



HAL
open science

Medical follow-up optimization: A Monte-Carlo planning strategy

Benoîte de Saporta, Aymar Thierry d'Argenlieu, Régis Sabbadin, Alice Cleynen

► **To cite this version:**

Benoîte de Saporta, Aymar Thierry d'Argenlieu, Régis Sabbadin, Alice Cleynen. Medical follow-up optimization: A Monte-Carlo planning strategy. 2024. hal-04382747

HAL Id: hal-04382747

<https://hal.science/hal-04382747>

Preprint submitted on 9 Jan 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Distributed under a Creative Commons Attribution 4.0 International License

Medical follow-up optimization: A Monte-Carlo planning strategy*

Benoîte de Saporta[†] Aymar Thierry d'Argenlieu[‡] Régis Sabbadin[§]
 Alice Cleynen[¶]

Abstract

Designing patient-specific follow-up strategy is a crucial step towards personalized medicine in cancer. Tools to help doctors deciding on treatment allocation together with next visit date, based on patient preferences and medical observations, would be particularly beneficial. Such tools should be based on realistic models of disease progress under the impact of medical treatments, involve the design of (multi-)objective functions that a treatment strategy should optimize along the patient's medical journey, and include efficient resolution algorithms to optimize personalized follow-up by taking the patient's history and preferences into account. We propose to model cancer evolution with a Piecewise Deterministic Markov Process where patients alternate between remission and relapse phases with disease-specific tumor evolution. This model is controlled via the online optimization of a long-term cost function accounting for treatment side-effects, hospital visits burden and disease impact on the quality of life. Optimization is based on noisy measurements of blood markers at visit dates. This optimization problem is extremely difficult. It has recently been modeled as an infinite dimensional continuous space Markov Decision Process, approximated by a discrete-space problem in order to be solved exactly. Here, instead, we leverage the Partially-Observed Monte-Carlo Planning algorithm to solve the full continuous-time, continuous-state problem, taking advantage of the nearly-deterministic nature of cancer evolution. We show that this approximate solution approach of the exact model performs better than the counterpart exact resolution of the discrete model, while allowing for more versatility in the cost function model, hence a patient-specific follow-up. Our findings in terms of modeling and our efficient simulation-based optimisation approach to produce follow-up strategies can efficiently and easily be adapted to a large number of other diseases, thus being useful to doctors and patients.

*We acknowledge the support of the French Agence Nationale de la Recherche (ANR), under grant ANR-21-CE40-005 (project HSMM-INCA), of European Union's Horizon 2020 research and innovation program (Marie Skłodowska-Curie grant agreement No 890462) and the support of MESO@LR-Platform at the University of Montpellier.

[†]IMAG, Univ Montpellier, CNRS, Montpellier, France

[‡]IMAG, Univ Montpellier, CNRS, Montpellier, France and IP Paris, Palaiseau, France

[§]Univ Toulouse, INRAE-MIAT, Toulouse, France

[¶]IMAG, Univ Montpellier, CNRS, Montpellier, France and John Curtin School of Medical Research, Australian National University, Canberra, ACT, Australia

Contents

1	Introduction	3
2	Results	5
2.1	Controlled piecewise deterministic Markov processes form a universal class of versatile models for patient follow-up	5
2.2	Cost functions encode the diverse impacts of treatment on the patient’s quality of life	6
2.3	Adapted Partially Observed Monte-Carlo Planning is particularly well suited for controlled PDMPs	6
2.4	Following up a patient with adapted POMCP is easy and fast in practice	8
2.5	Adapted POMCP can be tuned to outperform dynamic programming	8
2.5.1	Study 1: Impact of the parameters’ values on POMCP’s performance	10
2.5.2	Study 2: Adapted POMCP outperforms the dynamic programming approach	12
3	Discussion	13
4	Methods	14
4.1	Datasets, parameters and code availability	14
4.2	Reminder on controlled PDMPs	17
4.3	Reminder on Partially Observed Monte-Carlo Planning	17
4.4	Adapted POMCP algorithm to the case of controlled PDMPs	19
5	Supporting information	21
5.1	State of the art	21
5.2	Supplementary simulation results on POMCP parameters	22

1 Introduction

In long-term diseases such as cancer, patients alternate between remission and relapse phases and are monitored along time through non-invasive check-ups such as blood samples [24, 42]. Based on these noisy indirect disease measurements of some markers, practitioners must decide on treatment allocation, sometimes with little knowledge on the process dynamics (e.g. aggressiveness of the relapse) which may differ between patients [31, 39, 40]. Long retrospect of medical practice has allowed the definition of milestones to help practitioners in making follow-up decisions, but automated personalized criteria are yet to be defined to improve individual patient follow-up.

The ability to monitor patients in the least invasive manner, according to their personal preferences (more check-ups to enforce relapse detection, less aggressive treatments for better quality of life, etc) is a crucial step towards better care, but requires fine knowledge of diseases dynamics and reliable prediction algorithms. One of the main requirements for such task is the definition of a universal model adapted to patient-specific parameters that could describe in an exhaustive manner the possible consequences of the practitioner’s decisions. Mathematical models have been developed to link the tumor markers to tumor sizes [30], or to predict evolution of tumor growth from initial measurements [32, 45], but online adaptive models predicting relapses and automating treatment strategies are still lacking. In particular, such a model should be able to reconcile the continuous time evolution of the disease, continuous values for the markers leading to any possible values within a given range, and the noisy observations at discrete visit dates. A good candidate is the class of Piecewise Deterministic Markov Process (PDMP) [12, 13, 37]. Indeed, PDMPs are non diffusive hybrid stochastic processes that can handle both continuous and discrete variables and their interactions in continuous time. The only source of stochasticity comes from the jumps of the process. They are thus simple to simulate and easy to interpret. Controlled PDMPs allow continuous time dynamics on continuous (or hybrid discrete and continuous) state spaces with decisions taken in continuous time [14, 15].

This paper is based on the analysis of a large cohort of Multiple Myeloma (MM) patient data from the Intergrroupe Francophone du Myélome (IFM) 2009 clinical trial [2]. We assume that there is a single cancer marker that remains at a nominal threshold ζ_0 throughout any remission phase, and that at patient relapse its level increases exponentially with multiple possible behaviors until treatment is administered, or a threshold D is reached and the patient dies. To set up the context, we will assume that the study begins at time $t_0 = 0$ when the patient enters her first remission state, and we will denote t_1, t_2, \dots , her visit dates, defined over time by the practitioner, until some time horizon H is reached, or the patient dies. The time lapse between visits may not be constant, so that different patients may have different visit numbers and dates. More precisely, we will assume that at each visit time, the practitioner may choose to schedule the next visit in either 15, 30 or 60 days. Such decision may be based on the previous and current marker measurements, which we denote Y_0, Y_1, \dots . Note that the exact value of the marker is hidden as measurements are corrupted by noise, and measurements are only collected at visit dates. Together with the next visit date, the practitioner may chose to modify the patient’s current treatment, fixing it to one of the two available treatments, a and b , or to no treatment at all, denoted \emptyset . An example of patient follow-up data is presented in Figure 1 a).

We model the underlying continuous-time dynamics of the patient health by a controlled PDMP $(X_t)_{0 \leq t \leq H}$ [1, 14, 15, 17, 18, 35]. We propose to optimally control the process, that is to choose online the next treatment and visit date, based on present and past observations and decisions by minimising a cost function which is calibrated to balance the burden of deteriorated quality of life

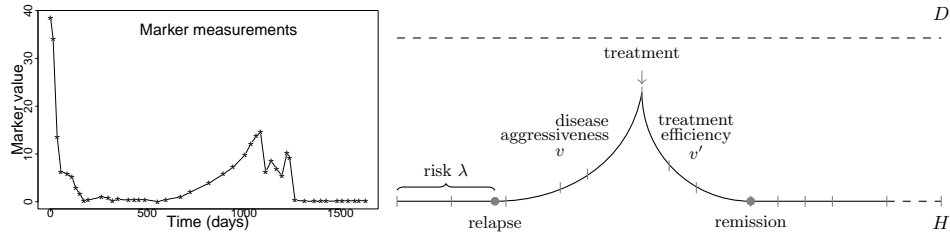


Figure 1: **Example of patient follow-up data, PDMP model.** **a)** Marker values are measured at each patient visits over a certain period of time. Data from the *Intergroupe Francophone du Myélome* 2009 clinical trial, courtesy of the *Centre de Recherche en Cancérologie de Toulouse*. **b)** PDMP model, representation of the marker level of a patient. The risk function λ controls the time to relapse, while parameters v and v' control the aggressiveness of the disease and the efficiency of the treatment respectively.

under treatment (including hospital visits) with the risk of dying from the disease.

Previous work has focused on discretizing this problem in order to solve it approximately through Dynamic Programming (DP) iterations [10]. More specifically, the optimal control problem for the PDMP has first been expressed as a Partially Observed Markov Decision Process (POMDP) [3]. This step is simply done by considering decision dates as stages of the POMDP. Note that the time lapse between decisions is thus not constant, the continuous time dynamics is encoded in the specific parametrization of the transition kernel, and the POMDP still has a continuous state space, with continuous observation space. The problem is then classically converted into a fully observed Markov Decision Process (MDP) [3, 9] on the *belief or filter space*. The filter process represents the probability distribution of the hidden values of the patient current state given the past and present observations. Second, the state space of the controlled PDMP has been discretized, so that an approximation of the filter process could be computed, charging only finitely many states. This approximate filter is called *conditional filter* in the sequel. Third, the belief space has been discretized in order to solve the MDP via dynamic programming iterations on a finite space.

In the current article, instead of discretizing the state and belief spaces, we use a *Monte-Carlo Tree Search* approach to (approximately) solve the controlled PDMP problem by simulation. More precisely, we propose an adaptation of the Partially Observed Monte-Carlo Planning (POMCP) algorithm [41], originally designed to solve discrete time / finite state and observation spaces POMDP, to the case of controlled PDMP. The novel challenge is that controlled PDMP involve continuous time as well as continuous state and observation spaces. We show empirically that this simulation-based approach outperforms the discretization-based approach both in terms of computation time and quality of returned policies. Thus, this approach is promising for providing an automated decision aiding tool for practitioners. To our knowledge, this is the first method to address this challenging question. Current decision making is typically based on heuristic rules derived from expert clinical knowledge [21, 29, 43].

