Strategic Conflict in Labor Market Separations

A Game-Theoretic Exploration of Hostility, Uncertainty, and Retention Decisions

Marcus Anthony Lisman

Advisor: Professor W. Bentley MacLeod

Submitted in partial fulfillment of the requirements for the Bachelor of Science in Computer Science and Economics

Department of Computer Science and Economics

Yale University

1 May 2025

Abstract

This thesis explores the persistence of dismissal conflicts in labor markets through a dynamic game-theoretic and computational lens. Despite the theoretical efficiency of mutually beneficial separation mechanisms, real-world employment relationships frequently devolve into adversarial dismissals. Drawing on contract theory, behavioral economics, and recent empirical insights, this study introduces a formal model to examine how strategic hostility, incomplete information, and belief updating jointly sustain inefficient separations over time. The core framework features a firm repeatedly deciding whether to retain or dismiss a worker whose productivity evolves stochastically via a two-state Markov process. A key innovation is the modeling of hostility as an endogenous cost: When severance falls short of the worker's private cost of dismissal, the firm incurs an additional non-pecuniary penalty. To account for the bounded rationality of firms and the uncertainty about future productivity and conflict risks, the thesis implements a Q-learning algorithm to simulate the evolution of policy under imperfect information. Through comparison with dynamic programming benchmarks, the study shows that firms with limited information often retain unproductive matches longer or adopt firing strategies that underperform socially optimal policies. Hostility significantly reduces firing rates, alters equilibrium behavior, widens the gap between firm profit and social surplus, and benefits the long-term benefits of the worker. Reinforcement learning experiments reveal how updating beliefs about productivity persistence and hostility risk shape the long-term dynamics of separation decisions. This research contributes theoretically by formalizing the interaction between strategic conflict and learning and methodologically by applying reinforcement learning to model labor market behavior. It also offers practical implications for policy design aimed at mitigating adversarial dismissals, reducing informational frictions, and aligning severance structures with perceived separation costs to improve overall labor market efficiency. In general, the thesis deepens our understanding of why adversarial dismissals remain widespread despite the availability of more efficient alternatives.

Acknowledgements

This thesis is dedicated to my family, friends, thesis advisor, the CSEC department, professors, administrators, and everyone else who made this opportunity possible. Thank you for making the last four years at Yale so special. I will take all of the memories and knowledge with me as I begin the next chapter of my life.

A special thanks to Professor W. Bentley MacLeod for his mentorship and guidance through my last semester at Yale while serving as my advisor for this thesis. Without his expertise and feedback, this project would not have been possible.

And to my parents Herman and Clara Lisman and my brother Matthew Lisman, thank you for always supporting my endeavors and for giving me the opportunities to succeed. I love you.

Contents

1 Introduction

Economic relationships are often shaped by strategic interactions between individuals, firms, and institutions. Although economic theory frequently assumes that rational actors seek to maximize utility within well-defined market structures, real-world interactions are often characterized by frictions, conflicts, and deviations from theoretically efficient outcomes. Understanding these deviations is essential for developing more accurate models of economic behavior and informing policy interventions.

One domain where such inefficiencies are particularly pronounced is the labor market, where conflicts between employers and employees arise in various forms. Labor market separations—ranging from voluntary resignations to dismissals—constitute critical decision points that influence firm productivity, worker welfare, and overall market efficiency. Despite the availability of more cooperative separation mechanisms, adversarial dismissals remain prevalent. The persistence of conflict in these interactions raises fundamental questions about the incentives, beliefs, and constraints that shape the behavior of the firm and the worker.

This thesis explores the strategic determinants of conflict in economic relationships, with a particular focus on labor market separations. Using a game-theoretic framework reinforced by computational modeling and empirical insights, this study aims to uncover the underlying drivers of dismissal conflicts and assess why firms and workers fail to adopt more efficient separation mechanisms. By integrating concepts from contract theory, behavioral economics, and reinforcement learning, this project contributes to a deeper understanding of how economic agents navigate uncertainty, negotiate outcomes, and respond to institutional constraints.

In examining these issues, this study advances theoretical perspectives on labor market frictions and provides practical insights for policymakers and firms seeking to mitigate conflict and improve labor market efficiency. The following sections will provide some background, review the relevant literature, and formally outline the research questions and the methodological approach undertaken in this thesis.

