Artificial Intelligence 196 (2013) 26–52

Contents lists available at SciVerse ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Physical search problems with probabilistic knowledge ☆

Noam Hazon [a,*,1], Yonatan Aumann [b], Sarit Kraus [b], David Sarne [b]

[a] Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA
[b] Department of Computer Science, Bar-Ilan University, Ramat Gan, Israel

ARTICLE INFO

Article history:
Received 31 August 2011
Received in revised form 30 September 2012
Accepted 24 December 2012
Available online 3 January 2013

Keywords:
Graph search
Economic search

ABSTRACT

This paper considers the problem of an agent or a team of agents searching for a resource or tangible good in a physical environment, where the resource or good may possibly be obtained at one of several locations. The cost of acquiring the resource or good at a given location is uncertain (a priori), and the agents can observe the true cost only when physically arriving at this location. Sample applications include agents in exploration and patrol missions (e.g., an agent seeking to find the best location to deploy sensing equipment along its path). The uniqueness of these settings is in that the cost of observing a new location is determined by distance from the current one, impacting the consideration for the optimal search order. Although this model captures many real world scenarios, it has not been investigated so far.

We analyze three variants of the problem, differing in their objective: minimizing the total expected cost, maximizing the success probability given an initial budget, and minimizing the budget necessary to obtain a given success probability. For each variant, we first introduce and analyze the problem with a single agent, either providing a polynomial solution to the problem or proving it is NP-complete. We also introduce a fully polynomial time approximation scheme algorithm for the minimum budget variant. In the multi-agent case, we analyze two models for managing resources, shared and private budget models. We present polynomial algorithms that work for any fixed number of agents, in the shared or private budget model. For non-communicating agents in the private budget model, we present a polynomial algorithm that is suitable for any number of agents. We also analyze the difference between homogeneous and heterogeneous agents, both with respect to their allotted resources and with respect to their capabilities. Finally, we define our problem in an environment with self-interested agents. We show how to find a Nash equilibrium in polynomial time, and prove that the bound on the performance of our algorithms, with respect to the social welfare, is tight.

1. Introduction

Frequently, in order to successfully complete its task, an agent may need to *explore* (i.e., search) its environment and choose among different available options. For example, an agent seeking to purchase a product over the Internet needs to query several electronic merchants in order to learn their posted prices; a robot searching for a resource or a tangible good needs to travel to possible locations where the resource is available and learn the configuration in which it is available, as

☆ This paper extends two earlier conference papers (Aumann et al., 2008 [6]; Hazon et al., 2009 [31]).

* Corresponding author.
E-mail addresses: noamh@cs.cmu.edu (N. Hazon), aumann@cs.biu.ac.il (Y. Aumann), sarit@cs.biu.ac.il (S. Kraus), sarned@cs.biu.ac.il (D. Sarne).

[1] This work was done while the author was at Bar-Ilan University.

0004-3702/$ – see front matter
http://dx.doi.org/10.1016/j.artint.2012.12.003

well as the difficulty of obtaining it there. In these environments, the benefit associated with an opportunity is revealed only upon observing it. The only knowledge available to the agent prior to observing the opportunity is the probability associated with each possible value of each prospect.

While in virtual environments the exploration can sometimes be considered costless, in physical environments traveling and observing typically entails a cost. Furthermore, traveling to a new location may increase or decrease the distance to other locations, so the cost associated with exploring other unexplored locations changes. For example, consider a Rover robot with the goal of mining a certain mineral. Potential mining locations may be identified based on satellite imaging, each location associated with some uncertainty regarding the difficulty of mining there. In order to assess the amount of battery power required for mining at a specific location, the robot needs to physically visit there. The robot's battery is thus used not only for mining the mineral but also for traveling from one potential location to another. Consequently, an agent's strategy in an environment associated with search costs should maximize the *overall* benefit resulting from the search process, defined as the value of the option eventually used, minus the costs accumulated along the process, rather than merely finding the best valued option.

In physical environments, it is common to use a team of agents rather than a single agent. Extending the single agent solution to multi-agent strategy may require subdividing the search space among the different agents. However, if agents have means of communication, then they may not wish to become too distant, as they can call upon each other for assistance. For example, even if a Rover does not have sufficient battery power for mining at a given location, it may be useful for it to travel to the site in order to determine the exact mining cost, and call for other robots that do have the necessary battery power. In this case, the scheduling of the robots' travel times is key, and must be carefully planned. If the agents are not fully cooperative, a selfish behavior should also be considered. Each one of the agents will try to minimize its traveling costs while still achieving the group's goal.

Finally, agents may be of different types, or with different amounts of resources. For example, Rover robots may be entering the mission with differing initial battery charges. They may also differ in their capabilities, like a team of Rovers in which some were specifically designed for mining missions, and thus require less battery power for the same mining task.

This paper aims at taking the first steps in understanding the characteristics of such physical search environments, both for the single and multi agent cases, and developing efficient exploration strategies for the like. Our main focus is on the case where the opportunities are aligned along a path, as in the case of perimeter patrol [60,19,2,3]. We note that many single and multi-agent coverage algorithms convert their complex environment into a simple long path [52,25,32]. Furthermore, we show that the problem in more general metric spaces is NP-complete, even for a tree graphs. For exposition purposes, in the remainder of the paper we use the classical procurement application where the goal of the search is purchasing a product and the value of each observed opportunity represents a price. Of course, this is only one example of the general setting of exploration in a physical environment, and the discussion and results of this paper are relevant to any such setting, provided that exploration and fulfilling the task consume the same type of resource.

We consider three variants of the problem, differing in their objective. The first (*Min-Expected-Cost*) is the problem of an agent that aims to minimize the expected total cost of completing its task. The second (*Max-Probability*) considers an agent that is given a budget for the task (which it cannot exceed) and aims to maximize the probability it will complete the task (e.g., reach at least one opportunity with a budget large enough to successfully buy the product). In the last variant (*Min-Budget*) the agent is required to guarantee a pre-defined probability of completing the task, and aims to minimize the overall budget that will be required to achieve the said success probability. We also consider the multi-agent extensions of these variants. While the first variant fits mostly product procurement applications, the two latter variants fit well into applications of robots engaged in remote exploration, operating with a limited amount of battery power (i.e., a budget).

1.1. Summary of results

We first consider the single agent case. We prove that in general metric spaces all three problem variants are NP-hard. Thus, as mentioned, we focus on the setting where all locations are located along a path. For this setting we provide polynomial algorithms for the *Min-Expected-Cost* problem. We show the other two problems (*Min-Budget* and *Max-Probability*) to be NP-complete even for the path. Thus, we consider further restrictions and also provide an approximation scheme. We show that both problems are polynomial if the number of possible prices is constant. Even with this restriction, we show that these problems are NP-complete on a tree graph. For the *Min-Budget* problem, we provide an FPTAS (fully-polynomial-time-approximation-scheme), that provides a $(1+\epsilon)$ approximation for any $\epsilon > 0$, in time $O(\text{poly}(n\epsilon^{-1}))$, where n is the size of the input.

For the multi-agent case, we first analyze a shared budget model, where all the resources and costs are shared among all the agents. We show that if the number of agents is fixed, then all of the single-agent algorithms extend to k-agents, with the time bounds growing exponentially in k. Therefore the computation of the agents' strategies can be performed whenever the number of agents is relatively moderate, a common scenario in many physical environments where several agents cooperate in exploration and search. If the number of agents is part of the input then the multi-agent versions of *Min-Budget* and *Max-Probability* are NP-complete even on the path and even with a single price.

We then investigate a model of private budgets, where each agent has its own initial budget. We again assume that the number of possible prices is bounded. In this case, we separately consider the setting where agents can communicate and the setting where they cannot. For non-communicating agents we show a polynomial algorithm for the *Max-Probability*

Table 1
Summary of results: n is the input size, m – the number of points (store locations), d – the number of different possible prices, w.p. (with phone) – the version of *Min-Expected-Cost* with the ability to purchase by phone (see Section 1.3), $\bar{d} = d+1$, k – the number of agents, n/a – the problem was not defined in that case or there is no need for a solution, f – the polynomial function which is defined in Lemma 22.

(a) Single agent

	Min-Expected-Cost	*Max-Probability*	*Min-Budget*
General metric spaces	NP-hard	NP-complete	NP-complete
Trees		NP-complete	NP-complete
Path			
Single price	n/a	$O(m)$	$O(m)$
d prices	$O(d^3m^4)$ ($O(d^2m^2)$ w.p.)	$O(m2^d(\frac{e \cdot m}{2d})^{2d})$	$O(m2^d(\frac{e \cdot m}{2d})^{2d})$
General case	$O(d^3m^4)$ ($O(d^2m^2)$ w.p.)	NP-complete	NP-complete
$(1+\epsilon)$ approximation	n/a		$O(n\epsilon^{-6})$

(b) Multi-agent, shared budget, on the path

	Min-Expected-Cost	*Max-Probability*	*Min-Budget*
k agents	$O(d^{2k+1}2^k(\frac{m}{k})^{4k})$ ($O(d^2 2^k(\frac{m}{k})^{2k})$ w.p.)	$O(m2^{kd}(\frac{em}{2kd})^{2kd})$	$O(m2^{kd}(\frac{em}{2kd})^{2kd})$
General case		NP-complete	NP-complete
$(1+k\epsilon)$ approximation	n/a		$O(n\epsilon^{-6k})$

(c) Multi-agent, private budget, on the path

	Max-Probability	*Min-Budget*$^{\text{identical}}$	*Min-Budget*$^{\text{distinct}}$
No-communication			
k fixed	$O(m^3k^2)$	$O(m^3k^2n)$	$O(m2^{kd}(\frac{em}{2d})^{2kd})$
k parameter	$O(m^3k^2)$	$O(m^3k^2n)$	NP-complete
With-communication			
k fixed		$f\left(m2^{k\bar{d}}\left(\frac{em}{2d}\right)^{2k\bar{d}}, k, d, k\right) \in P$	
k parameter			

problem that is suitable for any number of agents. For the *Min-Budget* problem with non-communicating agents, we present a polynomial algorithm for the case that all agents must be allotted identical resources, but show that the problem is NP-hard for the general case (unless the number of agents is fixed). Next we consider agents that can communicate, and can call upon each other for assistance. As noted above, in this case the scheduling of the different agents' moves must also to be carefully planned. We present polynomial algorithms for both the *Max-Probability* problem and the *Min-Budget* problem that work for any constant number of agents (but become non-polynomial when the number of agents is not constant).

We then move to the self-interested agents setting, where the agents seek to obtain an item but each one of them tries to minimize the use of its own private budget for traveling. We define two games, a sequential game, *Min-Expected-Cost-Game*, and simultaneous game, *Min-Budget-Game*. We show that when the number of possible prices is bounded and the number of agents is fixed, the strategy that maximizes the social welfare can be found in polynomial time. We also show that in *Min-Budget-Game* this strategy is a Nash equilibrium, but this is not always the case in *Min-Expected-Cost-Game*. We thus present a polynomial algorithm that guarantees finding a strategy which is a Nash equilibrium. Furthermore, we show an upper bound on the algorithm's performance, and prove that it is tight.

Finally, we extend our results to the case of heterogeneous agents with different capabilities, and discuss the assumptions that we made throughout the paper. Table 1 presents a summary of the results. Empty entries represent open problems.

1.2. Related work

Models of a single agent search process with prior probabilistic knowledge have attracted the attention of many researchers in various areas, mainly in economics and operational research, prompting several reviews over the years [40,42]. These search models have developed to a point where their total contribution is referred to as *search theory*. Probably the most famous problem within this field is the "secretary problem", which has a remarkably long list of articles that have been dedicated to its variations (see [22] for an extensive bibliography).[2] Nevertheless, these economic-based search models, as well as their extensions over the years into multi-agent environments [35,46,36,48], assume that the cost associated with observing a given opportunity is stationary (i.e., does not change along the search process). While this permissive assumption facilitates the analysis of search models, it frequently does not capture the real situation in the physical world, as illustrated in Fig. 1. Therefore, in this paper, we assume that the cost associated with observing a given opportunity may change along the search process. The use of changing search costs suggests an optimal search strategy structure different from the one used in traditional economic search models; other than merely deciding when to terminate its search, the agent also needs to integrate exploration sequence considerations into its decision process.

[2] While the "secretary problem" is a classical optimal-stopping online problem, it does not involve search costs and the goal is to maximize the probability of finding the best candidate rather than minimizing the expected overall cost in our case.

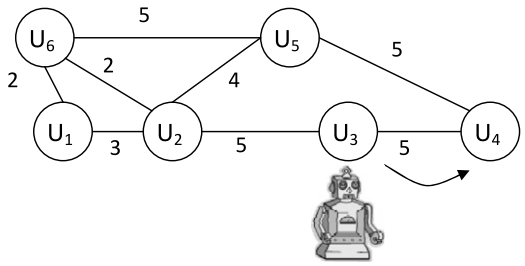

Fig. 1. An example of how the cost associated with observing a given opportunity may change along the search process. When the agent is located at u_3, the cost of observing the prices at u_2 and u_5 are 5 and 9, respectively. When the agent moves to u_4, the cost of observing the prices at u_2 and u_5 become 10 and 5, respectively.

Search with changing search costs has been previously considered in the computer science domain in the contexts of prize-collecting traveling salesman problems [9], the orienteering problem [55] and the graph searching problem [39].

In Prize-Collecting Traveling Salesman Problems (PC-TSP) we are given a graph with non-negative "prize" values associated with each node, and a salesman needs to pick a subset of the nodes to visit in order to minimize the total distance traveled while maximizing the total prize collected. Since there is a tradeoff between the cost of a tour and how much prize it spans, several variants have been developed. All the variants of PC-TSP are NP-hard, as they are generalizations of the famous Traveling Salesman Problem (TSP). One variant of PC-TSP is the k-TSP, where every node has a prize of one and the goal is to minimize the total distance traveled, while visiting at least k nodes. Over the years, several constant-factor approximations have been developed for the k-TSP [5,28,16,8,29]. The Orienteering Problem (OP) is another well-studied variant of PC-TSP, where the goal is to maximize the total prize collected, while keeping the distance traveled below a certain threshold. Several stochastic variants of the OP have been considered. Campbell et al. [17] investigated the Orienteering Problem with Stochastic Travel and Service times (OPSTS). Another stochastic variant of OP is the Orienteering Problem with Stochastic Profits (OPSP) [34].

These variants of the TSP and OP, while related, fundamentally differ from our model in that the traveling budget and the prizes in these models are distinct, with different "currencies". Thus, using up travel budget does not, and cannot affect the prize collected at a node. In our work, in contrast, traveling and buying use the same resource (e.g. battery power). This fundamental difference is perhaps best exemplified when considering the situation on a path. On the path, solving the TSP, PC-TSP, OP and OPSP is trivial, while, as we show, two of the problems we consider remain NP (and none seems totally trivial). As such, also the methods developed for solving PC-TSP and OP variants, which focus on more general metric spaces, are less relevant to our problem, which focuses on the path metric.

In the Graph Searching Problem (GSP), an agent seeks a single item that resides at some node of a graph, and a distribution is defined over the probabilities of finding the item at each node of the graph. The goal is to minimize the expected cost, as in our *Min-Expected-Cost* problem. The GSP was shown to be strictly related to a classic well-studied problem, the Minimum Latency Problem (MLP) [47], also called the traveling repairman problem [1], the school-bus driver problem [59], and the delivery man problem [23,43]. In this problem an agent is supposed to visit the nodes of a graph in a way that minimizes the sum of the latencies to the nodes, where the latency of a node is the distance traveled by the agent before visiting the node. The minimum latency problem was shown to be NP-complete even when the metric space is induced by a tree [51], but can be solved in linear time when the underlying graph is a path [1,27]. In the operations research community, there are several exact exponential time algorithms for the MLP, e.g. [61,23,50,14,41]. Researchers have also evaluated various heuristic approaches [58,56]. In the computer science community, there is a large branch of research dealing with approximate solutions to the MLP. For general metrics, Blum et al. [15] gave the first constant factor approximation. This was improved by Goemans and Kleinberg [30], and later by Chaudhuri et al. [18]. Koutsoupias et al. [39] provided a constant factor approximation for the unweighted case (i.e. for a shortest path metric on an unweighted graph), and Arora and Karakostas [4] gave a quasi-polynomial $O(n^{O(\log n)})$ time approximation scheme for weighted trees and points in $\mathbb{R}^d$. The MLP was also generalized to multi-agent settings (with k repairmen) by Fakcharoenphol et al. [20,21]. Koutsoupias et al. [39] and later Ausiello et al. [7] showed how to extend results obtained for the MLP to the GSP. For example, in some cases, approximation developed for the MLP can be applied to the GSP. Extensions of the GSP to scenarios where the item is mobile are of the same character [26,38]. The GSP models fundamentally differ from our model in that in GSP there is a single item located somewhere in the graph, which needs to be found, and thus a single binary probability at each node. In our model, the item may be simultaneously available at many nodes, but at different prices, and the for each node there is a distribution over the multiple prices (as in economic search theory). Additionally, in our model traveling and purchasing consume the same resource, an element lacking from the GSP model.

This paper thus tries to bridge the gap between classical economic search theory (which is mainly suitable for virtual or non-dimensional worlds) and the changing search cost constraint imposed by operating in physical multi-agent environments.

From a broad perspective, our search problems lay within the field of planning under uncertainty. Some of the important and relevant models in this field are Markov Decision Process (MDP) [12,45], Decentralized Markov Decision Process (DEC-

MDP) [13] and Stochastic Games [49]. In all of these models (Stochastic Games being the most general, MDP is the most restricted) the goal is to maximize expected cumulative reward over a finite or infinite number of steps. This is the objective with the *Min-Expected-Cost* problem, and we indeed use an MDP formulation to solve the *Min-Expected-Cost* problem on the path (see Section 2.1.2). The other two problems we consider are not concerned with expected rewards, and hence the MDP formulations, and variants, are inapplicable.

