Towards Recommendations for Value Sensitive Sustainable Consumption

Thomas Asikis
University of Zurich, Behavioral Game Theory
Andreasstrasse 15, 8050 Zürich
thomas.asikis@uzh.ch

Abstract

Excessive consumption can strain natural resources, harm the environment, and widen societal gaps. While adopting a more sustainable lifestyle means making significant changes and potentially compromising personal desires, balancing sustainability with personal values poses a complex challenge. This article delves into designing recommender systems using neural networks and genetic algorithms, aiming to assist consumers in shopping sustainably without disregarding their individual preferences. We approach the search for good recommendations as a problem involving multiple objectives, representing diverse sustainability goals and personal values. While using a synthetic historical dataset based on real-world sources, our evaluations reveal substantial environmental benefits without demanding drastic personal sacrifices, even if consumers accept only a fraction of the recommendations.

1 Introduction

Sustainable Development Goals [52] (SDGs) outlined by the United Nations (UN) encompass various sustainability criteria crucial for the long-term well-being of societies and the environment [29]. Environmental and societal sustainability goals entail multiple constraints and objectives affecting everyday decision-making [37, 10]. However, reconciling these objectives might involve intricate trade-offs; for instance, minimizing CO_2 emissions in one activity might inadvertently increase its water footprint. Adopting a value sensitive design [53] necessitates considering personal values, including preferences, tastes, and budget constraints.

Researchers anticipate Artificial Intelligence (AI) to play a pivotal role in tackling the complex challenges posed by the SDGs [56]. However, AI applications pose new challenges in aligning with value sensitive design, notably concerning unequal resource distribution, loss of individual autonomy and privacy, and increased emissions from computations [56, 23]. Value sensitive design in AI systems aims to tackle these challenges by integrating personal and societal values as objectives or constraints [53].

Prior approaches in value sensitive recommender systems have concentrated on individual mobile platform and website applications [4]. Such approaches may benefit from more comprehensive quantitative assessments of their collective impact. Such systems have the potential to guide individuals toward more sustainable consumption consciously [24]. While existing recommender systems often focus on specific aspects like diets or single product recommendations [1, 51, 50], our work builds on modern multi-objective design principles to address emerging complex trade-offs using optimization techniques [67, 65, 43, 44, 22, 38, 1]

The main contributions of the current article are to (i) propose and formalize a real-world multi-objective optimization problem for recommendations of personalized sustainable baskets, (ii) create and analyze a novel synthetic dataset based on sustainability factors and consumer real-world data, (iii) propose effective recommender baselines and quantify their impact on environmental criteria, and (iv) introduce a novel deep learning framework for mixed integer programming and multi-objective optimization problems.

2 Preliminaries

Related Work: Classic approaches in recommender systems fall into two primary categories, often combined [48]: (i) Collaborative filtering, which centers on user-item interactions and recommends items based on user similarities. (ii) Content-based (and context-aware) filtering applications mainly consider item attributes and item-item similarities. Applications in both categories often overlook the environmental aspects of products, focusing solely on past user shopping behavior without sustainability considerations.

Extending these approaches involves incorporating environmental and personal product selection criteria and using advanced methods to find similar products or users [18, 41, 26]. However, this becomes feasible only when the computation allows for a manageable enumeration of choices and similarity criteria, especially at the individual product level. On the basket level, sequence recommendations can lead to a combinatorial explosion, making it unfeasible to iterate all possible baskets across various criteria within a reasonable time [47]. Expanding the search and feature spaces, typically when considering shopping baskets that satisfy multiple personal and environmental criteria, poses challenges requiring improved metrics [40] and more advanced sampling techniques for training [59, 60].

Recommending sustainable baskets also requires considering resolving several trade-offs [46] between multiple criteria such as climate, health, and responsible consumption. Therefore, we consider a recommendation approach that actively searches for Pareto optimal baskets across consumer criteria by constructing them via optimization [18]. Multi-objective optimization, applied in the food industry, production, and supply chains, emerges as a relevant strategy for us [43, 2, 39, 57]. Selecting the optimal number of products for a consumer basket under budget and value sensitive constraints resembles a multi-objective and multi-dimensional knapsack problem [36]. Solving such problems via constraint optimization and linear/dynamic programming may prove challenging, requiring more efficient polynomial time approximations [32]. Linear programming relaxation methods may struggle with complex problems [5, 58]. Regularization methods resembling scalarization techniques [65] offer an alternative for solving these problems but require extensive fitting and intricate interpretation of Lagrange coefficients for each objective [64]. Classic evolutionary algorithms (EAs) [65], which can also be combined with scalarization methods, are frequently considered in the relevant literature [22, 14]. Our focus in this article lies on EAs, which seem adept at efficiently handling complex multi-objective optimization problems with multiple (more than 5) objectives and intricate trade-offs.

Dataset: Transaction data, product prices, and purchased quantities were retrieved by "The Complete Journey Dataset" by the Dunnhumby grocery store [17]. The proposed models are evaluated in weekly basket purchases that happen over 85 weeks for the top 500 GHG emission-producing households with 28400 historical (intended) baskets. Environmental impact indicators for product types are taken from related work. [45]. Nutrition information from Food Agricultural Organization Food Balance Sheets [30] is downloaded from FAO website [19]. Historically purchased weekly baskets from the transaction data are compared against recommended ones. Weekly baskets contain the sum of all the purchases for a given household within a week. We perform systematic preprocessing steps (see Appendix A) to align product quantities and units between datasets and remove non-matching products and small baskets. The data sources are publicly available, but Dunnhumby still needs to approve the release of the synthetic dataset. Reproduction steps with code will be open-sourced.

Problem Setting: Value sensitive sustainable recommendations are formalized as a multi-objective optimization problem of selecting combinations of discrete (non-negative integer) quantities over $N = 132$ distinct products. The weekly intended/historical basket is a vector of *non-negative integer product quantities* $\boldsymbol{x}^*_{k,q} \in \mathbb{N}^N_{\geq 0}$ for a specific household k at week q. Each element of the vector corresponds to a distinct product, e.g., the i-th element represents the number of units of product i to be purchased. Week q and household k indices are omitted from the basket vector for brevity. The intended basket $\boldsymbol{x}^*$ is considered the initial solution to the current problem. We also consider intended baskets to represent consumer product preferences and nutritional goals. We define an ordered set C of $|C| = 11$ of possible recommendation factors, with the factor index denoted as j. Then, we calculate a factor coefficient $c_{i,j}$ per product i unit, denoting the total amount of nutritional, monetary, or environmental quantities per product unit. The current dataset contains no negative values for coefficients $c_{i,j}$, e.g., no products are considered carbon sinks. Therefore, for a basket vector $\boldsymbol{x}$, one can calculate the total quantity for a specific feature as $v_j(\boldsymbol{x}) = \sum_{i=1}^{N} c_{i,j} x_i$.

Objective Formulation: Regarding objectives, consumer taste is expressed via the cosine similarity between intended and recommended baskets and the total intended basket price. We assume high similarity denotes a higher likelihood of a purchase under a counterfactual hypothesis, in which the consumer would consider recommended baskets before purchase. Regarding price, health, and environmental objectives, we consider basket ratio $\rho_j(\boldsymbol{x}, \boldsymbol{x}') = v_j(\boldsymbol{x})/v_j(\boldsymbol{x}')$ between recommendation and intended baskets $\boldsymbol{x}, \boldsymbol{x}'$. Objectives, features, and other relevant concepts are found in Table 2 and described further in Appendix B. For example, an optimization trade-off appears between objectives J_1 and J_2, as the intended basket optimizes similarity objective J_1, but not minimal cost objective J_2.