2 Results

2.1 Controlled piecewise deterministic Markov processes form a universal class of versatile models for patient follow-up

Despite the discrete-time acquisition of the marker measurements, we choose to model the dynamics of the patient’s health by a continuous-time controlled Piecewise Deterministic Markov Process (PDMP). The formalism of PDMPs is both light and versatile [12, 14, 36, 37]. It allows to describe the dynamics of the disease with only three biologically relevant parameters: the disease’s risk function, λ , that dictates how often the patient is likely to relapse, the aggressiveness of the disease v that dictates how fast the marker level will increase during relapses, and the treatment efficiency v' that dictates how fast the marker level decreases under treatment.

In the formalism of PDMP, this is formulated by an exponential flow Φ which slope parameter (v or v') depends on patient condition and treatment, the risk function λ , and a transition kernel Q , that dictates how the patient’s state evolves at relapses, here preventing the marker values from jumping abruptly at patient condition changes. This is illustrated in Fig 1 b).

We consider a common risk function (identical for all patients) which we allow to depend on the time since the last remission date as well as the cancer marker level. We assume that the aggressiveness of the disease can be patient-dependent. It is either high or low and we model this as two different diseases. This leads us to introduce two different treatments, each efficient for one of those diseases and slowing the progression of the other. It is also an option not to treat for a given period.

We introduce three variables m, ζ, u , where the mode m corresponds to the overall condition of the patient ($m = 0$: remission, $m = 1$: disease 1, $m = 2$: disease 2, $m = 3$: death of the patient), $\zeta \in [\zeta_0, D]$ is the level of the marker, where ζ_0 is the nominal value and D the death level and $u \geq 0$ is the time since the last change of overall condition (added for technical reasons to deal with non-constant risk functions). The precise definition of the controlled PDMP and its parameters (λ, Φ, Q) are given in the Methods section. The complete state of the patient is thus encoded by $s = (m, \zeta, u)$. We denote X_0, \dots, X_n the process values at the observation dates t_0, \dots, t_n .

The overall condition of the patient m , the level of the marker ζ and the relapse dates (together with the time u since the last change of condition) are not directly observed and thus cannot be used by the clinician to select a treatment. At each visit of the patient to the medical center, we assume that the practitioner receives a noisy observation of the marker level $y = \zeta + \epsilon$ where ϵ is some Gaussian noise. The practitioner also knows the time t since the beginning of the patient follow-up. The complete observation available to the practitioner is thus encoded by $\omega = (y, t)$. The practitioner also has access to an indicator that the patient is still alive as treatment and follow-up stop at the death of the patient.

Based on the collection of present and past measurements and decisions, the practitioner selects both a time delay r until the next visit to the medical center and a treatment ℓ to hold until this next visit. Note that in our framework measurements are only made at visit dates. A decision is thus a pair $d = (\ell, r)$, where $\ell \in \{\emptyset, a, b\}$, and $r \in \{15, 30, 60\}$. Given a fixed arbitrary decision policy, simulating controlled patient trajectories is easy: see Algorithm 1 given in the Methods Section.

2.2 Cost functions encode the diverse impacts of treatment on the patient’s quality of life

For the practitioner, controlling the disease is equivalent to choosing the best available treatment as well as the best next visit date in order to minimize its impact on the patient’s quality of life along time. Defining the impact of treatment on the quality of life is a difficult task as it will typically depend on the treatment’s side effects, the number of visits, the burden of living with a disease and the remaining life expectancy.

This paper proposes a mathematical definition of the impact of the treatment on quality of life in terms of a cost function that takes into account those different aspects. For a decision $d = (\ell, r)$ comprising a treatment allocation ℓ and a time to next visit r , and for a current marker level ζ at time t_k , and future marker level ζ' at time $t_{k+1} = t_k + r$, we define

$$c(\zeta, d, \zeta') = C_V + \kappa|\zeta' - \zeta_0|r + \beta r \mathbf{1}_{\{\zeta = \zeta_0, \ell \neq \emptyset\}} + M \mathbf{1}_{\{\zeta' = D\}}, \quad (1)$$

where C_V is a visit cost, κ is non-negative scale factor penalizing high marker values, β is a penalty for applying an unnecessary treatment and M is the death cost.

This cost function thus takes into account a visit cost, to prevent patients from undergoing too many screening tests, a cost depending on the marker value at the next visit, to encourage treatment and calibrate visit dates, a cost for degradation of quality of life due to treatments, in particular if they are not appropriate, and a cost for dying.

Calibrating cost parameters C_V , κ , β , and M is a very difficult task, which is allowed to be patient-dependent (some patients may even express a wish to be sedated rather than undergo very long and painful treatments), and treatment strategies are bound to be parameters-dependent.

When cast as a controlled PDMP with this cost function, the practitioner’s problem is mathematically equivalent to solving a (continuous state space) Partially Observable Markov Decision Process (POMDP) [10, 25], which expected value optimisation can be stated as

$$V = \inf_{\pi \in \Pi} \mathbb{E}_{X_0}^{\pi} \left[\sum_{n=0}^{N_{\pi}-1} c(X_n, d_n, X_{n+1}) \right], \quad (2)$$

where V is called the *optimal policy value* and represents the lowest possible expected total cost, Π is the set of admissible policies (yielding decisions depending only on current and past observations), N_{π} is the patient-specific number of visits within the time-horizon of the study when using policy π , $d_n = \pi(Y_0, t_0, \dots, Y_n, t_n)$ is the decision (ℓ, r) taken at the n -th visit date t_n according to policy π , and $(Y_n)_{n \geq 0}$ represents the marker observation process for the controlled-PDMP/POMDP. Solving this problem amounts to computing (a good approximation of) the optimal policy value and identifying an admissible policy π^* that reaches (a value close to) the minimum.

2.3 Adapted Partially Observed Monte-Carlo Planning is particularly well suited for controlled PDMPs

The Partially Observed Monte-Carlo Planning (POMCP) algorithm [41] is an efficient simulation-based algorithm that has been designed for real-time planning in large finite state-space POMDPs. In this paper we show that even-though it has not been designed to handle continuous state and observation spaces, we can adapt it to solve controlled PDMPs, thanks to their efficient simulation property, without resorting to the computation of complex integrals for computing transition probabilities.

The objective of POMCP is to reduce the complexity of dynamic programming, which requires the construction of the entire decision tree (including the probabilities of every possible outcome with every possible decision at every future time-point), by sampling the tree in a principled way so as to compute the current optimal action. POMCP is thus an *online* algorithm, which re-estimates the optimal strategy at each new data acquisition¹.

The POMCP algorithm relies on two main properties. The first one is the ability to simulate trajectories, so as to progressively build the decision tree and update filters Θ at every intermediate node h of the tree. Recall that a filter is a probability distribution representing the (approximate) distribution of the current hidden state given the observations. The standard POMCP algorithm uses a specific family of simulation-based filters Θ^p called *particle filters* specified below. Filters are used to sample sets of plausible states.

The second property is the requirement to provide *estimates* of the expected value of the policy in leaves of the current exploration tree, in order to guide exploration and build the decision policy.

In the Methods Section we detail the algorithm (Algorithm 2) and show that POMCP is particularly well suited for controlled PDMPs. We simply point out here why trajectories simulation and policy evaluation are particularly efficient in POMCP, in the case of a controlled PDMP.

1. Simulation is particularly straightforward with PDMPs [15, 22, 28], requiring only to simulate the jump times and exploit the deterministic behavior between jumps, see Algorithm 1. In our medical framework, it is made even more simple since only few jumps are allowed. When little knowledge is available about the underlying process, a classic approach is to resort to *particle filters* Θ^p [16]. A particle filter Θ^p at step n is a discrete uniform probability distribution with finite support B^p (where B^p may have repeated atoms). It is updated at step $n + 1$ through simulations: states s from B^p are updated through a one-step simulation to a new state s' , and selected to be added to B^p if the simulated observation is close to the true one. As an alternative filter to compare to, we propose to use a *conditional filter* Θ^c derived from the exact filter (that is the conditional distribution of the hidden state given the observations) from [10]. The exact conditional filter is updated through a recurrence formula involving ratios of integrals over the state space. By discretizing the state space, one can construct the approximation Θ^c of the exact filter. Unlike the particle filter that has a dynamically changing support with a uniform mass function, this conditional filter has a fixed support (the discretized space) with changing mass functions that are updated through analytical ratios of weighted sums.
2. To estimate the future expected cost at some node of the tree, POMCP requires to simulate many full trajectories from the current node to a leaf of the tree. This requires to apply an arbitrary strategy to pick actions at every future nodes for which a decision has not yet been optimized. This arbitrary strategy is called a *rollout strategy* in the POMCP framework. The most naive rollout strategy consists in uniformly randomly selecting decisions from the decision set $\{\emptyset, a, b\} \times \{15, 30, 60\}$. We consider instead a mode-based rollout strategy, which consists in choosing action \emptyset in mode 0 (no treatment if the simulated patient is in remission), action a in mode 1 and b in mode 2 (most efficient treatment if the simulated patient has relapsed) and a fixed next visit date of 15 days. This rollout strategy, while not being necessarily optimal (depending on the cost function it might be optimal not to treat at the beginning of a relapse, or to treat preventively when in remission), exploits knowledge of the cost function,

¹POMCP does not "forget" previously computed strategies, but updates them using new simulated samples after every new observation is received.

hence yields better estimates of action costs at time t . Note also that this mode-based policy is not applicable for real patients, since their mode is not observed. It is only applicable to simulated patients. This is fine since POMCP’s rollout strategy is only used through simulations to estimate costs.

POMCP has a number of tuning parameters (number of simulations, number of particles in the filter, exploration vs exploitation rates) which are described (as well as the impact of varying their values) in the following section.

