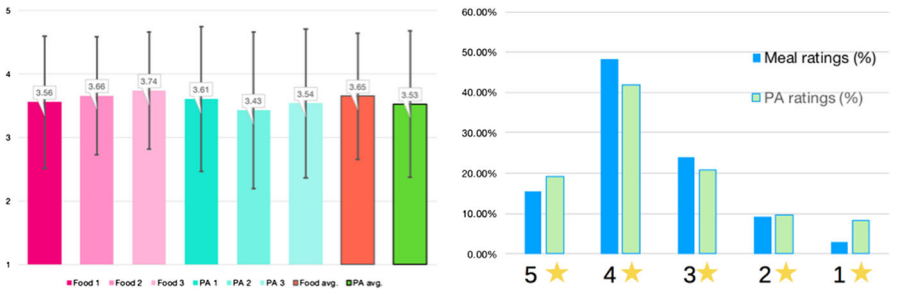

Fig. 10 *Left* Average user satisfaction with $K = 3$ meal and PA recommendations in a 5-point scale. *Right* Percentage distribution of ratings given by users to single meals and single PAs in recommendations received

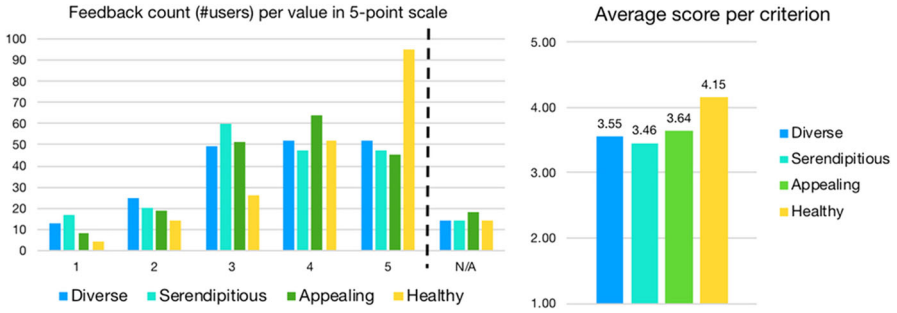

Fig. 11 *Left* Rating distribution (number of users) of values in the 5-point feedback scale, considering four criteria for evaluating recommendations. *Right* Average rating provided by users to recommendations on each criterion

across all three meals (resp. PAs). The plot on the right-hand side of Fig. 10 shows the overall distribution of 1-to-5 ratings given by users to the recommendations. The results show, in general, a prevalence of positive ratings over negative ones, particularly towards meals, showing slightly better values than PA in terms of both average ratings and rating distribution. Deviations around the average value are shown as consistent between meal and PA being recommended, suggesting that there is a similar consensus between both aspects in terms of users' perception of the recommendations.

Finally, in order to assess the perception of recommendations received "as a whole", users were requested their subjective opinion regarding four quality criteria using again a 5-point scale with values ranging between 1 and 5: (i) *diversity*, where higher ratings mean more diverse and less repetitive recommendations, (ii) *serendipity*, with higher ratings meaning more serendipitous and less expected recommendations, (iii) *attractiveness*, with higher ratings meaning more appealing recommendations that suit their preferences, and (iv) *health* with higher ratings indicating that recommendations are perceived as healthier. These final questions were optional, hence not all users answered all four of them. Figure 11 summarises the feedback collected for the four questions as a rating distribution (left) and the average score per question/criterion (right).

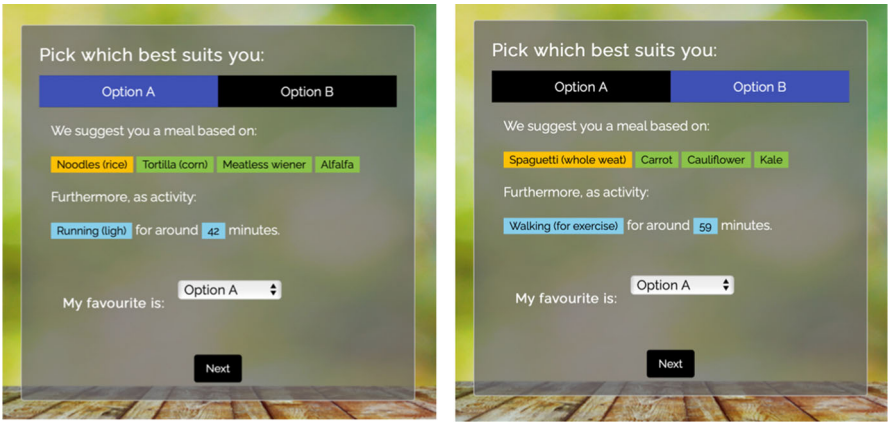

Fig. 12 Challenge study example. User selected option A: EvoRecSys recommendation