1.1 Background

Labor market separations are pivotal decision points that shape employment dynamics, firm profitability, and worker welfare. Although classical economic models emphasize voluntary separations and rational bargaining processes that yield efficient outcomes, real-world labor markets frequently exhibit adversarial dismissals, litigation, and strategic delays in separation decisions. These frictions signal deeper inefficiencies that persist even in environments where mutual gains from cooperation are theoretically possible.

Over the past several decades, employment protection laws have been implemented in various forms to balance flexibility for employers with security for workers. In the United States, wrongful discharge doctrines have emerged through judicial interpretation, introducing legal uncertainty and increasing the potential costs of firing[1]. In contrast, many European countries impose formalized severance payments, advance notice requirements, or contractual protections against unjust dismissal[2]. These protections, while intended to enhance job security and promote fairness, can inadvertently distort labor market behavior, leading firms to adjust production techniques, delay separations, or rely more heavily on precarious employment arrangements[3].

The persistence of adversarial separations despite the availability of cost-minimizing alternatives—such as Separations by Mutual Agreement (SMA)—raises a central puzzle. Empirical work such as Carry and Schoefer's *Conflict in Dismissals*[4] documents that firms and workers often engage in costly legal disputes and prolonged conflicts instead of pursuing cooperative exits. This pattern suggests that separations are influenced not only by observable costs and benefits, but also by less visible forces such as institutional rigidity, behavioral frictions, asymmetric beliefs about legal outcomes and the cost of unemployment, and emotional or strategic hostility between parties.

Hostility, in particular, may arise when one party perceives the separation as unfair or insufficiently compensated, resulting in non-monetary disutility and retaliatory behavior. Such responses may impose further costs on the firm—either directly through legal confrontation or indirectly through reputational damage and internal morale effects. Yet despite these risks, firms may still pursue contentious separations, suggesting that hostility itself may be endogenous to the employment relationship and strategically anticipated.

Furthermore, recent theoretical and computational work underscores how frictions in-

[1]David H Autor, William R Kerr, and Adriana D Kugler. *Do Employment Protections Reduce Productivity? Evidence from U.S. States*. Working Paper 12860. National Bureau of Economic Research, 2007. DOI: 10.3386/w12860. URL: http://www.nber.org/papers/w12860.

[2]Edward P. Lazear. "Job Security Provisions and Employment". In: *The Quarterly Journal of Economics* 105.3 (1990), pp. 699–726. ISSN: 00335533, 15314650. URL: http://www.jstor.org/stable/2937895 (visited on 04/20/2025); Stephen Nickell. "Unemployment and Labor Market Rigidities: Europe versus North America". In: *The Journal of Economic Perspectives* 11.3 (1997), pp. 55–74. ISSN: 08953309. URL: http://www.jstor.org/stable/2138184 (visited on 04/20/2025).

[3]Olivier Blanchard and Augustin Landier. "The Perverse Effects of Partial Labour Market Reform: Fixed-Term Contracts in France". In: *The Economic Journal* 112.480 (2002), F214–F244. ISSN: 00130133, 14680297. URL: http://www.jstor.org/stable/798373 (visited on 04/20/2025); Giuseppe Bertola. "Labor Turnover Costs and Average Labor Demand". In: *Journal of Labor Economics* 10.4 (1992), pp. 389–411. URL: https://EconPapers.repec.org/RePEc:ucp:jlabec:v:10:y:1992:i:4:p:389-411.

[4]Pauline Carry and Benjamin Schoefer. *Conflict in Dismissals*. Working Paper 33245. National Bureau of Economic Research, 2024. DOI: 10.3386/w33245.

troduced by employment protection laws may fundamentally alter firm incentives[5]. Legal structures can increase firms' perceived risks of separation, incentivizing them to preemptively avoid commitments or retain unproductive matches longer than optimal. Strategic considerations—such as deterring opportunism, preserving discipline, or avoiding reputational damage—also may play a role, complicating the assumption that agents will always act to minimize direct costs[6].

This thesis builds on these insights by developing a game-theoretic and reinforcement learning model of labor market separations. The model seeks to explain how conflict persists in separations and under what conditions firms opt for adversarial dismissals over cooperative agreements. By incorporating strategic incentives, belief updating, and endogenous hostility, the model offers a dynamic framework to understand why efficient separations remain underutilized in practice and what role hostility actually plays in the worker-firm relationship.