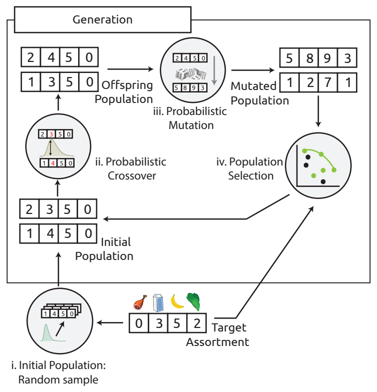
Scope	j	Feature	Unit	Target	Min. Objective
Taste	1	Cosine similarity	-	Max.	$J_1 = 1 - \boldsymbol{x}^\top \boldsymbol{x}^* / \|\boldsymbol{x}\| \, \|\boldsymbol{x}^*\|$
	2	Cost	Dollars (\$)	Min.	$J_2 = \rho_2(\boldsymbol{x}, \boldsymbol{x}^*)$.
Health	3	Energy	kCal	Pres.	$J_{j\in\{3,4,5\}} = (1 - \rho_j(\boldsymbol{x}, \boldsymbol{x}^*))^2$
	4	Protein	grams (g)	Pres.	
	5	Fat	grams (g)	Pres.	
Environment	6	GHG emissions	CO_2 kg eq.	Min.	$J_{j>5}(\boldsymbol{x}, \boldsymbol{x}^*) = \rho_j(\boldsymbol{x}, \boldsymbol{x}^*)$
	7	Acidification	SO_2 kg eq.	Min.	
	8	Eutrophication	PO_4^{3-} kg eq.	Min.	
	9	Land use	m^2	Min.	
	10	Water usage	L	Min.	
	11	Stressed water usage	L	Min.	

TABLE 2: Concepts relevant to objectives for basket selection. The "Target" column describes whether the optimization goal is to minimize, maximize, or preserve the intended basket value. The "Min. Objective" column shows the corresponding minimization objective used in the models.

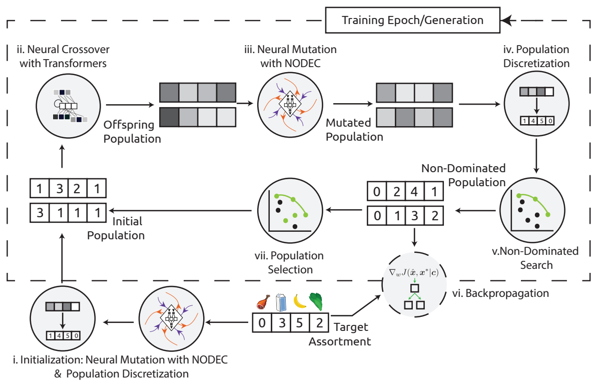
The multi-objective minimization problem for the M aforementioned objectives $J_j = J_j(\boldsymbol{x}, \boldsymbol{x}^*)$ can be summarized as $\min_{\boldsymbol{x}} (J_1, J_2, ...J_M), \boldsymbol{x} \in \hat{\mathrm{X}}$ for a set of feasible baskets $\hat{\mathrm{X}}$. An optimization algorithm $f(\mathrm{X}_0; \boldsymbol{w}) = \mathrm{X}$ with parameter vector $\boldsymbol{w}$ takes an initial set of baskets X_0 and calculates a recommended set of baskets X. The goal of such an algorithm is to find a non-dominated set of baskets. A basket $\boldsymbol{x}$ dominates $\boldsymbol{x} \prec \boldsymbol{x}'$ another basket $\boldsymbol{x}'$ if $J_j(\boldsymbol{x}) \leq J_j(\boldsymbol{x}')$ for all $j \in C$ and $J_j(\boldsymbol{x}) < J_j(\boldsymbol{x}')$ holds at least for one j [15]. If no other basket dominates $\boldsymbol{x}$, $\boldsymbol{x}$ is referred to as non-dominated.

3 Models

Multi-objective recommender systems are standard in literature [65, 42] and have been applied in sustainability-related problems from producer perspective [43] and recommendation of local businesses [44]. Evolutionary algorithms are widely used for multi-objective optimization [16] with $M \geq 2$ objectives. Although the current problem resembles a multi-task recommendation problem [65] that is often treated with scalarization methods [65], the high dimensionality of the problem settings indicates that EAs are more appropriate to consider for its solution. Furthermore, scalarization methods can be combined with evolutionary algorithms [22] to guide the evolution towards optimization of specific objectives. Figure 1 provides a brief overview of the benchmark evolutionary algorithms. We also employ a neural network approach, namely Gradient Guided Genetic Algorithms (G3A), which simultaneously allows for backpropagation and training over multiple baskets. We describe more technical details in Appendix C. While GAs optimize single-problem instances, G3As can be trained to generalize on unknown instances. We plan to use such properties of G3As further in future work.



(a) GA


Training Epoch/Generation
ii. Neural Crossover with Transformers
iii. Neural Mutation with NODEC
iv. Population Discretization
Offspring Population
Mutated Population
Non-Dominated Population
Initial Population
vii. Population Selection
v.Non-Dominated Search
vi. Backpropagation
Target Assortment
i. Initialization: Neural Mutation with NODEC & Population Discretization

(b) G3A

FIGURE 1: With classic probabilistic GAs (a), a population of basket vectors is improved iteratively through "generations". The key components are population sampling (a.i), crossover (a.ii), mutation (a.iii), and population selection of non-dominated solutions (a.iv). In G3A we employ transformers [54] for the crossover (b.ii) and neural ordinary differential equation controllers [3] for the initial population sample (b.i) and mutation operators (b.iii). Next, continuous quantities are discretized via straight-though estimators [6] to resemble discrete product quantities (component b.iv).

4 Results

Two multi-objective optimization algorithms are compared with G3A, namely MO-NES and reference point NSGA-II [14] (RNSGA-II). Several hyper-parameter configurations were tested per method, often taking several days to finish. G3A is parameterized to generate $B = 8$ recommended baskets for a single instance, whereas RNSGA-II and MO-NES generate $B = 10$ recommendations for a single intended basket.

Model	Runtime	Emissions	Mean / Min Improv.	Mean Optimality (CI)
	seconds	kg CO_2 eq.	kg CO_2 eq.	%
G3A (GPU)	1.89 ± 1.22	$(2.07 \pm 1.44)10^{-8}$	31.49 / 0.46	98.0 (97.9, 98.1)
MO-NES (CPU)	0.20 ± 0.01	$(2.16 \pm 0.14)10^{-9}$	21.03 / 0.41	94.8 (94.6, 94.9)
RNSGA-II (CPU)	0.46 ± 0.06	$(6.95 \pm 2.41)10^{-10}$	34.04 / 0.45	98.6 (98.5, 98.6)

TABLE 4: Execution time and GHG emissions (mean $\pm$ standard deviation) measured with python library `codecarbon` [49] over a sample of 100 intended baskets from different households. The mean optimality rate is reported across all intended baskets.

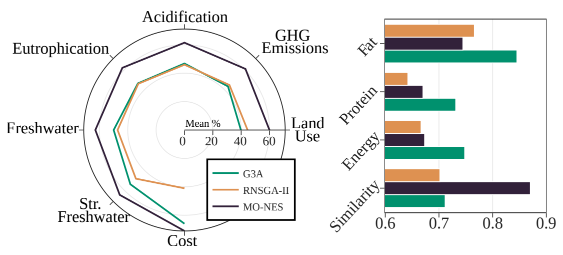
Algorithmic Comparison: First, the ability of baselines to produce find non-dominated solutions for the problem is evaluated in a short amount of time. Each model is tested individually per intended basket to evaluate performance for a single recommendation instance. We measure total runtime, average GHG emissions per execution, average GHG emission improvement in case of recommendation acceptance, and mean/min optimality ratio, i.e., the percentage of suggested solutions that are non-dominated by other method's solutions for the same recommendation. Our findings are summarized in Table 4, and we observe that all methods are highly efficient in retrieving non-dominant solutions. G3A is not batched, and thus, GPU speedups are not considered. For all methods, accepting a single recommendation outweighs vastly the emissions produced by runtime.