We believe these are promising results for various reasons. Firstly, all four rating distributions show a moderately skewed trend towards higher ratings, demonstrating that the average ratings obtained are good representatives of a minority of negative feedback, with no polarised majority opinions around the two extremes of the rating scale. Health is the most positively assessed criterion by most users, with a significantly higher average rating than the other three (4.15). It is also the only criterion in which the majority of users gave the highest rating, and hence, the proposed model succeeds in delivering meal-PA bundles perceived by users as healthy. Most users reported recommendations as appealing (4) or very appealing (5), which is also an encouraging result in terms of balancing healthy recommendations with adaptability to the user preferences. Diversity and serendipity show, in average, slightly closer results to the neutral value (3), although the majority of ratings are still distributed across the {3, 4, 5} rating interval. This suggests that while the collaborative filtering approach integrated in the GA helps producing diverse and serendipitous recommendations, there might still be areas for improving these aspects in future versions of the model or in new ones, motivating a more thorough exploration of the GA components, its fitness function, and any other RS techniques to be investigated and integrated in EvoRecSys.

6.2 Challenge study: EvoRecSys vs. collaborative filtering

Following the first study, 44 volunteers accepted an invitation to take part in a follow-up study to subjectively compare recommendations generated by EvoRecSys and recommendations generated by the second baseline system used in Sect. 5.3, which can be understood as collaborative filtering (CF) only. Users begin by entering their preferences (see Fig. 9), and are then shown 5 pairs of "blind" recommendations (e.g. see Fig. 12), one generated by EvoRecSys and one generated by CF alone. For each pair, the user is then challenged to select their *preferred* recommendation, without being told how the recommendations are generated. The ordering (A or B) of pairs

is randomly shuffled between EvoRecSys and CF to ensure that there is no selection bias based on ordering of options shown.

For this stage of the user study, EvoRecSys was configured to recommend one bundle, using parameters $popSize = 150$, $maxGen = 100$, $probCross = 0.6$ and $probMut = 0.1$. For the CF-recommendation, we initially created a population of individuals. The best individual in this initial population (generation 0) is then taken, and the CF-based mutation operator is applied (see Sect. 4.2.4). In this way, CF-recommendations are built using collaborative filtering, but without evolutionary optimisation.

In total, we conducted 220 pairwise challenges ($n = 44 \times 5 = 220$). The recommendation option generated by EvoRecSys was preferred 124 times, while the recommendation generated by CF was preferred 96 times. If we consider the null hypothesis that recommendations generated by each system are equally likely to be selected by users, then we can test this hypothesis by using a binomial distribution with $p = 0.5$ (probability of each option being selected at random), $n = 220$ (number of repeated trials), and $x = 124$ (number of times that EvoRecSys option is selected). We get probability $P(X \geq x) = 0.034$. Therefore, results suggest that EvoRecSys recommendations are preferred by users and we are able to reject the null hypothesis at the 0.05 significance level.

7 Discussion and lessons learnt

Our efforts to reformulate the recommendation problem as a multi-objective optimisation problem driven by a GA can be summarised as successful in the light of the experimental results. In general terms, recommendations have been positively rated by the majority of users who participated in the study. Furthermore, after a careful experimental setting up of the model parameters, the model achieved recommendations that are *tailored, consistent* to integrated knowledge and domain guidelines, *diverse*, and *acceptable*. All of these are fundamental requirements to meet according to the extant RS foundations (Aggarwal 2006). On the other side, although we showed an implemented model founded on specific design decisions, it must be noted that EvoRecSys deserves further exploration of other recommender principles and user preference/interaction aspects left outside the scope of this work. This, together with the results of our study, suggest that the EvoRecSys framework and its conceptual architecture should be subject to further study by the research community, thereby opening new pathways of research within the field of recommender systems for health and/or based on evolutionary computing.

Although the proposed framework and model have reported favourable results, they constitute to the best of our knowledge the first research efforts for health RS in this direction. Consequently, several challenges and areas for improvement have been identified during the framework design, model development, and experimental studies. The most relevant such directions are:

1. *Complementary datasets and interpretable recommendations*: One of the proven strengths of EvoRecSys is its GA ability to construct configurable recommen-

dations that accurately adapt to the users' needs and preferences, personalising fine-grained aspects such as serving sizes in meals and PA duration. However, these recommendations—specifically the suggested meals based on food items—may sometimes be less interpretable than, for example, recommending a recipe (Musto et al. 2020). For this reason, an immediate aspect deserving study is how to incorporate datasets that facilitate more meaningful food recommendations such as recipes, ready meals from a supermarket, regional food, or specific groceries. An interesting question to study here is the effect of bridging precise and highly optimised meals generated by EvoRecSys with static but more understandable recipes/products (e.g. from third-party datasets) that are similar. This would also help developing bespoke models focused on determined demographic sectors.