[5]Fernando Alvarez and Marcelo Veracierto. "Severance payments in an economy with frictions". In: *Journal of Monetary Economics* 47.3 (2001), pp. 477–498. URL: https://ideas.repec.org/a/eee/moneco/v47y2001i3p477-498.html; Lars Ljungqvist. "How Do Lay-off Costs Affect Employment?" In: *The Economic Journal* 112.482 (2002), pp. 829–853. ISSN: 00130133, 14680297. URL: http://www.jstor.org/stable/798534 (visited on 04/20/2025).

[6]W. Bentley MacLeod. *Great Expectations: Law, Employment Contracts, and Labor Market Performance*. Working Paper 16048. National Bureau of Economic Research, 2010. DOI: 10.3386/w16048. URL: http://www.nber.org/papers/w16048; James M. Malcomson. "Contracts, Hold-Up, and Labor Markets". In: *Journal of Economic Literature* 35.4 (1997), pp. 1916–1957. ISSN: 00220515. URL: http://www.jstor.org/stable/2729883 (visited on 04/20/2025).

1.2 Literature Review

Understanding the persistence of dismissal conflicts in labor markets requires integrating insights from game theory, labor economics, and behavioral economics. A growing body of research has examined strategic decision-making in employment separations, the role of relational contracts, and the dynamics of conflict resolution.[7]Despite significant empirical and theoretical advances, there are important gaps in explaining why firms and workers persist in costly adversarial dismissals instead of adopting mutually beneficial agreements. Standard models often assume frictionless separation or efficient bargaining, but real-world labor markets display persistent inefficiencies and adversarial separations that defy these predictions. This thesis builds on existing literature by formalizing dismissal conflicts through a game-theoretic model that incorporates key behavioral and strategic frictions, allowing for a deeper understanding of how these conflicts evolve over time.

A foundational study in this area is Carry and Schoefer's *Conflict in Dismissals*[8], which provides empirical evidence on the prevalence and drivers of conflictual separations in the French labor market. They find that only 12% of the potential dismissals are resolved through a mutually agreed-upon severance offer called a Separation by Mutual Agreement (SMA), with the remaining 88% involving adversarial separations driven by three primary factors: **hostility between employers and employees**, **firms' use of dismissals as a discipline device**, and **asymmetric beliefs about legal outcomes after litigation**. These findings challenge conventional economic models that assume cost-minimizing behavior and efficient bargaining, instead highlighting the role of behavioral frictions and strategic considerations in shaping separation decisions. Additionally, it introduces the role of hostility as a non-pecuniary but strategic cost. However, while their study provides robust empirical evidence, it lacks a formal theoretical framework to explain why these frictions persist across interactions and over time. This gap motivates the need for a structured model that explicitly incorporates learning dynamics, hostility costs, and institutional constraints.

The role of implicit agreements and long-term relationships has been extensively explored in the literature on relational contracts. Brown, Falk, and Fehr[9] show that in the absence of third-party enforcement, stable employment relationships emerge endogenously and sustain cooperation through informal mechanisms such as rent-sharing and the threat of termination. These findings align with MacLeod and Malcomson's foundational models[10], which

[7]W. Bentley MacLeod and James M. Malcomson. "Implicit Contracts, Incentive Compatibility, and Involuntary Unemployment". In: *Econometrica* 57.2 (1989), pp. 447–480. ISSN: 00129682, 14680262. URL: http://www.jstor.org/stable/1912562 (visited on 04/20/2025); Martin Brown, Armin Falk, and Ernst Fehr. "Relational Contracts and the Nature of Market Interactions". In: *Econometrica* 72.3 (2004), pp. 747–780. ISSN: 00129682, 14680262. URL: http://www.jstor.org/stable/3598834 (visited on 04/20/2025); Pedro Dal Bó and Guillaume R. Fréchette. "On the Determinants of Cooperation in Infinitely Repeated Games: A Survey". In: *Journal of Economic Literature* 56.1 (2018), pp. 60–114. DOI: 10.1257/jel.20160980. URL: https://www.aeaweb.org/articles?id=10.1257/jel.20160980; Carry and Schoefer, *Conflict in Dismissals*.

