Proceedings of the Thirty-Third International Conference on Automated Planning and Scheduling (ICAPS 2023)

Combining Heuristic Search and Linear Programming to Compute Realistic Financial Plans

Alberto Pozanco, Kassiani Papasotiriou, Daniel Borrajo*, Manuela Veloso

J.P. Morgan AI Research

{alberto.pozancolancho, kassiani.papasotiriou, daniel.borrajo, manuela.veloso}@jpmorgan.com

Abstract

Defining financial goals and formulating actionable plans to achieve them are essential components for ensuring financial health. This task is computationally challenging, given the abundance of factors that can influence one's financial situation. In this paper, we present the Personal Finance Planner (PFP), which can generate personalized financial plans that consider a person's context and the likelihood of taking financially related actions to help them achieve their goals. PFP solves the problem in two stages. First, it uses heuristic search to find a high-level sequence of actions that increase the income and reduce spending to help users achieve their financial goals. Next, it uses integer linear programming to determine the best low-level actions to implement the high-level plan. Results show that PFP is able to scale on generating realistic financial plans for complex tasks involving many low level actions and long planning horizons.

Introduction

Setting financial goals and planning ahead are crucial for achieving financial health whether for individuals, households or companies. For individuals, financial planning involves managing monetary resources through expenditures, investments and savings, while considering various life events, risks and goals. The benefits of financial planning have been studied and quantified using economic well-being indicators in both empirical (Warschauer and Sciglimpaglia 2012; Farinella, Bland, and Franco 2017) and theoretical settings (Hanna and Lindamood 2010).

Consulting a personal finance professional is the most common way of seeking financial advice, as they can help clients make decisions about investments, budgeting and other courses of action to achieve their goals. Such services are often very expensive and thus inaccessible to many people. Alternatives to speaking to an advisor include personal finance assessment tools and questionnaires which offer semi-personalized advice to users based on their input. Automated methods for financial planning often rely on expert systems or similar rule-based approaches (Kindle et al. 1989; Phillips, Nielson, and Brown 1992; Althnian 2021). The main weakness of these approaches is that they do not provide detailed solutions (i.e., plans with monthly actions). They also do not consider the feasibility of the recommended plans based on the user financial habits.

In this paper we present the Personal Finance Planner (PFP), which generates realistic plans that achieve users' financial goals. Due to the large action space, (i.e., there is a potentially great number of income and expenses sources), PFP solves the problem hierarchically in two stages, by exploiting the task's structure. First, it uses heuristic search to find a high-level sequence of income increase and spending decrease actions at each month that achieve the financial goal. Then, it uses integer linear programming (ILP) to decide how to implement the prescribed high-level plan by composing the right low-level actions to be applied at each month. In this paper, we primarily focus on personal finance planning. But our framework can also be applied to assist with financial planning tasks for households and companies.

Financial Planning Tasks

We aim to find realistic plans that allow users to transit from their current financial state to a state that fulfills their financial goals. Formally:

Definition 1. *A **Financial Planning Task** is a tuple FPT = $\langle \mathcal{F}, \mathcal{A}, I, G \rangle$ where $\mathcal{F}$ is a set of real-valued variables, $\mathcal{A}$ is a discrete set of joint actions, I is the initial state and G is a goal state.*

The variables in $\mathcal{F}$, that represent each state, are the time step t, the account balance B, and a set of monthly income and expenses sources or *types* $\mathcal{T}$. These income and expenses types can be seen as the different budgets or categories individuals can act upon, such as job income or housing expenses. The account balance B at time step t_{i+1} is computed as the sum of the balance in t and the sum of all the monthly income sources in t, minus the sum of all the monthly expenses sources in t.

A state s of a FPT task is an assignment over the real-valued numeric variables in $\mathcal{F}$. We will use s_X to access the value of a numeric variable $X \in \mathcal{F}$ of a state s. Numeric variables can appear in conditions $\xi \rhd k$, where ξ is an arithmetic expression, $\rhd \in \{\leq, \geq, =, <, >\}$, and k is a constant. The goal state is a conjunction of conditions taking the form $\langle s_{\mathcal{F}}, s_t \rangle \rhd k$, i.e., they specify the value of a real-valued variable in a given time step. In the rest

*On leave from Universidad Carlos III de Madrid

of the paper we will assume goals are formed by a single condition stating the account balance in a given time step. For example, a goal could be $\{\langle s_B, 4\rangle \geq 2000\}$, i.e., to have at least \$2000 in the account's balance in 4 months time.