There are also other fields in AI (and in general) that consider problems which are closely related to our problems. One such field is the work in path planning when there is uncertain knowledge about the environment. In these settings the agent reveals the traversability of the edges only upon reaching them. The Focused Dynamic A* (D*) algorithm [53] is the most popular heuristic search method that repeatedly determines a shortest path from the agent's current position to the goal, and is able to replan quickly as the knowledge of the terrain changes. Consequent alternatives to D* are D* lite [37] and DSA* [54], which are also able to handle the case where the goal changes over time in addition to the traversability of edges. Unlike our model, in this line of work there are neither uncertainties regarding values nor costs at the possible goal locations that need to be considered. Therefore, D* lite and DSA* algorithms cannot be used to solve our problem.

Another related line of work is scheduling, and in particular scheduling under uncertainty, which considers similar types of objectives [10,11,24]. Much research has been carried out in this field (see [33] for a survey), considering the problem of optimal allocation of resources to activities over time, where some of the parameters are uncertain. Our problems can also be seen as a scheduling problem, as we assign the agents to visit different stores over time. However, in the scheduling domain the chosen plan does not affect the way that the uncertain parameters (i.e., the processing time or the job length) are determined. In our case, the cost of visiting each store (i.e., the processing time) and the probability of buying there (i.e., the probability distribution over the job length) depends on the selected plan. Moreover, in many scheduling under uncertainty problem formulations the underlying deterministic problem is itself NP-hard, and research focus is on developing heuristics to cope with the probabilistic version of the problem. In our case there is no underlying deterministic problem – if all the prices are known, *Min-Budget* and *Max-Probability* are not defined and *Min-Expected-Cost* has a trivial solution. In addition, we concentrate in analyzing the cases where there is an exact polynomial solution or at least a proven approximation (with guarantees on the distance from the optimal solution).

1.3. Terminology and definitions

We are provided with a set of m points – $S = \{u_1, \ldots, u_m\}$, which represent the locations where the item may be available, which we call *stores*, together with a distance function $dis : S \times S \to R^+$, which determines the travel costs between any two locations.[3] We are also provided with the agents' initial locations, which are assumed WLOG (without loss of generality) to be at one of the stores (the product's price at this store may be ∞). With a single agent, there is one initial location, u_s; with k agents we are provided with a vector of initial locations $(u_s^{(1)}, \ldots, u_s^{(k)})$. In addition, we are provided with a cost probability function $p^i(c)$ – stating the probability that the cost of obtaining the item at store i is c. Let D be the set of distinct prices with non-zero probability, and $d = |D|$. We assume that the actual price at a store is only revealed once an agent reaches the store. Given these inputs, the goal is roughly to obtain the product at the minimal total cost, including both travel costs and purchase price. Since we are dealing with probabilities, this rough goal can be interpreted in three different concrete formulations:

1. *Min-Expected-Cost*: minimize the expected cost of obtaining the product.
2. *Min-Budget*: given a success probability p_{succ} minimize the budget necessary to guarantee obtaining the product with probability at least p_{succ}.
3. *Max-Probability*: given a total budget B, maximize the probability of obtaining the product.

In all the above problems, the optimization problem entails determining the strategy (order) in which to visit the different stores, and when to terminate the search. In *Min-Budget* and *Max-Probability*, the search is always terminated when the product is available for a price no greater than the remaining budget, and the agent is located at the store where the product is purchased. In *Min-Expected-Cost* the budget is not allocated in advance and therefore the agent may decide to buy the product at one store even when currently located at a different store. In this case, the agent will have to return to the specific store, thus paying the "return" costs. This model (which includes returning costs) is the basic *Min-Expected-Cost* model we consider. In addition, following standard assumption in economic search literature, we also consider a model in which there are no returning costs. In a physical environment this can be justified if the product can be purchased by phone (following the first physical visit at the store). We analyze both variants of the problem, and refer to them as *Min-Expected-Cost* and *Min-Expected-Cost*$^{\text{phone}}$ problems, respectively.

Technically, it is sometimes easier to work with the failure probability instead of the success probability. In order to compute the failure probability there is a need to only multiply the failure probabilities in each node that has been visited. For the success probability, one needs to use addition and multiplication. Therefore, instead of maximizing p_{succ} we may phrase our objective as minimizing the failure probability.

[3] We therefore have a metric space.

2. Single agent

We start by analyzing the single agent case, for general distance functions (e.g. the stores are located in a general metric space). Unfortunately, in these settings, all three of the above mentioned problems are NP-hard; *Min-Budget* and *Max-Probability* remain hard even if the metric space is a tree. Thus, we focus on the case that the stores are all located on a single path. We would like to emphasize that even in a general metric spaces the stores are along some path that can be traced through them. However, the agent can freely move from every store to every other store directly (with an associated travel cost). When we say that the stores are located on a path we mean that the agent's movement is more restricted – if the agent is located in a store it can move directly only to the two adjacent neighbors of its current location. We denote these problems *Min-Budget* (path), *Max-Probability* (path), and *Min-Expected-Cost* (path), respectively. In this case we can assume that, WLOG all points are on the line, and do away with the distance function *dis*. Rather, the distance between u_i and u_j is simply $|u_i - u_j|$. Furthermore, WLOG we may assume that the stores are ordered from left-to-right, i.e. $u_1 < u_2 < \cdots < u_m$. For exposition purposes, the analysis throughout the paper includes only sketch of proofs for hardness. The detailed proofs are given in Appendices A and B.

2.1. Minimize-Expected-Cost

We prove that the *Min-Expected-Cost* variant is hard for general metric spaces. To prove this we first convert the problem into its decision version. In *Min-Expected-Cost-Decide* we are given a set of points S, a distance function $dis : S \times S \to R^+$, an agent's initial location u_s, a price-probability function $p^{\cdot}(\cdot)$, and a maximum expected cost M. The problem is to decide whether there is a policy to obtain the product with an expected cost of at most M.

2.1.1. Hardness in general metric spaces

Theorem 1. *For general metric spaces* Min-Expected-Cost-Decide *is NP-hard.*

The reduction used is from Hamiltonian path, and we build the instance in such a way that the agent needs to visit all the stores, but only once, to obtain the product with the target expected cost.

We note that in the proof the number of possible prices, d, does not depend on the input. Thus, for general metric spaces *Min-Expected-Cost-Decide* is hard even if d is bounded. Furthermore, even if we assume that an agent can purchase the product by phone after leaving the store, *Min-Expected-Cost-Decide* is still hard (the proof is essentially identical, except that we remove the returning cost of $2n$ from M).

Before continuing with the analysis, we point out that if the cost of arriving to a location is fixed (e.g., in a virtual environment, such as e-commerce, where the cost of visiting each store does not depend on the last visited store), *Min-Expected-Cost* can be solved in polynomial time. The optimal search strategy in this case, as proved by Weitzman [57], is based on setting a reservation value (i.e., a threshold) to each store according to the distribution characterizing its value and the cost of revealing that value. The searcher should continue the search as long as the best value obtained so far is above the lowest reservation value among those associated with stores that have not been explored yet. Formally, the reservation value R_i of a store associated with a fixed cost Fc_i and a distribution $p^i(c)$ can be extracted from the following equation:

$$Fc_i = \sum_{c<R_i} (R_i - c)p^i(c) \tag{1}$$

Intuitively, R_i is the value where the searcher is precisely indifferent: the expected marginal benefit from revealing the actual value of a store (right-hand side) exactly equals the cost of doing so (left-hand side). The use of variable exploration costs due to physical constraints that need to be factored in thus dramatically increases the complexity of determining the agents' optimal strategies, precluding simple solutions such as the above.

2.1.2. Solution for the path

When all stores are located on a path, *Min-Expected-Cost* and *Min-Expected-Cost*$^{\text{phone}}$ problems can be modeled as a finite-horizon Markov Decision Process (MDP), as follows. For exposition purposes, we start with *Min-Expected-Cost*$^{\text{phone}}$. Note that on the path, at any point in time the points/stores visited by the agent constitute a contiguous interval, which we call the *visited interval*. Clearly, the algorithm need only make decisions at store locations. Furthermore, decisions can be limited to times when the agent is at one of the two stores at the edges of the *visited interval*. At each such location, the agent has only three possible actions: "go right" – extending the visited-interval one store to the right, "go left" – extending the visited-interval one store to the left, or "stop" – stopping the search and buying the product by phone at the best price so far. Also note that *after* the agent has already visited the interval $[u_\ell, u_r]$, how exactly it covered this interval does not matter for any future decision; the costs have already been incurred. Accordingly, the states of the MDP are quadruplets $[\ell, r, e, c]$, such that $\ell \leqslant s \leqslant r$, $e \in \{\ell, r\}$, and $c \in D$, representing the situation that the agents visited stores u_ℓ through u_r, it is currently at location u_e, and the best price encountered so far is c. The terminal states are $Buy(c)$ and all states of the

form $[1, m, e, c]$, and the terminal cost is c. For all other states there are two or three possible actions – "go right" (provided that $r < m$), "go left" (provided that $1 < \ell$), or "stop". The cost of "go right" on the state $[\ell, r, e, c]$ is $(u_{r+1} - u_e)$, while the cost of "go left" is $(u_e - u_{\ell-1})$. The cost of "stop" is always 0. Given the state $[\ell, r, e, c]$ and move "go right", there is probability $p^{r+1}(c')$ to transition to state $[\ell, r+1, r+1, c']$, for $c' < c$. With the remaining probability, the transition is to state $[\ell, r+1, r+1, c]$. Transition to all other states has zero probability. Transitions for the "go left" action are analogous. Given the state $[\ell, r, e, c]$ and the action "stop", there is probability 1 to transition to state $Buy(c)$. This fully defines the MDP. The optimal strategy for finite-horizon MDPs can be determined using dynamic programming (see [45, Ch. 4]). In our case, the number of entries in the dynamic programming table is at most $m \cdot m \cdot 2 \cdot d$ (since this is an upper bound on the number of possible states) and it takes at most $O(d)$ steps to compute each entry. Therefore, the complexity of solving *Min-Expected-Cost*$^{\text{phone}}$ is $O(d^2m^2)$ steps (using $O(dm^2)$ space).

For *Min-Expected-Cost* (no phone) we will need a larger MDP, but the basic structure is similar. First note that if an interval $[u_\ell, u_r]$ has been visited, and the item not yet purchased, then any future purchase within the interval (if there should be such a purchase) will be with the agent coming from outside the interval into the interval, and moving directly to a store for purchasing. In addition, there is a unique store u_{x_r}, such that any searcher coming from anywhere to the right of u_r and purchasing within $[u_\ell, u_r]$ purchases at u_{x_r}, and similarly a unique store u_{x_ℓ}, for purchases coming from the left of u_ℓ. The reason is that the cost of purchasing at a store is the sum of price and distance. So, any fixed additional distance does not change the minimum. Thus, if the price at u_{x_r} is lowest for someone coming from u_r, it is also the minimum for anyone coming from $u_{r'}$ to the right of u_r. Therefore, the states of the MDP for *Min-Expected-Cost* are septuplets $[\ell, r, e, c_\ell, x_\ell, c_r, x_r]$, representing the situation that the agent visited stores u_ℓ through u_r, it is currently at location u_e, $e \in \{\ell, r\}$, and the best price encountered so far for someone coming from the left (respectively right) is $c_\ell(c_r)$, which can be found at store $u_{x_\ell}(u_{x_r})$. The terminal states are $Buy(c, e, x)$ with a terminal cost of $c + |u_e - u_x|$, and all states of the form $[1, m, e, c_\ell, x_\ell, c_r, x_r]$ with a terminal cost of $c_\ell + |u_{x_\ell} - u_1|$ if $e = 1$ and $c_r + |u_m - u_{x_r}|$ if $e = m$. The actions are the same as in the MDP for *Min-Expected-Cost*$^{\text{phone}}$, but the transition probabilities are different. Given the state $[\ell, r, e, c_\ell, x_\ell, c_r, x_r]$ and move "go right", there is probability $p^{r+1}(c')$ to transition to state $[\ell, r+1, r+1, c_\ell, x_\ell, c', r+1]$, for $c' < (c_r + |u_{r+1} - u_{x_r}|)$. With the remaining probability, the transition is to state $[\ell, r+1, r+1, c_\ell, x_\ell, c_r, x_r]$. Transition to all other states has zero probability. Transitions for the "go left" action are analogous. Given the state $[\ell, r, e, c_\ell, x_\ell, c_r, x_r]$ and the action "stop", there is probability 1 to transition to state $Buy(c_\ell, e, x_\ell)$ if $e = \ell$ and $Buy(c_r, e, x_r)$ if $e = r$. This fully defines the MDP. Using the same analysis as before, we get that the complexity of solving *Min-Expected-Cost* is $O(d^3m^4)$ steps (using $O(d^2m^4)$ space).

2.2. Min-Budget and Max-Probability

2.2.1. NP completeness

Unlike the *Min-Expected-Cost* problem, the other two problems are NP-complete even on a path. To prove this we again convert the problems into their decision versions. In the *Min-Budget-Decide* problem, we are given a set of points S, a distance function $dis : S \times S \to R^+$, an agent's initial location u_s, a price-probability function $p^{\cdot}(\cdot)$, a minimum success probability p_{succ} and maximum budget B. We have to decide whether a success probability of at least p_{succ} can be obtained with a budget of at most B. The exact same formulation also constitutes the decision version of the *Max-Probability* problem.

Theorem 2. *The* Min-Budget-Decide *problem is NP-complete even on a path.*

The reduction is from 0–1 Knapsack. We build the instance such that the agent needs to go back and forth along the line in a zigzag movement. Before any direction switch the agent needs to decide weather to keep moving to a close store, to "collect" some more probabilities, or to keep its budget and head to the other direction. Each such decision corresponds to the decision of whether to spend some space and insert an item to the knapsack, or save it to the next item.

Thus, we either need to consider restricted instances or consider approximations. We do both.

2.2.2. Restricted case: Bounded number of prices

We consider the restricted case when the number of possible prices, d, is bounded. For brevity, we focus on the *Min-Budget* (path) problem. The same algorithm and similar analysis work also for the *Max-Probability* (path) problem. Consider first the case where there is only one possible price c_0. At any store i, either the product is available at this price, with probability $p_i = p^i(c_0)$, or not available at any price. In this setting we show that the problem can be solved in $O(m)$ steps. This is based on the following lemma, stating that in this case, at most one direction change is necessary.

Lemma 3. *Consider a price c_0 and suppose that in the optimal strategy starting at point u_s the area covered while the remaining budget is at least c_0 is the interval $[u_\ell, u_r]$. Then, WLOG we may assume that the optimal strategy is either $(u_s \rightarrowtail u_r \rightarrowtail u_\ell)$ or $(u_s \rightarrowtail u_\ell \rightarrowtail u_r)$.*

Proof. Any other route would yield a higher cost to cover the same interval. □

Using this observation, we immediately obtain an $O(m^3)$ algorithm for the single price case: for each interval $[u_\ell, u_r]$, consider both possible options and for each compute the total cost and the resulting probability. Choose the option which

Algorithm 1 OptimalPolicyForSinglePrice(Success probability p_{succ}, single price c_0).

$u_r \leftarrow$ leftmost point on right of u_s s.t. $1 - \prod_{i=s}^{r}(1-p_i) \geqslant p_{succ}$
$\ell \leftarrow s$
$B^{RL}_{\min} \leftarrow |u_r - u_s|$
while $\ell \geqslant 0$ and $r \geqslant s$ **do**
  $B \leftarrow 2|u_r - u_s| + |u_s - u_\ell|$
  if $B < B^{RL}_{\min}$ **then**
    $B^{RL}_{\min} \leftarrow B$
  $r \leftarrow r - 1$
  while $\ell \geqslant 0$ and $1 - \prod_{i=\ell}^{r} 1 - p_i < p_{succ}$ **do**
    $\ell \leftarrow \ell - 1$
$u_\ell \leftarrow$ rightmost point to left of u_s s.t. $1 - \prod_{i=\ell}^{s} 1 - p_i \geqslant p_{succ}$
$r \leftarrow s$
$B^{LR}_{\min} \leftarrow |u_s - u_l|$
while $r \leqslant m$ and $\ell \leqslant s$ **do**
  $B \leftarrow 2|u_s - u_\ell| + |u_r - u_s|$
  if $B < B^{LR}_{\min}$ **then**
    $B^{LR}_{\min} \leftarrow B$
  $\ell \leftarrow \ell + 1$
  while $r \leqslant m$ and $1 - \prod_{i=\ell}^{r} 1 - p_i < p_{succ}$ **do**
    $r \leftarrow r + 1$
return $\min\{B^{RL}_{\min}, B^{LR}_{\min}\} + c_0$

requires the lowest budget but still has a success probability of at least p_{succ}. With a little more care, the complexity can be reduced to $O(m)$. First note that since there is only a single price c_0, we can add c_0 to the budget at the end, and assume that the product will be provided at stores for free, provided that it is available. Now, consider the strategy of first moving right and then switching to the left. In this case, we need only consider the *minimal* intervals that provide the desired success probability, and for each compute the necessary budget. This can be performed incrementally, in a total of $O(m)$ operations for all such minimal intervals, since at most one point can be added and one deleted at any given time. Similarly for the strategy of first moving left and then switching to the right. The details are provided in Algorithm 1.