[8]Carry and Schoefer, *Conflict in Dismissals*.

[9]Brown, Falk, and Fehr, "Relational Contracts and the Nature of Market Interactions".

[10]MacLeod and Malcomson, "Implicit Contracts, Incentive Compatibility, and Involuntary Unemployment"; W. Bentley MacLeod and James M. Malcomson. *Implicit Contracts, Incentive Compatibility, and Involuntary Unemployment: Thirty Years On.* Tech. rep. IZA - Institute of Labor Economics, 2023. URL: http://www.jstor.org/stable/resrep65865 (visited on 04/20/2025).

explain how incomplete contracts and asymmetric information lead to wage rigidity and involuntary unemployment. However, even though these models illuminate the strategic basis of cooperation and breakdowns, they do not directly address the persistence of conflictual separations in the presence of viable cooperative alternatives.

Theoretical perspectives on dismissal costs further complicate the standard intuition that severance policies always inhibit flexibility by considering institutional factors that shape dismissal behavior. Lazear[11] presents a canonical model showing that severance pay reduces firm hiring more than firing, leading to higher unemployment. Nickell[12] expands this perspective through cross-continental comparison, finding that Europe's stronger employment protections are associated with higher structural unemployment than in North America. However, these rigidities are not uniformly inefficient. Alvarez and Veracierto[13] demonstrate that when labor market frictions are taken into account, severance payments can improve welfare and reduce volatility of unemployment and excessive churn by dampening inefficient separations. Bertola[14] refines this view by showing that turnover costs can increase average employment due to rational firm behavior under uncertainty. These results underscore that the effects of firing costs depend heavily on institutional context, bargaining structure, and firm strategy — considerations central to this thesis.

Complementing these models are more recent empirical studies on the effects of employment protection. Autor et al.[15] show that stricter wrongful discharge laws in U.S. states reduce firm entry and productivity growth, as firms adjust by avoiding risky employment matches. Acharya, Baghai, and Subramanian[16] support this with evidence from the industry that stronger dismissal protections promote innovation by alleviating the risk of employee miscarriage. These papers underscore a central tension: employment protections may create inefficiencies in some domains while promoting long-term value creation in others. Deciding which bucket a firm falls into depends largely on institutional levers such as severance or the dictating legal environment.

Hostility as an endogenous cost has received less attention in formal models but plays a critical role in explaining conflictual separations. Carry and Schoefer[17] estimate that more than half of dismissals would convert to SMAs in the absence of hostility alone. This behavioral channel is consistent with evidence from organizational studies, such as MacLeod, Valle Lara, and Zehnder[18], which show that conflict, when institutionalized, can serve as a strategic enforcement mechanism, but can also escalate inefficiencies when mismanaged.

[11] Lazear, "Job Security Provisions and Employment".

[12] Nickell, "Unemployment and Labor Market Rigidities: Europe versus North America".

[13] Alvarez and Veracierto, "Severance payments in an economy with frictions".

[14] Bertola, "Labor Turnover Costs and Average Labor Demand".

[15] Autor, Kerr, and Kugler, *Do Employment Protections Reduce Productivity? Evidence from U.S. States*.

[16] Viral V. Acharya, Ramin P. Baghai, and Krishnamurthy V. Subramanian. "Wrongful Discharge Laws and Innovation". In: *The Review of Financial Studies* 27.1 (2014), pp. 301–346. ISSN: 08939454, 14657368. URL: http://www.jstor.org/stable/24464827 (visited on 04/20/2025).

[17] Carry and Schoefer, *Conflict in Dismissals*.

[18] W. Bentley MacLeod, Victoria Valle Lara, and Christian Zehnder. "Worker Empowerment and Subjective Evaluation: On Building an Effective Conflict Culture". In: *Management Science* (2024), pp. 1–26. DOI: 10.1287/mnsc.2022.03085. eprint: https://doi.org/10.1287/mnsc.2022.03085. URL: https://doi.org/10.1287/mnsc.2022.03085.

Halac[19] further explores how firms may strategically invest in relational incentives to sustain cooperation, but such arrangements are vulnerable to power imbalances that fuel conflict. These insights support the idea that hostility is not merely a psychological residue but a strategic outcome shaped by institutional design and historical interaction patterns. These insights justify modeling hostility as a strategic and endogenous cost arising from the failure to sustain relational equilibrium rather than just as a preference shock.