Joint actions $\mathcal{A}$ are composed of a set of actions A that modify a financial state. We will use A_T to denote the subset of actions that modify a given budget type $T \in \mathcal{T}$. Each action $a \in A$ is defined in terms of its preconditions (pre(a)), which are conjunctions of numeric conditions over $\mathcal{F}$; and effects (eff(a)), which increase/decrease the value of numeric variables in $\mathcal{F}$ by a given percentage. In the rest of the paper we assume the same percentages effects for all the budget types. The actions execution is defined as a function $\gamma : S \times A \rightarrow S$ that specifies the state that results of applying an action in a given state. In addition to preconditions and effects, each action is assigned a *likelihood* value –a real number between 0 and 1 that reflects the feasibility of a user executing that action. For example, an action suggesting a job income increase of 0% will usually have a likelihood of 1, since no financial effort is required from the user. Instead, an action suggesting a health expenses decrease of 10% will have a likelihood close to 0, since this implies a more demanding change to the users' financial habits. In practice, these likelihoods can be given or inferred from data.

At each time step, a discrete set of joint actions $\mathcal{A}$ can be applied to modify the financial state. Each joint action $J \in \mathcal{A}$ is a set composed of an *action* $a \in A$ for each income and expense type in $\mathcal{T}$. Therefore, $|J| = |\mathcal{T}|$. We overload γ to define the execution of a joint action $J \in \mathcal{A}$ as $\gamma : S \times J \rightarrow S$. Joint actions can be applied in a state s when $\bigcup_{a \in J}$ pre(a) are satisfied in s. Their effects are given as the union of all its actions' effects $\bigcup_{a \in J}$ eff(a), plus advancing the time step and updating the balance B by summing the incomes and subtracting the expenses budgets in the new state. The likelihood of joint actions is computed as the product of its actions likelihood, likelihood$(J) = \Pi_{a \in J}$likelihood(a). Therefore, joint actions involving many demanding actions will have a lower likelihood.

The solution to a FPT is a plan, which consists of a sequence of joint actions that allow the user to transit from the initial state I to a state s where all goal conditions in G are true, $s \models G$. Therefore, a plan $\pi = \langle J_1, J_2, \ldots, J_n\rangle$ solves a FPT iff $\forall J_i \in \pi, J_i \in \mathcal{A}$, and $\gamma(\ldots\gamma(\gamma(I, J_1), J_2)\ldots), J_n) \models G$. A plan π solves a FPT optimally if there is no other plan with higher likelihood product of its joint actions:

$$\prod_{J \in \pi} \text{likelihood}(J) \tag{1}$$

We face two obstacles when trying to use search algorithms to (optimally) solve the problem as defined in Expression 1: (1) plan optimality is defined as a product, while the evaluation functions used within search algorithms typically use additive functions (as the g function used within A*); and (2) plan optimality is defined as a maximization task (maximize likelihood), while common search algorithms aim to minimize a given function. To overcome the first problem, we compute the logarithm of each action's likelihood score so we can sum them. To overcome the second problem, we transform the maximization task into a minimization task by subtracting the logarithm of the likelihood score from one. This process was previously proposed by Jimenez et al. (2006) to compile probabilistic planning tasks into deterministic ones. By performing these two transformations, now we have the following additive cost function that search algorithms can minimize:

$$c(J) = 1 - log_e(\text{likelihood}(J)) \tag{2}$$

Given a plan π and its cost $c(\pi)$, we can compute its likelihood by applying the following operation:

$$\text{likelihood}(\pi) = e^{-(c(\pi)-|\pi|)} \tag{3}$$

PFP: Hierarchically Combining Heuristic Search and Linear Programming to Solve FPT

The state space of a FPT grows as we increase the number of time steps we want to plan ahead (depth); or the number of income and expenses types and the actions we consider for each type (breadth). The former is constrained in practice to relatively small values, since the current use case we are considering focuses on short or medium term financial goals. However, the latter leads to a huge joint action space that renders useless any attempt to solve the problem monolithically. In our previous attempt to solve this problem (Pozanco, Papasotiriou, and Borrajo 2022), standalone automated planning and heuristic search approaches were not able to scale, even when only considering just two budgets and three actions. Currently, we are considering a more ambitious setting where we have nine different budgets ($|\mathcal{T}| = 9$), and five different actions that can be applied over each budget. This yields a joint action space $\mathcal{A}$ of 5^9 actions. One way of reducing this joint action space would be to work at the action space level by imposing a sequential order of actions at every month, (i.e. first act on food expenses, then health expenses, then travel expenses, etc.) and finally move to the next month. But this approach has two main drawbacks: we would be increasing the depth of the search tree a lot; and designing heuristics would be more complex.