2.4 Following up a patient with adapted POMCP is easy and fast in practice

The previous paragraphs set the grounds for optimizing the long-term follow up of patients. In practice, we will assume a patient will enter the follow-up study once she enters the remission phase after an initial round of treatment. The practitioner may hence assume that her current state is known, *i.e.* $s_0 = (0, \zeta_0, 0)$ and the initial value of both the particle and conditional filters is the Dirac mass at s_0 . The initial observation is $\omega_0 = (\zeta_0, t_0)$. The adapted POMCP algorithm is run to obtain the optimal decision d_0 , which the practitioner can use (if she decides to) to allocate treatment and decide on the next visit date t_1 .

At visit n , the patient will come back for some new marker measurement, so that the n -th observation value $\omega_n = (y_n, t_n)$ is obtained. The practitioner will have access to her full history, $h_n = \langle \omega_0 d_0 \omega_1 d_1 \cdots d_{n-1} \omega_n \rangle$ as well as her last belief filter, Θ_{n-1}^c or Θ_{n-1}^p . An initial update of the filter is performed, either using the recursion formula for Θ_n^c from Θ_{n-1}^c and ω_n , or by particle filtering through rejection sampling for Θ_n^p from Θ_{n-1}^p and ω_n . The adapted POMCP algorithm is then ran to obtain the optimal current decision d_n , which the practitioner can use (or not) to allocate treatment and decide on the next visit date t_{n+1} . This is illustrated in Figure 2.

2.5 Adapted POMCP can be tuned to outperform dynamic programming

The simulation study presented in this section has been conducted based on real data obtained from the *Centre de Recherche en Cancérologie de Toulouse* (CRCT). Multiple myeloma (MM) is the second most common haematological malignancy in the world and is characterised by the accumulation of malignant plasma cells in the bone marrow. Classical treatments are based on chemotherapies, which, if appropriate, act fast and efficiently bring MM patients to remission in a few weeks. However almost all patients eventually relapse more than once and the five-year survival rate is around 50%.

We have obtained data from the *Intergroupe Francophone du Myélome* 2009 clinical trial [2] which has followed 748 French MM patients from diagnosis to their first relapse on a standardized protocol for up to six years. At each visit a blood sample was obtained to evaluate the amount of monoclonal immunoglobulin protein in the blood, a marker for the disease progression. An example of patient dataset is given in Figure 1.

Based on these data, we calibrated our PDMP model as described in the Methods section, and we performed simulations to evaluate the performance of the POMCP strategy to select the combination of treatment and next visit date at each time point of the trajectories (these time-points being themselves selected by the algorithm). The performance of the approach was measured by a Monte-Carlo estimate of the expectation and confidence interval of its value as well as the runtime

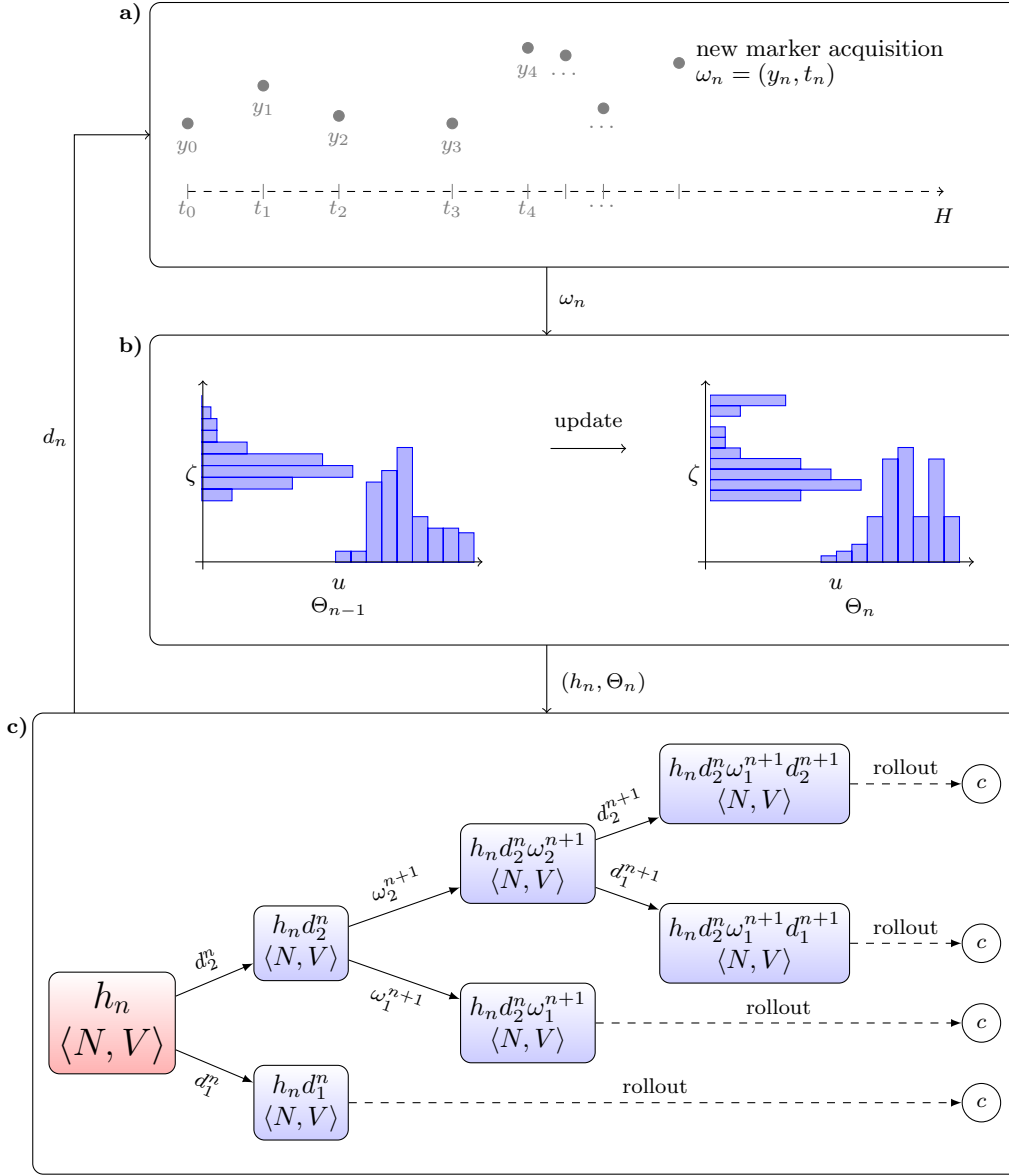


Figure 2: **Practice of patient follow-up.** **a)** At each new visit the patient has a new marker measurement, and the practitioner receives a new observation $\omega_n = (y_n, t_n)$. **b)** The filter is updated with the new observation, either through particle rejection sampling (*particle filter*) or via a recursion formula (*conditional filter*). **c)** The decision tree is partially explored via simulation through an adapted POMCP algorithm using the updated filter. The algorithm returns the optimal decision d_n , combination of a time to next visit (defining t_{n+1}) and treatment to allocate (influencing y_{n+1}).

of the online computation of a complete trajectory. For each disease parameters’ configuration 500 simulations were performed to estimate these values. Codes and parameters are available at <https://github.com/acleynen/pomcp4pdmp> [11].

2.5.1 Study 1: Impact of the parameters’ values on POMCP’s performance

We evaluated the impact of the value of 6 parameters on the performance: (i) the *filter* chosen (conditional or particle), (ii) the *rollout* procedure chosen, (iii) the exploitation/exploration tradeoff parameter α' , (iv) the number n_{search} of simulations in the online POMCP procedure, (v) the number K of initial states to sample from at each of the n_{search} simulations, and (vi) the internal POMCP precision parameter \mathcal{D} to select particles in the particle filter. Those parameters are described at length in the Methods Section, see Algorithm 2. The github page [11] contains tables with results for every sets of parameters’ values that were tested. In this section we describe the most important results.

We performed all parameter comparisons for both the conditional and the particle filters. The conditional particle filter is a discrete probability distribution on a finite fixed support B^c of size 184 with 81 states in condition $m = 0$, 31 states in condition $m = 1$, 71 states in condition $m = 2$ and one state in condition $m = 3$. The choice of these states is discussed in [10]. To adapt this filter to the POMCP environment, at each iteration n we start by randomly sampling K states s from B^c with the distribution given by Θ_n^c . For the particle filter, this number K directly corresponds to the number of particles in the filter, hence the size of B^p . Note that for the conditional filter the support B^c does not change over time, whereas for the particle filter B^p keeps the same size but possibly contains different states at each iteration.

Mode-based rollout outperforms naive rollout. We found that the uniform rollout procedure (selecting decisions randomly) produced very poor results compared to the mode-based rollout procedure and hence we only present results for the mode-based rollout policy here.

POMCP is robust to the exploration/exploitation trade-off. We found that the exploitation/exploration trade-off parameter had little influence on the overall performance, with the exception of extreme values ($\alpha' = 0.99$, almost no exploration, and $\alpha' = 0.2$, almost no exploitation, both leading to poorer performance). We also tried several adaptive strategies to select the value of α' depending on the confidence in the belief (measured in terms of entropy) which did not improve the results. The following results are therefore discussed for a fixed value of $\alpha' = 0.5$, but the reader may refer to the Supplementary Information Tables 4 and 5 for additional results on these parameters.

Increasing the number of exploratory simulations yields the best performance gain.

For a fixed number of sampled belief states K and precision \mathcal{D} , increasing the number of simulations n_{search} improved the performance of the algorithm while decreasing the variance in the simulations for both filter types (see the top left panel of Figure 3 for $K = 500$, $\mathcal{D} = 0.01$). In the case of the conditional filter, the runtime increases linearly, from 10^3 seconds per trajectory with $n_{\text{search}} = 100$ simulations to 10^4 seconds per trajectory with $n_{\text{search}} = 1000$. In the case of the particle filter, the runtime is 3 times as much for 100 simulations since when $n_{\text{search}} < K$ additional simulations have

to be performed to compute the particle filter at time $n + 1$. The difference decreases to only 1.2 times the runtime of the conditional filter for $n_{\text{search}} = 1000$.