Next, consider the case that there may be several different available prices, but their number, d, is fixed. We provide a polynomial algorithm for this case (though exponential in d). First note that in the *Min-Budget* problem, we seek to minimize the initial budget B necessary so as to guarantee a success probability of at least p_{succ} given this initial budget. Once the budget has been allocated, however, there is no requirement to minimize the actual expenditure. Thus, at any store, if the product is available for a price no greater than the remaining budget, it is purchased immediately and the search is over. If the product has a price beyond the current available budget, the product will not be purchased at this store under any circumstances. Denote $D = \{c_1, c_2, \ldots, c_d\}$, with $c_1 > c_2 > \cdots > c_d$. For each c_i there is an interval $I_i = [u_\ell, u_r]$ of points covered while the remaining budget was at least c_i. By definition, for all i, $I_i \subseteq I_{i+1}$. Thus, consider the *incremental* area covered with remaining budget c_i, $\Delta_i = I_i - I_{i-1}$ (with $\Delta_1 = I_1$). Each Δ_i is a union of an interval at left of u_s and an interval at the right of u_s (both possibly empty). The next lemma, which is the multi-price analogue of Lemma 3, states that there are only two possible optimal strategies to cover each Δ_i:

Lemma 4. *Consider the optimal strategy and the incremental areas Δ_i ($i = 1, \ldots, d$) defined by this strategy. For $c_i \in D$, let u_{ℓ_i} be the leftmost point in Δ_i and u_{r_i} the rightmost point. Suppose that in the optimal strategy the covering of Δ_i starts at point u_{s_i}. Then, WLOG we may assume that the optimal strategy is either $(u_{s_i} \rightarrowtail u_{r_i} \rightarrowtail u_{\ell_i})$ or $(u_{s_i} \rightarrowtail u_{\ell_i} \rightarrowtail u_{r_i})$. Furthermore, the starting point for covering Δ_{i+1} is the ending point of covering Δ_i.*

Proof. The areas Δ_i fully determine the success probability of the strategy. Any strategy other than the ones specified in the lemma would require more travel budget, without enlarging any Δ_i. □

Thus, the optimal strategy is fully determined by the leftmost and rightmost points of each Δ_i, together with the choice for the ending points of covering each area. We can thus consider all possible cases and choose the one with the lowest budget which provides the necessary success probability. There are $\frac{m^{2d}}{(2d)!} \leqslant (\frac{em}{2d})^{2d}$ ways for choosing the external points of the Δ_i's, and there are a total of 2^d options to consider for the covering of each. For each option, computing the budget and probability takes $O(m)$ steps. Thus, the total time is $O(m2^d(\frac{em}{2d})^{2d})$. Similar algorithms can also be applied for the *Max-Probability* (path) problem. In all, we obtain:

Theorem 5. Min-Budget (*path*) *and* Max-Probability (*path*) *can be solved in $O(m)$ steps for a single price and $O(m2^d(\frac{em}{2d})^{2d})$ for d prices.*

In summary, the effect of bounding the number of prices was indeed surprising. Indeed, as we showed in Lemma 4, the number of prices determines incremental intervals for covering by the agent. Each price induces only one interval and

there are only two optimal ways to cover each interval. Therefore, if the number of prices is fixed, even if we increase the number of stores we can still enumerate and check all the possible ways to cover the intervals optimally. Unfortunately, moving beyond the path, *Min-Budget-Decide* turns hard, even with a bounded number of possible prices, and even on a tree.

Theorem 6. *The* Min-Budget-Decide *problem is NP-complete on a tree, even with a bounded number of prices.*

The reduction is from 0–1 Knapsack. In the proof we build a star tree, where the agent is located at the root and all the other stores around him. We use only two possible prices, 0 and inf, so the only difference between stores is their distance from the root (which corresponds to the knapsack items size) and the probability of purchasing the product (which corresponds to the knapsack item value).

2.2.3. Min-Budget approximation

Next, we provide an FPTAS (fully-polynomial-time-approximation-scheme) for the *Min-Budget* (path) problem. The idea is to force the agent to move in quantum steps of some fixed size δ. In this case the tour taken by the agent can be divided into *segments*, each of size δ. Furthermore, the agent's decision points are restricted to the ends of these segments, except for the case where along the way the agent has sufficient budget to purchase the product at a store, in which case it does so and stops. We call such a movement of the agent a *δ-resolution tour*. Note that the larger δ the less decision points there are, and the complexity of the problem decreases. Given $0 < \epsilon < 1$, we show that with a proper choice of δ we can guarantee a $(1+\epsilon)$ approximation to the optimum, while maintaining a complexity of $O(n\,\mathrm{poly}(1/\epsilon))$, where n is the size of the input.

Our algorithm is based on computing for (essentially) each initial possible budget B, the maximal achievable success probability, and then pick the minimum budget with probability at least p_{succ}. Note that once the interval $[\ell, r]$ has been covered without purchasing the product, the only information that matters for any future decision is (i) the remaining budget, and (ii) the current location. The exact (fruitless) way in which this interval was covered is, at this point, immaterial. This, "memoryless" nature calls, again, for a dynamic programming approach. We now provide a dynamic programming algorithm to compute the optimal δ-resolution tour. WLOG assume that $u_s = 0$ (the initial location is at the origin). For integral i, let $w_i = i\delta$. The points w_i, which we call the *resolution points*, are the only decision points for the algorithm. Set L and R to be such that w_L is the rightmost w_i to the left of all the stores and w_R the leftmost w_i to the right of all stores. We define two tables, $\mathsf{fail}[\cdot,\cdot,\cdot,\cdot]$ and $\mathsf{act}[\cdot,\cdot,\cdot,\cdot]$, such that for all ℓ, r, $L \leqslant \ell \leqslant 0 \leqslant r \leqslant R$, $e \in \{\ell, r\}$ (one of the end points), and budget B, $\mathsf{fail}[\ell, r, e, B]$ is the minimal failure probability achievable for purchasing at the stores *outside* $[w_\ell, w_r]$, assuming a remaining budget of B, and starting at location w_e. Similarly, $\mathsf{act}[\ell, r, e, B]$ is the best act to perform in this situation ("left", "right", or "stop"). Given an initial budget B, the best achievable success probability is $(1 - \mathsf{fail}[0, 0, 0, B])$ and the first move is $\mathsf{act}[0, 0, 0, B]$. It remains to show how to compute the tables. The computation of the tables is performed from the outside in, by induction on the number of remaining points. For $\ell = L$ and $r = R$, there are no more stores to search and $\mathsf{fail}[L, R, e, B] = 1$ for any e and B. Assume that the values are known for i remaining points, we show how to compute for $i+1$ remaining points. Consider $\mathsf{cost}[\ell, r, e, B]$ with $i+1$ remaining points. Then, the least failure probability obtainable by a decision to move right (to w_{r+1}) is:

$$F_R = \left(1 - \sum_{c \leqslant B-\delta} p^{r+1}(c)\right) \mathsf{fail}[\ell, r+1, r+1, B-\delta]$$

Similarly, the least failure probability obtainable by a decision to move left (to $w_{\ell-1}$) is:

$$F_L = \left(1 - \sum_{c \leqslant B-\delta} p^{\ell-1}(c)\right) \mathsf{fail}[\ell-1, r, \ell-1, B-\delta]$$

Thus, we can choose the act providing the least failure probability, determining both $\mathsf{act}[\ell, r, e, B]$ and $\mathsf{fail}[\ell, r, e, B]$. In practice, we compute the table only for B's in integral multiples of δ. This can add at most δ to the optimum. Also, we may place a bound $B^\delta_{\max}$ on the maximal B we consider in the table. In this case, we start filling the table with $w_L = -B^\delta_{\max}/\delta$ and $w_R = B^\delta_{\max}/\delta$, the furthest point reachable with budget $B^\delta_{\max}$.

Next, we show how to choose δ and prove the approximation ratio. Set $\lambda = \epsilon/9$. Let $\alpha = \min\{|u_s - u_{s+1}|, |u_s - u_{s-1}|\}$ – the minimum budget necessary to move away from the starting point, and $\beta = m^2|u_m - u_1| + \max\{c\colon\ \exists i, p^i(c) > 0\}$ – an upper bound on the total usable budget. We start by setting $\delta = \lambda^2\alpha$ and double it until $\delta > \lambda^2\beta$, performing the computation for all such values of δ. For each such value of δ, we fill the tables (from scratch) for all values of B's in integral multiples of δ up to $B^\delta_{\max} = 2\lambda^{-2}\delta$. We now prove that for at least one of the choices of δ we obtain a $(1+\epsilon)$ approximation.

Consider a success probability p_{succ} and suppose that optimally this success probability can be obtained with budget B_{opt} using the tour T_{opt}. By *tour* we mean a list of actions ("right", "left" or "stop") at each decision point (which, in this case, are all store locations). We convert T_{opt} to a δ-resolution tour, T^δ_{opt}, as follows. For any $i \geqslant 0$, when T_{opt} moves for the first time to the right of w_i then T^δ_{opt} moves all the way to w_{i+1}. Similarly, for $i \leqslant 0$, when T_{opt} moves for the first time to the left of w_i then T^δ_{opt} moves all the way to w_{i-1}.

Note that T^{δ}_{opt} requires additional travel costs only when it "overshoots", i.e. when it goes all the way to the resolution point while T_{opt} would not. This can either happen (i) in the last step, or (ii) when T_{opt} makes a direction change. Type (i) can happen only once and costs at most δ. Type (ii) can happen at most once for each resolution point, and costs at most 2δ. Suppose that T^{δ}_{opt} makes t *turns* (i.e. t direction changes). Then, the total additional travel cost of the tour T^{δ}_{opt} over T_{opt} is at most $(2t+1)\delta$. Furthermore, if we use T_{opt} with budget B_{opt} and T^{δ}_{opt} with budget $B_{opt}+(2t+1)\delta$ then at any store, the available budget under T^{δ}_{opt} is at least that available with T_{opt}. Thus, T^{δ}_{opt} is a δ-resolution tour that with budget at most $B_{opt}+(2t+1)\delta$ succeeds with probability $\geqslant p_{succ}$. Hence, our dynamic algorithm, which finds the optimal such δ-resolution tour will find a tour with budget $B^{\delta}_{opt} \leqslant B_{opt}+(2t+2)\delta$ obtaining at least the same success probability. Note that we include one additional δ, for the integral multiples of δ in the tables.

Since T^{δ}_{opt} has t-turns, T_{opt} must also have t-turns, with targets at t *distinct* resolution segments. For any i, the i-th such turn (of T_{opt}) necessarily means that T_{opt} moves to a point at least $(i-1)$ segments away, i.e. a distance of at least $(i-1)\delta$. Thus, for B_{opt}, which is at least the travel cost of T_{opt}, we have[4]:

$$B_{opt} \geqslant \sum_{i=1}^{t}(i-1)\delta = \frac{(t-1)(t)}{2}\delta \geqslant \frac{t^2}{4}\delta \tag{2}$$

On the other hand, since we consider all options for δ in multiples of 2, there must be a $\hat{\delta}$ such that:

$$\lambda^{-2}\hat{\delta} \geqslant B_{opt} \geqslant \frac{\lambda^{-2}}{2}\hat{\delta} \tag{3}$$

Combining (2) and (3) we get that $t \leqslant 2\lambda^{-1}$. Thus, the approximation ratio is:

$$\frac{B^{\hat{\delta}}_{opt}}{B_{opt}} \leqslant \frac{B_{opt}+2(t+1)\hat{\delta}}{B_{opt}} \leqslant 1+\frac{2(t+1)\hat{\delta}}{\lambda^{-2}\hat{\delta}/2} \tag{4}$$

$$\leqslant 1+\left(8\lambda+4\lambda^2\right) \leqslant 1+\epsilon \tag{5}$$

Also, combining (3) and (5) we get that

$$B^{\hat{\delta}}_{opt} \leqslant B_{opt}(1+\epsilon) \leqslant 2\lambda^{-2}\hat{\delta} = B^{\hat{\delta}}_{\max}$$

Hence, the tables with resolution $\hat{\delta}$ consider this budget, and $B^{\hat{\delta}}_{opt}$ will be found.

It remains to analyze the complexity of the algorithm. For any given δ there are $B^{\delta}_{\max}/\delta = 2\lambda^{-2}$ budgets we consider and at most this number of resolution points at each side of u_s, for each, there are two entries in the table. Thus, the size of the table is $\leqslant 8\lambda^{-6} = O(\epsilon^{-6})$. The computation of each entry takes $O(1)$ steps (see the discussion above). We consider δ in powers of 2 up to $\beta \leqslant 2^n$, where n is the size of the input. Thus, the total computation time is $O(n\epsilon^{-6})$. We obtain:

Theorem 7. *For any $\epsilon > 0$, the* Min-Budget (*path*) *problem can be approximated with a $(1+\epsilon)$ factor in $O(n\epsilon^{-6})$ steps.*

3. Multi-agent, shared budget

Since even the single agent case is hard for general metric spaces, with the multi-agent case we focus solely on situations in which all the stores are on a single path. We assume k agents operating in the same underlying physical setting as in the single agent case, i.e. a set of *stores* S and a price probability function for each store. We assume that the goal is not individualized; the agents seek to obtain only one item and having multiple goods is not beneficial. Furthermore, since the agents are fully collaborative, they do not care which agent will obtain the item.

We begin by analyzing the *shared budget* multi-agent model, where all the resources and costs are shared among all the agents. In theory, agents may move in parallel, but since minimizing time is not an objective, we may assume WLOG that at any given time only one agent moves. When an agent reaches a store and finds the price at this location, the optimal strategy should tell whether to purchase the product (and where) and if not what agent should move next and to where. Therefore, in the *k-Shared-Min-Expected-Cost* problem the agents try to minimize the expected total cost, which includes the travel costs of all agents plus the final purchase price (which is one of the prices that the agents have sampled). In *k-Shared-Min-Budget* and *k-Shared-Max-Probability*, the initial budget is for the use of all the agents, and the success probability is for any of the agents to purchase, at any location. Since all the agents use the same budget in this model, inter alia, for traveling costs, we assume the agents can communicate with each other to coordinate their moves. In *k-Shared-Min-Budget* and *k-Shared-Max-Probability* the agents only need to announce to the other agents when they reach a specific store. In *k-Shared-Min-Expected-Cost* the agents also need to communicate the price they find at the location they have reached.

[4] Assuming that $t > 1$. If $t = 0, 1$ the additional cost is small by (3).

In general, the algorithms for the single-agent case (for the path) can be extended to the multi-agent case, with the additional complexity of coordinating between the agents. The proofs are relegated to Appendix B, as they are very similar to the single agent case. We obtain:

Theorem 8. *With k agents,* k-Shared-Min-Expected-Cost^{phone} *can be solved in* $O(d^2 2^k (\frac{m}{k})^{2k})$, *and* k-Shared-Min-Expected-Cost *can be solved in* $O(d^{2k+1} 2^k (\frac{m}{k})^{4k})$.

Theorem 9. *With k agents,* k-Shared-Min-Budget *and* k-Shared-Max-Probability *with d possible prices can be solved in* $O(m 2^{kd} (\frac{em}{2kd})^{2kd})$.

Theorem 10. *With k agents, for any* $\epsilon > 0$, k-Shared-Min-Budget *can be approximated to within a factor of* $(1 + k\epsilon)$ *in* $O(n\epsilon^{-6k})$ *steps* (*for an arbitrary number of prices*).

While the complexity in the multi-agent case grows exponentially in the number of agents, in most physical environments where several agents cooperate in exploration and search, the number of agents is relatively moderate. In these cases the computation of the agents' strategies is efficiently facilitated by the principles of the algorithmic approach presented in this paper.

If the number of agents is not fixed (i.e. part of the input) then, the complexity of all three variants grows exponentially. Most striking perhaps is that *k-Shared-Min-Budget* and *k-Shared-Max-Probability* are NP-complete even on the path with a *single price*. This is in contrast to the single agent case where the single price case can be solved in $O(n)$ steps. To prove this we again formulate the problems into a decision version – *k-Shared-Min-Budget-Decide* – given a set of points S on the path, initial locations for all agents $(u_s^{(1)}, \ldots, u_s^{(k)})$, a price-probability function $p^{\cdot}(\cdot)$, a minimum success probability p_{succ} and a maximum budget B, decide if success probability of at least p_{succ} can be achieved with a maximum budget B.

Theorem 11. k-Shared-Min-Budget-Decide *is NP-complete even on the path with a* single price.

The reduction is (again) from 0–1 Knapsack. We build the instance such that the number of agents corresponds to the number of possible knapsack items. Each agent can only visit one store, since we set the distances to the other stores above the (shared) budget. As before, the probabilities at the stores correspond to the knapsack items values, and the distances correspond to the knapsack items sizes. The problem is to decide which agents will move given the initial budget, which corresponds to the decision which items to insert to the knapsack given its size.

4. Multi-agent, private budget

We now investigate a model of *private budgets*, wherein each agent j has its own initial budget B_j (unlike the previous shared budget model). If the objective is to minimize the total expected cost, the private budgets model is equal to the shared budget model since the agents are cooperative. Therefore in this case we have two concrete problem formulations:

1. *k-Private-Max-Probability*: given initial budgets B_j, for each agent j, maximize the probability of obtaining the item.
2. *k-Private-Min-Budget*: given a target success probability p_{succ}, minimize the agents' initial budgets necessary to guarantee acquisition of the item with a probability of at least p_{succ}.

Since the corresponding single-agent problems are hard even for the path, we again assume that the number of possible prices, d, is bounded. In the *k-Private-Min-Budget* problem it is also important to distinguish between two different agent models:

- *Identical budgets*: the initial budgets of all the agents must be the same. The problem is to minimize this initial budget, and we denote the problem as *k-Private-Min-Budget*$^{\text{identical}}$.
- *Distinct*: the agents' initial budgets may be different. In this case the problem is to minimize the average initial budget, and we denote the problem as *k-Private-Min-Budget*$^{\text{distinct}}$.

4.1. Non-communicating agents

We first consider the case where agents cannot communicate with each other. In this case agents cannot assist each other. Hence a solution is a *strategy* comprised of a set of ordered lists, one for each agent, determining the sequence of stores this agent must visit.