The persistence of conflict in repeated economic relationships is further illuminated by the experimental and theoretical literature on infinitely repeated games. Dal Bó and Fréchette[20] survey an extensive body of experiments and find that while cooperation is theoretically sustainable under certain discount factors, it often fails due to strategic uncertainty and coordination failure. Abreu, Pearce, and Milgrom[21] emphasize the destabilizing effect of imperfect information transmission in repeated partnerships, showing how delays or noise in feedback can erode trust as players fail to update beliefs in synchrony. These results directly apply to employment separations, where firms and workers often lack common knowledge about the state of the firm's productivity and how that might change, legal risks, reputational costs, or future bargaining prospects. As such, conflicts may persist even in environments where efficient cooperation is theoretically feasible.

MacLeod and Malcomson's broader work on contracts and hold-up problems[22] highlights how incomplete enforceability leads to strategic underinvestment and adversarial behavior. Autor[23] complements this theoretical insight with evidence from U.S. states, showing that stronger employment protections can reduce firm entry but also increase investment in worker-firm relationships. Acharya et al.[24] extend this idea to innovation, showing that wrongful discharge protections reduce hold-up risk and stimulate innovation by increasing workers' bargaining power and job security. Together, these studies reinforce the idea that conflict and cooperation are not merely opposites but rather are dynamic responses to the institutional rules of the game.

Additional contributions from empirical labor economics provide context for these theoretical developments. Blanchard and Landier[25] show that partial labor market reforms in France led to increased use of fixed-term contracts and employment instability, highlighting the unintended consequences of incomplete institutional design. Messina and Vallanti[26] find that layoff costs can dampen hiring incentives and exacerbate job protection dualism.

[19] Marina Halac. "Investing in a relationship". In: *The RAND Journal of Economics* 46.1 (2015), pp. 165–185. ISSN: 07416261. URL: http://www.jstor.org/stable/43895586 (visited on 04/20/2025).

[20] Dal Bó and Fréchette, "On the Determinants of Cooperation in Infinitely Repeated Games: A Survey".

[21] Dilip Abreu, Paul Milgrom, and David Pearce. "Information and Timing in Repeated Partnerships". In: *Econometrica* 59.6 (1991), pp. 1713–1733. ISSN: 00129682, 14680262. URL: http://www.jstor.org/stable/2938286 (visited on 04/20/2025).

[22] W. Bentley MacLeod and James M. Malcomson. "Investments, Holdup, and the Form of Market Contracts". In: *The American Economic Review* 83.4 (1993), pp. 811–837. ISSN: 00028282. URL: http://www.jstor.org/stable/2117580 (visited on 04/27/2025).

[23] Autor, Kerr, and Kugler, *Do Employment Protections Reduce Productivity? Evidence from U.S. States*.

[24] Acharya, Baghai, and Subramanian, "Wrongful Discharge Laws and Innovation".

[25] Blanchard and Landier, "The Perverse Effects of Partial Labour Market Reform: Fixed-Term Contracts in France".

[26] Julian Messina and Giovanna Vallanti. "Job Flow Dynamics and Firing Restrictions: Evidence from Europe". In: *Economic Journal* 117.521 (2007), pp. 279–301. URL: https://EconPapers.repec.org/RePEc:ecj:econjl:v:117:y:2007:i:521:p:279-301.

Meanwhile, Autor, Kerr, and Kugler[27] demonstrate that employment protection laws reduce total factor productivity by distorting firm-level adjustment decisions. These results echo the broader concern that institutional friction, once introduced, can produce path-dependent conflict dynamics and support the need for a model that explains how conflict persists and how it shapes firm responses to those frictions.

Finally, the use of reinforcement learning (RL) in strategic contexts also opens new theoretical ground. To capture how firms and workers adapt over time, this thesis leverages reinforcement learning. Reinforcement learning offers a powerful framework to model these dynamics of adaptation, intuition, and bounded rationality. Leslie and Collins[28] show that agents using Q-learning in multi-agent games often fail to converge to Nash equilibrium, particularly when opponents are also adapting. This suggests that firms and workers, through trial and error, may repeatedly select suboptimal strategies, reinforcing conflictual behavior as both sides update beliefs and adjust strategies iteratively. This thesis draws inspiration from Halac's exploration of time-inconsistent contracts[29], and from the recent advances in applying RL to inverse problems[30]. By allowing for strategic experimentation, learning from past interactions, and belief updating about unknown parameters such as worker hostility or productivity persistence, reinforcement learning allows us to simulate the evolution of separation strategies in a dynamic labor market.