In this paper we have chosen a different approach and propose the **Personal Finance Planner (PFP)** which leverages a hierarchical approach that exploits the structure of the task to overcome the joint action space size challenge. Making large problems tractable through hierarchical decomposition have been extensively studied in the search and planning communities (Knoblock 1990; Bercher, Alford, and Höller 2019), and our approach is inspired by these works. First, we use heuristic search to find a high-level sequence of income increase and expense decrease joint actions at each month that achieve the financial goal. Then, we use ILP to decide how to implement the joint actions prescribed by the high-level plan by choosing the best low-level combination of actions to be applied at each month.

High-Level Planning through Heuristic Search

In order to compute a high-level plan, we need to: (1) define how to create the high-level problem space from the original FPT; and (2) define the heuristics that will drive the search in the high-level task.

Algorithm 1: Abstract Action Space Computation

Input: A, N
Output: $\mathcal{A}'$
1: $A_{inc}, A_{exp} \leftarrow$ GROUPBYABSTRACTTYPES(A)
2: minInc, maxInc $\leftarrow$ GETMINMAXEFF(A_{inc})
3: minExp, maxExp $\leftarrow$ GETMINMAXEFF(A_{exp})
4: minIncCost, maxIncCost $\leftarrow$ GETMINMAXCOST(A_{inc}, minInc, maxInc)
5: minExpCost, maxExpCost $\leftarrow$ GETMINMAXCOST(A_{exp}, minExp, maxExp)
6: $A'_{inc} \leftarrow$ LI(minInc, maxInc, minIncCost, maxIncCost, N)
7: $A'_{exp} \leftarrow$ LI(minExp, maxExp, minExpCost, maxExpCost, N)
8: $\mathcal{A}' \leftarrow \{A'_{inc} \times A'_{exp}\}$
9: **return** $\mathcal{A}'$

Algorithm 2: Greedy Heuristic (GH)

Input: $s, G, \mathcal{A}'$, Admissible
Output: h
1: $h \leftarrow \infty$, remainingTimeSteps $\leftarrow G_t - s_t$
2: **for** $J \in$ SORTBYCOST($\mathcal{A}'$) **do**
3: $s' \leftarrow$ EXECUTE(remainingTimeSteps, J, s)
4: **if** $s' \models G$ **then**
5: **if** Admissible = True **then**
6: $h \leftarrow c(J)$
7: **else**
8: $h \leftarrow c(J) \times$ remainingTimeSteps
9: **return** h
10: **return** h

Defining the High Level Problem Space The account balance is computed as a function of the sum of incomes minus the sum of expenses. Therefore, we can generate an abstract problem space by reducing the size of $\mathcal{T}$ and only consider two *abstract action types*: income increase actions A_{inc} and expense decrease actions A_{exp}. However, we need to exercise caution when creating this abstraction so it fulfills the downward refinement property (DRP) (Bacchus and Yang 1991), i.e., given that a low level solution exists, every abstract solution can be refined to a low level solution without backtracking across abstract levels.