The success probability of a strategy is the probability that at least one of the agents will succeed in its task. Technically, in this case, it is easier to calculate the complementary failure probability: the probability that all the agents will not succeed in their tasks. For example, suppose that the stores and agents are located as illustrated in Fig. 2, and consider the depicted strategy. This strategy fails if for both agents and each of the stores they visit the cost of the item is higher than

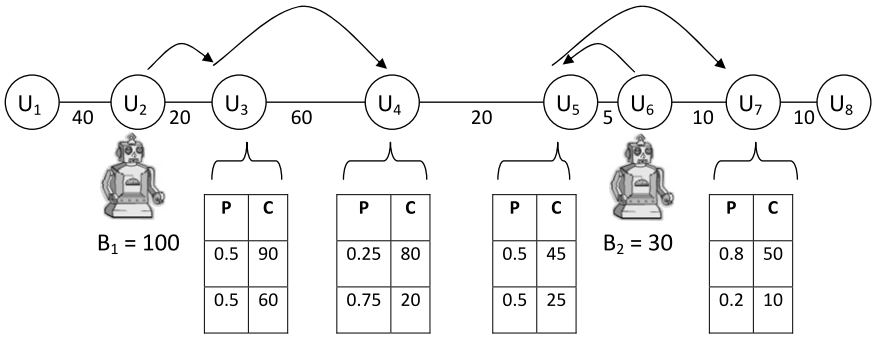

Fig. 2. A possible input with a suggested strategy. The numbers on the edges represent traveling costs. The table at each store u_i represents the cost probability function $p_i(c)$. The strategy of each agent is illustrated by the arrows.

their remaining budget. This will happen with probability $(\frac{1}{2} \cdot \frac{1}{4}) \cdot (\frac{1}{2} \cdot \frac{4}{5}) = \frac{1}{20}$. Hence, the success probability of this strategy is $\frac{19}{20}$.

We begin by considering the *k-Private-Max-Probability* problem. We prove:

Theorem 12. *In the no communication case if the number of possible costs is constant then* k-Private-Max-Probability *can be solved in polynomial time for any number of agents.*

The proof is based on the following definitions and lemmata. The key idea is that in most cases the stores will be visited by only one agent in the optimal strategy. However, there are some cases where the same store will be visited by more than one agent. We identify these cases and show that there are only a fixed number of them. We are thus able to provide a dynamic programming algorithm to find the optimal strategy.

Note that multiple strategies may result in the same success probability. In this case we say that the strategies are *equivalent*. In particular there may be more than one optimal strategy.

Definition 13. Let S be a strategy. Agents i and $\bar{i}$ are said to be *separated* by S if each store that is reached by i is not reached by $\bar{i}$.

Lemma 14. *If agents i and $\bar{i}$ are not separated by any optimal strategy, then in at least one optimal strategy at least one of these agents must pass the initial location of the other.*

Proof. WLOG assume that i is on the right side of $\bar{i}$. Consider an optimal strategy S. Let r be the rightmost store that is reached by i and $\bar{l}$ the leftmost store that is reached by $\bar{i}$. Assume by contradiction that none of the agents passes the initial location of the other in S. Thus, there is at least one store between their initial locations that is reached by both agents. WLOG assume that $\bar{i}$ reaches at least one store with a higher budget than i's remaining budget when reaching it, and denote by $\bar{r}^*$ the rightmost such store. Consider the following modified strategy: i goes according to S till the stage it has to reach $\bar{r}^*$. If i did not reach r yet then instead of reaching $\bar{r}^*$ it goes all the way straight to r. Otherwise, it stops just before reaching $\bar{r}^*$. $\bar{i}$ goes according to S till the stage it has to reach $\bar{r}^*$. If $\bar{i}$ did not reach $\bar{l}$ yet then after reaching $\bar{r}^*$ it goes all the way straight to $\bar{l}$. Otherwise, it stops after reaching $\bar{r}^*$. Agents i and $\bar{i}$ are separated by this strategy and it has at least the same success probability as S, in contradiction. □

Based on this lemma, we now show that if two agents are not separated by any optimal policy, their movement has a specific structure.

Lemma 15. *Suppose that agents i and $\bar{i}$ are not separated by any optimal strategy. Let S be an optimal strategy. Suppose that in S agent i passes the initial location of agent $\bar{i}$ and agent $\bar{i}$ does not stay in its initial location. Then, there is an optimal strategy such that one of the following holds*:

- *$\bar{i}$ moves only in one direction which is opposite to the final movement's direction of i. Furthermore, if the final movement's direction of i is right (left) then $\bar{i}$ passes the leftmost (rightmost) store that is reached by i.*
- *Either i or $\bar{i}$ does not move.*

Proof. WLOG assume that i is on the right side of $\bar{i}$. Let $[l, r]$ be the interval of stores covered by i. Since i passes the initial location of $\bar{i}$, l is located on the left of $u_s^{(\bar{i})}$ and r is located on the right of $u_s^{(\bar{i})}$.

First we show that we may assume that $\bar{i}$ reaches at least one store outside the interval $[l, r]$. If it does not, consider the following two cases. If i's remaining budget at each store is always as high as $\bar{i}$'s remaining budget then $\bar{i}$ does not have to move and the theorem holds. Otherwise, let $\bar{r}^*$ be the rightmost store where $\bar{i}$'s remaining budget is higher than

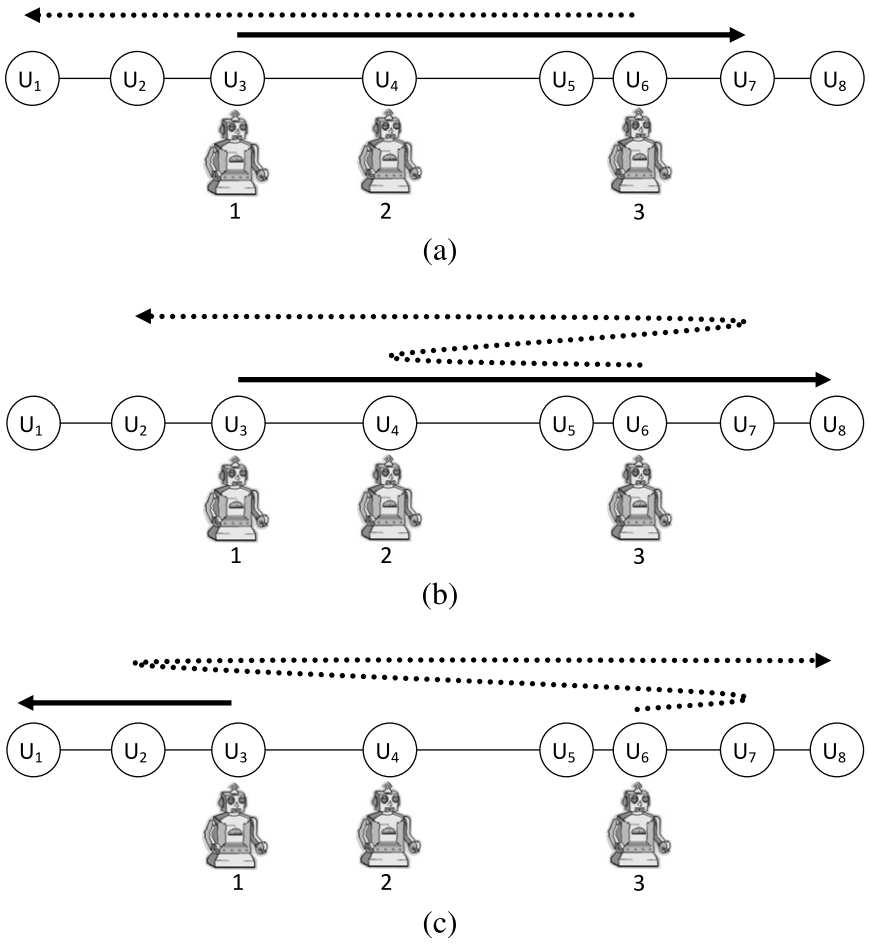

Fig. 3. The only three cases where a pair of agents may not be separated.

i's remaining budget. If $\bar{r}^*$ is on the left side of i's initial location, then as in the proof of Lemma 14, the agents can be separated. If $\bar{r}^*$ is on the right side of i's initial location and it equals r, there is no need for i to reach r since at each store in $[u_s^{(i)}, r]$, $\bar{i}$ has at least the same budget as i. Thus, there is an optimal strategy where either i does not move or it moves only to the left, so $\bar{i}$ passes the rightmost store that is reached by i. If $\bar{r}^*$ is on the right side of i but on the left side of r then there is no need for $\bar{i}$ to go beyond $\bar{r}^*$. Since it has more budget than i at this location, $\bar{i}$ can move to l while i moves to r. Thus, again, there is an optimal strategy where either i does not move or it moves only to the right, so $\bar{i}$ passes the leftmost store that is reached by i. Thus, we may assume that $\bar{i}$ reaches at least one store outside the interval $[l, r]$.

WLOG assume that i's final movement's direction is left and suppose that $\bar{i}$ reaches at least one store outside the interval $[l, r]$ to the left of l. If $\bar{i}$'s budget at l is higher than i's remaining budget there, then it is also higher at $u_s^{(\bar{i})}$, and again the agents can be separated. If $\bar{i}$'s budget at l is not higher than i's remaining budget, then $\bar{i}$ does not have to move since i can reach the same stores to the left of l.

Now suppose that $\bar{i}$ moves to the right (which is the opposite direction of i's final movement) and passes $u_s^{(i)}$, but it also changes its direction. The only reason for $\bar{i}$ to change directions is to reach a store on the left side of its initial location, with a higher budget than i has at this store, or to reach a store that i does not reach at all. In both cases $\bar{i}$ must reach each store in $[l, u_s^{(i)}]$ with at least the same budget as i has at the same location, so either S is not optimal, or we can modify S by letting only $\bar{i}$ to move while i does not move at all. □

Using these lemmata we observe that for any two agents, there is only a constant number of possible cases where the agents are not separated by the optimal strategies. Fig. 3 illustrates the three core cases (the others are symmetrical). Here, agents 1 and 3 are non-separated agents. Note that every agent between them, like agent 2, does not have to move at all in the optimal strategy.

Therefore we can use a dynamic programming approach to find an optimal strategy whereby all the agents are separated, but we also check the non-separated strategies individually.

Recall that in our problem the objective is to maximize the success probability, given the initial budgets. Technically, it is easy to work with the failure probability instead of the success probability.

Definition 16. $\mathsf{fail}[u_i, j]$ is the minimal failure probability if the only reachable stores are in the interval $[u_1, u_i]$, and only agents $1, \ldots, j$ are allowed to move. $\mathsf{act}[u_i, j]$ is the optimal strategy achieving $\mathsf{fail}[u_i, j]$, under the same conditions.[5]

Note that where $u_i < u_s^{(j)}$, $\mathsf{fail}[u_i, j]$ is not defined. Given $\mathsf{act}[u_i, j]$, $\mathsf{fail}[u_i, j]$ can be easily computed in $O(m)$ steps. For technical reasons we add another agent, 0, with a budget of zero and set its initial location to the leftmost store, i.e. $u_s^{(0)} = u_1$. $\mathsf{fail}[u_i, 0] = 1$ for all i, and this agent doesn't affect the failure probability of any policy.

We are now ready to prove Theorem 12, by showing a polynomial time algorithm for *k-Private-Max-Probability*.

Proof of Theorem 12. We use dynamic programming to calculate $\mathsf{fail}[u_m, k]$ and $\mathsf{act}[u_m, k]$. For $\mathsf{fail}[u_i, 1]$ and $\mathsf{act}[u_i, 1]$, which is the single agent case, we employ the polynomial algorithm obtained from Theorem 5.

Given any agent $\bar{j}$ we first consider the case where $u_i = u_s^{(\bar{j})}$. In this case in the optimal strategy $\bar{j}$ moves only to the left, or not at all. Let $u_l^{(\bar{j})}$ be the leftmost store visited by $\bar{j}$ with the optimal strategy for the given interval, and agent l be the one such that $u_s^{(l)} \leqslant u_l^{(\bar{j})}$ (l may equal 0). Each agent t such that $l < t < \bar{j}$ does not move in the optimal strategy. Otherwise, agents t and $\bar{j}$ are not separated and according to Lemma 15 agent t must pass the rightmost store $u_s^{(\bar{j})}$, which is not possible. The same argument shows that each agent t such that $t \leqslant l$ does not reach $u_l^{(\bar{j})}$. Therefore $\mathsf{act}[u_i, \bar{j}]$ is composed of $\mathsf{act}[u_{l-1}^{(\bar{j})}, l]$, which are already known, together with the movement of agent $\bar{j}$ to $u_l^{(\bar{j})}$. Thus, computing $u_l^{(\bar{j})}$ takes $O(m)$ steps.

Next, consider the case where $u_i > u_s^{(\bar{j})}$. In this case, in the optimal strategy $\bar{j}$ may move in both directions, or not move at all. Let $u_l^{(\bar{j})}$ be the leftmost store visited by $\bar{j}$ with the optimal strategy for this interval, and agent l is the one such that $u_s^{(l)} \leqslant u_l^{(\bar{j})}$. First note that each agent t, $t \leqslant l$, and $\bar{j}$ are separated by the optimal policy, or $\bar{j}$ does not move. Otherwise, according to Lemma 14 t must pass the initial location of $\bar{j}$ but according to Lemma 15 $\bar{j}$ must reach a store outside the interval $[u_s^{(l)}, u_s^{(\bar{j})}]$ which does not occur. Since $\bar{j}$ passes the initial locations of every agent t, $l < t < \bar{j}$, if one of them moves it goes only in the opposite direction of the final movement direction of $\bar{j}$ according to Lemma 15, and as illustrated in Fig. 3. Since they all must move in the same direction, according to the same lemma at most one of them moves in the optimal policy. Therefore, to compute $\mathsf{act}[u_{\bar{i}}, \bar{j}]$ we check only the following options, and choose the best one:

1. $\bar{j}$ does not move, and $\mathsf{act}[u_{\bar{i}}, \bar{j}] = \mathsf{act}[u_{\bar{i}}, \bar{j} - 1]$.
2. Each agent t, $l < t < \bar{j}$, does not move. Thus, $\mathsf{act}[u_{\bar{i}}, \bar{j}]$ is composed of $\mathsf{act}[u_{l-1}^{(\bar{j})}, l]$, with the optimal movement of agent $\bar{j}$ in the interval $[u_l^{(\bar{j})}, u_i]$.

The previous two options assume that $\bar{j}$ and every other agent are separated. Otherwise:

3. One agent t, $l < t < \bar{j}$, moves. Let $u_l^{(\bar{t})}$ be the leftmost store visited by either agent t or $\bar{j}$, with the optimal strategy, and agent l is the one such that $u_s^{(l)} \leqslant u_l^{(\bar{t})}$. $\mathsf{act}[u_{\bar{i}}, \bar{j}]$ is composed of $\mathsf{act}[u_{l-1}^{(\bar{t})}, l]$, with the optimal movement of the two agents $\bar{j}$ and t in the interval $[u_l^{(\bar{t})}, u_i]$.

There are at most m possible options for $u_l^{(\bar{j})}$. In each option we check for at most k agents m possible options for $u_l^{(\bar{t})}$. Therefore for each agent j and store u_i $\mathsf{act}[u_i, j]$ can be found in $O(m^2 k)$ steps, and $\mathsf{act}[u_m, k]$ can be found in $O(m^3 k^2)$ time steps using $O(mk)$ space. □

We can use the algorithm for the *k-Private-Max-Probability* problem to obtain a polynomial time algorithm for the *k-Private-Min-Budget*$^{\text{identical}}$ problem:

Theorem 17. *In the no communication setting, if the number of costs is constant,* k-Private-Min-Budget^{identical} *can be solved in polynomial time for any number of agents.*

Proof. By Theorem 12, given a budget $\bar{B}$, we can calculate the maximum achievable success probability. Thus we can run a binary search over the possible values of $\bar{B}$ to find the minimal one that still guarantees a success probability p_{succ}.

[5] There may be more than one strategy with the same failure probability, $\mathsf{act}[u_i, j]$ is one of them.

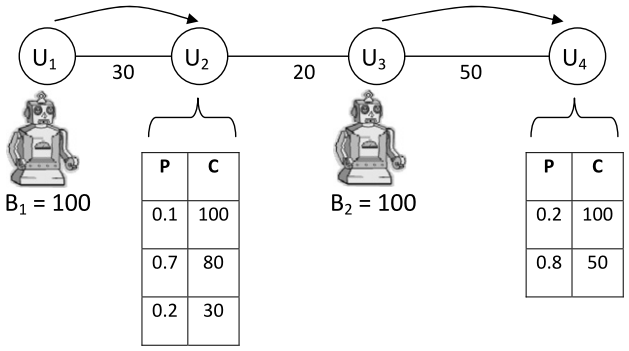

Fig. 4. A possible input with suggested moves. The numbers on the edges represent traveling costs. The table at each store u_i represents the cost probability function $p_i(c)$. The moves are illustrated by the arrows.

The maximum required budget is $2 \cdot |u_1 - u_m| + \max D$, which is part of the input. Thus the binary search will require a polynomial number of steps. □

Surprisingly, the results for *k-Private-Min-Budget*$^{\text{distinct}}$ are different. We note that in the *k-Private-Min-Budget*$^{\text{distinct}}$ problem the objective is to minimize the average budget, which is the same as minimizing the total budget. Therefore, if the number of agents is fixed, we can use the same polynomial time algorithm that is used in the proof of Theorem 9 (except that the visited intervals of the agents are not disjoint). If the number of agents is a parameter, the hardness of *k-Private-Min-Budget*$^{\text{distinct}}$ follows from that of the *k-Shared-Min-Budget* problem. We obtain:

Theorem 18. *If the number of agents is fixed,* k-Private-Min-Budget^{distinct} *with no communication can be solved in* $O(m2^{kd}(\frac{em}{2d})^{2kd})$. *If the number of agents is a parameter,* k-Private-Min-Budget^{distinct} *with no communication is NP-complete even for a single possible cost.*

4.2. Communicating agents

Once communication is added agents can call upon each other for assistance and the relative scheduling of the agents' moves must also be considered. In this case a *solution* is an ordered list of *moves*, where each move is a pair stating an agent and its next destination.