Taken together, this literature highlights key drivers of dismissal conflicts—ranging from relational contracting and strategic uncertainty to adaptive learning and institutionalized hostility. On one hand, employment separations are shaped by relational contracts, legal frameworks, and strategic signaling. On the other, they evolve dynamically as firms and workers learn from past outcomes, misjudge adversarial costs, or respond to institutional incentives. This thesis bridges these concepts by developing a model of labor separations that integrates strategic decision-making, reinforcement learning, and institutional constraints to explain why firms and workers persist in adversarial separations despite the availability of more efficient alternatives. By formalizing these dynamics and allowing for asymmetric beliefs, endogenous hostility, and incomplete knowledge of productivity, this thesis advances theoretical perspectives on labor market frictions. It also provides practical insights for policymakers and organizations seeking to mitigate conflict and its associated costs, improve employment outcomes, and create a more economically efficient environment.

[27] Autor, Kerr, and Kugler, *Do Employment Protections Reduce Productivity? Evidence from U.S. States*.

[28] David Leslie and Edmund Collins. "Individual Q -Learning in Normal Form Games". In: *SIAM J. Control and Optimization* 44 (Jan. 2005), pp. 495–514. DOI: 10.1137/S0363012903437976.

[29] Halac, "Investing in a relationship".

[30] Divyansh Garg et al. "IQ-Learn: Inverse soft-Q Learning for Imitation". In: *CoRR* abs/2106.12142 (2021). arXiv: 2106.12142. URL: https://arxiv.org/abs/2106.12142.

1.3 Explanation and Motivation of Research Questions

Despite the theoretical efficiency of cost-minimizing employment separations, real-world labor markets frequently exhibit persistent conflicts in termination decisions. Traditional models suggest that firms should fire low-productivity workers when separation costs are manageable and retain them when not. Additionally, these models predict that rational agents will opt for mutually beneficial separation agreements, but empirical evidence[31] indicates that firms and workers often choose more adversarial and costly dismissal processes. This clean separation logic breaks down when firms face uncertainty, both about the future and about the costs of separation, such as worker hostility. In such settings, separation behavior becomes more complex, path-dependent, and shaped by belief updating rather than perfect foresight.

This thesis seeks to explain why conflict persists even in relatively simple labor environments by focusing on three key forces: (1) **endogenous hostility**, (2) **uncertain productivity transitions**, and (3) **learning dynamics in the absence of full information**. We posit that firms do not always know ex ante how long a low-productivity state will last or whether a fired worker will impose additional unobservable costs (e.g., retaliation, litigation, reputational harm). In this environment, optimal separation decisions cannot be derived from static optimization alone—they must emerge from a learning process that incorporates both immediate costs and anticipated future consequences. Additionally, there is a lack of a comprehensive framework that explains how these factors interact dynamically, especially the cost of hostility, and why firms and workers repeatedly engage in costly conflict instead of adapting toward more cooperative outcomes. Understanding the mechanisms that sustain adversarial separations is crucial for both theoretical labor economics and policy design, as persistent dismissal conflicts can lead to inefficiencies in job mobility, increased litigation costs, and negative externalities for firms and workers.

To formalize these ideas, this thesis develops a dynamic model in which a firm employs a worker whose productivity follows a two-state Markov process, $\alpha_t \in \{H, L\}$, where H represents high productivity and L low. At each period, the firm decides whether to retain or fire the worker. If the worker is fired, the firm must pay severance $\bar{s}$, but if $\bar{s} < k$, where k is the worker's private cost of being fired, the firm incurs an additional **hostility cost** equal to $k - \bar{s}$. Importantly, this cost is endogenous and arises only when the separation is perceived as unjust or insufficiently compensated.