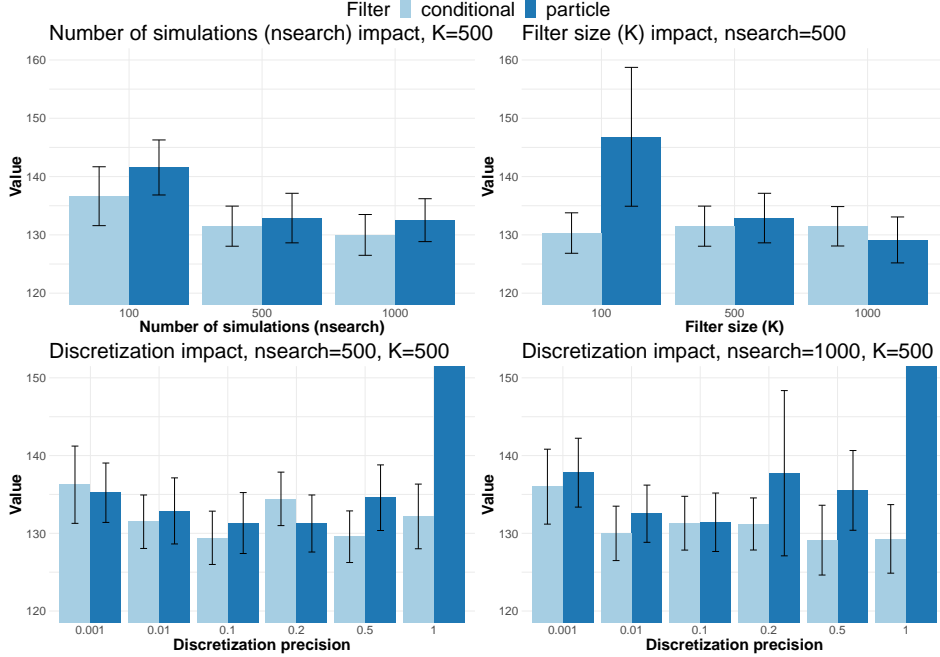


Figure 3: **Impact of POMCP parameters on the estimated value function.** Top left: increasing the number of simulations for filters with 500 initial states improves the average trajectory costs. Top right: increasing the number of atoms in the filter improves the performance of the particle filter but not the conditional filter.

The particle filter requires a large belief state to achieve high performance. As expected, for a fixed number of exploratory simulations n_{search} and precision \mathcal{D} , increasing the number K of particles in the particle filter led to a tremendous improvement for the particle filter together with a significantly decreased variance in the trajectory costs, while it had no impact on the conditional filter (see top right panel of Figure 3 for $n_{\text{search}} = 500$, $\mathcal{D} = 0.01$). Similarly, K has no runtime impact for the conditional filter, while it leads to exponential increase of the runtime for the particle filter.

POMCP for controlled PDMPs requires one additional tuning: the precision of the tree observation nodes. The bottom two panels of Figure 3 illustrate the impact of the precision \mathcal{D} for a fixed filter size ($K = 500$) and two different numbers of simulations ($n_{\text{search}} = 500$, left, and $n_{\text{search}} = 1000$, right). For a smaller number of simulations, decreasing the precision improves the result up to $\mathcal{D} = 0.1$, and then worsens them again. This tendency is still observed for the particle filter when n_{search} increases, while the performance of the conditional filter is optimal with the loosest precision, $\mathcal{D} = 1$. Those results are the consequence of two factors: as detailed in the

Methods Section, each simulation creates a novel node in the tree exploration, where a node is a set of potential future trajectories with their estimated costs. When the precision is very fine, each simulation produces a different future observation and hence the estimation relies on one-step forward simulations which may miss future events. On the other hand, when the precision is very loose, each simulation will build on the previous simulation to explore a step forward, yielding very long trees with few branches. This will also miss the variability of different outcomes. The second factor comes from the way the filters are constructed. Particle filters rely on comparing simulations to observations. When the precision is very loose, almost all simulations will be accepted, creating a strong bias in the belief of the current state, that will propagate from time-point to time-point. As the conditional filter update does not rely on simulations, hence neither on precision, there is no propagation of uncertainty from step to step, and when the number of exploratory simulations n_{search} is large enough to guarantee some diversity in the tree exploration, estimating the cost of each decision from longer trajectories will provide better results.

Finally, one may note that the gain in using conditional filters is mostly apparent in extreme parameter scenarios, for instance with very low number of particles K , with very high precision rates, etc. Provided the user has enough computing budget, both filters tend to provide very similar results.

2.5.2 Study 2: Adapted POMCP outperforms the dynamic programming approach

In this study we compare the results of three resolution strategies calibrated with their optimal parameters on biological relevant outcomes: the death rate, the Progression-Free Survival (PFS) time, that is the time from entry in the study to the first relapse, the time spent under treatment, the number of visits to the hospital, and the cost. Those quantities were normalized so that they range between 0 and 1, and such that an optimal result is 0. To do so, death rate was normalized so that a random treatment strategy yields 1 (here 5% of patients); the PFS was transformed as $1 - (\text{PFS}/H)$ (where we recall that H is the study horizon), so that a patient who does not relapse has normalized PFS equal to 0; the time spent under treatment was normalized by H , the number of visits was normalized as $\frac{N_{\text{visit}} - 40}{160 - 40}$ since over the horizon, a visit every 15 days produces 160 visits, whereas a visit every 60 days produces 40 visits; and finally the cost was normalized as $\frac{C - v_0}{C_{\text{random}} - v_0}$ where v_0 is the the best approximation of the optimal value obtained through discretizations in [10], and C_{random} is the average cost of the random strategy.

Here again, we simulated 500 trajectories with each strategy under the same cost parameters. The results are summarized in the Radar plot of Figure 4, and additional visual information on average trajectory cost are given in the barplots. In the Radar plot representation, a perfect strategy should delimitate the inner circle.

Compared strategies are the discretization/DP approach (DP) from [10] that relies on exact resolution by dynamic programming of the discretized POMPD, the adapted POMCP with the conditional filter (POMCP-Conditional) and the adapted POMCP with the particle filter (POMCP-Particles). Interestingly, the combination of POMCP with the conditional filter yields the lowest average trajectory cost and the shortest average time spent under treatment, by slightly increasing the number of visits and reducing PFS compared to the DP approach. However, the particle-based POMCP approach (relying fully on simulations with no other exploitation of the underlying model) yields cost almost as good as the previous two approaches, with increased number of visits but longest progression-free survival time. Importantly, out of 500 simulations, one of the trajectories

ended with a patient dying.



Figure 4: **a)** Radar plot comparing performances of 4 solution strategies on death rate, progression-free survival (PFS), time spent under treatment, average number of visits per patient, and average trajectory cost. An optimal strategy would be the inner-circle. **b)** Barplot of trajectory cost for 500 simulations under three main strategies: POMCP with particle filter, POMCP with conditional filter and Dynamic Programming on discretized processes.

3 Discussion

In this paper, a mechanistic PDMP model for cancer evolution and treatment has been presented. PDMPs are very flexible tools that allow to model routine screening data with very few parameters with biological meaning. When embedded in a control framework, PDMPs usually suffer from the need to compute intractable integrals and thus resort to several layers of approximations with heavy computational burden. In our case, there are two major difficulties in solving the optimization problem for the controlled PDMP. The first one is related to the partial observations of the process, since the practitioner only observes some noisy measurement of the marker at visit dates, and not the overall condition of the patient nor the relapse dates. The second one comes from the continuous state space and continuous time dynamics of the process, which prevent direct use of exhaustive exploration solution strategies such as dynamic programming [6].

In a previous work [10], we have dealt with those difficulties by defining an equivalent fully observable Markov decision process on an enlarged state space through the use of *conditional filters*, the conditional distributions of the hidden process given the observations. Then we discretized the state space of the original process, in order to obtain *finite support* filters and discretized again these finite support approximate filters to obtain finitely many (belief) states. The fully discretized model can then be solved by dynamic programming.

Here we investigated another original solution approach exploiting filter objects under a different (simulation-based) dimension reduction strategy. We show that the inherent generator function of the PDMP can be exploited to make use of simulation-based solution strategies such as POMCP

with excellent performance. Provided the number of simulations is large enough (either to explore the outcome space, or to construct consistent belief states), this approach can even outperform discretization approaches that exploit the knowledge of the underlying model. We have also proposed to combine both approaches, using discretization based conditional filters and simulation based solution strategy, resulting in a more robust algorithm (in particular less sensitive to the choice of POMCP parameters and with more stable variance), but with little performance gain.

The main advantage of the discretization/DP approach is that solutions are pre-computed for all new patients. This is especially useful under the assumption that all patients have the same dynamics with the same parameters. Its main drawback is that the model presented here is at maximum complexity for such an approach. In particular, it will become intractable if one wants to take into account more disease markers or more modes and treatments.

Conversely, the simulation-based approach can extend to any complexity provided simulations can be performed easily and fast. In addition, one of the main advantages of simulation-based approaches such as that presented here is that cost parameters can be modified with each new patient, yielding a patient-based procedure closer to precision medicine. This remains quite theoretical, as in practice calibrating cost parameters is a very difficult task, but with experience practitioners may be able to encode personal preferences, such as shorter life with better quality, or longer life at the price of more treatments, etc.

In this work, we have adapted the POMCP algorithm to solving a controlled PDMP with *known* model, too complex to be solved through exact dynamic programming. We made the assumption that the patient-disease model was known, which is a daring assumption. Therefore, one next step of our approach is to extend it by considering an unknown model and applying *Reinforcement Learning* methods [44]. A fundamental problem in Reinforcement Learning is the difficulty of deciding whether to select actions in order to learn a better model of the environment, or to exploit current knowledge about the rewards and effects of actions [26]. This is especially true in disease control problems.