The computation of the new action and joint action space is described in Alg. 1. It receives as input the original set of actions A and the number of buckets $N \in \mathbb{N} \geq 2$, which determines the level of abstraction of the new action space. The algorithm first groups the actions in A into the two different abstract types: A_{inc} (those that increase income) and A_{exp} (those that decrease expenses). The algorithm then computes the min and max increase (min/max effect of A_{inc} actions) and decrease (min/max effect of A_{exp} actions) effect as well as their cost (lines 2-5). The cost is computed through the GETMINMAXCOST function, which sums the cost of all the actions that produce the given max/min income(expense) increase(decrease). This information is used to generate the new action space of each abstract type in the linear interpolation (LI) function (lines 6-7). This function takes the action with the min and max increase (decrease) effect over the income (expenses) abstract types, and generates N new actions by linearly interpolating these two points and returning $N - 2$ points equally spaced between the min and max values. For example, assuming minInc $= 0\%$, maxInc $= 10\%$, minIncCost $= 1$, maxIncCost $= 3$ and $N = 3$, this function would return the two input actions plus an interpolated action a with eff$(a) = 5\%$ and $c(a) = 2$. Finally, the algorithm returns the set of high-level joint actions. We also compact the state representation by summarizing all the budget types in $\mathcal{T}$ into two abstract types $\mathcal{T}' = \langle \text{Inc}, \text{Exp} \rangle$. We refer as $\mathcal{F}' = \langle t, B, \text{Inc}, \text{Exp} \rangle$ to the new abstract state representation, and FPT$' = \langle \mathcal{F}', \mathcal{A}', I, G \rangle$ as the new high-level task. As a reminder, we are assuming goals related to increasing the account balance. Alg. 1 guarantees $\mathcal{A}'$ does not contain actions that can increase the balance more than the actions in $\mathcal{A}$.

Therefore, any plan solving FPT$'$ will use income increases and expense decreases that can be replicated by actions in $\mathcal{A}$ to achieve the goal and solve the FPT. Thus, PFP$'$ satisfies the DRP.

Computing a High-Level Plan We use A* (Hart, Nilsson, and Raphael 1968) to solve FPT$'$. The cost of reaching a state s is computed using Eq. 2. In order to estimate the cost of reaching the goal from s, we propose a heuristic based on solving a relaxation of the problem where only the same joint action can be applied at every step. In other words, the number of potential plans is limited to the number of joint actions. Alg. 2 outlines the computation of this heuristic.

The algorithm receives as input the current state (s) and goals (G), the available high level joint actions ($\mathcal{A}'$), and a parameter that indicates whether the heuristic should be admissible or not. It first computes the number of remaining time steps from s to reach G. Next, the algorithm iterates over the list of joint actions (sorted according to their cost as per Eq. 1), executing the given joint action J from s for the number of remaining steps, and yielding a state s'. If the goal is satisfied in s', the algorithm finishes and returns the heuristic estimate h. This heuristic value will depend on the admissibility parameter. If we are interested in an admissible heuristic (GH$_a$), Alg. 2 returns the cost of executing that joint action, $c(J)$. If we want a more informative but inadmissible heuristic (GH$_i$), Alg. 2 returns the cost of executing that action times the number of remaining time steps. The output of using A* with the defined heuristic will be a high level plan π' consisting of high level joint actions $J' \in \mathcal{A}'$ that solve FPT$'$ prescribing a *summary* of how individuals should modify their income and expenses on each month.

Low-Level Planning through ILP

We compute an actionable low-level plan from π' by sequentially solving an optimization task at each time step (e.g. month). This process is shown in Alg. 3. It receives as input the high level plan, the original actions, and the initial state; and returns a low-level plan π which is the solution of a FPT. The algorithm iterates over π', extracting the high-level joint action J' prescribed for each month. This joint action is given as input to the ILP model below, which computes the low-level joint action J that *better* replicates J'.

$$\text{minimize} \sum_{a \in A} x_a \times c(a) \tag{4}$$

Configuration		3 months		6 months		9 months		12 months	
Heuristic	#Buckets N	$c(\pi)$	Time (s)	$c(\pi)$	Time (s)	$c(\pi)$	Time (s)	$c(\pi)$	Time (s)
GH_i	2	111.0 ± 12.4	0.3 ± 0.0	53.3 ± 8.2	0.6 ± 0.0	43.8 ± 0.0	0.9 ± 0.0	26.1 ± 7.5	1.2 ± 0.0
	3	101.6 ± 18.9	0.3 ± 0.0	47.7 ± 3.4	0.6 ± 0.0	30.3 ± 0.2	0.9 ± 0.1	20.6 ± 3.3	1.2 ± 0.0
	4	98.3 ± 21.1	0.3 ± 0.0	45.3 ± 5.3	0.6 ± 0.0	25.7 ± 0.0	0.9 ± 0.0	19.6 ± 2.6	1.3 ± 0.1
	5	98.3 ± 21.1	0.4 ± 0.1	43.3 ± 4.8	0.6 ± 0.1	29.6 ± 2.3	0.9 ± 0.1	18.1 ± 1.6	1.2 ± 0.0
GH_a	2	111.0 ± 12.4	0.3 ± 0.0	40.3 ± 7.9	0.6 ± 0.0	30.8 ± 2.5	0.9 ± 0.0	26.1 ± 7.5	1.3 ± 0.0
	3	101.6 ± 18.9	0.3 ± 0.0	35.5 ± 4.7	0.7 ± 0.0	21.2 ± 0.4	1.6 ± 0.0	20.6 ± 3.3	7.0 ± 1.7
	4	98.3 ± 21.1	0.3 ± 0.0	36.0 ± 2.9	1.4 ± 0.1	25.4 ± 1.0	19.5 ± 1.5	–	–
	5	98.3 ± 21.1	0.4 ± 0.1	31.0 ± 3.4	4.5 ± 1.3	–	–	–	–