The success probability of a solution is now calculated according to the order of moves. For example, suppose that the stores and agents are located as illustrated in Fig. 4.

Consider the following solution: agent 2 first goes to u_4 and then agent 1 goes to u_2. Agent 2 is the only one which can succeed at u_4, with a probability of 0.8. With probability of 0.2 it will not succeed and agent 1 has a probability of 0.2 to succeed at u_2. Hence, the success probability is $0.8 + 0.2 \cdot 0.2 = 0.84$. If we switch the order of the moves we get a probability of 0.9 to succeed at u_2 with the first move, since agent 2 will be called for assistance if the cost required is less than 100. If not, agent 2 will move to u_4 as before. Hence, this solution success probability is $0.9 + 0.1 \cdot 0.8 = 0.98$.

When the number of agents is not fixed, *k-Private-Max-Probability*, *k-Private-Min-Budget*$^{\text{identical}}$ and *k-Private-Min-Budget*$^{\text{distinct}}$ are not known to be solvable in polynomial time. However, in many physical environments where several agents cooperate in exploration and search, the number of agents is relatively small. In this case we can show that all the three problems can be solved in polynomial time. We show:

Theorem 19. *In the setting of communicating agents, if the number of agents and the number of different costs is fixed then* k-Private-Max-Probability, k-Private-Min-Budget^{identical} *and* k-Private-Min-Budget^{distinct} *can be solved in polynomial time.*

For brevity, we focus on the *k-Private-Max-Probability* problem. The same algorithm and similar analysis work also for the other two problems.

First note that as in *Max-Probability*, in *k-Private-Max-Probability* we need to maximize the probability of obtaining the item given the initial budgets B_i, but there is no requirement to minimize the actual resources consumed (in contrast to *k-Shared-Max-Probability*). Thus, at any store, if agents can obtain the item for a cost no greater than its remaining budget, the search is over. Furthermore, if the cost is beyond the agent's available budget, but there is another agent with a sufficient budget to both travel from its current location and to obtain the item, then this agent is called upon and the search is also over. Otherwise, the item will not be obtained at this store under any circumstances. Thus, the basic strategy structure, which determines which agent goes where, remains the same. Unless the search has to be terminated, the decision of one agent where to go next is not affected by the knowledge gained by others. Using a similar argument as in the proof of Theorem 9, we get the following result. For brevity, we denote $\bar{d}$ instead of $d + 1$.

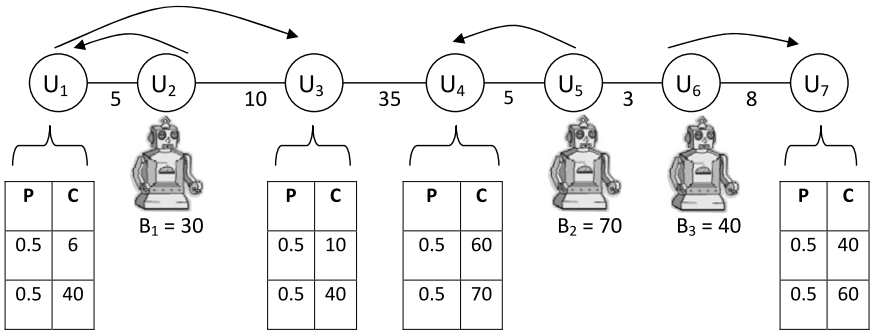

Fig. 5. A possible input with a suggested strategy. The numbers on the edges represent traveling costs. The table at each store u_i represents the cost probability function $p_i(c)$. The moves of each agent are illustrated by the arrows.

Proposition 20. *For k agents, one needs only to consider $O(m2^{k\bar{d}}(\frac{em}{2\bar{d}})^{2k\bar{d}})$ number of options for the set of moves of the agents.*

Proof. Let $c_1 > c_2 > \cdots > c_d$ be the set of costs. For each agent j and for each c_i there is an interval $I_i^{(j)} = [u_\ell, u_r]$ of points covered while *the agent's* remaining budget is at least c_i. Furthermore, for each j and for all i, $I_i^{(j)} \subseteq I_{i+1}^{(j)}$. Thus, consider for each agent the *incremental* area covered when *its* remaining budget is c_i but less than c_{i-1}, $\Delta_i^{(j)} = I_i^{(j)} - I_{i-1}^{(j)}$ (with $\Delta_1^{(j)} = I_1^{(j)}$). Each $\Delta_i^{(j)}$ is a union of an interval at left of $u_s^{(j)}$ and an interval at the right of $u_s^{(j)}$ (both possibly empty). Since there is communication, an agent may continue to reach a store even if it does not have any chance of obtaining the item there, in order to reveal the cost for the use of other agents. Thus, the optimal strategy may define also an interval $I_{\bar{d}}^{(j)} = [u_\ell, u_r]$ of points covered while the remaining budget of j is greater than 0. By Lemma 31 (see Appendix B), the moves of each agent are fully determined by the leftmost and rightmost stores of each $\Delta_i^{(j)}$, together with the choice for the ending points of covering each area. Therefore, for each j there are at most $\frac{(m)^{2\bar{d}}}{(2\bar{d})!} \leqslant (\frac{em}{2\bar{d}})^{2\bar{d}}$ possible choices for the external stores of the $\Delta_i^{(j)}$'s, and there are a total of $2^{\bar{d}}$ options to consider for the covering of each. Thus, the total number of options for the set of moves is $O(2^{k\bar{d}}(\frac{em}{2\bar{d}})^{2k\bar{d}})$, which is polynomial (in m). □

It thus remains to consider the scheduling between the moves, i.e. their *order*. Theoretically, with n moves there are $n!$ different possible orderings. We show, however, that for any given set of moves, we need only to consider a polynomial number of possible orderings.

Consider a given set of moves M, determining the sets $\Delta_i^{(j)}$. Note that for each agent, M fully determines the order of the moves of this agent. A subset M' of M is said to be a *prefix* of M, if for each agent the moves in M' are a prefix of the moves of this agent in M. A subset M' is a suffix of M if $M - M'$ is a prefix. We now inductively define the notion of a *cascading order*:

1. The trivial order on moves of a single agent is cascading.
2. Let M be a set of moves, and let c_{i_0} be the highest cost (of the product) that any agent can pay. An order S on M is *cascading* if M and S can be decomposed in the form $M = M_{pre} \cup M_{mid} \cup M_{post}$ and $S = S_{pre} \circ S_{mid} \circ S_{post}$, such that:
   - M_{pre} is a prefix of M consisting only of moves of agents with budget less than c_{i_0} and S_{pre} is a cascading order on M_{pre}.
   - There exists an agent j' with budget at least c_{i_0} such that M_{mid} consists of all the moves of j' in $\Delta_{i_0}^{(j')}$ and S_{mid} is the (one possible) order on these moves.
   - M_{post} are the remaining moves in M and S_{post} is a cascading order on them.

To visualize this definition, consider the example in Fig. 5. Given the depicted set of moves, the highest cost that an agent can pay is 60, by agent 2 at u_4. Therefore, a possible cascading order S can decompose M such that $M_{pre} =$ {agent 3 goes to u_7}, $M_{mid} =$ {agent 2 goes to u_4} and $M_{post} =$ {agent 1 goes to u_1 and to u_3}. Since each group contains only moves of a single agent, S is a cascading order on M. Another possible cascading order S' can decompose M such that $M_{pre} =$ {agent 1 goes to u_1 and to u_3, agent 3 goes to u_7}, $M_{mid} =$ {agent 2 goes to u_4} and $M_{post} = \{\}$. S' is a cascading order on M if $S' = S'_{pre} \circ S'_{mid} \circ S'_{post}$, where S'_{mid} and S'_{post} are trivial, and $S'_{pre} =$ {agent 1 goes first, agent 3 goes second}. We now prove (by induction) that cascading orders are optimal.

Lemma 21. *For any set of moves M there exists a cascading order with optimal success probability.*

Proof. The proof is by induction on the number of agents and the number of moves in M. If there is only one agent moving in M then the order is cascading. Otherwise, consider any other order S on M and let A_{i_0} be the set of agents with budget at least c_{i_0}. Let j' be the first agent in A_{i_0} to cover its $\Delta_{i_0}^{j'}$ and let t_0 be the time it completes covering it. M_{pre} includes all the moves taken by agents not in A_{i_0} prior to t_0; M_{mid} includes all the moves of j' in $\Delta_{i_0}^{(j')}$; and M_{post} the rest of the moves in M. We show that we do not decrease the success probability by first making all moves of M_{pre} then all those of M_{mid}, and finally those of M_{post}. By the inductive hypothesis S_{pre}, S_{mid} and S_{post} are optimal for M_{pre}, M_{mid} and M_{post}, respectively and the result follows.

Before t_0 all agents in A_{i_0} have a higher budget than any agent not in A_{i_0}. Thus, before t_0 agents of A_{i_0} will never call upon those not in A_{i_0}. Thus, if we let the agents that are not in A_{i_0} take their moves first the success probability will not decrease. We can thus allow all of the moves of M_{pre} to be performed first.

Also, before t_0 no agents of A_{i_0} needs to call upon each other for assistance (since they are all in the same resource bracket). Thus, we may allow them to take their moves independently without decreasing the success probability. In particular, we can allow j' to complete its covering of $\Delta_{i_0}^{(j')}$ before any other member of A_{i_0} moves. Thus, we get that first having the moves of M_{pre} and then of M_{mid} does not decrease the success probability. The moves of M_{post} are the remaining moves. □

Finally we show that the number of cascading orders is polynomial:

Lemma 22. *For fixed k and d and any set of moves M there are a polynomial number of cascading orders on M.*

Proof. Let $f(n,k,d,\ell)$ be the number of cascading orders with k agents, n moves, d costs and ℓ agents in A_{i_0}. We prove by induction that f is a polynomial in n. Since $\ell \leqslant k$, the result follows. Clearly, for any ℓ, $f(n,k,0,\ell) = \ell!$ (all of which are useless). Then, by the definition of cascading orders $f(n,k,d,\ell) \leqslant \ell n^{k-\ell} f(n,k-\ell,d-1,k-\ell) f(n,k,d,\ell-1)$ (the n^k being for the choice of M_{pre}). By the inductive hypothesis $f(n,k-\ell,d-1,k-\ell)$ and $f(n,k,d,\ell-1)$ are polynomials in n. Thus, so is $f(n,k,d,\ell)$. □

Together with Proposition 20 we get that the total number of options to consider is polynomial, proving the *k-Private-Max-Probability* part of Theorem 19. The proof for the other two problems is similar.

5. Self-interested agents

In this section we consider the strategic behavior that may occur when the agents are self-interested. We assume k agents, operate in the same underlying physical setting as in the previous multi-agent case with private budgets, i.e. the stores are all on a single path, the number of possible prices, d, is bounded, and there is a fixed number of agents. However in the self-interested agents setting, the agents seek to obtain the item but do not want to spend their individual budgets on travel costs; we assume the purchase price is equally shared among all the agents. In this case we define two games, a simultaneous game, *Min-Budget-Game*, and a sequential game, *Min-Expected-Cost-Game*.

5.1. Min-Budget-Game

In the *Min-Budget-Game* we are given a target success probability p_{succ}, and each agent's objective is to minimize its initial budget necessary to guarantee that the item will be acquired with a probability of at least p_{succ}. To avoid the case where each agent will set its initial budget at zero, we set the utility of not guaranteeing the success probability p_{succ} so low that it will always be worthwhile to attain it. We assume the game is a simultaneous game; the agents can only choose their initial budgets. After this phase, the agents calculate the (collaborative) strategy that will maximize their success probability (given their chosen budgets) and follow it. The only decision point in this game is when an agent needs to choose its budget.

Since the number of agents and the number of different costs is fixed, the optimal solution for *k-Private-Min-Budget*$^{\text{distinct}}$ can be found in polynomial time, whether the agents can or cannot communicate (Theorem 19). Let B_i^{alg} be the initial budget that was assigned to agent i by the algorithm from Theorem 19. This solution of *k-Private-Min-Budget*$^{\text{distinct}}$, which is optimal, can be directly translated into a strategy, denote Opt_{soc}: each agent i should individually choose its initial budget to be B_i^{alg}. Obviously, Opt_{soc} maximizes the social welfare and it can be computed in polynomial time. Furthermore, Opt_{soc} is also a Nash equilibrium [44, p. 14]. Clearly, for each agent i, there is no incentive to deviate and to choose a budget for itself which is larger than B_i^{alg}, since with B_i^{alg} the success probability p_{succ} is already guaranteed (assuming the other agents will not deviate). On the other hand, since the algorithm of Theorem 19 is optimal, there is no incentive for each agent i to deviate and choose a budget for itself which is smaller than B_i^{alg}, as p_{succ} will not be achieved (recall that the utility of not guaranteeing the success probability is very low). We obtain:

Theorem 23. *In the* Min-Budget-Game*, the strategy that maximizes the social welfare,* Opt_{soc}*, can be found in polynomial time and it is also a Nash equilibrium.*

5.2. Min-Expected-Cost game

In the *Min-Expected-Cost-Game* each agent's objective is to minimize its total expected cost. As in the previous game, to avoid the case where each agent will not want to make any move, we set the utility of not obtaining the item so low that it is always worth traveling to at least one store to purchase the product. The *Min-Expected-Cost-Game* is a sequential game and the rules are as follows. At each time step, only one of the agents is allowed to move to the next store, but it can also decide not to move at all. Then there is a decision phase, where every agent is allowed to buy the product, to opt-out, or to do nothing. If at least one agent decides to buy the product, it is purchased and then the game is over (even if other agents decide to opt-out). No matter how many agents decide to buy the product, only the one with minimal price is purchased. If no agent decides to buy the product and at least one agent decides to opt-out then the game is over without buying the product. Otherwise, the decision phase ends and the game proceeds by allowing the next agent (according to a fixed, pre-defined cyclic order) to move. The pre-defined order of movement phases well-define the game, but it has no essential meaning; the agents have the option not to move during their turns, so actually any order of movements may occur.

In order to find the strategy that will maximize the social welfare, Opt_{soc}, we need to run the algorithm from Theorem 8. In our setting, it will run in polynomial time. However, unlike in the *Min-Budget-Game*, the solution found cannot be directly translated into a strategy. First, we need to translate the movements. At any stage, if the algorithm for *k-Shared-Min-Expected-Cost* decides that a specific agent should move, for instance agent i, then the strategy for *Min-Expected-Cost-Game* defines that until it is agent i's turn to move, any other agent will not move during its movement phase, and all the agents will do nothing during the decision phase. We also need to handle the case where one agent does not move according to this strategy. For this purpose, we determine that in any case where one of the agents deviates from its determined policy in the movement phase, the other agents purchase the product during the decision phase that follows. If it is not possible, i.e. the product is not available where the agents are located, the other agents opt-out during the decision phase. The translation of the algorithm's decision to buy is straightforward; the strategy defines that in the corresponding decision phase all the agents decide to buy, and an agent that cannot buy opt-out. We also do not need to handle the case where one agent deviates in a decision phase, since the game will be over in that case. In conclusion, Opt_{soc}, the strategy for *Min-Expected-Cost-Game* that maximizes the social welfare, can be found in polynomial time using the algorithm from Theorem 8. However, Opt_{soc} is not always a Nash equilibrium, as will be shown in Example 24. For ease of notation, when describing Opt_{soc} or any other strategy we omit the movement and decision phases when the agents do nothing.

Example 24. Suppose that the stores and agents are located as illustrated in Fig. 6. The traveling costs between u_2 and u_3 and between u_4 and u_5 are so high, that the only reasonable moves are according to the illustrated arrows. Opt_{soc} for this example is that agent 1 will go to u_2. If the price is 6 the product will be purchased. Otherwise, agent 3 will go to u_6 and if the price is 6 the product will be purchased. Otherwise, agent 2 will go to u_4 and the product will be purchased at the minimal sampled price (which can be 12 or 27). The expected cost of this strategy is 14.375, but it is not a Nash equilibrium. Clearly, if the product was not purchased after the moves of agents 1 and 3, then the minimal sampled price will be 27. At this stage, if agent 2 deviates and decides not to move, the product will be purchased and the private cost of agent 2 will be 9 (the purchase price is equally shared among all the agents). If agent 2 proceeds according to Opt_{soc}, its expected cost will be $4 + 0.5 \cdot 4 + 0.5 \cdot 9 = 10.5 > 9$. Therefore, agent 2 will have an incentive to deviate from Opt_{soc}.

If we switch the movement order of agents 2 and 3, the expected cost will be higher, 15.125, but this strategy is a Nash equilibrium. Clearly, agent 1 will not deviate during its turn since the other agents will opt-out. Agent 2 will not deviate during its turn since its private cost will be 10, and if it will follow the strategy its expected cost will be $4 + 0.5 \cdot 4 + 0.5 \cdot (0.5 \cdot 2 + 0.5 \cdot 9) = 8.75 < 10$ (assuming the other agents will not deviate). Agent 3 will not deviate during its turn either, since its private cost will be 10, and if it will follow the strategy its expected cost will be $4 + 0.5 \cdot 2 + 0.5 \cdot 9 = 9.5 < 10$.

Example 24 demonstrates that Opt_{soc} is not always a Nash equilibrium. We now show a polynomial algorithm that always returns a strategy which is a Nash equilibrium. Furthermore, we show an upper bound on the algorithm's performance, and prove that it is tight.