Hostility in this framework is not a random shock but a strategic consequence of undercompensation. This concept aligns with the empirical findings of Carry and Schoefer[32], who estimate that hostility alone accounts for a significant share of missed opportunities for efficient separation. It is also supported by experimental work on relational breakdowns and conflict norms that shows that conflict can emerge from frayed expectations rather than outright misbehavior.

In addition to hostility, firms also face **uncertainty about productivity persistence**. Since productivity evolves according to a Markov process with unknown transition probabilities p, a firm observing a low-productivity state must decide whether to retain the worker in hopes of recovery or fire and endure separation costs. The firm may not know in advance

[31] Carry and Schoefer, *Conflict in Dismissals*.
[32] Ibid.

how persistent the low state is, and it must learn through repeated interactions whether the expected duration of unproductivity justifies the cost of waiting.

To capture these learning dynamics, this thesis embeds the firm's decision problem in a Q-learning framework. The firm begins without full knowledge of the productivity transition matrix or the distribution of worker types (i.e., the likelihood that $k > \bar{s}$), and must learn an optimal firing policy through repeated interaction. Over time, the firm refines its estimates of (a) the expected duration of low productivity, and (b) the expected cost of hostility, leading to evolving separation behavior. This reinforcement learning framework allows us to study optimal policy in a static environment and how suboptimal behaviors may persist or emerge due to noisy experience and bounded rationality[33].

Therefore, the main research question this thesis addresses is:

> **How does hostility and uncertainty about future productivity affect the firm's separation policy?**

This central question gives rise to the following sub-questions:

- **Under what conditions does the firm retain a worker in the low state L despite negative expected profit?** We analyze how the threat of hostility and the anticipated duration of the low state affect this decision.
- **How does uncertainty about the Markov transition probability p influence the firm's policy over time?** We simulate cases where the firm does not know how long it will remain in L and must learn whether waiting is worth the cost.
- **How do informational regimes (full knowledge vs. partial knowledge) influence the alignment between firm profit, firing rates, and social surplus?** We compare the firm's learned policies under each regime to benchmark outcomes under dynamic programming with complete information.

In addressing these questions, this thesis contributes a formal and computational account of how dismissal decisions evolve in uncertain, frictional environments. It bridges the empirical observation of inefficient separations with a theoretical model grounded in learning, strategic retention, and conflict costs. By highlighting the role of endogenous hostility and belief formation, this study offers new insight into why firms may delay separation, even when the current state suggests firing is optimal, and how suboptimal retention or conflict-prone behavior can persist over time.

Beyond its theoretical contributions, this study holds important policy implications. If dismissal conflicts are driven by structural factors such as outcome uncertainty or entrenched hostility, then interventions aimed at reducing these frictions, such as clearer legal frameworks, alternative dispute resolution mechanisms, or incentive-aligned severance policies, may help mitigate adversarial separations. By formally characterizing the strategic incentives underlying dismissal behavior, this thesis offers insights that can inform labor market policies aimed at improving employment relations and reducing inefficiencies.

[33] Leslie and Collins, "Individual Q -Learning in Normal Form Games"; Garg et al., "IQ-Learn: Inverse soft-Q Learning for Imitation".

1.4 Significance of the Study

This thesis makes theoretical and methodological contributions to the study of labor market separations by offering an explanation for the persistence of dismissal conflicts in the presence of seemingly superior and more efficient alternatives. Although the existing literature has examined employment protections, strategic separations, and relational contracts, few models explicitly capture how firms learn over time under uncertainty and how endogenous hostility can shape separation policy in a dynamic environment.

From a theoretical standpoint, this thesis advances the literature by formalizing a dynamic, micro-founded model that integrates endogenous non-monetary conflict costs with productivity transitions governed by a Markov process. It introduces a novel mechanism—hostility triggered by under-compensated separations—that directly affects the firm's incentives to fire. This mechanism shifts the analytical focus away from static cost-benefit calculations toward dynamic belief updating, where firms must weigh current losses against uncertain future gains while learning about the costs of separation through experience. By doing so, it extends the application of game theory to dynamic labor market settings, offering a structured explanation for why dismissal conflicts persist even when cooperative alternatives are available.