Impact analysis: We now seek recommendations that would be likely accepted by the user by filtering out recommendations that perform poorly across most objectives or produce highly dissimilar baskets, i.e., when cosine_sim$(\boldsymbol{x}, \boldsymbol{x}^*) \leq 0.5$. The whole period of shopping is re-sampled 5000 times with 25% of the baskets replaced with recommendations to estimate mean impact.

Figure 2a indicates that RNSGA-II outperforms other baselines in terms of cost, while G3A shows higher performance in terms of nutritional values, where values closer to 1.0 are more desired. MO-NES shows higher performance in terms of cosine similarity, where higher is better in the reported plots. RNSGA-II and G3A compete tightly for environmental impact, with RNSGA-II clearly outperforming in terms of water footprints and G3A clearly outperforming in terms of land usage. All three baselines offer different profiles of non-dominated solutions across objectives, namely that they find different types of good recommendations across objective combinations. Using RNSGA-II has the most significant effect on the selected basket, involving removing the highest number of products. Next up, G3A typically eliminates around 51% of products while introducing 21% new ones, on average. Finally, MO-NES is the least intrusive baseline, respecting consumer basket choices the most. Figure 2b showcases the comparison between mean basket values for period samples with G3A and without any recommender.

FIGURE 2: Evaluation of recommendations.

(a) The mean ratio of quantities between recommended and intended basket.



(b) What do I sacrifice?

Personal Impact: G3A (Intended)	
Cost	35.2 (38.7)
Fat (g)	698 (726)
Protein (g)	911 (971)
Energy (kCal)	25400 (27100)
Added (%)	21
Removed (%)	54

(c) How much do we save?

G3A Environmental Impact	
Land Use (km^2)	1.02
GHG Emissions (kt)	0.35
Acidification (t)	1.16
Eutrophication (t)	1.24
Freshwater (10^7L)	22.8
Str. Freshwater (10^9L)	1.06

The total impact for accepting 25% of the G3A recommendations (see Appendix Figure 4 for the rest) is showcased in Figure 2c, where the mean values of total reduction on all environmental quantities across samples are reported. In conclusion, an accepted fraction of proposed recommendations can yield a substantial effect on environmental impact while also respecting personal preferences.

5 Conclusion

The article introduces a framework for multi-objective recommender systems, focusing on personalized sustainable shopping recommendations. It evaluates real-world data to design shopping baskets that balance sustainability and user preferences. By comparing novel approaches with existing methods, this study demonstrates how even partial adoption of sustainable recommendations can significantly benefit the environment. This framework opens avenues for advancing multi-objective optimization for personalized sustainable consumption in various recommendation system settings.

Acknowledgements

This work/paper has been partially supported by the 'Co-Evolving City Life - CoCi' project through the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 833168). The author would like also like to acknowledge partial support by NCCR Automation, a National Centre of Competence in Research, funded by the Swiss National Science Foundation (grant number 180545). Finally, the author would also like to thank Dr. Nino Antulov-Fantulin, Dr. Lucas Boettcher, Prof. Dirk Helbing, Prof. Petros Koumoutsakos and Prof. Avishek Anand and the reviewers for their useful feedback on the manuscript.

References

[1] Ricardo Abejón et al. "Multi-Objective Optimization of Nutritional, Environmental and Economic Aspects of Diets Applied to the Spanish Context". In: *Foods* 9.11 (2020), p. 1677.

[2] Masoud Alinezhad et al. "A fuzzy multi-objective optimization model for sustainable closed-loop supply chain network design in food industries". In: *Environment, Development and Sustainability* (2022), pp. 1–28.

[3] Thomas Asikis, Lucas Böttcher, and Nino Antulov-Fantulin. "Neural ordinary differential equation control of dynamics on graphs". In: *Physical Review Research* 4.1 (2022), p. 013221.

[4] Thomas Asikis et al. "How value-sensitive design can empower sustainable consumption". In: *Royal Society open science* 8.1 (2021), p. 201418.

[5] Abbas Bazzi. *Strengths and Limitations of Linear Programming Relaxations*. Tech. rep. EPFL, 2017.

[6] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. "Estimating or propagating gradients through stochastic neurons for conditional computation". In: *arXiv preprint arXiv:1308.3432* (2013).

[7] J. Blank and K. Deb. "pymoo: Multi-Objective Optimization in Python". In: *IEEE Access* 8 (2020), pp. 89497–89509.

[8] Lucas Böttcher, Nino Antulov-Fantulin, and Thomas Asikis. "AI Pontryagin or how artificial neural networks learn to control dynamical systems". In: *Nature Communications* 13.1 (2022), pp. 1–9.

[9] Lucas Böttcher, Thomas Asikis, and Ioannis Fragkos. "Control of Dual-Sourcing Inventory Systems Using Recurrent Neural Networks". In: *INFORMS Journal on Computing* (2023).

[10] Abhishek Chaudhary, David Gustafson, and Alexander Mathys. "Multi-indicator sustainability assessment of global food systems". In: *Nature communications* 9.1 (2018), p. 848.

[11] Ricky T. Q. Chen et al. "Neural Ordinary Differential Equations". In: *Advances in Neural Information Processing Systems* (2018).

[12] Ricky TQ Chen, Brandon Amos, and Maximilian Nickel. "Learning Neural Event Functions for Ordinary Differential Equations". In: *International Conference on Learning Representations*. 2020.

[13] Kalyanmoy Deb and Himanshu Jain. "An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point-Based Nondominated Sorting Approach, Part I: Solving Problems With Box Constraints". In: *IEEE Transactions on Evolutionary Computation* 18.4 (2014), pp. 577–601. DOI: `10.1109/TEVC.2013.2281535`.

[14] Kalyanmoy Deb and J Sundar. "Reference point based multi-objective optimization using evolutionary algorithms". In: *Proceedings of the 8th annual conference on Genetic and evolutionary computation*. 2006, pp. 635–642.

[15] Kalyanmoy Deb et al. "A fast and elitist multiobjective genetic algorithm: NSGA-II". In: *IEEE transactions on evolutionary computation* 6.2 (2002), pp. 182–197.

[16] Kalyanmoy Deb et al. "A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II". In: *International conference on parallel problem solving from nature*. Springer. 2000, pp. 849–858.

[17] *Dunnhumby - The complete journey dataset*. Available at `https://www.dunnhumby.com/source-files/` (last accessed: September 2021). 2021.

[18] David Elsweiler, Hanna Hauptmann, and Christoph Trattner. “Food Recommender Food recommenderSystems”. In: *Recommender Systems Handbook*. Ed. by Francesco Ricci, Lior Rokach, and Bracha Shapira. New York, NY: Springer US, 2022, pp. 871–925. ISBN: 978-1-0716-2197-4. DOI: `10.1007/978-1-0716-2197-4_23`. URL: `https://doi.org/10.1007/978-1-0716-2197-4_23`.

[19] *FAO – Food Agriculture Organization, Food Balance Sheets*. Available at `http://www.fao.org/3/X9892E/X9892e05.htm` (last accessed: September 2021). 20211.

[20] Carlos M Fonseca, Luís Paquete, and Manuel López-Ibánez. “An improved dimension-sweep algorithm for the hypervolume indicator”. In: *2006 IEEE international conference on evolutionary computation*. IEEE. 2006, pp. 1157–1163.

[21] Guillermo Fuertes et al. “Chaotic genetic algorithm and the effects of entropy in performance optimization”. In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 29.1 (2019), p. 013132.

[22] Tobias Glasmachers, Tom Schaul, and Jürgen Schmidhuber. “A natural evolution strategy for multi-objective optimization”. In: *International Conference on Parallel Problem Solving from Nature*. Springer. 2010, pp. 627–636.

[23] Dirk Helbing and Evangelos Pournaras. “Build digital democracy A WISE KING?” In: *Nature* 527.7576 (2015), pp. 33–34. ISSN: 0028-0836. DOI: `10.1038/527033a`.