4 Methods

4.1 Datasets, parameters and code availability

In order to propose a simulation study as realistic as possible we have used real data to infer the parameters of the design. The data come from the follow-up of 748 multiple myeloma patients registered in the 2009 IFM clinical trial described in the Results Section. An example of data is given in Figure 1. From this data, we opted for the exponential form of the dynamics in the disease states with boundaries $\zeta_0 = 1$ and $D = 40$ for remission and death levels, simply calibrated as the minimal and maximal values in the data set. Then, as described in the Results Section, 3 parameters had to be calibrated, and all hyperparameters are explicitly given in the github repository [11].

For the risk function λ , we choose to distinguish the *standard relapse* (from remission to disease state) from the *therapeutic escape* (from a disease state under appropriate treatment to the other disease state). We then further separate the risk by disease and treatment, so that for any treatment ℓ and any state $s = (m, \zeta, u)$, one has $\lambda^\ell(s) = \lambda_m^\ell(\zeta, u)$, where the form of λ_m^ℓ is specified in Table 1. For the *standard relapse*, the risk μ_i for disease i was chosen as piecewise increasing linear functions calibrated such that the risk of relapsing increases until some duration τ_1 (average of standard relapses occurrences), then remains constant, and further increases between say τ_2 and τ_3 years (to model late or non-relapsing patients). This function and corresponding density are illustrated in

Table 1: Risk function of the controlled continuous time PDMP.

	$\ell = \emptyset$	$\ell = a$	$\ell = b$
$m = 0$	$\lambda_m^\ell(\zeta, u) = (\mu_1 + \mu_2)(u)$	$\lambda_m^\ell(\zeta, u) = \mu_2(u)$	$\lambda_m^\ell(\zeta, u) = \mu_1(u)$
$m = 1$	$\lambda_m^\ell(\zeta, u) = 0$	$\lambda_m^\ell(\zeta, u) = \mu'(\zeta)$	$\lambda_m^\ell(\zeta, u) = 0$
$m = 2$	$\lambda_m^\ell(\zeta, u) = 0$	$\lambda_m^\ell(\zeta, u) = 0$	$\lambda_m^\ell(\zeta, u) = \mu'(\zeta)$
$m = 3$	$\lambda_m^\ell(D, u) = 0$	$\lambda_m^\ell(D, u) = 0$	$\lambda_m^\ell(D, u) = 0$

Table 2: Flow Φ of the controlled continuous time PDMP. For any treatment ℓ , initial state $s = (m, \zeta, u)$, and duration t , $\Phi^\ell(m, \zeta, u, t) = (m, \Phi_m^\ell(\zeta, t), u + t)$ is the state of the patient after a time t starting from state s at time 0 if no change of condition or treatment occurred.

	$\ell = \emptyset$	$\ell = a$	$\ell = b$
$m = 0$	$\Phi_m^\ell(\zeta, t) = \zeta$	$\Phi_m^\ell(\zeta, t) = \zeta$	$\Phi_m^\ell(\zeta, t) = \zeta$
$m = 1$	$\Phi_m^\ell(\zeta, t) = \zeta e^{v_1^0 t}$	$\Phi_m^\ell(\zeta, t) = \zeta e^{-v_1' t} = \zeta e^{v_1^a t}$	$\Phi_m^\ell(\zeta, t) = \zeta e^{v_1^b t}$
$m = 2$	$\Phi_m^\ell(\zeta, t) = \zeta e^{v_2^0 t}$	$\Phi_m^\ell(\zeta, t) = \zeta e^{v_2^a t}$	$\Phi_m^\ell(\zeta, t) = \zeta e^{-v_2' t} = \zeta e^{v_2^b t}$
$m = 3$	$\Phi_m^\ell(\zeta, t) = \zeta$	$\Phi_m^\ell(\zeta, t) = \zeta$	$\Phi_m^\ell(\zeta, t) = \zeta$

Figure 5 for disease b .

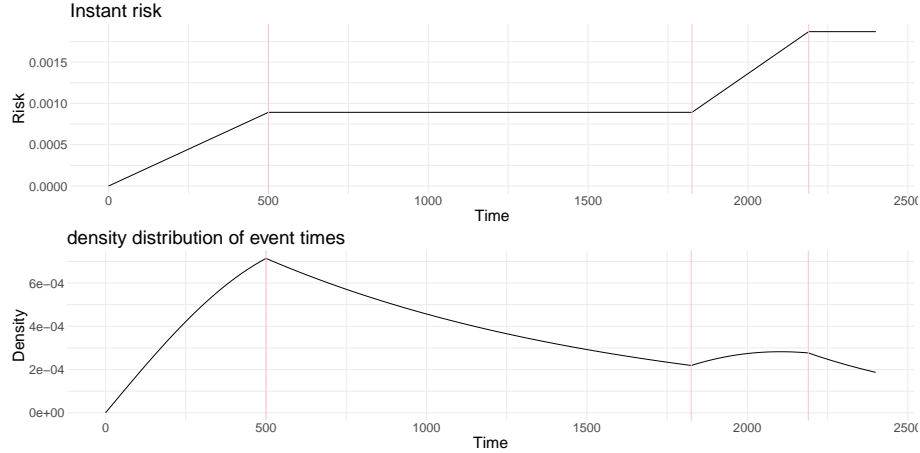


Figure 5: Risk and Density functions for standard relapse from remission condition to disease b condition (similar shapes for standard relapse to disease a).

For the therapeutic escape, we chose to fit a Weibull survival distribution of the form

$$\mu'(\zeta) = (\tilde{\beta}\zeta)^{\tilde{\alpha}},$$

with $-1 < \tilde{\alpha} < 0$ to account for a higher relapse risk when the marker decreases. We arbitrarily chose $\tilde{\alpha} = -0.8$ and calibrated $\tilde{\beta} = 1000$ such that only about 5% of patients experience a therapeutic escape.

The aggressiveness of the disease/treatment efficiency v/v' may depend on both treatment ℓ and mode m . Specific values are thus denoted by v_m^ℓ/v'_m , see Table 2. After setting aside patients that do not relapse (about 20%), we first estimated remission and relapse times using maximal slope difference, and then fitted exponential regression models on each segment. We then clustered

Table 3: Markov transition kernel Q for the controlled PDMP. matrix $Q_m^\ell(m')$ for the controlled PDMP. For any treatment ℓ and initial state $s = (m, \zeta, u)$ a jump sends the patient to state $s' = (m', \zeta', u')$ sampled from the distribution $Q(\cdot|s, \ell)$. The following constraints are satisfied: $\zeta' = \zeta$, $u' = 0$ and m' is sampled from the discrete distribution Q_m^ℓ .

$\ell = \emptyset$	
$m = 0$	$Q_m^\ell(m') = \mathbb{1}_{(m' \in \{1,2\})} \frac{\mu_{m'}(u)}{\mu_1(u) + \mu_2(u)}$
$m = 1$	$Q_m^\ell(m') = \mathbb{1}_{(m'=0)}$ (possible only if $\zeta = \zeta_0$)
	$Q_m^\ell(m') = \mathbb{1}_{(m'=3)}$ (possible only if $\zeta = D$)
$m = 2$	$Q_m^\ell(m') = \mathbb{1}_{(m'=0)}$ (possible only if $\zeta = \zeta_0$)
	$Q_m^\ell(m') = \mathbb{1}_{(m'=3)}$ (possible only if $\zeta = D$)
$\ell = a$	
$m = 0$	$Q_m^\ell(m') = \mathbb{1}_{(m'=2)}$
$m = 1$	$Q_m^\ell(m') = \mathbb{1}_{(m'=2)}$ (possible only if $\zeta > \zeta_0$)
	$Q_m^\ell(m') = \mathbb{1}_{(m'=0)}$ (possible only if $\zeta = \zeta_0$)
$m = 2$	$Q_m^\ell(m') = \mathbb{1}_{(m'=3)}$ (possible only if $\zeta = D$)
$\ell = b$	
$m = 0$	$Q_m^\ell(m') = \mathbb{1}_{(m'=1)}$
$m = 1$	$Q_m^\ell(m') = \mathbb{1}_{(m'=3)}$ (possible only if $\zeta = D$)
$m = 2$	$Q_m^\ell(m') = \mathbb{1}_{(m'=1)}$ (possible only if $\zeta > \zeta_0$)
	$Q_m^\ell(m') = \mathbb{1}_{(m'=0)}$ (possible only if $\zeta = \zeta_0$)

patients based on their relapse coefficient and chose the number of clusters using the slope heuristic on residual sum of squares. We obtained two groups and used the average values to obtain $v_1^\emptyset = 0.02$ (22% of patients) and $v_2^\emptyset = 0.006$ (78% of relapsing patients). We then computed the average of the treatment parameters for each group and obtained $v_1^a = 0.077$ and $v_2^a = 0.025$. The data do not present patient relapsing under treatment, and therefore we could not estimate v for therapeutic escapes or inappropriate treatments. Because we assume the aggressiveness in those circumstances should be smaller than under standard relapse, we chose $v_1^b = 0.01$ and $v_2^b = 0.003$.

By separating the risk λ by disease, the kernel function Q is automatically fitted, since we assume that the marker level does not jump at relapses, and the mode is selected by the risk clocks leading to the jump, whichever rings first, see Table 3. Finally, we arbitrarily selected a centered Gaussian distribution with noise parameter $\sigma^2 = 1$ for the observation process.

We resorted to extensive simulations study to select cost parameters that seemed reasonable (very few patients dying over our study horizon, on average not more than a fourth of the follow-up time spent under treatment, etc). We arbitrarily fixed the visit cost C_V to 1. We then fixed the death cost M to 110 so that with visits every 15 days and early relapse, a patient would rather die than spend the entire horizon under treatment with numerous visits. We then fixed $\beta = 0.1$ so that the penalty of applying an unnecessary treatment would count as 1.5, 3 or 6 times the visit cost depending on the choice of next visit date r . Finally, we selected $\kappa = 1/6$ from extensive simulations so that for low marker observations (typically when it is hard to decide between relapse or remission with high noise) it might be preferable to wait for new data acquisition rather than treat by default. All codes and parameters are available at <https://github.com/acleynen/pomcp4pdmp> [11].