Theorem 25. *There is a polynomial algorithm for finding a Nash equilibrium for the* Min-Expected-Cost-Game.

Proof. The algorithm works as follows. It divides all the buying costs by k, and then solves the finite-horizon MDP as in the proof of Theorem 8. The solution is then translated into a strategy for the *Min-Expected-Cost-Game*, in the same way we translated the optimal solution of *k-Shared-Min-Expected-Cost* to Opt_{soc}. We denote this strategy by Opt_{Nash}. Since the pre-process takes $O(d)$ operations the algorithm for finding Opt_{Nash} is polynomial.

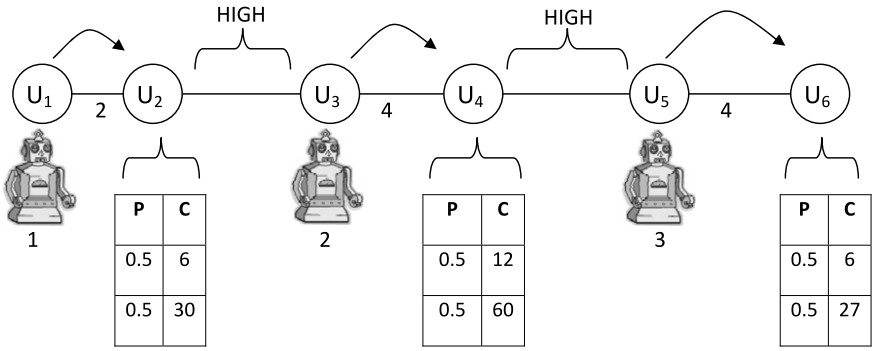

Fig. 6. A possible input with suggested moves. The numbers on the edges represent traveling costs. The table at each store u_i represents the cost probability function $p_i(c)$. The reasonable moves are illustrated by the arrows.

Any strategy S consists of traveling costs and buying costs, denoted by $\{t_i\}$ and $\{b_i\}$, respectively. We can then write the expected cost of S as $\mathrm{E}[S] = \sum_i (p_i^t \cdot t_i) + \sum_i (p_i^b \cdot b_i)$, where p_i^t, p_i^b are the associated probabilities. We also write $t_i \in j$ if the traveling cost t_i was credited to the movement of agent j.

We now analyze the steps of Opt_{Nash}. First note that if the product is not available to any agent, there is no incentive to deviate since the other agents will opt-out and the game will be over. We thus assume that the product is available. The last step of Opt_{Nash} is a decision step, where the product is purchased. By definition, there is an incentive to purchase the product in this step. In any other decision phase the strategy of Opt_{Nash} is not to purchase the product. However, if agent j deviates in a movement phase, the product will be purchased in the decision phase that follows. Therefore, we only need to consider the movement phases. Now, consider agent j and a movement phase r, and suppose that j needs to move in r. If j deviates (does not move), his expected cost will be c/k, the best price available now divided by the number of agents. Since Opt_{Nash} is optimal (with respect to the modified buying costs),

$$c/k \geqslant \sum_{i \geqslant r} (p_i^t \cdot t_i) + \sum_{i \geqslant r} (p_i^b \cdot b_i/k) \tag{6}$$

In addition,

$$\sum_{i \geqslant r} (p_i^t \cdot t_i) \geqslant \sum_{i \geqslant r, t_i \in j} (p_i^t \cdot t_i) \tag{7}$$

Combining (6) and (7) we get,

$$c/k \geqslant \sum_{i \geqslant r, t_i \in j} (p_i^t \cdot t_i) + \sum_{i \geqslant r} (p_i^b \cdot b_i/k)$$

where the right term is the expected cost of agent j if it follows Opt_{Nash}. Therefore, agent j has no incentive to deviate. The same analysis shows that j does not have an incentive to deviate if it does need to move in r. □

Opt_{Nash} is a Nash equilibrium, but it does not maximize the social welfare. Furthermore, there may be another Nash equilibrium which will yield a larger social welfare. For example, recall the settings in Example 24. In these settings, Opt_{Nash} policy is that agent 1 will go to u_2. If the price is 6 the product will be purchased. Otherwise, agent 3 will go to u_6 and buy the product at the minimal price (6 or 27). This is indeed a Nash equilibrium with an expected cost of 15.25. However, we already showed a better Nash equilibrium with an expected cost of 15.125. We now prove an upper bound on the performance of Opt_{Nash}; the expected cost of Opt_{Nash} is no more than k times worse than the expected cost of Opt_{soc}.

Theorem 26. $\mathrm{E}[\mathrm{Opt}_{Nash}] \leqslant k \cdot \mathrm{E}[\mathrm{Opt}_{soc}]$.

Proof. Suppose that $\mathrm{E}[\mathrm{Opt}_{Nash}] > k \cdot \mathrm{E}[\mathrm{Opt}_{soc}]$. Therefore,

$$\mathrm{E}[\mathrm{Opt}_{Nash}] = \sum_i (p_i^t \cdot t_i) + \sum_i (p_i^b \cdot b_i) > k \cdot \left[\sum_j (p_j^t \cdot t_j) + \sum_j (p_j^b \cdot b_j) \right] = k \cdot \mathrm{E}[\mathrm{Opt}_{soc}]$$

Then,

$$\sum_i (p_i^t \cdot t_i/k) + \sum_i (p_i^b \cdot b_i/k) > \sum_j (p_j^t \cdot t_j) + \sum_j (p_j^b \cdot b_j)$$

Since $t_i > t_i/k$ and $b_j > b_j/k$ then,

$$\sum_i (p_i^t \cdot t_i) + \sum_i (p_i^b \cdot b_i/k) > \sum_j (p_j^t \cdot t_j) + \sum_j (p_j^b \cdot b_j/k)$$

The left and right terms are the expected costs of Opt_{Nash} and Opt_{soc}, respectively, where all the buying costs are divided by k. Therefore, Opt_{Nash} is not optimal in these settings. Contradiction. □

As for the lower bound, consider the following example.

Example 27. For any $\epsilon > 0$, suppose that the price at $u_2 = u_s^{(1)}$ is k with a probability of 1, and the price at the leftmost store, u_1 is 0 with a probability of 1. The traveling cost from u_2 to u_1 is $1+\epsilon$. In all other stores the price is very high and the traveling costs between any other store to u_2 is also very high, for instance $2k$. Opt_{soc} for this example is that agent 1 will go left and buy the product at u_1. The cost of this strategy is $1+\epsilon$, but it is not a Nash equilibrium. Clearly, agent 1 will prefer to buy the product in its initial location, u_2, since its own cost will be 1, instead of $1+\epsilon$ in Opt_{soc}. The total cost from this strategy will be k, and it is the only Nash equilibrium. Therefore, for any algorithm that finds a strategy S which is a Nash equilibrium, $\mathrm{E}[S] \in \Omega(\frac{k}{1+\epsilon} \cdot \mathrm{E}[\mathsf{Opt}_{soc}])$, and the bound from Theorem 26 is tight.

6. Heterogeneous agents

The analysis so far assumes that all agents are of the same type, with identical capabilities. Specifically, the cost of obtaining the item at any given store is assumed to be the same for all agents. However, agents may be of different *types* and hence with different capabilities. For example, some agents may be equipped with a drilling arm, which allows them to consume less battery power while mining. In this section we consider situations of *heterogeneous agents*, and show that the results can be extended to such settings.

While agents may have different capabilities, in many cases it is reasonable to assume that if one agent is more capable than the other at one location, it is also more capable at all other locations (or at least not less capable). Hence the following definition:

Definition 28. We say that agents are *inconsistent* if there exist budgets B, B', agents j, j', and locations i, i', such that at location i with budget B

$$\Pr[j \text{ can obtain the item}] < \Pr\left[j' \text{ can obtain the item}\right]$$

but at location i' with budget B'

$$\Pr[j \text{ can obtain the item}] > \Pr\left[j' \text{ can obtain the item}\right]$$

We now show that the results of Section 4.1 can be extended to heterogeneous agents.

Theorem 29. *In the private budget and no communication setting, if the number of different costs for each agent is constant, then* k-Private-Max-Probability *and* k-Private-Min-Budget^{identical} *can be solved in polynomial time with any number of heterogeneous agents, provided that the agents are consistent.*

The algorithm is essentially the same dynamic programming algorithm described in Section 4.1. The consistency assumption is necessary for Lemmata 14 and 15 to remain true.

In any other case, we can do away with the consistency assumption. Clearly, however, we do need to assume that upon reaching a site, agents can assess the cost for obtaining the item for all other agents. Otherwise, communication would be meaningless. We obtain:

Theorem 30. *In the setting of communicating agents, with a constant number of agents, and a constant number of different costs for each agent,* k-Shared-Min-Expected-Cost, k-Shared-Min-Expected-Cost^{phone}, k-Shared-Max-Probability, k-Shared-Min-Budget, k-Private-Max-Probability, k-Private-Min-Budget^{identical} *and* k-Private-Min-Budget^{distinct} *can be solved in polynomial time even with inconsistent heterogeneous agents.*

The algorithms and proofs remain essentially the same as those for the case of homogeneous agents.

7. Discussion

In this paper we mainly analyzed the case where the stores are located along a path (either closed or non-closed). We see fair applicability of the proposed model to real-life applications. The most important/appropriate application, as discussed in the paper, is robot patrolling (see [60,19,2,3]). The reason is that a common patrolling scheme is patrolling along the surroundings of the area of interest, and the robot's movement is, by the task definition, restricted to be along a path even if there are shortcuts. However, there are additional important families of applications that the model can be mapped to – exploration along a path. Typical applications of this kind include:

- Finding a place for camping along a path – consider an expedition or a group of campers that follows a trail or travel along a river. When deciding on where to set their night camp, the group should consider different locations along their trail/river, each associated with some uncertainty related to the benefit from spending the night there.
- Positioning scouts – consider a border-control squad, which needs to position itself for scouting (e.g., based on some information that illegal infiltrators will arrive during the next few hours). Many possible locations along the border segment can potentially be used by the squad, each offering a different visibility level and accessibility to different point from where the infiltrators may arrive, which are a priori unknown (e.g., due to visibility conditions, terrain conditions and human factors).
- Deciding on a restaurant – consider a group that wants to dine together in one of the numerous restaurants on Balboa Blvd in Newport Beach. Assume that the true utility from dining in any of the restaurants can be observed only once getting to it (e.g., after observing the menu, how crowd it is and getting an impression of the general atmosphere).
- Deciding where to place bets – consider a visitor arriving to Las Vegas Blvd, interested in gambling in one of the casinos there. Similar to the restaurants' example, the gambler will be able to learn about the utility from gambling (not the expected payoff, which is likely to be similar in all places, but rather the gambling experience in terms of atmosphere, crowd, excitement, etc.) in any given casino only upon visiting it.
- Buying souvenirs – consider a tourist visiting the pier in Key West, Florida. Walking along the pier, the tourist will find many souvenir stores, selling practically the same items, however in different prices. In this case, the tourist should consider the tradeoff between the potential saving in cost and the alternative cost of time when going back and forth, visiting the different stores.
- Finding the best place to install a spying device at some place along a communication line – assume there are several possible locations along the line that are applicable for installing the equipment, each characterized by a different chance of being discovered or with a different chance of success.

All the above applications are characterized by a physical search along a line with potential locations that can be explored, where there is a distribution of potential gains/utilities for each location. Indeed, numerous physical environments may only be represented by a planar graph. Theorems 1 and 6 show that physical search problems are hard even on planar graphs and trees, even with a single agent, but finding a heuristic is of practical interest nonetheless. It seems that the first steps in building such a heuristic will be to utilize our results. For example, one should try to avoid repeated coverage as much as possible and restrict the number of cases where such coverage is necessary, as we showed in Theorem 12. Another idea is to convert the complex graph structure into a path, where each site on the path represents a region of strongly-connected nodes on the original graph. Many graphs which represent real physical environments consist of some regions with strongly-connected nodes, but few edges connect these regions (for example, cities, have many roads inside but are connected by only a few highways). A heuristic algorithm for these graphs may use our algorithm to construct a strategy for the sites along the path, and use an additional heuristic for visiting the sites inside a region.

We also considered the case where mining costs are rounded/estimated to one of a constant number of possible options. We believe that this assumption is appropriate since the given input for our problems includes prior probabilistic knowledge. Usually, this data comes from some sort of estimation so it is reasonable to assume that the number of options is fixed; If there are too many possible values, an accurate estimation is hard to achieve. Nevertheless, if the number of costs will not be a constant it can be rounded to a fixed number of costs, which yields a PTAS (polynomial-time approximation scheme) for our problems.

We also assume that the agents seek only one item. As soon as more than one item is needed, our results do not hold, and seemingly the problems become NP-complete.

8. Conclusions and future work

This paper considers single and multi-agent physical search problems, with prior probabilistic knowledge. This integration of changing search cost into economic search models is important, as it improves the realism and applicability of the modeled problem. At the same time, it also dramatically increases the complexity of determining the agents' optimal strategies, precluding simple solutions such as easily computable reservation values (see for example "Pandora's problem" [57], that was briefly discussed in Section 2.1.1). Indeed, we showed that our problems are hard on a metric space, sometimes even if it is a tree. We then focused on the path case, presenting polynomial algorithms for the two variants of *Min-Expected-Cost*

problem, and proving hardness for *Max-Probability* and *Min-Budget* problems. We provided FPTAS for *Min-Budget* in that case, and showed that both problems are polynomial if the number of possible prices is bounded.

For the multi-agent case, we analyzed shared and private budget models. In the case of the shared budget, we showed how all of the single-agent algorithms extend to k-agents, with the time bounds growing exponentially in k. We proved that this is also the case with the private budget model, if the agents can communicate. In the case of the private budget with no communicating agents, we presented a polynomial algorithm that is suitable for any number of agents. We also extended the analysis to heterogeneous agents.

Finally, we considered the self-interested agents setting, showing how to find a Nash equilibrium for *Min-Budget-Game* and *Min-Expected-Cost-Game* in polynomial time. In both cases, we showed an upper bound on the ratio between this solution to the optimal one (the one which maximizes the social welfare) and proved that it is tight.

For future work, there are still many interesting open problems. With a single agent, the complexity of *Min-Expected-Cost* problem on a tree is yet to be explored. This case is interesting since it can be shown that *Min-Expected-Cost* is easy for a specific tree, namely star graph, where d is bounded. In the shared budget model, the complexity of *k-Shared-Min-Expected-Cost* problem where k is part of the input is still open. In the private budget model, the complexity of all the problems with a non-constant number of communicating agents is open. In addition, there are interesting extensions to consider. We showed that most of our results can be extended to heterogeneous agents, with different *buying* capabilities. The next step is to analyze our results with heterogeneous agents with different *traveling* capabilities. Another direction is to add time constraint, which will possibly result with completely different optimal strategies. Currently, we assume that communication between the agent is free and reliable; a possible extension is to integrate communication costs to our model, and to handle the consequences of non-reliable communication. The *Min-Budget-Game* can also be extended; instead of defining the utility of achieving p_{succ} as a step function ("high" if achieving p_{succ}, and "low" if not), it could be defined as a linear function of p_{succ}. Finally, metric spaces beyond the line remain an open challenge. As discussed in Section 7, the techniques and results given in this paper can facilitate the development of approximations and/or heuristics for the general metric space.

Appendix A. Proofs for Section 2

Theorem 1. *For general metric spaces* Min-Expected-Cost-Decide *is NP-hard.*

Proof. The proof is by reduction from Hamiltonian path, defined as follows. Given a graph $G = (V, E)$ with $V = \{v_1, \ldots, v_n\}$, decide whether there is a simple path $(v_{i_1}, v_{i_2}, \ldots, v_{i_n})$ in G covering all nodes of V. The reduction is as follows. Given a graph $G = (V, E)$ with $V = \{v_1, \ldots, v_n\}$, set S (the set of stores) to be $S = \{u_s\} \cup \{u_1, \ldots, u_n\}$, where u_s is the designated start location, and $\{u_1, \ldots, u_n\}$ correspond to $\{v_1, \ldots, v_n\}$. The distances are defined as follows. For all $i, j = 1, \ldots, n$, $dis(u_s, u_i) = 2n$, and $dis(u_i, u_j)$ is the length of the shortest path between v_i and v_j in G. Set $M = 2n + \sum_{j=1}^{n} 2^{-j}(j-1) + 2^{-n}(n! + n - 1 + 2n)$. For all i, $p^i(0) = 0.5$, and $p^i(M) = 0.5$, and for u_s, $p^s(n!) = 1$.

Suppose that there is a Hamiltonian path $H = (v_{i_1}, v_{i_2}, \ldots, v_{i_n})$ in G. Then, the following policy achieves an expected cost of exactly M. Starting in u_s move to u_{i_1} and continue traversing according to the Hamiltonian path. If at any point u_i along the way the price is 0, purchase and stop. Otherwise continue to the next node on the path. If at all points along the path the price is M, return to u_s and purchase there, where the price is $n!$. The expected cost of this policy is as follows. The price of the initial step (from u_s to u_{i_1}) is a fixed $2n$. For each j, the probability to obtain price 0 at u_{i_j} but not before is 2^{-j}. The cost of reaching u_{i_j} from u_{i_1} is $j - 1$. The probability that no u_j has a price of 0 is 2^{-n}, in which case the purchase price is $n!$, plus $n - 1$ wasted steps along the Hamiltonian path and a cost of $2n$ for returning to u_s. The total expected cost is thus exactly M.