[24] Dirk Helbing et al. “Ethics of Smart Cities: Towards Value-Sensitive Design and Co-Evolving City Life”. In: *Sustainability* 13.20 (2021). ISSN: 2071-1050. DOI: `10.3390/su132011162`. URL: `https://www.mdpi.com/2071-1050/13/20/11162`.

[25] Dan Hendrycks and Kevin Gimpel. “Gaussian error linear units (gelus)”. In: *arXiv preprint arXiv:1606.08415* (2016).

[26] Jui-Ting Huang et al. “Embedding-based retrieval in facebook search”. In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2020, pp. 2553–2561.

[27] Hisao Ishibuchi et al. “Evolutionary many-objective optimization by NSGA-II and MOEA/D with large populations”. In: *2009 IEEE International Conference on Systems, Man and Cybernetics*. IEEE. 2009, pp. 1758–1763.

[28] Eric Jang, Shixiang Gu, and Ben Poole. “Categorical reparameterization with gumbel-softmax”. In: *arXiv preprint arXiv:1611.01144* (2016).

[29] R. W. Kates et al. “Sustainability Science”. In: *Science* 292.5517 (2001), pp. 641–642. ISSN: 00368075. DOI: `10.1126/science.1059386`.

[30] A Kelly, W Becker, and E Helsing. “Food balance sheets”. In: *Food and health data: their use in nutrition policy-making. Copenhagen: WHO Regional Office for Europe* (1991), pp. 39–47.

[31] Ed Klotz and Alexandra M Newman. “Practical guidelines for solving difficult mixed integer linear programs”. In: *Surveys in Operations Research and Management Science* 18.1-2 (2013), pp. 18–32.

[32] Ariel Kulik and Hadas Shachnai. “There is no EPTAS for two-dimensional knapsack”. In: *Information Processing Letters* 110.16 (2010), pp. 707–710.

[33] Kyuseop Kwak, Sri Devi Duvvuri, and Gary J Russell. “An analysis of assortment choice in grocery retailing”. In: *Journal of Retailing* 91.1 (2015), pp. 19–33.

[34] Ziwei Li and Sai Ravela. “Neural Networks as Geometric Chaotic Maps”. In: *IEEE Transactions on Neural Networks and Learning Systems* (2021).

[35] Hui Lu et al. “The effects of using chaotic map on improving the performance of multiobjective evolutionary algorithms”. In: *Mathematical Problems in Engineering* 2014 (2014).

[36] Thibaut Lust and Jacques Teghem. “The multiobjective multidimensional knapsack problem: a survey and a new approach”. In: *International Transactions in Operational Research* 19.4 (2012), pp. 495–520.

[37] Talha Manzoor et al. “Optimal control for sustainable consumption of natural resources”. In: *IFAC Proceedings Volumes* 47.3 (2014), pp. 10725–10730.

[38] R.T. Marler and J.S. Arora. “Survey of multi-objective optimization methods for engineering”. In: *Structural and Multidisciplinary Optimization* 26.6 (2004), pp. 369–395. ISSN: 1615-1488. DOI: `10.1007/s00158-003-0368-6`.

[39] Ali Mirdar Harijani, Saeed Mansour, and Behrooz Karimi. “A multi-objective model for sustainable recycling of municipal solid waste”. In: *Waste Management & Research* 35.4 (2017), pp. 387–399.

[40] Kevin Musgrave, Serge Belongie, and Ser-Nam Lim. “A metric learning reality check”. In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXV 16*. Springer. 2020, pp. 681–699.

[41] Priyanka Nigam et al. “Semantic product search”. In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2019, pp. 2876–2885.

[42] “NNIA-RS: A multi-objective optimization based recommender system”. In: *Physica A: Statistical Mechanics and its Applications* 424 (2015), pp. 383–397. ISSN: 0378-4371. DOI: `https://doi.org/10.1016/j.physa.2015.01.007`. URL: `https://www.sciencedirect.com/science/article/pii/S0378437115000096`.

[43] Arnault Pachot et al. “Multiobjective recommendation for sustainable production systems”. In: *MORS workshop held in conjunction with the 15th ACM Conference on Recommender Systems (RecSys), 2021*. Amsterdam, Netherlands, Sept. 2021. URL: `https://hal.archives-ouvertes.fr/hal-03349092`.

[44] Gourab K Patro et al. “Towards safety and sustainability: Designing local recommendations for post-pandemic world”. In: *Fourteenth ACM Conference on Recommender Systems*. 2020, pp. 358–367.

[45] Joseph Poore and Thomas Nemecek. “Reducing food’s environmental impacts through producers and consumers”. In: *Science* 360.6392 (2018), pp. 987–992.

[46] Prajal Pradhan et al. “A systematic study of sustainable development goal (SDG) interactions”. In: *Earth’s Future* 5.11 (2017), pp. 1169–1179.

[47] Massimo Quadrana, Paolo Cremonesi, and Dietmar Jannach. “Sequence-Aware Recommender Systems”. In: *ACM Comput. Surv.* 51.4 (2018). ISSN: 0360-0300. DOI: `10.1145/3190616`. URL: `https://doi.org/10.1145/3190616`.

[48] Francesco Ricci, Lior Rokach, and Bracha Shapira. “Introduction to recommender systems handbook”. In: *Recommender systems handbook*. Springer, 2010, pp. 1–35.

[49] Victor Schmidt et al. *CodeCarbon: Estimate and Track Carbon Emissions from Machine Learning Computing*. `https://github.com/mlco2/codecarbon`. 2021. DOI: `10.5281/zenodo.4658424`.

[50] Alain Starke. “RecSys Challenges in achieving sustainable eating habits.” In: *HealthRecSys@RecSys*. 2019, pp. 29–30.

[51] Sabina Tomkins et al. “Sustainability at scale: towards bridging the intention-behavior gap with sustainable recommendations”. In: *Proceedings of the 12th ACM Conference on Recommender Systems*. ACM. 2018, pp. 214–218.

[52] UN. *Global Sustainable Development Report: 2015 edition*. Tech. rep. United Nations, 2015, p. 202.

[53] Jeroen Van den Hoven. “ICT and value sensitive design”. In: *The information society: Innovation, legitimacy, ethics and democracy in honor of Professor Jacques Berleur SJ*. Springer, 2007, pp. 67–72.

[54] Ashish Vaswani et al. “Attention is all you need”. In: *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2017, pp. 5998–6008.

[55] Yash Vesikar, Kalyanmoy Deb, and Julian Blank. “Reference Point Based NSGA-III for Preferred Solutions”. In: *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*. 2018, pp. 1587–1594. DOI: `10.1109/SSCI.2018.8628819`.

[56] Ricardo Vinuesa et al. “The role of artificial intelligence in achieving the Sustainable Development Goals”. In: *Nature communications* 11.1 (2020), pp. 1–10.

[57] Zhaoyuan Wang et al. “Food package suggestion system based on multi-objective optimization: A case study on a real-world restaurant”. In: *Applied Soft Computing* 93 (2020), p. 106369.

[58] Jake Weiner et al. “Automatic decomposition of mixed integer programs for lagrangian relaxation using a multiobjective approach”. In: *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*. 2020, pp. 263–270.

[59] Chao-Yuan Wu et al. “Sampling matters in deep embedding learning”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2840–2848.

[60] Xu Xie et al. “Contrastive learning for sequential recommendation”. In: *2022 IEEE 38th international conference on data engineering (ICDE)*. IEEE. 2022, pp. 1259–1273.

[61] Xuefeng F Yan, Dezhao Z Chen, and Shangxu X Hu. “Chaos-genetic algorithms for optimizing the operating conditions based on RBF-PLS model”. In: *Computers & Chemical Engineering* 27.10 (2003), pp. 1393–1404.