4.2 Reminder on controlled PDMPs

Here is a description of how to simulate a trajectory of a controlled PDMP between two consecutive visits to the medical center. The controlled PDMP parameters are λ , the disease risk function (distribution of duration until the next jump i.e. condition change), Q , the Markov kernel, defining the stochastic transition to the state reached after the next jump and $\{v_m^\ell\}^2$, the parameters of the exponential deterministic behavior of the marker between two jumps, defined from the current mode and treatment applied.

Algorithm 1 Simulation of a trajectory between two consecutive decision times of a controlled PDMP.

```

1: procedure SIMULATEPDMP( $m, \zeta, u, \ell, r$ )
2:    $t \leftarrow 0$ 
3:    $v \leftarrow v_m^\ell$ 
4:   while  $t < r$  do
5:      $S \sim \lambda$ 
6:      $S \leftarrow \min\{S, t^*(m, \zeta, u, \ell)\}$ 
7:     if  $t + S > r$  then
8:       return  $m, \zeta \exp(vr), u + r$ 
9:     else
10:       $t \leftarrow t + S$ 
11:       $\zeta \leftarrow \zeta \exp(vS)$ 
12:       $u \leftarrow u + S$ 
13:       $m \sim Q(\cdot | m, \zeta, u, \ell)$ 
14:       $u \leftarrow 0$ 
15:       $v \leftarrow v_m^\ell$ 

```

Procedure SIMULATEPDMP takes as input an initial position $X_t = s = (m, \zeta, u)$ with mode m , marker level ζ , time since the last jump u , and a decision $d = (\ell, r)$ with treatment to be applied ℓ for a duration r until the next visit to the medical center and returns the state $X_{t+r} = s' = (m', \zeta', u')$ of the process at time $t + r$ given that treatment ℓ was applied. At line 5, $S \sim \lambda$ means that S is sampled from the distribution with risk function λ , which means that it has the following survival function

$$\mathbb{P}(S > t) = e^{-\int_0^t \lambda(m, \zeta \exp(v\tau), u + \tau) d\tau}.$$

At line 6, $t^*(m, \zeta, u, \ell)$ is the (deterministic) time to reach either the nominal value ζ_0 or the death level D from the current point (m, ζ, u) (if no change of condition or treatment occurs). The third variable u representing the time since the last jump allows transitions described in SIMULATEPDMP to be Markovian, i.e. to be independent of the previous transitions.

4.3 Reminder on Partially Observed Monte-Carlo Planning

In this section we give a description of the original POMCP algorithm³. The algorithm is called iteratively at each observation step n and requires several entries in order to output a decision d_n to be used by the operator. The inputs include a set of possible decisions (denoted A), a history

²For the sake of unified notation, we denote here $v_1^a = v_1'$, $v_2^b = v_2'$, and $v_0^\emptyset = v_0^a = v_0^b = 0$

³A Python implementation of the POMCP algorithm can be found here: <https://github.com/GeorgePik/POMCP>

$h_n = \langle \omega_0 d_0 \omega_1 \dots d_{n-1} \omega_n \rangle$ of successive observations and decisions up to step n , including the current observation ω_n , an approximate (particle) *filter* $\Theta^p(h_n)$ with support $B^p(h_n)$, a simulator $\mathcal{G}(s, d)$ of state and observation at step $n+1$ together with their cost given a state s and decision d at step n , a stopping criterion `TIMEOUT` and an arbitrary `ROLLOUT` strategy to provide a heuristic evaluation of an history, whenever needed.

Algorithm 2 Original POMCP algorithm [41]

<pre> 1: procedure POMCP(h_n) 2: repeat 3: if $h_n = \emptyset$ then 4: $s \sim \Theta_0^p$ 5: else 6: $s \sim \Theta^p(h_n)$ 7: SIMULATE(s, h_n) 8: until TIMEOUT() 9: $d^* \leftarrow \arg \min_d V(h_n d)$ 10: $V(h_n) \leftarrow \min_d V(h_n d)$ 11: return d^* </pre>	<pre> 18: procedure SIMULATE(s, h) 19: if $s.t \geq H$ then return 0 20: if $h \notin \mathcal{T}$ then 21: for all $d \in A$ do 22: $\mathcal{T}(hd) \leftarrow \langle N_{\text{init}}, V_{\text{init}}, \emptyset \rangle$ 23: $C \leftarrow \text{ROLLOUT}(s, h)$ 24: return C 25: $d^* \leftarrow \arg \min_d V(hd) - \alpha \sqrt{\frac{\log(N(h))}{N(hd)}}$ 26: $(s', \omega, c) \sim \mathcal{G}(s, d^*)$ 27: $C \leftarrow c + \text{SIMULATE}(s', hd^* \omega)$ 28: $B^p(h) \leftarrow B^p(h) \cup \{s\}$ 29: $N(h) \leftarrow N(h) + 1$ 30: $N(hd^*) \leftarrow N(hd^*) + 1$ 31: $V(hd^*) \leftarrow V(hd^*) + \frac{C - V(hd^*)}{N(hd^*)}$ 32: return C </pre>
<pre> 12: procedure ROLLOUT(s, h) 13: if $s.t = H$ then 14: return 0 15: $d \sim \pi_{\text{rollout}}(h)$ 16: $(s', \omega, c) \sim \mathcal{G}(s, d)$ 17: return $c + \text{ROLLOUT}(s', hd\omega)$ </pre>	

The POMCP algorithm involves several data structures:

- Simulated states $s = (m, \zeta, u)$.
- Decisions $d = (\ell, r)$, belonging to a finite decision space A , preferably small.
- Observations $\omega = (y, t)$. The original algorithm assumes that they belong to a finite observation space Ω of limited size.
- Histories $h = \langle \omega_0 d_0 \omega_1 d_1 \dots, d_{n-1} \omega_n d' \omega' d'' \omega'' \dots \rangle$. Histories represent sequences of decisions and observations of variable lengths. They are the concatenation of the sequence of past observations/decisions $h_n = \langle \omega_0 d_0 \omega_1 d_1 \dots d_{n-1} \omega_n \rangle$ plus an arbitrary sequence of *future* observations/decisions, built using the *rollout strategy* and a simulation model of the POMCP (line 28, $h \leftarrow hd^* \omega$).
- \mathcal{T} is a tree data structure rooted at the initial history h_n . Each node of \mathcal{T} will correspond to an extended history, ended by either a decision or an observation. Each simulation step creates a novel node in the tree, and histories h are attached to \mathcal{T} by appending the corresponding novel decision/observation to the parent node's history. In addition, we also attach to each node (now denoted $\mathcal{T}(h)$ or $\mathcal{T}(hd)$) (i) integer numbers $N(h)$ or $N(hd)$ where $N(h)$ corresponds to

the number of times the history h has been simulated and $N(hd)$ to the number of times d has been selected after h was encountered, (ii) real numbers $V(hd)$ corresponding to an estimate of the *cost value* of hd , that is the expected sum of future costs obtained if the optimal policy is applied after h has been encountered and d selected, until the final decision step and (iii) $\Theta^p(h)$, called a particle filter, which is a discrete uniform distribution on a set $B^p(h)$ of states s compatible with the current history h .

When SIMULATE is called with entry a history h that already belongs to \mathcal{T} , it updates the values of $N(h)$ and $B^p(h)$ as well as the values $N(hd)$ and $V(hd)$ for all the successor nodes⁴ hd . When SIMULATE is called with entry a history h which does not yet belong to \mathcal{T} (as is the case initially for h_n), it appends h as well as all its successor nodes hd to \mathcal{T} and initializes their values $N(h)$, $B^p(h)$, $N(hd)$ and $V(hd)$.

Procedure SIMULATE is based on a *generator function*, $(s', \omega, c) \sim \mathcal{G}(s, d)$ that generates a successor (hidden) state s' , an observation ω and an immediate cost c , from decision d applied in current (hidden) state s . Repeated calls to \mathcal{G} are used to progressively expand \mathcal{T} .

Simulation sequences and updates are performed, starting from h_n , until TIMEOUT() function requires to stop (generally, after an arbitrary number n_{search} of trajectories have been simulated or a fixed amount of time has been spent). Then, the decision d^* which maximizes $V(h_n d^*)$ is applied to the real-world system, and a real-life observation $\omega \in \Omega$ is obtained. The new real-world history becomes $h_{n+1} = h_n d^* \omega$ and \mathcal{T} is pruned, so that the new tree is rooted⁵ in h_{n+1} .

POMCP proposes strategies to select the input rollout and filters when the user has no knowledge on the process. A typical rollout strategy may simply involve selecting the decisions randomly from the set A . The following particle filter update procedure, included in procedure SIMULATE, is suggested:

- If $h = \emptyset$, sample $s \sim \Theta_0^p$. In the initial step of the algorithm, we simulate random particles from an arbitrary belief state.
- If $h \neq \emptyset$, sample $s \sim \Theta^p(h)$ where $\Theta^p(h)$ is the uniform discrete distribution on the finite nonempty set $B^p(h)$. Indeed, if $h \neq \emptyset$, this means that procedure SIMULATE(s', h) has already been called at least once for some s' and thus $B^p(h) \neq \emptyset$ ($B^p(h)$ contains at least s').
- Sample $(s', \omega, c) \sim \mathcal{G}(s, d^*)$. This step is performed in line 27 of the POMCP algorithm.
- if $|\omega - \omega'| = 0$, update $B^p(hd^* \omega) \leftarrow B^p(hd^* \omega) \cup \{s'\}$.

As the number of samples increases, the supports of the filters $B^p(h)$ will contain more and more particles and converge to the empirical distribution $\mathbb{P}(\cdot|h)$ of hidden states given the observed trajectory h . When the state space is finite, $\Theta^p(h)$ can be seen as an histogram approximation of $P(\cdot|h)$.

4.4 Adapted POMCP algorithm to the case of controlled PDMPs

It seems natural to apply a POMCP algorithm to optimize a PDMP control strategy in the context of disease control. Indeed, PDMPs have natural *generator* functions since algorithm SIMULATEPDMP

⁴It is a property of the algorithm that whenever $h \in \mathcal{T}$, $hd \in \mathcal{T}$ as well.