Conversely, suppose that there is no Hamiltonian path in G. Clearly, since the price at u_s is so large, any optimal strategy must check all nodes/stores $\{u_1, \ldots, u_n\}$ before purchasing at u_s. Since there is no Hamiltonian path in G, any such exploration would be strictly more expensive than the one with a Hamiltonian path. Thus, the expected cost would be strictly more than M. □

Theorem 2. *The* Min-Budget-Decide *problem is NP-complete even on a path.*

Proof. Given an optimal policy it is easy to compute its total cost and success probability in $O(n)$ steps, therefore *Min-Budget-Decide* is in NP. The proof of NP-hardness is by reduction from the 0–1 Knapsack problem, defined as follows. Given a knapsack of capacity $C > 0$ and N items, where each item has value $v_i \in \mathbb{Z}^+$ and size $s_i \in \mathbb{Z}^+$, determine whether there is a selection of items ($\delta_i = 1$ if selected, 0 if not) that fits into the knapsack, i.e. $\sum_{i=1}^{N} \delta_i s_i \leqslant C$, and the total value, $\sum_{i=1}^{N} \delta_i v_i$, is at least V.

Given an instance of 0–1 Knapsack we build an instance for the *Min-Budget-Decide* problem as follows. We assume WLOG that all the points are on the line. Our line consists of $2N + 2$ stores. N stores correspond to the knapsack items, denoted by $u_{k_1}, \ldots, u_{k_N}$. The other $N + 2$ stores are denoted $u_{g_0}, u_{g_1}, \ldots, u_{g_{N+1}}$, where u_{g_0} is the agent's initial location. Let $T = 2 \cdot \sum_{i=1}^{N} s_i$ and $maxV = N \cdot \max_i v_i$. For each odd i, u_{g_i} is to the right of u_{g_0} and $u_{g_{i+2}}$ is to the right of u_{g_i}. For each even i ($i \neq 0$), u_{g_i} is to the left of u_{g_0} and $u_{g_{i+2}}$ is to the left of u_{g_i}. We set $|u_0 - u_1| = |u_0 - u_2| = T$ and for each $i > 0$

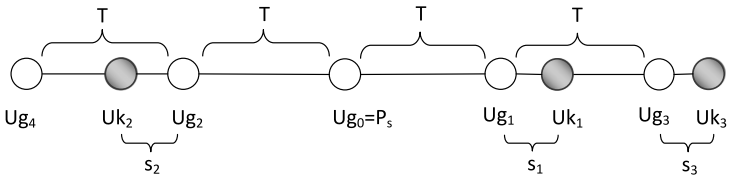

Fig. 7. Reduction of 0–1 Knapsack to *Min-Budget-Decide* problem used in the proof of Theorem 2, for $N = 3$.

also, $|u_{g_i} - u_{g_{i+2}}| = T$. If N is odd (even) u_{k_N} is on the right (left) side of u_{g_i} and it is the rightmost (leftmost) point. As for the other u_{k_i} points, u_{k_i} is located between u_{g_i} and $u_{g_{i+2}}$, if i is odd, and between $u_{g_{i+2}}$ and u_{g_i} otherwise. For both cases, $|u_{g_i} - u_{k_i}| = s_i$. See Fig. 7 for an illustration.

We set $B = T \cdot \sum_{j=1}^{N+1} j + 2C + 1$ and for each i set $X^i = T \cdot \sum_{j=1}^{i} j + 2 \cdot \sum_{j=1}^{i-1} s_j$. At store $u_{g_{N+1}}$ either the product is available at the price of 1 with probability $1 - 2^{-maxV}$, or not available at any price. On any other store u_{g_i}, either the product is available at the price of $B - X^i$ with the same probability, or not available at all. At any store u_{k_i}, either the product is available at the price of $B - X^i - s_i$, with probability $1 - 2^{-v_i}$, or not available at any price. Finally, we set $p_{succ} = 1 - 2^{-maxV \cdot (N+1)} \cdot 2^{-V}$.

Suppose there is a selection of items that fit the knapsack with a total value of at least V, and consider the following policy: go right from u_{g_0} to u_{g_1}. Then for each $i = 1, 2, \ldots, N$, if $\delta_i = 0$ (item i was not selected) change direction and go to the other side to $u_{g_{i+1}}$. Otherwise, continue in the current direction to u_{k_i} and only then change direction to $u_{g_{i+1}}$. This policy's total travel cost is $\sum_{i=1}^{N}(i \cdot T + \delta_i \cdot 2s_i) + (N+1) \cdot T = T \cdot \sum_{i=1}^{N+1} i + 2C = B - 1$, thus the agent has enough budget to reach all u_{g_i}, and u_{k_i} with $\delta_i = 1$. When the agent reaches u_{g_i}, $i < N + 1$ it has already spent on traveling cost exactly $T \cdot \sum_{j=1}^{i} j + 2 \cdot \sum_{j=1}^{i-1}(\delta_j \cdot s_j) \leqslant X^i$ so the agent has a probability of $1 - 2^{-maxV}$ to purchase the product at this store. When it reaches $u_{g_{N+1}}$ it is on the end of its tour and since the agent's total traveling cost is $B - 1$, here it also has a probability of $1 - 2^{-maxV}$ to purchase the product. When it reaches u_{k_i} it has already spent exactly $T \cdot \sum_{j=1}^{i} j + 2 \cdot \sum_{j=1}^{i-1}(\delta_j \cdot s_j) + s_i \leqslant X^i + s_i$ so the agent has a probability of $1 - 2^{-v_i}$ to purchase the product in this store. In total, the success probability is $1 - (2^{-maxV \cdot (N+1)} \cdot \prod_{i=1}^{N} 2^{-v_i \cdot \delta_i}) \geqslant 1 - (2^{-maxV \cdot (N+1)} \cdot 2^{-V}) = p_{succ}$ as required.

Suppose there is a policy, *plc* with a total travel cost that is less than or equal to B, and its success probability is at least p_{succ}. Hence, *plc*'s failure probability is at most $1 - p_{succ} = 2^{-maxV \cdot (N+1)} \cdot 2^{-V}$. Since $maxV = N \cdot \max_i v_i$, *plc* must reach all the $N + 1$ stores u_{g_i} with enough budget. Hence, *plc* must go right from u_{g_0} to u_{g_1} and then to each other u_{g_i} before $u_{g_{i+1}}$. Therefore *plc* goes in a zigzag movement from one side of u_s to the other side and so on repeatedly. *plc* also has to select some u_{k_i} to reach with enough budget. Thus, *plc* has to reach these u_{k_i} right after the corresponding store u_{g_i}. We use $\gamma_i = 1$ to indicate the event in which *plc* selects to reach u_{k_i} right after u_{g_i}, and $\gamma_i = 0$ to denote the complementary event. *plc*'s total traveling cost is less than or equal to $B - 1$ to be able to purchase the product also at the last store, $u_{g_{N+1}}$, so $T \cdot \sum_{j=1}^{N+1} j + 2 \cdot \sum_{j=1}^{N} \gamma_j \cdot s_j \leqslant T \cdot \sum_{j=1}^{N+1} j + 2C$. Thus, $\sum_{j=1}^{N} \gamma_j \cdot s_j \leqslant C$. Also, $p_{succ} = 1 - 2^{-maxV \cdot (N+1)} \cdot 2^{-V} \leqslant 1 - 2^{-maxV \cdot (N+1)} \cdot \prod_{i=1}^{N} 2^{-v_i \cdot \gamma_i} \Rightarrow 2^{-V} \leqslant \prod_{i=1}^{N} 2^{-v_i \cdot \gamma_i} \Rightarrow V \geqslant \sum_{i=1}^{N} v_i \cdot \gamma_i$. Setting $\delta_i = \gamma_i$ gives a selection of items that fit the knapsack. □

Theorem 6. *The* Min-Budget-Decide *problem is NP-complete on a tree, even with a bounded number of prices.*

Proof. Membership in NP is immediate as in the proof of Theorem 2. The proof of NP-hardness is by reduction from the 0–1 Knapsack problem.

Given an instance of 0–1 Knapsack we build an instance for the *Min-Budget-Decide* problem as follows. We have $N + 2$ stores. N stores corresponds to the knapsack items, denoted by $u_{k_1}, \ldots, u_{k_N}$. The other 2 stores are u_0 and u_e, where u_0 is the agent's initial location. The stores are placed on a star, which is a tree with one internal node, u_0, and $N + 1$ leaves. The distance to any u_{k_i} is defined according to the item size, $dis(u_0, u_{k_i}) = s_i/2$, and $dis(u_0, u_e) = C$. At any store u_{k_i}, either the product is available at the price of 0 with probability $1 - 2^{-v_i}$, or not available at any price. At store u_0 the product is not available, and at store u_e either the product is available at the price of 0 with probability $1 - 2^{-maxV}$, $maxV = N \cdot \max_i v_i$, or not available at any price. Finally, we set $p_{succ} = 1 - 2^{-maxV} \cdot 2^{-V}$, and $B = 2 \cdot C$.

Suppose there is a selection of items that fit the knapsack with a total value of at least V, and consider the following policy: for each $i = 1, 2, \ldots, N$, if $\delta_i = 1$ (item i was selected) go from u_0 to u_{k_i} and then back to u_0. Finally, go from u_0 to u_e. This policy's travel cost is $\sum_{i=1}^{N}(\delta_i \cdot s_i) + C \leqslant 2 \cdot C = B$. If the product is available at any store, its price is 0. Thus, the success probability of this policy is $1 - (2^{-maxV} \cdot \prod_{i=1}^{N} 2^{-v_i \cdot \delta_i}) \geqslant 1 - (2^{-maxV} \cdot 2^{-V}) = p_{succ}$ as required.

Suppose there is a policy, *plc* with a total travel cost that is less than or equal to B, and its success probability is at least p_{succ}. Hence, *plc*'s failure probability is at most $1 - p_{succ} = 2^{-maxV} \cdot 2^{-V}$. Since $maxV = N \cdot \max_i v_i$, *plc* must reach store u_e. *plc* also has to select some u_{k_i} to reach, but since $dis(u_0, u_e) = C$ and $B = 2 \cdot C$, *plc* must reach these u_{k_i} before reaching u_e. We use $\gamma_i = 1$ to indicate the event in which *plc* selects to reach u_{k_i}, and $\gamma_i = 0$ to denote the complementary event. *plc*'s traveling cost before going to u_e is less than or equal C, to be able reach u_e, so $\sum_{j=1}^{N} \gamma_j \cdot s_j \leqslant C$. Also, $p_{succ} =$

$1 - 2^{-maxV} \cdot 2^{-V} \leqslant 1 - 2^{-maxV} \cdot \prod_{i=1}^{N} 2^{-v_i \cdot \gamma_i} \Rightarrow 2^{-V} \leqslant \prod_{i=1}^{N} 2^{-v_i \cdot \gamma_i} \Rightarrow V \geqslant \sum_{i=1}^{N} v_i \cdot \gamma_i$. Setting $\delta_i = \gamma_i$ gives a selection of items that fit the knapsack. □

Appendix B. Proofs for Section 3

Theorem 8. *With k agents,* k-Shared-Min-Expected-Cost^{phone} *can be solved in* $O(d^2 2^k (\frac{m}{k})^{2k})$, *and* k-Shared-Min-Expected-Cost *can be solved in* $O(d^{2k+1} 2^k (\frac{m}{k})^{4k})$.

Proof. We start with *k-Shared-Min-Expected-Cost*$^{\text{phone}}$. Since the stores are on the path, at any point in time the points/stores visited by the agents constitute a set of k disjoint contiguous intervals, which we call the *visited intervals*. Clearly, the algorithm need only make decisions at store locations. Furthermore, decisions can be limited to times when the agents are at one of the two stores edges of the *visited interval*. At each such location, each agent has only three possible actions: "go right" – extending its visited-interval one store to the right, "go left" – extending its visited-interval one store to the left, or "stop" – stopping the search and buying the product at the best price so far. Also note that *after* each agent i has already visited its interval $[u_{\ell^{(i)}}, u_{r^{(i)}}]$, how exactly it covered this interval does not matter for any future decision; the costs have already been incurred. Accordingly, the states of the MDP are quadruplets $[L, R, E, c]$, such that $L = (\ell^{(1)}, \ell^{(2)}, \ldots, \ell^{(k)})$, $R = (r^{(1)}, r^{(2)}, \ldots, r^{(k)})$, $E = (e^{(1)}, e^{(2)}, \ldots, e^{(k)})$ and $c \in D$. For each agent i, $\ell^{(i)} \leqslant s^{(i)} \leqslant r^{(i)}$ and $e^{(i)} \in \{\ell^{(i)}, r^{(i)}\}$. Every such state represents the situation that each agent i visited stores $u_{\ell^{(i)}}$ through $u_{r^{(i)}}$, it is currently at location $u_{e^{(i)}}$, and the best price encountered so far is c. Since the intervals are disjoint, $r^{(i)} < \ell^{(i+1)}$ for every i.

The terminal states are $Buy(c)$ and all states where all the stores were visited. The terminal cost is c. For all other states there are at most $2k+1$ possible actions – "agent i go right" (provided that $r^{(i)} < \ell^{(i+1)}$ and $r^{(i)} < m$), "agent i go left" (provided that $r^{(i-1)} < \ell^{(i)}$ and $1 < \ell^{(i)}$), or "stop". The cost of "agent i go right" is $(u_{r^{(i)}+1} - u_{e^{(i)}})$, while the cost of "agent i go left" is $(u_{e^{(i)}} - u_{\ell^{(i)}-1})$. The cost of "stop" is always 0. Given a vector V, let $V^i(j)$ be the same vector but with value j at index i. Given the state $[L, R, E, c]$ and move "agent i go right", there is probability $p^{r^{(i)}+1}(c')$ to transition to state $[L, R^i(r^{(i)}+1), E^i(r^{(i)}+1), c']$, for $c' < c$. With the remaining probability, the transition is to state $[L, R^i(r^{(i)}+1), E^i(r^{(i)}+1), c]$. Transition to all other states has zero probability. Transitions for the "agent i go left" actions are analogous, while with the action "stop" there is probability 1 to transition to state $Buy(c)$. This fully defines the MDP. The optimal strategy for finite-horizon MDPs can be determined using dynamic programming (see [45, Ch. 4]). In our case, the complexity can be brought down to $O(d^2 2^k (\frac{m}{k})^{2k})$ steps (using $O(d 2^k (\frac{m}{k})^{2k})$ space).

We now move to *k-Shared-Min-Expected-Cost*. Like in the single agent case, if an interval $[u_{\ell^{(i)}}, u_{r^{(i)}}]$ has been visited by agent i and the item not yet purchased, then any future purchase within the interval (if there should be such a purchase) will be with an agent coming from outside the interval into the interval, and moving directly to a store for purchasing. Note that even if the interval $[u_{\ell^{(i)}}, u_{r^{(i)}}]$ has been visited by agent i, the purchaser may also be one of its immediate neighbors (i.e., agents $i-1$ and $i+1$). In addition, there is a unique store $u_{x_r^{(i)}}$, such any purchaser coming from anywhere to the right of $u_{r^{(i)}}$ and purchasing within $[u_{\ell^{(i)}}, u_{r^{(i)}}]$ purchases at $u_{x_r^{(i)}}$, and similarly a unique store $u_{x_\ell^{(i)}}$, for purchases coming from the left of $u_{\ell^{(i)}}$. Therefore, the states of the MDP for *k-Shared-Min-Expected-Cost* are septuplets $[L, R, E, C_\ell, X_\ell, C_r, X_r]$, such that $L = (\ell^{(1)}, \ell^{(2)}, \ldots, \ell^{(k)})$, $R = (r^{(1)}, r^{(2)}, \ldots, r^{(k)})$, $E = (e^{(1)}, e^{(2)}, \ldots, e^{(k)})$, $C_\ell = (c_{\ell^{(1)}}, c_{\ell^{(2)}}, \ldots, c_{\ell^{(k)}})$, $X_\ell = (x_{\ell^{(1)}}, x_{\ell^{(2)}}, \ldots, x_{\ell^{(k)}})$, $C_r = (c_{r^{(1)}}, c_{r^{(2)}}, \ldots, c_{r^{(k)}})$, and $X_r = (x_{r^{(1)}}, x_{r^{(2)}}, \ldots, x_{r^{(k)}})$. Every such state represents the situation that each agent i visited stores $u_{\ell^{(i)}}$ through $u_{r^{(i)}}$, it is currently at location $u_{e^{(i)}}$, $e^{(i)} \in \{\ell^{(i)}, r^{(i)}\}$, and the best price encountered so far in $[u_{\ell^{(i)}}, u_{r^{(i)}}]$ when coming from the left (respectively right) is $c_{\ell^{(i)}}(c_{r^{(i)}})$, which can be found at store $u_{x_{\ell^{(i)}}}(u_{x_{r^{(i)}}})$.

Given a state $[L, R, E, C_\ell, X_\ell, C_r, X_r]$, let $BuyAt(i)$ be the minimum cost of purchasing within the visited interval of agent i, i.e., $BuyAt(i) = \min\{c_{e^{(i)}} + |u_{e^{(i)}} - u_{x_{e^{(i)}}}|, c_{r^{(i)}} + |u_{e^{(i+1)}} - u_{r^{(i)}}|, c_{l^{(i)}} + |u_{e^{(i-1)}} - u_{l^{(i)}}|\}$. The terminal states are $Buy(i)$ with a terminal cost of $BuyAt(i)$, and all states where all the stores were visited, with a terminal cost of $\min_{i \in k} BuyAt(i)$. The actions are the same as in the MDP for *k-Shared-Min-Expected-Cost*$^{\text{phone}}$, but the transition probabilities are different. Given the state $[L, R, E, C_\ell, X_\ell, C_r, X_r]$ and move "agent i go right", there is probability $p^{r^{(i)}+1}(c')$ to transition to state $[L, R^i(r^{(i)}+1), E^i(r^{(i)}+1), C_\ell, X_\ell, C_r^i(c'), X^i(r^{(i)}+1)]$, for $c' < (c_{r^{(i)}} + |u_{r+1^{(i)}} - u_{x_{r^{(i)}}}|)$. With the remaining probability, the transition is to state $[L, R^i(r^{(i)}+1), E^i(r^{(i)}+1), C_\ell, X_\ell, C_r, X_r]$. Transition to all other states has zero probability. Transitions for the "agent i go left" action are analogous. Given the state $[L, R, E, C_\ell, X_\ell, C_r, X_r]$ and the action "stop", there is probability 1 to transition to state $Buy(i)$, $i = \arg\min_{i \in k} BuyAt(i)$. This fully defines the MDP. Using the same analysis as before, we get that the complexity of solving *k-Shared-Min-Expected-Cost* is $O(d^{2k+1} 2^k (\frac{m}{k})^{4k})$ steps (using $O(d^{2k} 2^k (\frac{m}{k})^{4k})$ space). □

Theorem 9. *With k agents,* k-Shared-Min-Budget *and* k-Shared-Max-Probability *with d possible prices can be solved in* $O(m 2^{kd} (\frac{em}{2kd})^{2kd})$.