2. *Highly configurable and diverse components:* Experiments on the GA parameter settings have demonstrated the importance of semantic coherency criteria to guarantee higher aptitude and quality in recommendations. Due to the nature of the techniques used at the core of the EvoRecSys framework, it is possible to flexibly define the architecture of the recommender engine. Based on this feature, more semantic rules such as compatibility among ingredients can be implemented in order to ensure coherency and diversity from the deepest level (food items within a bundle), to the highest level (food items between bundles). Furthermore, it is possible to add/remove new food item categories. For instance, desserts could be incorporated for building a more robust recommendation for two or three-course meals.
3. *Implicit dynamic data acquisition:* Additionally, improving how the users' preferences are modelled in order to acquire a more accurate insight about the user preferences and habits would be possible. The model implemented in this study relies on preferences explicitly provided by the user during their initial interaction with the system. However, more reliable recommendations could be built: (i) by acquiring new forms of data dynamically and over time, e.g. via daily feedback of physical activity logs, and (ii) discovering how these recommendations may align with the preferences and personal needs stated by the user if a mechanism that learns from the user feedback and her/his evolving behaviour towards recommendations is incorporated. In line with this research direction, we also consider it equally important to define more objective evaluation metrics and criteria for experimentally validating the models developed, especially in terms of quantifying the extent to which recommendations align with stated and/or implicitly modelled users' preferences.

Another relevant aspect to consider is that all experiments were performed on standard commodity hardware. Thus, the thresholds arranged in Sects. 5.1 and 5.2 were partly dependent of the computational power of the equipment available. A dedicated server with high performance hardware would enable us to consider much lower efficiency thresholds and therefore more reliable recommendations. Moreover, fast responses in real time and parallel handling of multiple user requests would be possible. Nevertheless, the experiments made provide a suitable methodological approach to follow for the experimental configuration and validation of models built upon the EvoRecSys conceptual framework.

8 Concluding remarks and future work

In this research, we have introduced EvoRecSys, a novel conceptual framework for recommendations in health-related domains, entirely based on the premise that it is possible to strike a balance between three main dimensions: (i) what the user *prefers*, (ii) what the user *needs*, and (iii) what the user sets as a *goal*. The framework is characterised by defining an evolutionary algorithmic approach that establishes the balance of these three components through a multi-objective optimisation problem. A distinctive feature of the framework is its ability to build highly configurable items in the form of meal-exercising bundles to be recommended rather than immutable items, which allows more tailored and reliable recommendations for the user. The proposed framework architecture is defined to be flexibly instantiated into different model implementations for recommendation across different application areas of health and well-being. In this paper, we presented an implementation of EvoRecSys into a general purpose meal-physical activity recommender to help achieving well-being goals. However, the proposed conceptual framework guidelines may also help when building models for people with special needs such as patients with chronic diseases, professional athletes, or people whose cultural background only allows them to eat specific food. In all cases, when a model is built, we encourage a validation process by health experts to consider the well-being goals and recommendations provided by the system; in particular, users should seek medical assistance to reach goals when there is an illness present.

This study has delivered a first proof of concept where GAs are exploited as the core technique of a recommender system instead of being a complementary part of a recommender engine driven by other currently used techniques. As a consequence of the promising results obtained in this research, future work directions have been outlined. For instance, the inclusion of a dietary and exercising diary for the user is one of the main developments that will help to improve the robustness of this framework by obtaining a better insight of users' behaviour.

Additionally, a more flexible graphical interface will improve the user experience. For instance, the possibility of creating new bundles combining any of the recommended meals with any of the PAs built by the system. Regarding interpretation of recommendations (see Sect. 7), it can be difficult for users to intuitively compare food portions to the nearest gram and exercise to the nearest minute, therefore showing users an average food portion or a valid portion interval (e.g. to the nearest 50 g) and presenting valid time ranges for exercise (e.g. to the nearest 5 min), will improve the user experience.

On a last note, a native mobile application would provide more freedom in terms of the implementation of a more friendly user interface, possibly linked to wearable devices for seamless capturing of data. For instance, heart ratio and number of steps per day would allow us to learn more about the physical status and habits of the user and therefore the recommendations of physical activities would be more aligned with the physical activity in real time.

Acknowledgements Hugo Alcaraz-Herrera's PhD is supported by The Mexican Council of Science and Technology (Consejo Nacional de Ciencia y Tecnología - CONACyT).

Appendix: Algorithms

To enable replication, we present EvoRecSys algorithms in full. Algorithm 2 presents the general workflow of EvoRecSys.