Table 1: Solution cost ($c(\pi)$) and total execution time (Time(s)) in problems with increasing complexity (number of months) for different configurations of PFP where we vary the heuristic (GH_i and GH_a) and #buckets.

Algorithm 3: Computing a Low-Level Plan

Input: π', A, I
Output: π
1: $\pi \leftarrow \emptyset$
2: **for** $J' \in \pi'$ **do**
3: $\quad J \leftarrow \text{ILP}(J', A, I)$
4: $\quad \pi \leftarrow \pi \cup J$
5: $\quad I \leftarrow \gamma(J, I)$
6: **return** π

subject to:

$$\sum_{a \in A_T} x_a = 1,\ T \in \mathcal{T} \tag{5}$$

$$\sum_{a \in A_{inc}} \text{IncomeDiff}(x_a, I) \geq \text{IncomeDiff}(J'_{inc}, I) \tag{6}$$

$$\sum_{a \in A_{exp}} \text{ExpensesDiff}(x_a, I) \geq \text{ExpensesDiff}(J'_{exp}, I) \tag{7}$$

We create one binary decision variable x_a for each action $a \in A$. These variables will take a value of 1 if action a is included in the joint action J, and 0 otherwise. The ILP minimizes the sum of the costs of the actions included in J (Eq. 4). Constraint 5 ensures that only one action of each type is included in J, i.e., the joint action does not suggest two expense decreases over the same budget. Constraint 6 ensures that the income difference achieved by the combination of actions $a \in A$ is greater than or equal to the income difference achieved by the joint action J', while Constr. 7 does the same for the expenses. After the ILP returns a joint action J, it is added to the plan π and the state is I updated.

Evaluation

We evaluate PFP in problems with 9 different budgets, divided into 2 income and 7 expense budgets. These budgets range from job income to food and housing expenses, and were selected by domain experts. We consider 5 different actions to be applied over each budget by either increasing or decreasing it by: $0, 2.5, 5, 7.5$ and 10%. Note that a 0% increase or decrease action is equivalent to a no-op action over that budget. Each action has a different likelihood provided by domain experts. This results in a joint action space $\mathcal{A}$ of 5^9 actions. We generate FPTs of increasing difficulty by varying the time horizon when the goal balance has to be achieved: $3, 6, 9$ and 12 months. We set the initial balance to $\$1000$, and the goal balance to be a random number between 2.0 and 2.2 times the initial balance. The sum of all income budgets is also $\$1000$, as well as the sum of all expense budgets, thus creating problem instances where no savings are generated if the initial financial state remains unchanged. We randomly distribute these numbers into the different income and expenses budgets. Search algorithms and heuristics have been implemented in Python 3.8. ILP tasks are solved using the PuLP library (Mitchell, OSullivan, and Dunning 2011) and the CBC solver (Forrest and Lougee-Heimer 2005). We solve each problem with the two different heuristics GH_a and GH_i and 4 different number of buckets: $2, 3, 4$ and 5. We stop there because otherwise the abstract search would be more granular than the original FPT. Experiments were run in Intel(R) Xeon(R) E3-1585L v5 @ 3.00GHz machines with 64GB of RAM and a 30s timeout, as PFP is meant to work in a real-time setting. Results for these configurations are reported in Tab. 1.