Proof. For brevity, we focus on the *k-Shared-Max-Probability* problem. The same algorithm and similar analysis work also *k-Shared-Min-Budget* problem. Let $c_1 > c_2 > \cdots > c_d$ be the set of costs. For each agent j and for each c_i there is an interval $I_i^{(j)} = [u_\ell, u_r]$ of points covered while the remaining budget is at least c_i. Furthermore, for each j and for all i, $I_i^{(j)} \subseteq I_{i+1}^{(j)}$. Thus, consider for each agent the *incremental* area covered with remaining budget c_i but less than c_{i-1}, $\Delta_i^{(j)} = I_i^{(j)} - I_{i-1}^{(j)}$

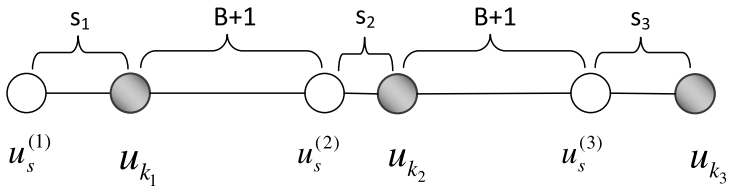

Fig. 8. Reduction of the 0–1 KNAPSACK problem to the *Multi-Min-Budget-Decide* problem used in the proof of Theorem 11, for $N = 3$.

(with $\Delta_1^{(j)} = I_1^{(j)}$). Each $\Delta_i^{(j)}$ is a union of an interval at left of $u_s^{(j)}$ and an interval at the right of $u_s^{(j)}$ (both possibly empty). The next lemma, which is the multi-agent Max-Probability analogue of Lemma 4 states that there are only two possible optimal strategies to cover each $\Delta_i^{(j)}$:

Lemma 31. *Consider the optimal solution and the incremental areas for each agent j, $\Delta_i^{(j)}$ $(i = 1, \ldots, d)$ defined by this solution. For $i \in 1, \ldots, d$, let $u_{\ell_i}^{(j)}$ be the leftmost store in $\Delta_i^{(j)}$ and $u_{r_i}^{(j)}$ the rightmost store. Suppose that in the optimal strategy the covering of $\Delta_i^{(j)}$ starts at location $u_{s_i}^{(j)}$. Then, WLOG we may assume that the optimal strategy for each j is either $(u_{s_i}^{(j)} \rightarrowtail u_{r_i}^{(j)} \rightarrowtail u_{\ell_i}^{(j)})$ or $(u_{s_i}^{(j)} \rightarrowtail u_{\ell_i}^{(j)} \rightarrowtail u_{r_i}^{(j)})$. Furthermore, the starting point for covering $\Delta_{i+1}^{(j)}$ is the ending point of covering $\Delta_i^{(j)}$.*

Proof. Any strategy other than the ones specified in the lemma would reach all the stores covered by the optimal solution with at most the same available budget. □

By the previous lemma, the moves of each agent are fully determined by the leftmost and rightmost stores of each $\Delta_i^{(j)}$, together with the choice for the ending points of covering each area. For each two agents j_1, j_2, the intervals of covered points are disjoint, i.e. $I_d^{(j_1)} \cap I_d^{(j_2)} = \emptyset$. Therefore, for each j there are at most $\frac{(\frac{m}{k})^{2d}}{(2d)!} \leqslant (\frac{em}{2kd})^{2d}$ possible choices for the external stores of the $\Delta_i^{(j)}$'s, and there are a total of 2^d options to consider for the covering of each. For each option, computing the budget and probability takes $O(m)$ steps. Thus, the total time is $O(m2^{kd}(\frac{em}{2kd})^{2kd})$, which is polynomial (in m). □

Theorem 10. *With k agents, For any $\epsilon > 0$,* k-Shared-Min-Budget *can be approximated to within a factor of $(1 + k\epsilon)$ in $O(n\epsilon^{-6k})$ steps (for arbitrary number of prices).*

Proof. For k agents, we extend the dynamic programming algorithm, which calculates fail$[\cdot, \cdot, \cdot, \cdot]$ and act$[\cdot, \cdot, \cdot, \cdot]$ tables, in the same way we extended the single agent algorithm in the proof of Theorem 8. We now save k disjoint intervals, thus the tables size becomes $O(\epsilon^{-6k})$. The rest of the approximation algorithm remains essentially the same. We still consider δ in powers of 2 up to $\beta \leqslant 2^n$, where n is the size of the input. Thus, the total computation time is $O(n\epsilon^{-6k})$. Since the approximation ratio in each interval is guaranteed to be $(1 + \epsilon)$, we get a total ratio of $(1 + k\epsilon)$. □

Theorem 11. k-Shared-Min-Budget-Decide *is NP-complete even on the path with a* single price.

Proof. An optimal policy defines for each time step which agent should move and in which direction. Since there are at most $2m$ time steps, it is easy to compute the success probability and the total cost in $O(m)$ steps, therefore the problem is in NP. The NP-hard reduction is from the 0–1 KNAPSACK problem.

We assume WLOG that all the points are on the line. We use N agents and our line consists of $2N$ stores. N stores correspond to the knapsack items, denoted $u_{k_1}, \ldots, u_{k_N}$. The other N points are the starting point of the agents, $\{u_s^{(i)}\}_{i=1,\ldots,N}$. We set the left most point to $u_s^{(1)}$ and the right most point to u_{k_N}. For all $1 \leqslant i \leqslant N - 1$ set u_{k_i} right after $u_s^{(i)}$ and $u_s^{(i+1)}$ right after u_{k_i}. Set $|u_s^{(i)} - u_{k_i}| = s_i$ and $|u_{k_i} - u_s^{(i+1)}| = B + 1$. See Fig. 8 for an illustration.

The price at all the nodes is $c_0 = 1$ and $p^{k_i}(1) = 1 - 2^{-v_i}$. Finally, set $B = C + 1$ and $p_{succ} = 1 - 2^{-V}$.

For every agent i, the only possible move is to node p_{k_i}, denote by $\gamma_i = 1$ if agent i moves to p_{k_i}, and 0 if not. Therefore, there is a selection of items that fit, i.e., $\sum_{i=1}^N \delta_i s_i \leqslant C$, and the total value, $\sum_{i=1}^N \delta_i v_i$, is at least V iff there is a selection of agents that move such that $\sum_{i=1}^N \gamma_i s_i \leqslant B$, and the total probability $1 - \prod_{i=1}^N \gamma_i 2^{-v_i}$, is at least $p_{succ} = 1 - 2^{-V}$. □

References

[1] F. Afrati, S. Cosmadakis, C. Papadimitriou, G. Papageorgiou, N. Papakonstantinou, The complexity of the traveling repairman problem, Theoretical Informatics and Applications 20 (1986) 79–87.

[2] N. Agmon, S. Kraus, G.A. Kaminka, Multi-robot perimeter patrol in adversarial settings, in: Proceedings of the 2008 IEEE International Conference on Robotics and Automation (ICRA-2008), 2008, pp. 2339–2345.

[3] N. Agmon, D. Urieli, P. Stone, Multiagent patrol generalized to complex environmental conditions, in: Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence (AAAI-2011), 2011, pp. 1090–1095.
[4] S. Arora, G. Karakostas, Approximation schemes for minimum latency problems, in: Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing (STOC-1999), 1999, pp. 688–693.
[5] S. Arora, G. Karakostas, A $2+\epsilon$ approximation algorithm for the k-MST problem, in: Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA-2000), 2000, pp. 754–759.
[6] Y. Aumann, N. Hazon, S. Kraus, D. Sarne, Physical search problems applying economic search models, in: Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (AAAI-2008), 2008, pp. 9–16.
[7] G. Ausiello, S. Leonardi, A. Marchetti-Spaccamela, On salesmen, repairmen, spiders, and other traveling agents, in: Algorithms and Complexity, Springer, Berlin/Heidelberg, 2000, pp. 1–16.
[8] B. Awerbuch, Y. Azar, A. Blum, S. Vempala, Improved approximation guarantees for minimum weight k-trees and prize-collecting salesmen, SIAM Journal on Computing 28 (1) (1999) 254–262.
[9] E. Balas, The prize collecting traveling salesman problem, Networks 19 (1989) 621–636.
[10] J.C. Beck, N. Wilson, Job shop scheduling with probabilistic durations, in: Proceedings of the Sixteenth European Conference on Artificial Intelligence (ECAI-2004), 2004, pp. 652–656.
[11] J.C. Beck, N. Wilson, Proactive algorithms for job shop scheduling with probabilistic durations, IEEE Transactions on Robotics 28 (1) (2007) 183–232.
[12] R. Bellman, A Markovian decision process, Indiana University Mathematics Journal 6 (4) (1957) 679–684.
[13] D. Bernstein, D. Givan, N. Immerman, S. Zilberstein, The complexity of decentralized control of Markov decision processes, Mathematics of Operations Research 27 (4) (2002) 819–840.
[14] L. Bianco, A. Mingozzi, S. Ricciardelli, The traveling salesman problem with cumulative costs, Networks 23 (2) (1993) 81–91.
[15] A. Blum, P. Chalasani, D. Coppersmith, B. Pulleyblank, P. Raghavan, M. Sudan, The minimum latency problem, in: Proceedings of the Twenty-Sixth Annual ACM Symposium on the Theory of Computing (STOC-1994), 1994, pp. 163–171.
[16] A. Blum, R. Ravi, S. Vempala, A constant-factor approximation algorithm for the k-MST problem, Journal of Computer and System Sciences 58 (1) (1999) 101–108.
[17] A.M. Campbell, M. Gendreau, B.W. Thomas, The orienteering problem with stochastic travel and service times, Annals of Operations Research 186 (1) (2011) 61–81.
[18] K. Chaudhuri, B. Godfrey, S. Rao, K. Talwar, Paths, trees, and minimum latency tour, in: Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS-2003), 2003, pp. 36–45.
[19] Y. Elmaliach, A. Shiloni, G.A. Kaminka, A realistic model of frequency-based multi-robot fence patrolling, in: Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2008), 2008, pp. 63–70.
[20] J. Fakcharoenphol, C. Harrelson, S. Rao, The k-traveling repairman problem, in: Proceedings of the Fourteenth Annual ACM–SIAM Symposium on Discrete Algorithms (SODA-2003), 2003, pp. 655–664.
[21] J. Fakcharoenphol, C. Harrelson, S. Rao, The k-traveling repairmen problem, ACM Transactions on Algorithms 3 (4) (2007) 40.
[22] T.S. Ferguson, Who solved the secretary problem?, Statistical Science 4 (3) (1989) 282–289.
[23] M. Fischetti, G. Laporte, S. Martello, The delivery man problem and cumulative matroids, Operations Research 41 (1993) 1055–1064.
[24] N. Fu, P. Varakantham, H.C. Lau, Towards finding robust execution strategies for RCPSP/max with durational uncertainty, in: Proceedings of the Twentieth International Conference on Automated Planning and Scheduling (ICAPS-2010), 2010, pp. 73–80.
[25] Y. Gabriely, E. Rimon, Spanning-tree based coverage of continuous areas by a mobile robot, Annals of Mathematics and Artificial Intelligence 31 (2001) 77–98.
[26] S. Gal, Search Games, Academic Press, 1980.
[27] A. García, P. Jodrá, J. Tejel, A note on the traveling repairman problem, Networks 40 (2002) 27–31.
[28] N. Garg, A 3-approximation for the minimum tree spanning k vertices, in: Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science (FOCS-1996), 1996, pp. 302–309.
[29] N. Garg, Saving an epsilon: a 2-approximation for the k-MST problem in graphs, in: Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing (STOC-2005), 2005, pp. 396–402.
[30] M. Goemans, J. Kleinberg, An improved approximation ratio for the minimum latency problem, in: Proceedings of the Seventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA-1996), 1996, pp. 152–158.
[31] N. Hazon, Y. Aumann, S. Kraus, Collaborative multi agent physical search with probabilistic knowledge, in: Proceedings of the Twenty-first International Joint Conference on Artificial Intelligence (IJCAI-2009), 2009, pp. 164–167.
[32] N. Hazon, G.A. Kaminka, Redundancy, efficiency, and robustness in multi-robot coverage, in: Proceedings of the 2005 IEEE International Conference on Robotics and Automation (ICRA-2005), 2005, pp. 735–741.
[33] W. Herroelen, R. Leus, Project scheduling under uncertainty: Survey and research potentials, European Journal of Operational Research 165 (2) (2005) 289–306.
[34] T. İlhan, S.M.R. Iravani, M.S. Daskin, The orienteering problem with stochastic profits, IIE Transactions 40 (4) (2008) 406–421.
[35] J. Kephart, A. Greenwald, Shopbot economics, Autonomous Agents and Multi-Agent Systems 5 (3) (2002) 255–287.
[36] J. Kephart, J. Hanson, A. Greenwald, Dynamic pricing by software agents, Computer Networks 32 (6) (2000) 731–752.
[37] S. Koenig, M. Likhachev, Fast replanning for navigation in unknown terrain, IEEE Transactions on Robotics 21 (3) (2005) 354–363.
[38] B.O. Koopman, Search and Screening: General Principles with Historical Applications, Pergamon Press, 1980.
[39] E. Koutsoupias, C.H. Papadimitriou, M. Yannakakis, Searching a fixed graph, in: Proceedings of the 23rd International Colloquium on Automata, Languages and Programming (ICALP-1996), 1996, pp. 280–289.
[40] S. Lippman, J. McCall, The economics of job search: A survey, Economic Inquiry 14 (1976) 155–189.
[41] A. Lucena, Time-dependent traveling salesman problem – the deliveryman case, Networks 20 (6) (1990) 753–763.
[42] J. McMillan, M. Rothschild, Search, in: R. Aumann, S. Amsterdam (Eds.), Handbook of Game Theory with Economic Applications, Elsevier, 1994, pp. 905–927 (Chapter 27).
[43] E. Minieka, The delivery man problem on a tree network, Annals of Operations Research 18 (1–4) (1989) 261–266.
[44] M.J. Osborne, A. Rubinstein, A Course in Game Theory, The MIT Press, 1994.
[45] M.L. Puterman, Markov Decision Processes: Discrete Stochastic Dynamic Programming, Wiley–Interscience, 1994.
[46] I. Rochlin, D. Sarne, M. Laifenfeld, Coordinated exploration with a shared goal in costly environments, in: Proceedings of the 20th European Conference on Artificial Intelligence (ECAI-2012), 2012, pp. 690–695.
[47] S. Sahni, T. Gonzales, P-complete problems and approximate solutions, in: Proceedings of the 15th Annual Symposium on Switching and Automata Theory (SWAT-1974), 1974, pp. 28–32.
[48] D. Sarne, S. Kraus, Managing parallel inquiries in agents' two-sided search, Artificial Intelligence 172 (4–5) (2008) 541–569.
[49] L.S. Shapley, Stochastic games, Proceedings of the National Academy of Sciences of the USA 39 (10) (1953) 1095–1100.
[50] D. Simchi-Levi, O. Berman, Minimizing the total flow time of n jobs on a network, IIE Transactions 23 (3) (1991) 236–244.

[51] R. Sitters, The minimum latency problem is NP-hard for weighted trees, in: Proceedings of the 9th Conference on Integer Programming and Combinatorial Optimization (IPCO-2002), 2002, pp. 230–239.
[52] S.V. Spires, S.Y. Goldsmith, Exhaustive geographic search with mobile robots along space-filling curves, in: First International Workshop on Collective Robotics, 1998, pp. 1–12.
[53] A. Stentz, The focussed D* algorithm for real-time replanning, in: Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI-1995), 1995, pp. 1652–1659.
[54] X. Sun, W. Yeoh, S. Koenig, Dynamic fringe-saving A*, in: Proceedings of the 8th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2009), 2009, pp. 891–898.
[55] T. Tsiligirides, Heuristic methods applied to orienteering, Journal of the Operational Research Society 35 (9) (1984) 797–809.
[56] I.R. Webb, Depth-first solutions for the deliveryman problem on tree-like networks: an evaluation using a permutation model, Transportation Science 30 (2) (1996) 134–147.
[57] Martin L. Weitzman, Optimal search for the best alternative, Econometrica 47 (3) (May 1979) 641–654.
[58] R.J.V. Wiel, N.V. Sahinidis, Heuristic bounds and test problem generation for the time dependent traveling salesman problem, Transportation Science 29 (2) (1995) 167–183.
[59] T. Will, Extremal results and algorithms for degree sequences of graphs, PhD thesis, University of Illinois at Urbana–Champaign, 1993.
[60] K. Williams, J. Burdick, Multi-robot boundary coverage with plan revision, in: Proceedings of the 2006 IEEE International Conference on Robotics and Automation (ICRA-2006), 2006, pp. 1716–1723.
[61] C. Yang, A dynamic programming algorithm for the travelling repairman problem, Asia-Pacific Journal of Operations Research 6 (10) (1989) 192–206.