Algorithm 2 EvoRecSys Algorithm

```
1: procedure EXECUTE EvoRecSys
2:     popize = 250                                          ▷ Number of individuals
3:     maxGen = 150                                          ▷ Number of generations
4:     userData                  ▷ Physical data, preferences and goal obtained by the web app
5:     TEE ← calculateTEE(userPhysicalData)
6:     userCalories ← calculateCalories(TEE, userGoal)       ▷ Algorithm 3
7:     initialise pop[popSize]                               ▷ Algorithm 4
8:     evaluate individuals(userPreferences)                 ▷ Algorithm 7
9:     obtain best individual                    ▷ Individual with minimum aptitude
10:    for gen 0 to maxGen do
11:        newPop[] ← execute tournament                     ▷ See Section 4.2.3
12:        execute crossover over newPop[]                   ▷ Algorithm 8
13:        execute mutation over newPop[]                    ▷ Algorithm 10
14:        pop[] = newPop[]                                  ▷ Replace old pop with new pop
15:        evaluate individuals
16:        obtain best individual
17:    end for
18: end procedure
```

First, we calculate the TEE value using Eq. (4) and user's physical data obtained through the web app. The TEE value and the fitness *goal* chosen by the user are passed as parameters to the function described in Algorithm 3 which calculates the number of tailored calories (see Sect. 4.1).

In order to lose approximately 500 g weekly, which is the minimum recommended without harm (Williamson et al. 1992), the allocated value for *calories* is 551.26 (see example in Sect. 4.2.1). In addition, this value is used if the user wants to gain weight. On the other hand, the allocated value of *muscle* is 0.15 because, on average, 15% of excess of intake calories is needed for a non-athlete person (Garthe et al. 2013). The returned value, *tailoredCal*, is then used by the genetic algorithm.

The step before the genetic algorithm starts consists in retrieving the food and PA data from the database. In this implementation, items whose preference value is 0 (the user neither eat it in the case of food items nor perform the physical activity) are not retrieved (see Sect. 4.1).

Algorithm 3 Calculate user's intake calories

```
1: procedure CALCULATE CALORIES(TEE, goal)
2:     meals = 3                                   ▷ Number of daily meals
3:     calories = 551.26                           ▷ Calories constant
4:     muscle = 0.15                               ▷ Muscle constant
5:     tailoredCal                                 ▷ Tailored calories for the user
6:     if goal = "losing weight" then
7:         tailoredCal = (TEE − calories)/meals
8:     end if
9:     if goal = "maintaining weight" then
10:        tailoredCal = TEE/meals
11:    end if
12:    if goal = "gaining weight" then
13:        tailoredCal = (TEE + calories)/meals
14:    end if
15:    if goal = "gaining muscle mass" then
16:        tailoredCal = (TEE + (TEE ∗ muscle))/meals
17:    end if
18:    return tailoredCal
19: end procedure
```

In Algorithm 2, Line 7, the population of size *popSize* is initialised, with each individual (containing B meal/PA bundles) created using the procedure described in Algorithm 4.

Algorithm 4 Create new individual

```
1: procedure CREATE INDIVIDUAL(foodItems, tailoredCal, PA_items, TEE)
2:     newIndividual                               ▷ Initialise a new individual
3:     for i = 0 to i = B do                       ▷ Create B bundles
4:         meal ← createMeal(foodItems, tailoredCal)   ▷ Algorithm 5; items from DB
5:         PA ← createPA(PA_items, TEE, weight, goal)  ▷ Algorithm 6; items from DB
6:         bundle ← insert(meal, PA)
7:         newIndividual ← insert(bundle)
8:     end for
9:     pop[] ← insert(newIndividual)               ▷ Insert into population
10: end procedure
```

The process focused on building a meal is described in Algorithm 5. The algorithm not only explores the search space in terms of food items, but also explores the search space in terms of portions. [40,60] is the range which is used to allocate a random percentage of the total number of calories to the main food item (see Line 3). The function called *tailorFood* (see Lines 6 and 12) creates a new food item interpolating the food item data received as the first parameter (see Sect. 4.1) using the number of calories received as the second parameter. Finally, *randomValues* (see Line 8) generates a list of three random numbers whose sum is the remain calories.

Algorithm 6 presents the process of building a PA. In order to preserve whether the excess number calories or the deficit of calories needed to achieve the chosen well-being goal, the TEE value is used rather than the value of *tailoredCal*. Similarly to the food tailoring function (see Algorithm 5, Lines 6 and 12), the function called

Algorithm 5 Create meal

```
1: procedure CREATE MEAL(foodItems, tailoredCal)
2:     new meal                                   ▷ Initialises a new meal
3:     mainPortion ← random(40, 60)               ▷ Random % of total calories
4:     mainCal = (tailoredCal ∗ mainPortion)/100
5:     randomMain ← random(foodItems)
6:     newMain ← tailorFood(randomMain, mainCal)
7:     remainCal ← tailoredCal − mainCal
8:     sideCalList ← randomValues(remainCal)
9:     newSideItems                               ▷ Initialises a new sides list
10:    for i = 0 to i = sideCalListLength do
11:        randomSide ← random(foodItems)
12:        newSide ← tailorFood(randomSide, sideCalList[i])
13:        newSideItems ← insert(newSide)
14:    end for
15:    newMeal ← insert(newMain)
16:    newMeal ← insert(newSideItems)
17:    return newMeal
18: end procedure
```

tailorPA (see Line 7) interpolates the PA data received as the first parameter using the recommended time as the second parameter for building a new PA item.