[62] Penghang Yin et al. “Understanding Straight-Through Estimator in Training Activation Quantized Neural Nets”. In: *International Conference on Learning Representations*. 2018.

[63] You Yong, Sheng Wanxing, and Wang Sunan. “Study of chaos genetic algorithms and its application in neural networks”. In: *2002 IEEE Region 10 Conference on Computers, Communications, Control and Power Engineering. TENCOM’02. Proceedings.* Vol. 1. IEEE. 2002, pp. 232–235.

[64] Hossein Zare and Masoud Hajarian. “Determination of regularization parameter via solving a multi-objective optimization problem”. In: *Applied Numerical Mathematics* 156 (2020), pp. 542–554.

[65] Yong Zheng and David (Xuejun) Wang. “A survey of recommender systems with multi-objective optimization”. In: *Neurocomputing* 474 (2022), pp. 141–153. ISSN: 0925-2312. DOI: `https://doi.org/10.1016/j.neucom.2021.11.041`. URL: `https://www.sciencedirect.com/science/article/pii/S0925231221017185`.

[66] Eckart Zitzler and Lothar Thiele. “Multiobjective optimization using evolutionary algorithms—a comparative case study”. In: *International conference on parallel problem solving from nature*. Springer. 1998, pp. 292–301.

[67] Yi Zuo et al. “Personalized recommendation based on evolutionary multi-objective optimization [research frontier]”. In: *IEEE Computational Intelligence Magazine* 10.1 (2015), pp. 52–62.

A Appendix: Data Preprocessing

This section contains more details on the data preprocessing procedure to generate the synthetic dataset. The Dunnhumby transaction data quantities are included in US imperial units and conversion to the metric system was done in the following manner: (i) Unit labels are identified and grouped together with regular expressions and manual corrections, e.g., "LB, lb, LBs" all represent the same label which denotes pounds. (ii) Weight and volumetric labels are separated and proper conversion coefficients are used to convert each unit to the corresponding metric unit used in the other datasets. (iii) Prices may change over time, so the mean price per unit is calculated through time and overall stores to generate the price features used by all algorithms. After processing, the dataset contains almost 885’000 transactions corresponding to approximately 167’000 baskets purchased by almost 2’500 households. The transactions were performed over a period of 709 days across 489 distinct stores. The proposed multi-objective recommender systems in the current article can also be applied to any dataset and basket selection problem that involves multiple objectives relevant to product characteristics.

All three source datasets contain different product type labels for each product. From "The Complete Journey Dataset" the "SUB_COMMODITY_DESC" column is treated as the product identifier. Each value of the column "SUB_COMMODITY_DESC" is matched against the "product category" column from the FAO FBS dataset and the dataset column "product" from Ref [45]. The resulting dataset contains transaction prices, purchased quantities, environmental impact values, and nutritional info per transaction.

The FAO product categorization is used to identify products, and thus the product set is considered to contain 132 distinct products. The proposed dataset[1] is used to motivate and estimate environmental impact, consumer preferences, purchasing costs, and nutritional values of recommended baskets.

For the dataset factors, e.g., price and emissions per unit, as the total quantities per unit may change, we may need to calculate aggregate values of these factors, such as the mean price over the same product. We also tested other estimators, such as the median values. Any measure of central tendency or the actual values at a current time and retailer store can be considered. In a real-world application,

[1]The relevant code will be made available publicly, but the release of the final dataset is pending confirmation from Dunnhumby.

where the purchased basket is not known, the consumer may provide an intended basket via a shopping list interface or an e-shop basket interface.

B Optimization Objectives

Personal Objectives: We consider consumer preference over similar baskets to the intended one as the first personal to optimize when the recommended basket $\boldsymbol{x}$ is as similar as possible to the intended basket $\boldsymbol{x}^*$. High similarity between a recommended basket $\boldsymbol{x}$ and the target/intended $\boldsymbol{x}^*$ indicates a higher likelihood of a purchase under a counterfactual hypothesis, in which the consumer would consider recommended baskets before purchase. The first objective function to minimize depends on the cosine similarity between the recommended and intended basket $J_1(\boldsymbol{x}, \boldsymbol{x}^*) = 1 - \boldsymbol{x}^\top \boldsymbol{x}^* / \|\boldsymbol{x}\| \, \|\boldsymbol{x}^*\|$.

The next personal value considered in optimization is a function of cost. In general, it is assumed that individuals would prefer to minimize expenses and select cheaper baskets that satisfy their taste. Next, the cost ratio between recommended and intended basket costs is calculated as an objective function: $J_2(\boldsymbol{x}, \boldsymbol{x}^*) = \rho_2(\boldsymbol{x}, \boldsymbol{x}^*)$. Here we observe a direct trade-off with objective J_1. For example, the intended basket optimizes J_1 but not J_2, which is optimized by an empty basket.

Next, we consider basket health/nutritional values. For each unit of product i and nutritional product feature j the nutritional quantity per unit $c_{i,j}$ is calculated. Three nutritional features are denoted by indices $j \in \{3, 4, 5\}$. The health objective functions, namely $J_j(\boldsymbol{x}, \hat{\boldsymbol{x}}) = (1 - \rho_j(\boldsymbol{x}, \boldsymbol{x}^*))^2 = (v_j(\boldsymbol{x}^*) - v_j(\boldsymbol{x})/v_j(\boldsymbol{x}^*))^2, j \in \{3, 4, 5\}$, use the intended basket's nutritional value as a baseline to evaluate the difference for each nutritional feature between recommended and intended baskets

Environmental Impact Objectives: Collective environmental values are also considered based on the provided data from Ref. [45]. In total, a set of six environmental impact criteria are considered for each product, as shown in Table 2, namely *greenhouse gas* (GHG) emissions, which contribute to climate change, *acidifying* pollution that decreases fertility and can cause desertification, *eutrophication* pollution, which destabilizes food chains in ecosystems, *water usage* that has several environmental effects, *stress-weighted water* usage that takes into account whether the water is taken from arid/dry lands, and *land usage*, which is important to resource allocation for farming and deforestation. The mean product features per unit are used as coefficients $c_{i,j}$ for calculating $v_{\text{j}}, j > 5$. Similar to the price objective, the ratio between the intended and recommended basket of each environmental impact feature is considered an objective function: $J_j(\boldsymbol{x}, \boldsymbol{x}^*) = \rho_j(\boldsymbol{x}, \boldsymbol{x}^*)$

C Appendix: Evolutionary Algorithms and Multi-Objective Optimization

Typically each basket, or solution[2] in the optimization context, $\boldsymbol{x}$ is mapped to an objective vector $\boldsymbol{\zeta}(\boldsymbol{x}) \in \mathbb{R}^M$, where each vector element represents an objective function value $\zeta_j = J_j(\boldsymbol{x})$. Often, such algorithms improve a set of an initial population of solutions X_τ by applying probabilistic operators on each solution vector $\boldsymbol{x}$, such as the *random crossover*. Random crossover randomly combines elements from different solutions $\boldsymbol{x}, \boldsymbol{x}'$ with probability p

$$x_i = \begin{cases} x_i' & \text{if } \delta < p \\ x_i & \text{otherwise} \end{cases}, \tag{1}$$

where δ is sampled from a probability density function $\delta \sim f$ with finite support $[0, 1]$. Another probabilistic mechanism is the *random mutation*, e.g., replacing an element of the solution with a random number sampled from a probability distribution $\kappa \sim f_{\text{discrete}}$ to an element of the solution

$$x_i = \kappa. \tag{2}$$

Each new solution is evaluated based on the corresponding objective vector $\boldsymbol{\zeta}(\boldsymbol{x})$, and a *selection* of solutions is performed. Typically, a non-dominated sorting is performed to select non-dominated solution candidates from new solutions and the initial population. The non-dominated sorting is performed recursively, i.e., each time a non-dominated set F_α is selected, the non-dominated

[2]The term solution will be used in the sections that describe models in accordance to literature.

solutions are assigned to a non-dominated front F_α and then removed from the population. A new non-dominated search is performed on the remaining population solutions to determine the non-dominated front $F_{\alpha+1}$. This process repeats until all solutions are assigned to a front. A possible selection mechanism would select all non-dominated solutions, i.e., the solutions in F_1. The selected solution candidates are preserved in a new population of solutions $\mathrm{X}_{\tau+1}$ and the whole process (crossover, mutation, selection) is repeated until a convergence criterion is met, e.g., no new solutions are preserved in a population after an iteration. We denote τ as a *generation* index. A widely used algorithm that follows the above strategy for multi-objective optimization is the Non-dominated Sorting Genetic Algorithm II (NSGA-II) [15].