⁵The interest of pruning \mathcal{T} instead of starting with an empty tree in h_{n+1} is to exploit past simulations in the computation of the next decision.

can be naturally augmented with an observation simulator and a cost function, in order to obtain a generator function \mathcal{G} as described above. We propose three adaptations of the original POMCP algorithm to exploit the particular framework of controlled PDMPs.

Rollout policies The original POMCP algorithm describes the possible rollout policies as *admissible* policies, meaning that actions choices should only depend on the history of past actions and observations. Instead, we exploit here the PDMP generator which provides both hidden states and observations. This allows to design rollout policies exploiting hidden states instead of noisy observations. In our medical framework, one may build interesting *rollout policies* exploiting the hidden mode of the disease to provide good heuristics to the simulation part of POMCP. In practice, we propose to compare the two following rollout policies:

1. The (admissible) uniform policy : $\pi_{\text{unif}}(\omega = (y, t)) \sim \mathcal{U}(\{\emptyset, a, b\} \times \{15, 30, 60\})$
2. The (non-admissible) mode policy : $\pi_{\text{mode}}(s = (m, \zeta, u)) = \pi_{\text{mode}}(m) = \begin{cases} \{\emptyset, 15\} & \text{if } m = 0, \\ \{a, 15\} & \text{if } m = 1, \\ \{b, 15\} & \text{if } m = 2. \end{cases}$

The mode policy, being based on the full observation of the process, is likely to underestimate the real cost of an optimal control policy, which is a useful property for the convergence of a heuristic search method [38].

In our simulation study we observed that a mode-based rollout policy can be particularly efficient.

Observation space The time, state and observation spaces of the disease control PDMP model are continuous. This means that the probability of simulating exactly the same history h twice is zero if we apply the POMCP procedure as such. Thus, the tree depth and the size of filters supports $B^p(h)$ may never exceed 1. This is particularly annoying since the POMCP algorithm convergence proof only holds when $\Theta(h)$ is close to the true empirical belief state, which (approximately) holds when the size of the support $B^p(h)$ tends to $+\infty$. Indeed, the POMCP procedure in [41] requires that $B^p(h)$ contains at least K particles, when h is non empty. In practice, for the controlled PDMP case, observations are made of pairs (y, t) of a continuous-value observed marker level and discrete time of current decision. Therefore, we discretize the observation space into a set of contiguous intervals and group together observations belonging to the same interval.

The continuous nature of the model also prevents the exact computation of the filter, hence we resort to the use of particle or conditional filters. The construction of the former is slightly adapted from the initial POMCP algorithm to fit our model. Indeed, as the true process still produces continuous-valued observations, the last action of the particle filter update procedure is modified to updating $B^p(hd^*\omega)$ with s' only if $|\omega - \omega'| < \mathcal{D}$ (with \mathcal{D} chosen by the user, typically of the same magnitude as the discretization precision). It may still happen that this procedure selects only states with wrong mode m at step n (i.e. $\Theta^p(h_n)$, which is only an approximation of the true filter, does not contain the true hidden mode m), and hence cannot generate compatible states at step $n + 1$ due to diverging dynamics of the process in the different modes. This is not a simple matter of statistical accuracy, but a very practical problem. When this happens, we say that $B^p(h_{n+1})$ is *deprived* of particles and we cannot go on applying POMCP to h_{n+1} .

This problem of particle deprivation can be mitigated by a few modifications of the standard particle filter construction:

- Assume that $B^p(h_{n+1})$ is empty or too small, whatever the number of simulations of $s_n \sim B^p(h_n)$ followed by a call to $\mathcal{G}(s_n, d_n^*)$ we perform. Then, we may go back in the history and resimulate $s_{n-1} \sim B^p(h_{n-1})$ followed by two successive calls to \mathcal{G} :

- $(s_n, \omega', c') \leftarrow \mathcal{G}(s_{n-1}, d_{n-1}^*)$ and, provided that $|\omega' - \omega_n| < \mathcal{D}$,
- $(s_{n+1}, \omega'', c'') \leftarrow \mathcal{G}(s_n, d_n^*)$, hoping that now, $|\omega'' - \omega_{n+1}| < \mathcal{D}$.

A particle s_{n+1} is then added to $B^p(h_{n+1})$ whenever the two above conditions are met.

- Assume that $B^p(h_{n+1})$ is non-empty but still too small ($|B^p(h_{n+1})| \ll K$) after the previous step was applied a large number of times. We may perform particle revigoration by resampling particles from $B^p(h_{n+1})$ and duplicate them.
- Finally, when everything fails, we may perform a large number of sampled transitions from $B^p(h_n)$ (e.g. $1000 \times K$) and keep the K particles s' in the generated samples (s', ω', c') with minimal distance $|\omega_{n+1} - \omega'|$, with some arbitrary distance definition.

In our experimental studies, we applied these three modifications in turn, whenever needed, until we got belief states $B^p(h_n)$ of cardinality at least K .

Filters We propose to modify the original *particle filter* of POMCP to incorporate the *conditional filter*. Hence the algorithm is modified as follows: starting from an initial arbitrary belief filter Θ_0^c for $h_0 = \emptyset$ as in the original POMCP, when the new history becomes $h_{n+1} = h_n d^* \omega$, we compute the new filter Θ_{n+1}^c as a deterministic function of Θ_n^c and d^*, ω (see [10] for its specific form) and sample K particles from Θ_{n+1}^c to generate a set of plausible hidden states.

Depending only on the current belief and the new observation (and not on simulations), this filter does not suffer from the propagation of approximations and particle deprivation. Moreover, the computational burden of the simulations is replaced by the computation of weighted sums which are particularly efficient in matrix programming languages.

5 Supporting information

5.1 State of the art

Artificial Intelligence in medicine The use of artificial intelligence methods in medicine has recently exploded, as was shown for example in [27]. Yet the vast majority of these works focus on diagnosis and prognosis, rather than treatment and follow-up. There are a few studies related to treatment of diseases, though. [7] provides a review of approaches to cancer treatment focused on medical decision support. In the context of *drug design* for cancer treatment, (deep) Reinforcement Learning (RL) approaches have been proposed recently [33, 34], but with the disadvantage of being black-box approaches, preventing the practitioner from access to an explainable model of disease evolution and treatment. Uncertainty quantification in models of assisted decision making have been proposed to alleviate this problem [4]. In the same line of work, [5] advocate learning of mechanistic models of cancer evolution/treatment in order to help decision making. However, the latter approach requires a large amount of data *prior* to applying any treatment action, in order to learn a model that may prove only partially valid as decisions influence the disease dynamics. The authors advocate that the approach may be rendered more efficient if learning phases are interleaved with actual decision phases.

An alternative view of controlled PDMPs Controlled PDMPs can be modelled as continuous space POMDPs, in the way we proposed in this paper. However, they can also be seen as a particular subclass of continuous-time (Partially Observed) *Semi-Markov Decision Processes* [23]. Fully-observed continuous-time Semi-Markov Decision Processes extend Markov Decision Processes by including random continuous durations of state transitions and by considering that decisions can only be made at transition times. Several reinforcement learning solution approaches have been proposed, both in the fully-observed [8, 19] and partially-observed [20] cases to solve these problems. An alternative approach to ours could be to cast the PDMP model of cancer treatment into the continuous-time SMDP model and look for specializations of the existing simulation based solution algorithms.

5.2 Supplementary simulation results on POMCP parameters

Here we provide raw results for a series of parameter we tried to tune to optimize the POMCP algorithm, but that did not seem to bring additional improvement in our framework. Table 4 shows the impact of the trade off parameter α' in a few scenarios for the particle filter.

Table 4: Raw results for the particle filter varying parameters α' , n_{search} and K . For each parameter set, $n = 500$ trajectories were simulated. The Value column is the average cost of the trajectories over the n trajectories, and $\hat{\sigma}$ its empirical variance. We also recorded the runtime of optimizing each trajectory (duration column).

Filter	π_{rollout}	n_{search}	K	α'	Value	$1.96\hat{\sigma}/\sqrt{n}$	duration	duration s.d
particle	π_{mode}	100	100	0.2	161.93	14.79	1730	982
particle	π_{mode}	100	100	0.5	156.01	13.01	1629	913
particle	π_{mode}	100	100	0.8	165.63	13.51	1714	1053
particle	π_{mode}	100	100	0.99	147.24	6.17	1641	971
particle	π_{mode}	100	500	0.2	141.10	9.83	2727	683
particle	π_{mode}	100	500	0.5	141.56	4.72	2771	699
particle	π_{mode}	100	500	0.8	134.98	4.44	2646	685
particle	π_{mode}	100	500	0.99	133.57	3.56	2640	611
particle	π_{mode}	500	100	0.2	146.30	8.27	4617	660
particle	π_{mode}	500	100	0.5	146.82	11.91	4047	670
particle	π_{mode}	500	100	0.8	145.87	13.01	3708	610
particle	π_{mode}	500	100	0.99	140.91	6.35	3454	614
particle	π_{mode}	500	500	0.2	135.99	4.00	5182	598
particle	π_{mode}	500	500	0.5	132.88	4.25	5068	643
particle	π_{mode}	500	500	0.8	136.14	8.08	5271	698
particle	π_{mode}	500	500	0.99	129.42	5.06	4946	724

To allow adaptive selection of the trade off parameter c we tried three dynamic procedures to exploit or explore more depending on our trust in the current patient state. To do so, we define the state entropy as $E_t = \sum_{m=0}^2 p_m \log(p_m)$, $p_j = \sum_{s=(m,\zeta,u) \in B} \mathbf{1}\{m = j\}$ and $E_{\text{max}} = \log(1/3)$ and consider the following:

- entropy: $\alpha_t = E_t/E_{\text{max}}$

- rev-entropy: $\alpha_t = 1 - E_t/E_{\max}$
- rev-entropy-2: $\alpha_t = 1 - E_t/2E_{\max}$.

However, none of those procedures improved the results, as illustrated in selected examples in Table 5.