Algorithm 6 Create PA

```
1: procedure CREATE PA(PA_items, TEE, weight, goal)
2:     metRef = 60                                ▷ MET reference value for an hour
3:     randomPA ← random(PA_items)
4:     metValue ← gets(randomPA)
5:     burnedCal = metValue ∗ weight              ▷ Burned calories per hour
6:     recommendedTime = (TEE ∗ metRef)/burnedCal
7:     new PA ← tailorPA(randomPA, recommendedTime)
8:     return newPA
9: end procedure
```

Regarding the evaluation of individuals (see Algorithm 2, Line 8), Algorithm 7 describes the procedure that takes place. Each individual is evaluated under four approaches (see Sect. 4.2.2). Each approach (or restriction) provides a numerical value between [0.0, 1.0]. Once all evaluations are finished, the arithmetic mean is calculated using the outputs of the approaches [see Eq. (6)] and finally it is set on the individual.

Executing tournament (see Algorithm 2, Line 11) is fully described in Sect. 4.2.3.

Concerning the crossover operator (see Algorithm 2, Line 12), Algorithm 8 describes the procedure. Two random individuals from the population are taken to recombine their items under a probability *crossProb*. This task is repeated *popSize* times.

The core procedure of the crossover operator is called *combine items* (see Algorithm 8, Line 6). Algorithm 9 presents how said procedure works. Likewise the function called *tailor food* (See Algorithm 5, Lines 6 and 12), the function called *mixFood* (see Lines 14 and 21) builds a new food item interpolating the food data received as the second parameter using the number of calories contained in the food data received

Algorithm 7 Evaluate individuals

```
1: procedure EVALUATE INDIVIDUALS(userPreferences)
2:     for i = 0 to i = popSize do
3:         hValue ← healthyFoodRestriction(pop[i])
4:         eValue ← exercisingRestriction(pop[i])
5:         cdValue ← consistencyDiversityRestriction(pop[i])
6:         upValue ← userPreferencesRestriction(userPreferences, pop[i])
7:         individualAptitude ← (hValue + eValue + cdValue + upValue)/4
8:         pop[i] ← set(individualAptitude)
9:     end for
10: end procedure
```

Algorithm 8 Crossover operator

```
1: crossProb = 0.6                                  ▷ Probability that crossover takes place
2: for i = 0 to i = popSize do
3:     if random(0.0, 1.0) ≤ crossProb then
4:         P1 ← population[i]                       ▷ Select individual to apply crossover
5:         P2 ← random(population[])                ▷ Select random individual from population
6:         newIndividual ← combine items(P1, P2)    ▷ Algorithm 9
7:         population[i] ← insert(newIndividual)
8:     end if
9: end for
```

as the first parameter. Regarding the PA combination process (see Line 31), it only inserts the PA of the second parent ($P2$) to the new bundle, conserving the meal of the first parent($P1$)

Finally, the mutation operator (see Algorithm 2, Line 13) is described in Algorithm 10. An individual from the population is taken to swap its items under a probability *mutProb*. This task is repeated *popSize* times. Furthermore, the CF task takes place (see Sect. 4.2.4) and the output, which consists in both food items and PA items, is available (see Line 2).

The principal procedure in the mutation operator is called *swapItems* (see Line 6). Algorithm 11 presents the behaviour of the said procedure. The function called *swapMain* (see Line 10) replaces the current main item inside the individual by a random main item taken from the set of similar items, interpolating the data from the similar food item using the calories intake from the current item. In the same manner, *swapSide* (see Line 15) replaces a side food item. Regarding the function called *swapPA* (see Line 21), it replaces the current PA by a random PA taken from the similar items, interpolating the data from the similar item using the intake calories from the current item.