C.1 RNSGA-II

A non-dominated sorting algorithm may produce a high number of non-dominated solutions that are not preferable, e.g., solutions that optimize a single objective very well and not the others. We aim to keep the population size B per generation constant, so a secondary selection operation must be performed. Random selection is often undesired in problems with multiple objectives [14]; thus, a more sophisticated technique is preferred. Some probabilistic evolutionary algorithms use a sorting operation to perform a secondary selection operation that guides the evolutionary processes towards preferred non-dominated solutions, e.g., non-dominated solutions that optimize specific combinations of the objectives very well. A typical example used as a baseline in the current study is reference point NSGA-II, abbreviated as RNSGA-II [14], which uses reference directions to guide evolution towards preferred solutions. In brief, one or more reference points are selected to guide the evolution. A reference point $\hat{\boldsymbol{\zeta}}$ is generated by providing the system with a vector of preferred objective values. Each candidate solution receives two ranks determined by the non-dominated sorting and a distance metric from each reference point, i.e., lower distance values receive lower ranks. Lower ranks are used to select the candidates for the next generation. This algorithm shows higher performance gains than NSGA-II to perform better on multi-objective problems with more than 2 objectives [14]. RSNGA-II is used as a baseline in the current study following the default implementation of [7].

A logistic map [21] is applied on the initial basket to generate the initial solution for RNSGA-II, improving performance considerably compared to other random initializations. Several reference point settings are tested for RNSGA-II. The current reference points provided to RNSGA-II are three: one that is calculated by using the infeasible optimum, where every loss is 0, one that minimizes all individual losses (e.g., all values for $j \leq 5$ are 0 and the rest are 1), and one that minimizes all environmental losses (e.g., all values for $j > 5$ are set to 0). Using less than two reference points often resulted in bad performance. Other reference point settings were tested on 100 intended baskets, such as using the intended basket or minimizing specific losses on smaller samples. However, whether this leads to better performance needs to be clarified. The current reference point setup was chosen as it provided the best-performing dominance ratio compared to other baselines. Integer exponential crossover and polynomial mutation are used for the genetic operators. Finally, other settings were tested with up to $B = 100$. However, they were omitted due to lower dominance ratio, slower convergence times, large number of solutions, and difficulty in determining subsets of reasonable solutions.

C.2 MO-NES

Another way to handle multi-objective optimization problems is the use of MO-NES [22], which uses a gradient-guided search algorithm to find non-dominated solutions by parametrizing a probabilistic model (relies on sampling). The algorithm optimizes the parameters of a model that samples solutions from underlying distributions. For each solution, a sample vector $\boldsymbol{z} \in \mathbb{R}^N$ is generated, where each element is sampled from a normal distribution $z_i \sim \mathcal{N}(0, 1)$. A new solution $\boldsymbol{x}'$ is calculated based on a parent solution $\boldsymbol{x}' = \boldsymbol{x} + \sigma \boldsymbol{A}\boldsymbol{z}$, where $\sigma \in \mathbb{R}$, $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ are the co-variance related terms. Samples from the previous population X_τ and the new candidates $\boldsymbol{x}'$ are combined into an intermediate population X'. Each solution $\boldsymbol{x} \in \mathrm{X}'$ is assigned a rank α based on the non-dominated sorting. A secondary rank β is assigned to each solution based on the value of a hyper-volume metric [22, 66] in descending order. To calculate the hyper-volume metric, a dominated reference point $\boldsymbol{\zeta}^{(0)} \in \mathbb{R}^M$ is selected in the objective space, such that all considered solutions $\boldsymbol{x} \in \mathrm{X}'$ dominate this point $\boldsymbol{\zeta}(\boldsymbol{x}) \prec \boldsymbol{\zeta}^{(0)}$. The hyper-volume metric [66] is used to calculate the hyper-volume between each solution and the dominated reference point, e.g., by using the proposed implementation of Ref. [20].

The hyper-volume metric is calculated on standardized loss values by subtracting the mean loss and dividing it by their standard deviation over all solutions. The covariance-related parameters $\boldsymbol{A}, \sigma$ are updated with a gradient update. A modified version of MO-NES, where solutions are rounded and negative values are clipped to 0 prior to evaluation, is used as a baseline in the current article. The initial value of each solution is sampled as $x_i = \text{ReLU}(x), x \sim \mathcal{N}(0, 0.2)$. Parameter $\sigma = 1/3$ and elements of A were initialized uniformly in $[0, 0.001]$. Following notation from Ref. [22], the learning rates for each parameter are $\eta_\sigma^+ = 0.01$, $\eta_\sigma^- = 0.01/5$ and $\eta_A = 0.01/4$. MO-NES trains up to 40 generations.

C.3 Gradient Guided Genetic Algorithm

Probabilistic algorithms may suffer from slow convergence [27], especially on high dimensional problems. Dependence on randomness and selection of random seeds may also be considered a challenge [21, 35]. Recently, deterministic chaos genetic algorithms have been proposed to calculate solutions in a deterministic and seemingly more efficient manner [61, 63]. Furthermore, chaotic maps seem very promising for sparse and highly dimensional problems as they can control entropy [21] and the performance of the optimization procedures. For example, genetic algorithms may show improved performance if a logistic map [21] is used to sample initial solutions around the intended basket. Nevertheless, chaos genetic algorithms do not use explicit feedback from the loss function, such as MO-NES [22], and often the selection of adequate chaotic maps requires extensive hyper-parameter optimization [21, 35]. This article investigates another potential design, where neural networks perform *mutation and crossover* operators or *initialize the population* instead of chaotic maps. Neural networks show promising capabilities to learn chaotic maps and strange attractors [34], and back-propagation can be used to learn the parameters of the neural networks and control the chaotic behavior to improve solutions across generations. This article proposes a novel gradient-guided genetic algorithm by combining design concepts from chaotic genetic algorithms and neural networks. G3A may evolve a population of solutions conditional to input data (such as the coefficient matrix) by performing gradient-guided genetic operations. An overview of G3A is provided based on Figure 1b.

C.3.1 Population Initialization

An initial population matrix $\boldsymbol{X}_0$ is calculated by applying the untrained neural mutation from $t = 0$ to $t = T$. B solutions are selected during initialization by sampling the mutation trajectory every $\Delta t = T/B$. During each generation, a population matrix $\boldsymbol{X}_\tau \in \mathbb{N}_0^{B\times N}$ is created, where each row represents a recommended solution.

C.3.2 Neural Crossover

A neural crossover operator is then applied to the population matrix and generates an offspring solution for each solution in the initial population. The main neural network component is a transformer network $f_{\text{transformer}} : \mathbb{N}_0^{B\times N} \to \mathbb{R}^{B\times B\times N}$ with Gaussian Error Linear Unit [25] (GeLU) activation functions as hidden layers [54]. Each parent solution $\boldsymbol{x}$ is compared with the rest of the population matrix $\boldsymbol{X}_\tau$. For each element x_i of the parent solution, the transformer generates an attention vector $\boldsymbol{g} \in \mathbb{R}^B$ over all solutions in the population. The element $\hat{x}_{i,b}$ is selected from the b-th parent in the population that received the maximum attention value from the transformer $b = \arg\max_b g_b$.