The high-level search is PFP's bottleneck, as it requires $> 80\%$ of the total execution time in all configurations. PFP scales better when using the inadmissible heuristic GH_i to solve the higher level task, returning solutions in around 1s. On the other hand GH_a returns better solutions in some scenarios, but cannot solve the high level tasks in problems with long time horizons or many buckets. As expected, we observe a trend across heuristics and time horizons where higher N's values translate to less costly, i.e., more likely plans. However, this trend is not always monotonic, as in some cases our linear interpolation might not accurately represent some of the non-linear relationships between the likelihoods of the original actions in the FPT. Finally, plans involving longer horizon goals tend to be less costly since they allow the planner to pick less aggressive actions.

Conclusions

In this paper we have presented PFP, a personal finance planner that generates realistic financial plans by hierarchically combining heuristic search and ILP. Experimental results show how PFP scales to large instances and returns results in real time. An ongoing qualitative evaluation of the planner is showing users are interested in having software that guides them through their financial planning.

Acknowledgements

This paper was prepared for informational purposes by the Artificial Intelligence Research group of JPMorgan Chase & Co. and its affiliates ("JP Morgan"), and is not a product of the Research Department of JP Morgan. JP Morgan makes no representation and warranty whatsoever and disclaims all liability, for the completeness, accuracy or reliability of the information contained herein. This document is not intended as investment research or investment advice, or a recommendation, offer or solicitation for the purchase or sale of any security, financial instrument, financial product or service, or to be used in any way for evaluating the merits of participating in any transaction, and shall not constitute a solicitation under any jurisdiction or to any person, if such solicitation under such jurisdiction or to such person would be unlawful.

The authors would like to acknowledge Naftali Cohen, Nicolas Marchesotti, Alice McCourt, Andrea Stefanucci, and Jessica Staddon for their input and suggestions at various key stages of the research.

References

Althnian, A. 2021. Design of a Rule-based Personal Finance Management System based on Financial Well-being. *International Journal of Advanced Computer Science and Applications*, 12(1).

Bacchus, F.; and Yang, Q. 1991. The Downward Refinement Property. In Mylopoulos, J.; and Reiter, R., eds., *Proceedings of the 12th International Joint Conference on Artificial Intelligence. Sydney, Australia, August 24-30, 1991*, 286–293. Morgan Kaufmann.

Bercher, P.; Alford, R.; and Höller, D. 2019. A Survey on Hierarchical Planning-One Abstract Idea, Many Concrete Realizations. In *IJCAI*, 6267–6275.

Farinella, J.; Bland, J.; and Franco, J. 2017. The Impact of Financial Education on Financial Literacy and Spending Habits. *International Journal of Business, Accounting, & Finance*, 11(1).

Forrest, J.; and Lougee-Heimer, R. 2005. CBC user guide. In *Emerging theory, methods, and applications*, 257–277. INFORMS.

Hanna, S. D.; and Lindamood, S. 2010. Quantifying the economic benefits of personal financial planning. *Financial Services Review*, 19(2).

Hart, P.; Nilsson, N.; and Raphael, B. 1968. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2): 100–107.

Jiménez, S.; Coles, A.; and Smith, A. 2006. Planning in probabilistic domains using a deterministic numeric planner. In *25th Workshop of the UK Planning and Scheduling Special Interest Group*.

Kindle, K. W.; Cann, R. S.; Craig, M. R.; and Martin, T. J. 1989. PFPS-Personal Financial Planning System. In *IAAI*.

Knoblock, C. A. 1990. Learning Abstraction Hierarchies for Problem Solving. In Shrobe, H. E.; Dietterich, T. G.; and Swartout, W. R., eds., *Proceedings of the 8th National Conference on Artificial Intelligence. Boston, Massachusetts, USA, July 29 - August 3, 1990, 2 Volumes*, 923–928. AAAI Press / The MIT Press.

Mitchell, S.; OSullivan, M.; and Dunning, I. 2011. PuLP: a linear programming toolkit for python. *The University of Auckland, Auckland, New Zealand*, 65.

Phillips, M. E.; Nielson, N. L.; and Brown, C. E. 1992. An Evaluation of Expert Systems. *Journal of Financial Counseling and Planning*, 3(1).

Pozanco, A.; Papasotiriou, K.; and Borrajo, D. 2022. PFPT: a Personal Finance Planning Tool by means of Heuristic Search and Automated Planning. *3rd Workshop on Planning for Financial Services (Finplan) at ICAPS'22*, 10.

Warschauer, T.; and Sciglimpaglia, D. 2012. The economic benefits of personal financial planning: An emperical analysis. *Financial Services Review*, 21(3).