Algorithm 9 Combine items

```
procedure COMBINE ITEMS(P1, P2)
    bundleCrossProb = 0.9                          ▷ Probability to recombine a bundle
    mealPAProb = 0.8                               ▷ Probability to recombine a meal
    mainSideProb = 0.25                            ▷ Probability to recombine a main item
    newIndividual                                  ▷ Initialises the offspring
    for i = 0 to i = numBundles do
        if random(0.0, 1.0) ≤ bundleCrossProb then
            mixedBundle                            ▷ Initialise new bundle
            if random(0.0, 1.0) ≤ mealPAProb then
                mixedMeal                          ▷ Initialise new meal
                if random(0.0, 1.0) ≤ mainSideProb then
                    mainItem_P1 ← gets(P1)
                    mainItem_P2 ← gets(P2)
                    mixedMain ← mixFood(mainItem_P1, mainItem_P2)
                    mixedMeal ← insert(mixedMain, sideItems_P1)
                    mixedBundle ← insert(mixedMeal, PA_P1)
                else
                    mixSides                       ▷ Initialise new food sides list
                    for j = 0 to j = sideItems_P1 Length do
                        if random(0.0, 1.0) ≤ 0.5 then
                            mixSide ← mixFood(sideItems[j]_P1, sideItems[j]_P2)
                            mixSides ← insert(mixSide)
                        else
                            mixSides ← insert(sideItems[j]_P1)
                        end if
                    end for
                    mixedMeal ← insert(mainItem_P1, mixSides)
                    mixedBundle ← insert(mixedMeal, PA_P1)
                end if
            else
                mixedBundle ← insert(meal_P1, PA_P2)
            end if
            newIndividual ← insert(mixedBundle)
        else
            newIndividual ← insert(Bundle[i]_P1)
        end if
    end for
    return newIndividual
end procedure
```

Algorithm 10 Mutation operator

```
mutProb = 0.1                                      ▷ Probability that mutation takes place
similarItems                                       ▷ CF output: similar items (food and PA)
for i = 0 to i = PopSize do
    if random(0.0, 1.0) ≤ mutProb then
        individual ← pop[i]
        newIndividual ← swapItems(individual, similarItems)        ▷ Algorithm 11
        pop[i] ← insert(newIndividual)
    end if
end for
```

Algorithm 11 Swap Items

```
1: procedure SWAP ITEMS(individual, similarItems)
2:     bundleMutProb = 0.9                          ▷ Probability to recombine a bundle
3:     mealPAProb = 0.8                             ▷ Probability to recombine a meal
4:     mainSideProb = 0.25                          ▷ Probability to recombine a main item
5:     for i = 0 to i = number of bundles do
6:         if random(0.0, 1.0) ≤ bundleMutProb then
7:             if random(0.0, 1.0) < mealPAProb then
8:                 if random(0.0, 1.0) ≤ mainSideProb then
9:                     randomMain ← get(similarItems)
10:                    swapMain(individual, randomMain)
11:                else
12:                    for j = 0 to j = sideItemsLength do
13:                        if random(0.0, 1.0) ≤ 0.5 then
14:                            randomSide ← get(similarItems)
15:                            swapSide(individual, randomSide)
16:                        end if
17:                    end for
18:                end if
19:            else
20:                randomPA ← get(similarItems)
21:                swapPA(individual, randomPA)
22:            end if
23:        end if
24:    end for
25:    return individual
26: end procedure
```

References

Achananuparp, P., Weber, I.: Extracting food substitutes from food diary via distributional similarity. In: Proceedings of the 2016 Workshop on Engendering Health with RecSys—HealthRecSys'16 (2016)

Aggarwal, C.C.: Recommender Systems: The Textbook, 1st edn. Springer, Berlin (2006)

Akkoyunlu, S., Manfredotti, C., Cornuéjols, A.: Investigating substitutability of food items in consumption data. In: ACM Int. Conf. RecSys17 (2017)

Alhijawi, B., Kilani, Y.: Using genetic algorithms for measuring the similarity values between users in collaborative filtering recommender systems. In: ICIS'16 Int. Conf. (2016)

Arizona State University, H.L.R.C.: The adult compendium of physical activities and additional resources (2011)

Beliakov, G., Pradera, A., Calvo, T.: Aggregation Functions: A Guide for Practitioners. Studies in Fuzziness and Soft Computing, 1st edn. Springer, Berlin (2007)

Berndsen, J., Lawlor, A., Smyth, B.: Running with recommendation. In: ACM Int. Conf. RecSys17 (2017)

Buchely, M.F., Ganguly, S., Aken, D.C.V., O'Malley, R., Lekakh, S., Chandrashekhara, K.: Experimental development of Johnson-Cook strength model for different carbon steel grades and application for single-pass hot rolling. Steel Res. Int. **74** (2020)

Caldeira, J., Marinho, L., Oliveira, R.S., Trattner, C.: Healthy menus recommendation: optimizing the use of the pantry. In: ACM Int. Conf. RecSys18 (2018)

Cui, L., Ou, P., Fu, X., Wen, Z., Lu, N.: A novel multi-objective evolutionary algorithm for recommendation systems. J. Parallel Distrib. Comput. **103**, 53–63 (2017)

Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: Nsga-ii. IEEE Trans. Evol. Comput. **6**(2), 182–197 (2002)

Dragone, P., Pellegrini, G., Vescovi, M., Tentori, K., Passerini, A.: No more ready-made deals: Constructive recommendation for telco service bundling. In: Proceedings of the 12th ACM Conference on Recommender Systems, RecSys'18, pp. 163–171 (2018)