Table 5: Simulation results with adaptative choice of the exploration/exploitation parameter α' . For each parameter set, $n = 500$ trajectories were simulated. The Value column is the average cost of the trajectories over the n trajectories, and $\hat{\sigma}$ its empirical variance. We also recorded the runtime of optimizing each trajectory (duration column).

Filter	n_{search}	α'	Value	$1.96\hat{\sigma}/\sqrt{n}$	duration
conditional	100	entropy	138.63	5.06	776
conditional	100	rev-entropy	131.94	3.70	786
conditional	100	rev-entropy-2	133.75	3.70	770
particles	100	entropy	142.17	9.88	2421
particles	100	rev-entropy	143.27	10.34	2473
particles	100	rev-entropy-2	135.28	3.82	2438
conditional	1000	entropy	131.78	4.70	8313
conditional	1000	rev-entropy	131.41	3.45	8432
conditional	1000	rev-entropy-2	132.73	3.56	8332
particles	1000	entropy	133.39	3.74	9994
particles	1000	rev-entropy	135.64	3.74	10047
particles	1000	rev-entropy-2	131.89	4.24	10028

References

1. Anthony Almudevar. A dynamic programming algorithm for the optimal control of piecewise deterministic markov processes. *SIAM Journal on Control and Optimization*, 40(2):525–539, 2001.
2. Michel Attal, Valerie Lauwers-Cances, Cyrille Hulin, Xavier Leleu, Denis Caillot, Martine Escoffre, Bertrand Arnulf, Margaret Macro, Karim Belhadj, Laurent Garderet, et al. Lenalidomide, bortezomib, and dexamethasone with transplantation for myeloma. *New England Journal of Medicine*, 376(14):1311–1320, 2017.
3. Nicole Bäuerle and Ulrich Rieder. *Markov decision processes with applications to finance*. Springer Science & Business Media, 2011.

4. Edmon Begoli, Tanmoy Bhattacharya, and Dimitri Kusnezov. The need for uncertainty quantification in machine-assisted medical decision making. *Nature Machine Intelligence*, 1(1):20–23, 2019.
5. Sébastien Benzekry. Artificial intelligence and mechanistic modeling for clinical decision making in oncology. *Clinical Pharmacology & Therapeutics*, 108(3):471–486, 2020.
6. Dimitri Bertsekas and John N Tsitsiklis. *Neuro-dynamic programming*. Athena Scientific, 1996.
7. Bhavneet Bhinder, Coryandar Gilvary, Neel S Madhukar, and Olivier Elemento. Artificial intelligence in cancer research and precision medicine. *Cancer discovery*, 11(4):900–915, 2021.
8. S.J Bradtke and M.O. Duff. Reinforcement learning methods for continuous-time markov decision problems. In *Advances in neural information processing systems*, pages 393–400, 1995.
9. Alice Cleynen and Benoîte de Saporta. Change-point detection for piecewise deterministic markov processes. *Automatica J. IFAC*, 797:234–247, 2018. arXiv:1709.09467, hal-01596670.
10. Alice Cleynen and Benoîte de Saporta. Numerical method to solve impulse control problems for partially observed piecewise deterministic markov processes. *arXiv preprint arXiv:2112.09408*, 2023.
11. Alice Cleynen and Aymar Thierry d’Argenlieu. Pomcp4pomdp, implementation of pomcp algorithm adapted for pdmp models, 2023.
12. Bertrand Cloez, Renaud Dessalles, Alexandre Genadot, Florent Malrieu, Aline Marguet, and Romain Yvinec. Probabilistic and piecewise deterministic models in biology. *ESAIM: Proceedings and Surveys*, 60:225–245, 2017.
13. Mark HA Davis. Piecewise-deterministic markov processes: A general class of non-diffusion stochastic models. *Journal of the Royal Statistical Society: Series B (Methodological)*, 46(3):353–376, 1984.
14. MHA. Davis. *Markov models and optimization*, volume 49 of *Monographs on Statistics and Applied Probability*. Chapman & Hall, London, 1993.
15. Benoîte de Saporta, François Dufour, and Huilong Zhang. *Numerical methods for simulation and optimization of piecewise deterministic Markov processes: application to reliability*. John Wiley & Sons, 2015.
16. Pierre Del Moral. Non linear filtering: Interacting particle solution. *Markov Processes and Related Fields*, 2:555–580, 03 1996.
17. MAH Dempster and JJ Ye. Impulse control of piecewise deterministic markov processes. *The Annals of Applied Probability*, pages 399–423, 1995.
18. Oswaldo Luiz do Valle Costa and François Dufour. *Continuous average control of piecewise deterministic Markov processes*. Springer, 2013.

19. K. Doya. Reinforcement learning in continuous time and space. *Neural computation*, 12(1):219–245, 2000.
20. Jianzhun Du, Joseph Futoma, and Finale Doshi-Velez. Model-based reinforcement learning for semi-markov decision processes with neural odes, 2020.
21. Christian Gerecke, Stephan Fuhrmann, Susanne Striffler, Martin Schmidt-Hieber, Hermann Einsele, and Stefan Knop. The diagnosis and treatment of multiple myeloma. *Deutsches Ärzteblatt International*, 113(27-28):470, 2016.
22. Daniel T Gillespie. Exact stochastic simulation of coupled chemical reactions. *The journal of physical chemistry*, 81(25):2340–2361, 1977.
23. Ron Howard. Semi-markovian decision processes. *Bulletin de l’Institut International de Statistique*, 40(1), 1964.
24. Sabrina Hundt, Ulrike Haug, and Hermann Brenner. Blood markers for early detection of colorectal cancer: a systematic review. *Cancer Epidemiology Biomarkers & Prevention*, 16(10):1935–1953, 2007.
25. Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.
26. Sammie Katt, Frans A Oliehoek, and Christopher Amato. Learning in pomdps with monte carlo tree search. In *International Conference on Machine Learning*, pages 1819–1827. PMLR, 2017.
27. Yogesh Kumar, Apeksha Koul, Ruchi Singla, and Muhammad Fazal Ijaz. Artificial intelligence in disease diagnosis: a systematic literature review, synthesizing framework and future research agenda. *Journal of Ambient Intelligence and Humanized Computing*, pages 1–28, 2022.
28. Vincent Lemaire, Michèle Thiéullen, and Nicolas Thomas. Thinning and multilevel Monte Carlo methods for piecewise deterministic (Markov) processes with an application to a stochastic Morris-Lecar model. *Adv. in Appl. Probab.*, 52(1):138–172, 2020.
29. Sagar Lonial, Lawrence H Boise, and Jonathan Kaufman. How i treat high-risk myeloma. *Blood, The Journal of the American Society of Hematology*, 126(13):1536–1543, 2015.
30. Amelie M Lutz, Juergen K Willmann, Frank V Cochran, Pritha Ray, and Sanjiv S Gambhir. Cancer screening: a mathematical model relating secreted blood biomarker levels to tumor sizes. *PLoS medicine*, 5(8):e170, 2008.
31. Madhav Nagpal, Shreya Singh, Pranshu Singh, Pallavi Chauhan, and Meesam Abbas Zaidi. Tumor markers: A diagnostic tool. *National journal of maxillofacial surgery*, 7(1):17, 2016.
32. Chiara Nicolò, Cynthia Périer, Melanie Prague, Carine Bellera, Gaëtan MacGrogan, Olivier Saut, and Sébastien Benzekry. Machine learning and mechanistic modeling for prediction of metastatic relapse in early-stage breast cancer. *JCO clinical cancer informatics*, 4:259–274, 2020.

33. Marcus Olivecrona, Thomas Blaschke, Ola Engkvist, and Hongming Chen. Molecular de-novo design through deep reinforcement learning. *Journal of cheminformatics*, 9(1):1–14, 2017.
34. Mariya Popova, Olexandr Isayev, and Alexander Tropsha. Deep reinforcement learning for de novo drug design. *Science advances*, 4(7):eaap7885, 2018.
35. Vincent Renault, Michèle Thioullien, and Emmanuel Trélat. Optimal control of infinite-dimensional piecewise deterministic Markov processes and application to the control of neuronal dynamics via optogenetics. *Netw. Heterog. Media*, 12(3):417–459, 2017.
36. Martin G. Riedler, Michèle Thioullien, and Gilles Wainrib. Limit theorems for infinite-dimensional piecewise deterministic Markov processes. Applications to stochastic excitable membrane models. *Electron. J. Probab.*, 17:no. 55, 48, 2012.
37. Ryszard Rudnicki and Marta Tyran-Kamińska. *Piecewise deterministic processes in biological models*, volume 1. Springer, 2017.
38. Stuart Russel, Peter Norvig, et al. *Artificial intelligence: a modern approach*, volume 256. Pearson Education Limited London, 2013.
39. Anne-Sofie Schrohl, Mads Holten-Andersen, Fred Sweep, Manfred Schmitt, Nadia Harbeck, John Foekens, and Nils Brunner. Tumor markers: from laboratory to clinical utility. *Molecular & Cellular Proteomics*, 2(6):378–387, 2003.
40. S Sharma. Tumor markers in clinical practice: General principles and guidelines. *Indian journal of medical and paediatric oncology*, 30(01):1–8, 2009.
41. David Silver and Joel Veness. Monte-carlo planning in large pomdps. *Advances in neural information processing systems*, 23, 2010.
42. Gabriella Sozzi, Davide Conte, Luigi Mariani, Salvatore Lo Vullo, Luca Roz, Claudia Lombardo, Marco A Pierotti, and Luca Tavecchio. Analysis of circulating tumor dna in plasma at diagnosis and during follow-up of lung cancer patients. *Cancer research*, 61(12):4675–4678, 2001.
43. A Keith Stewart, Paul G Richardson, and Jesus F San-Miguel. How i treat multiple myeloma in younger patients. *Blood, The Journal of the American Society of Hematology*, 114(27):5436–5443, 2009.
44. Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.
45. Jiangping Xu, Guillermo Vilanova, and Hector Gomez. A mathematical model coupling tumor growth and angiogenesis. *PloS one*, 11(2):e0149422, 2016.