This work's multi-head attention transformer networks contain 1 encoder and 1 decoder layer with GeLU activation functions and 11 heads. Replacing the crossover network with probabilistic operators or simpler neural network architectures has not yielded better results so far but is still a subject of study and future work. Both of the transformer encoder and decoder layers contain a hidden layer with 2048 hidden neurons and, layer norm layers in output and input and also dropout operations on neuron outputs[3]. A sigmoid activation is then applied to the attention values and each selected parent element is used to calculate an "offspring" solution element x_i' in the following manner:

$$x_i' = \text{sigmoid}(g_k)x_i + (1 - \text{sigmoid}(g_b))\hat{x}_{i,b}. \tag{3}$$

[3]according to the default implementation found in `https://pytorch.org/docs/stable/generated/torch.nn.Transformer.html` (accessed October 2021)

C.3.3 Neural Mutation

Next, a mutation operator neural network $u(\boldsymbol{x}(t)) : \mathbb{R}^{B\times N} \to \mathbb{R}^{B\times N}$ evolves a solution $\boldsymbol{x}(t)$ in continuous time t by applying the following neural ODE control

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{u}(\boldsymbol{x}(t)). \tag{4}$$

A neural ODE solve [11] scheme is used to calculate the continuous time evolution between subsequent genetic generations, e.g., $\boldsymbol{x}(0) \to \boldsymbol{x}(T)$. The underlying neural network has sinusoidal activation functions in the hidden layer, inspired by the sinusoidal iterator used in Ref. [35]. The evolution period is $[0, 1]$ and a single hidden layer with 256 neurons. To select the solutions that are preserved for the next generation, a finite number of mutated solutions is sampled uniformly across time for each solution $\boldsymbol{x}(0)$ of the current generation τ at predetermined time-steps within the ODE solver. The output activation of the neural network is a Rectifier Linear Unit (ReLU) activation [54], which removes negative product quantities from each solution.

Neural network weights and activation functions generate real-valued solutions. The proposed problems require discrete product quantities in the solution. Therefore, a discretization operation that allows gradient propagation is applied to each solution. G3A can be viewed as an extension of neural ODEs control [3, 8] with discrete events [12] to MIP problems.

C.3.4 Straight-through Discretization: Fractional Decoupling

Neural networks are known to operate in real value settings, as back-propagation requires the output of neural networks to be continuous and differentiable with regard to objective functions, so that the chain rule can be efficiently applied. Continuous outputs are incompatible with end-to-end learning mixed integer programming problem (MIP) solutions. Rounding neural network outputs creates a challenge when back-propagating error for calculating the gradient as rounding functions are not differentiable in its domain, particularly at the integer values

A potential approach is to train the neural networks in a real-valued manner and then apply rounding when evaluating the solutions, e.g., applying a linear programming relaxation scheme [31, 44]. Such relaxations may become problematic when considering shopping baskets over a wide variety of products. Neural networks with many outputs may assign a small positive quantity over hundreds of products to a single basket to optimize taste and environmental losses. In such case, many product quantities are rounded to 0, yielding empty or very sparse baskets as solutions. Another approach proposed in the literature is to use the Gumbel soft-max operator [28], which allows for gradient propagation via the aforementioned straight-through estimators [62]. Since the decision problem in question requires no upper bounds on purchased product quantities, using the Gumbel soft-max operator may yield high dimensional outputs that may require more time to train for large-scale problems.

An alternative approach, termed fractional decoupling, is proposed to efficiently calculate a gradient update and perform gradient descent via a straight-through estimator [6, 62, 9]. To perform fractional decoupling, one subtracts the fractional part h_i of a real-valued output y_i while treating the fractional part as constant, i.e., this allows no gradient propagation through the fractional part in the computational graph. This operation can be considered as a rounding straight-though estimator.

Illustrative Numerical Example of Fractional Decoupling

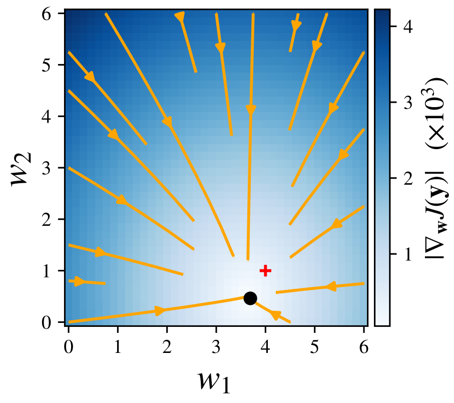
C.4 Numerical example of gradient flows with fraction decoupling

To illustrate the gradient flows after with discretization of fractional decoupling we provide an illustrative example, which compares gradient flows of a 2-parameter network both with and without fractional decoupling when solving the same problem. Given a predetermined coefficient vector $\hat{\boldsymbol{w}} = [w_1 \quad w_2] = [4 \quad 1]$ and normally distributed inputs $x_1, x_2 \sim \mathcal{N}(\mu = 0, \sigma = 1)$ we apply the following transformation:

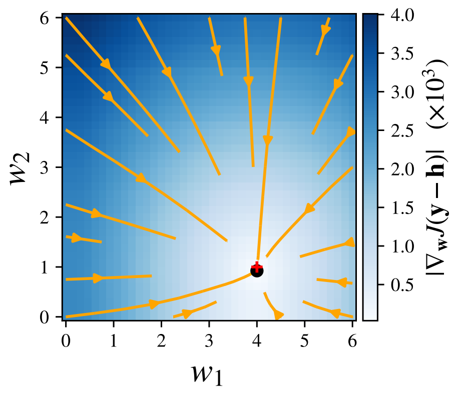
$$\hat{\boldsymbol{y}} = \lfloor \hat{\boldsymbol{w}} \odot \boldsymbol{x} \rfloor \tag{5}$$

Given a batch matrix of 50 input vectors $X \in \mathbb{R}^{50\times 2}$ and their corresponding label matrix $\hat{Y} \in \mathbb{R}^{50\times 1}$ we train two neural networks. A neural network with continuous outputs during training, where the floor function is applied only during inference:

$$f_1(\boldsymbol{x}) = y = \boldsymbol{w} \odot \boldsymbol{x} \tag{6}$$



(d) Continuous gradient descent.



(e) Fractional decoupling gradient descent.

FIGURE 3: Gradient direction (orange lines), optimal solution (red cross), and lowest gradient norm point (black disk).

and a separate neural network with a fractional decoupling term:

$$f_2(\boldsymbol{x}) = \boldsymbol{y} - \boldsymbol{h}. \tag{7}$$

The mean squared loss is minimized during learning over a single batch of 50 samples:

$$J(f_\cdot(\boldsymbol{x})) = \frac{1}{50}\sum_{1}^{50}\frac{1}{2}\left\|f_\cdot(\boldsymbol{x}) - \hat{\boldsymbol{y}}\right\|_2^2. \tag{8}$$

We take 40 weight values evenly spaced in the interval $[0, 6]$ for each neural network parameter $w_1\, w_2$ and generate all possible pairs (cartesian product). For each parameter vector of $\boldsymbol{w} = (w_1\, w_2)$ we calculate the gradients $\nabla_{\boldsymbol{w}J(f_1)}$ and $\nabla_{\boldsymbol{w}J(f_2)}$ respectively. The calculated gradients are illustrated in Figure 3, where fractional decoupling better approximates the underlying coefficient vector. Using fractional decoupling generally has little effect on the gradient direction and magnitude. The minimum magnitude gradient points for each neural network are expected to be local minima for J(see black disks in Figure 3). During inference fractional decoupling yields a lower loss value $J(\boldsymbol{y}-\boldsymbol{h})) \approx 0.02$ compared to the continuous neural network with floored output $J(\boldsymbol{y})) \approx 0.28$.