Dutta, D., Sil, J., Dutta, P.: A bi-phased multi-objective genetic algorithm based classifier. Expert Syst. Appl. **146** (2020)
Elliot, C.A., Hamlin, M.J.: Combined diet and physical activity is better than diet or physical activity alone at improving health outcomes for patients in New Zealand's primary care intervention. BMC Public Health **18**(230) (2018)
England, P.H.: Government dietary recommendations (2016)
Garthe, I., Raastad, T., Refsnes, P.E., Sundgot-Borgen, J.: Effect of nutritional intervention on body composition and performance in elite athletes. Eur. J. Sport Sci. **13**, 295–303 (2013)
Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning, 1st edn. Addison-Wesley Publishing Company, Boston (1989)
Gunasegaran, T., Cheah, Y.-N.: Evolutionary combinatorial optimization for word embedding (ECOWE) in sentiment classification. Malays. J. Comput. Sci. **3**, 34–45 (2019)
Guo, Z., Wang, M., Agyekum, A.A., Wu, J., Chen, Q., Zuo, M., El-Seedi, H.R., Tao, F., Shi, J., Qin Ouyang, X.Z.: Quantitative detection of apple watercore and soluble solids content by near infrared transmittance spectroscopy. J. Food Eng. **279** (2020)
Gómez-Uribe, C.A., Hunt, N.: The netflix recommender system: Algorithms, business value, and innovation. ACM Trans. Manag. Inf. Syst. **6** (2015)
Hassan, M., Hamada, M.: Genetic algorithms approaches for improving prediction accuracy of multi-criteria recommender systems. Int. J. Comput. Intell. Syst. **11** (2018)
Health Canada, H.M.O.C.: Nutrient value of some common foods (2008)
Holland, J.H.: Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence, 1st edn. U Michigan Press (1975)
Johns, D.J., Hartmann-Boyce, J., Jebb, S.A., Aveyard, P.: Diet or exercise interventions vs combined behavioral weight management programs: a systematic review and meta-analysis of direct comparisons. J. Acad. Nutr. Diet. **114**(10), 1557–1568 (2014)
Jong, K.D., Fogel, L., Schwefel, H.-P.: The Handbook of Evolutionary Computation, 97/1 edn. IOP Publishing Ltd and Oxford University Press (1997)
Karabadji, N.E.I., Beldjoudi, S., Seridi, H., Aridhi, S., Dhifli, W.: Improving memory-based user collaborative filtering with evolutionary multi-objective optimization. Expert Syst. Appl. **98**, 153–165 (2018)
Kilani, Y., Otoom, A.F., Alsarhan, A., Almaayah, M.: A genetic algorithms-based hybrid recommender system of matrix factorization and neighborhood-based techniques. J. Comput. Sci. **28**, 78–93 (2018)
Lee, S.H., Kim, E.K.: Accuracy of predictive equations for resting metabolic rates and daily energy expenditures of police officials doing shift work by type of work. Department of Food and Nutrition, Gangneung-Wonju National University (2012)
Lv, G., Hu, C., Chen, S.: Research on recommender system based on ontology and genetic algorithm. Neurocomputing **187**, 92–97 (2015)
Musto, C., Trattner, C., Starke, A., Semeraro, G.: Towards a knowledge-aware food recommender system exploiting holistic user models. In: Proceedings of 28th Conference on User Modeling, Adaptation and Personalization, UMAP'20 (2020)
Pilloni, P., Piras, L., Boratto, L., Carta, S., Fenu, G., Mulas, F.: Recommendation in persuasive ehealth systems: an effective strategy to spot users' losing motivation to exercise. In: RecSys'17: Proceedings of the 11th ACM Conference on Recommender Systems (2017)
Rapti, E., Karageorgos, A., Ntalos, G.: Adaptive constraint and rule-based product bundling in enterprise networks. In: 23th IEEE International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE-2014) (2014)
Reimer, U., Maier, E., Ulmer, T.: Automatic user adaptation for behavior change support. In: ACM Int. Conf. RecSys16 (2016)
Rezaei, M., Asadizadeh, M.: Predicting unconfined compressive strength of intact rock using new hybrid intelligent models. J. Min. Environ. **11**, 231–246 (2020)
Schäfer, H.: Personalized support for healthy nutrition decisions. In: RecSys'16: Proceedings of the 10th ACM Conference on Recommender Systems, pp. 455–458 (2016)
Shetty, P., Henry, C., Black, A.: Energy requirements of adults: an update on basal metabolic rates (BMRS) and physical activity levels (PALS). Eur. J. Clin. Nutr. **50**, 11–23 (1996)
Srinivas, M., Patnaik, L.M.: Adaptive probabilities of crossover and mutation in genetic algorithms. IEEE Trans. Syst. Man Cybern. **24**, 656–667 (1994)
Thaler, R.H., Sunstein, C.R.: Nudge: Improving Decisions About Health, Wealth, and Happiness. Yale University Press, New Haven (2008)