C.4.1 Selection

A non-dominated sorting is performed across all discretized solutions to determine the best solutions from each trajectory. The mean objective value per feature

$$\overline{\zeta}_j = \frac{\sum_{\boldsymbol{x}(t)\in F_1} \zeta_j(\boldsymbol{x}(t))}{|F_1|} \tag{9}$$

is calculated over all non-dominated solutions, i.e., all samples $\boldsymbol{x} \in F_1$, and then each element $\overline{\zeta}_j$ is used to calculate gradients and perform the parameter update. Mean objective values $\overline{\zeta}_j$ can be scaled before gradient calculation to match consumer preferences and guide the algorithm towards non-dominated solutions that perform better in specific objective values. To select the B solutions that are used as input population for the next generation, the hyper-volume and non-dominated ranks are used [22] as described in the MO-NES baseline in Appendix C.2.

C.4.2 Back-Propagation Through Evolution

The back-propagation through evolution starts by calculating the individual objective function values for each solution selected by the selection operator. For each objective, the mean value over all selected individuals is calculated. Experimental results indicated that using a different optimizer for each Neural Operator yields higher performance. The RMSProp optimizer is used with learning rate $\eta = 0.0001$ for the neural mutation operator and an RMSProp optimizer with learning rate

$\eta = 0.0001$ for the Neural Crossover Operator. The gradient is calculated iteratively per objective, and the loss is scaled 7 times for health objectives. Such scaling resembles a scalarization method [65] optimization, although a weighted sum may not used for gradient calculation. Not scaling the loss yielded recommendations that did not optimize health objectives well, as environmental and cost objectives were often positive-correlated and dominated the gradient upgrades. In general, each objective loss can also be scaled to match explicit consumer preferences. Consumers may rank or score the most important objectives, and such scores can be used as scaling factors for the objectives [4].

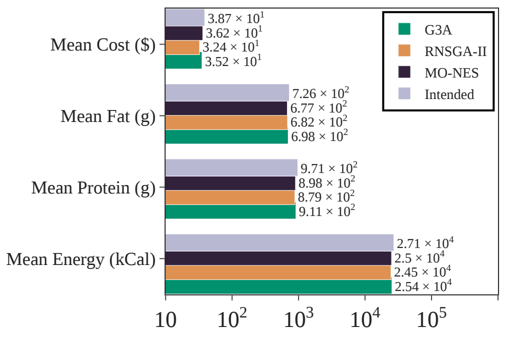
D Appendix: Experimental Evaluation

All baselines are evaluated in weekly basket purchases over 85 weeks for 500 households, and 28400 intended baskets are considered. In particular, the households are chosen based on their total greenhouse gasses (GHG) emissions, i.e., the top 500 emission producers are selected. For each recommendation, a ratio toward the intended basket's cost, environmental impact, or nutritional quantities is considered. Some ratio functions coincide with the proposed objective functions, but this is not the case for nutritional losses, as the normalized MSE showed better convergence but required scaling. Other GA baselines were also considered [13, 15, 55], but did not produce competitive results and thus are skipped for brevity.

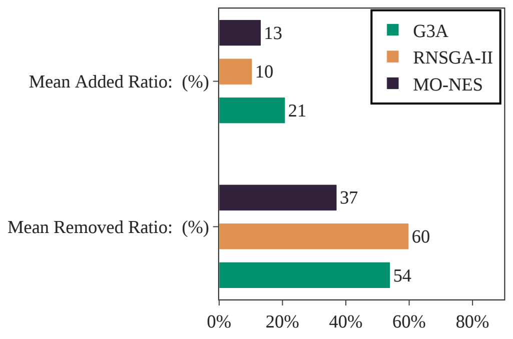
It is important to note that all three baselines were tested on a subset of potential hyper-parameters. Hyper-parameter optimization was performed for several days to the extent that each method could solve the problem effectively. From observed models, the best-performing parameterization per method was selected. In future work, G3A will be compared against other optimization methods on more established problems to determine performance in terms of optimality. Such a study was out of the scope of this article. The population sizes were chosen after evaluating different values. The sizes that generated well-performing solutions efficiently were preferred. G3A is parameterized to generate $B = 8$ recommendations per intended basket, whereas RNSGA-II and MO-NES generate $B = 10$ recommendations per intended basket.

Real-World Impact Comparison: A counterfactual scenario is evaluated to extend the comparison of G3A and estimate the impact on total reduction values. 5000 counterfactual trajectories are sampled for each model, each trajectory being 86 weeks long. For each trajectory, 25% of all intended baskets is assumed to be replaced with a recommendation. The recommendation which replaces the intended basket is chosen randomly[4]. Figure 4 illustrates the ability of all algorithms to achieve a considerable reduction of environmental impact compared to the intended basket. For example, deciding to replace 25% of intended baskets with a G3A recommendation leads to a reduction of approximately 35 metric kilo-tons of CO_2 eq. or approximately 1 billion liters of stressed freshwater for G3A. The current results indicate that G3A performs similarly to RNSGA-II regarding environmental impact, by removing less and adding more products. MO-NES instead produces recommendations that have the most negligible impact on the consumer basket.

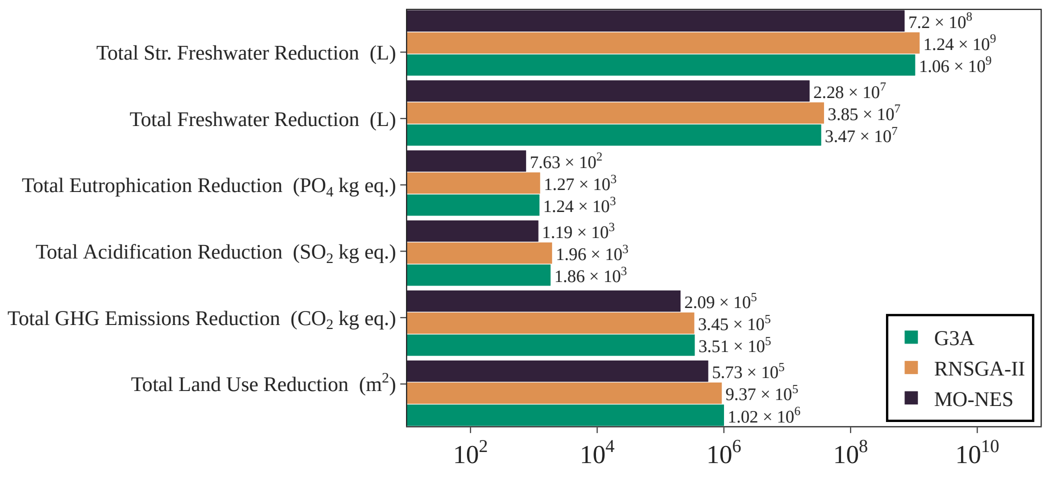
[4]In the current setting, a decision model for sampling, such as the one in [33] cannot be used because the transactions of the current dataset may be affected by marketing campaigns and other covariates. Furthermore, whether consumers were aware of sustainability issues when purchasing is not apparent. Thus, the modeling of environmental impact decision factors may be invalid. Therefore, designing a valid decision model to estimate the effect of a recommender system in this case is out of the scope of this article and could be considered as future work.



(a) Mean nutritional value and cost per basket per trajectory.



(b) Mean ratios of added and removed units per basket per trajectory.



(c) Mean reduction of the total environmental impact of accepting recommendations versus purchasing only intended baskets per sampled trajectory. Longer bars perform better.

FIGURE 4: Comparison of the impact per model over 5000 trajectories, where 25% of the intended basket purchases is randomly replaced with a recommendation. Mean nutritional quantities per basket and trajectory are reported (see a). Next (see b.), the mean value of added and removed units per basket is provided over all recommendations and trajectories, where intended baskets are omitted for the calculation. The total environmental impact and cost reduction are calculated per sample and then subtracted from the total quantities of the original trajectory (only intended baskets are purchased). Although confidence intervals are calculated, they are omitted as they are mostly too narrow and, thus, not visible in the log scale.