Whitley, D.: A genetic algorithm tutorial. Stat. Comput. **4**, 65–85 (1994)
Williamson, D.F., Serdula, M.K., Anda, R.F., Levy, A., Byers, T.: Weight loss attempts in adults: goals, duration, and rate of weight loss. Am. J. Public Health **82**, 1251–1257 (1992)
Wishnofsky, M.: Caloric equivalents of gained or lost weight. Am. J. Clin. Nutr. **6**, 542–546 (1958)
Wu, J.-C., Rodríguez, J. A.S., Pampín, H.J.C.: Session-based complementary fashion recommendations. In: RecSys'19: Proceedings of the 13th ACM Conference on Recommender Systems (2019)
Zanker, M., Aschinger, M., Jessenitschnig, M.: Constraint-based personalized configuring of product and service bundles. Int. J. Mass Cust. **3**(4), 407–425 (2010)
Zhang, B.-T., Kim, J.-J.: Comparison of selection methods for evolutionary optimization. Evol. Optim. Int. J. Internet **2**(1), 55–70 (2000)

Hugo Alcaraz-Herrera is a Ph.D. student (4rd year) at University of Bristol sponsored by The Mexican Council of Science and Technology (CONACyT). He obtained both B.Sc. and M.Sc. from the National Autonomous University of Mexico (UNAM). His current research is focused on Evolutionary Computing, particularly in Genetic Algorithms and Coveolutionary Genetic Algorithms. Furthermore, he has participated in Machine Learning projects applied in the Nutrition domain.

John Cartlidge is Associate Professor at the Department of Computer Science of the University of Bristol. He received a PhD in the field of evolutionary computation from the School of Computing at the University of Leeds in 2004. Subsequently, he worked for Hewlett-Packard Research Laboratories and the London Stock Exchange before cofounding a technology start-up that delivered enterprise software for dynamic pricing. After completing post-doctoral research posts in the areas of evolutionary finance and cloud computing, he became Assistant Professor at the University of Nottingham Ningbo China, before moving to the University of Bristol in 2017. John's primary research interests now lie in financial engineering, and he currently leads a Knowledge Transfer Partnership on automated spend analytics. Previously, he has published impact reports on the future of computer trading in financial markets for the UK's Government Office for Science, he has advised the Financial Conduct Authority on regulation technology, and he has acted as an expert witness in automated trading systems for the high courts in London.

Zoi Toumpakari is a Lecturer in Nutrition and Behaviour Change at the University of Bristol. Her research focuses on how features of the eating environment, like where and with whom people eat, influence their food intake and how these could be altered to improve the population's diet. In addition, her research explores the role of dietary and physical activity patterns in the prevention of cardiometabolic health. Recently she has received funding to use machine learning to automate food aggregation for nutrition and health research.

Max Western is a lecturer in behavioural science at the University of Bath. His research interests are on motivation and behaviour change relating to healthy lifestyles and active ageing. A key focus of Max's research is the use of digital technologies for supporting the uptake or maintenance of physical activity, having completed his PhD at the University of Bath in the use of wearable technology for supporting changes in behaviour through the provision of personalised multidimensional feedback.

Iván Palomares received two M.Sc. degrees, one in computer science (with Faculty and Nationwide Distinctions) from the University of Jaén, Spain; and one in Soft Computing and Intelligent Systems (with Honors) from the University of Granada, Spain, in 2009 and 2011, respectively. He received the Ph.D. degree in computer science (with Nationwide distinctions) from the University of Jaen, in 2014. He is a Distinguished Associate Professor at National Cheng Kung University (NCKU), Tainan, Taiwan; and Senior Research Scientist at the Andalusian Research Institute in Data Science and Artificial Intelligence (DaSCI), University of Granada, Spain. He has been a Turing Fellow with the Alan Turing Institute, London, UK, between 2018–2020. He has authored a book "Large Group Decision Making: Creating Decision Support Systems at Scale." His research interests include data-driven and intelligent approaches for recommender systems, personalization for health and wellbeing, leisure and tourism in smart cities, reciprocal recommender systems for social matching, large-scale decision making and consensus, prefer-

ence modelling and aggregation, and AI for Sustainable Development Goals. Dr. Palomares's research results have been published in top journals and conference proceedings, including IEEE TRANSACTIONS ON FUZZY SYSTEMS; IEEE TRANSACTIONS ON SYSTEMS, MAN AND CYBERNETICS: SYSTEMS; European Journal of Operational Research; Applied Soft Computing; International Journal of Intelligent Systems; Information Fusion, Knowledge-Based Systems; Applied Intelligence; Renewable & Sustainable Energy Reviews, amongst others.