In the field of labor economics, this study deepens the analysis of employment separations by formally characterizing how hostility and uncertain beliefs about future outcomes shape firm-worker interactions. Previous research has primarily examined dismissal conflicts through empirical lenses, identifying key drivers but lacking a unifying theoretical model. By bridging this gap, this study enhances the understanding of labor market frictions and contributes to the broader literature on employment relationships, contract theory, and dispute resolution mechanisms.

Methodologically, the use of reinforcement learning, specifically Q-learning, allows the model to simulate firm behavior in informational environments where critical parameters are unknown or only partially observable. This framework captures bounded rationality and strategic experimentation in a way that traditional dynamic programming models with perfect foresight cannot. By varying the informational regime, the model evaluates how firms adapt, how misjudgments emerge, and how policy misalignment can persist despite incentives for efficiency.

This study also contributes conceptually by reframing dismissal conflicts as a problem of dynamic miscoordination rather than simple legal or moral failure. The findings suggest that persistent conflict does not necessarily stem from irrationality or malice, but from experience-driven learning under uncertainty about productivity paths and separation outcomes. Hostility, in this view, is not noise—it is a strategic and interpretable signal within the employment relationship.

Beyond its theoretical contributions, this research offers practical insights for labor policy and organizational design. If inefficiencies in separation behavior arise due to uncertainty and endogenous conflict costs, policy interventions might aim to reduce ambiguity in productivity forecasting (e.g., through feedback systems or performance signals) or to minimize hostility through well-designed severance structures and dispute resolution frameworks. Moreover, the model provides a quantitative foundation for evaluating how interventions like severance guarantees or mediation procedures might influence firm retention strategies and long-run employment efficiency. A reinforcement learning model may help policymakers anticipate and address behavioral trends in employment separations, leading to more effective labor policies that promote stability and efficiency in workplace relations.

By connecting a computational learning framework with labor market frictions, this thesis bridges the gap between observed dismissal behavior and formal economic modeling. Its findings contribute to both academic scholarship and practical policy design by offering insights that may help firms, workers, and policymakers in managing employment separations. Its contributions lie in explaining why efficient separation mechanisms remain underutilized and in suggesting how better information, clearer expectations, and experience-aware institutions might move labor markets closer to efficient, mutually beneficial outcomes.

1.5 Scope and Limitations

This thesis focuses on understanding the persistence of dismissal conflicts in labor markets through a game-theoretic and reinforcement learning framework. It aims to explain why firms and workers continue to engage in adversarial separations despite the availability of more cooperative alternatives. The research primarily examines labor market separations where strategic interactions between firms and workers play a central role.

The scope of this study is limited to individual dismissal decisions rather than collective layoffs or broader macroeconomic labor trends. While economic downturns, industry-specific shocks, and large-scale labor market regulations undoubtedly influence dismissal patterns, the focus here is on firm-worker interactions at the micro level. By isolating these interactions, the study seeks to provide a detailed theoretical and computational understanding of the mechanisms that drive dismissal conflicts.

Methodologically, this study employs a combination of game theory and reinforcement learning to model how firms and workers adapt their separation strategies over time. However, this approach entails certain limitations. First, while reinforcement learning allows for an examination of strategic adaptation, it relies on simulated decision-making rather than direct empirical validation through observed firm-worker interactions. The results remain theoretical and require future empirical testing for full validation.

Another limitation concerns data availability. The study does not rely on proprietary firm-level dismissal records or direct labor court case data, which could provide a richer empirical basis for testing the model. Instead, it draws on existing empirical studies[34], which document key drivers of dismissal conflicts but do not offer real-time decision-making data. As a result, while the model can replicate observed patterns, it does not offer direct empirical estimations of firm-specific dismissal strategies.

Additionally, the study does not explicitly incorporate cultural, psychological, or institutional variations in labor market regulations across different countries. While dismissal conflicts occur in various legal and economic contexts, this research assumes a generalizable strategic framework applicable to developed labor markets with formal dismissal procedures. Future research could extend this model by incorporating country-specific labor laws, social norms, and union dynamics.

This thesis also assumes that firms and workers are boundedly rational agents who update their strategies based on past experiences. However, it does not explicitly model external influences such as third-party mediators, legal consultants, or government interventions in dismissal disputes. The inclusion of such external factors could refine the model's predictions but is beyond the current study's scope.

Moreover, the modeling framework used in this thesis assumes a stylized two-state Markov process to represent worker productivity. This abstraction is useful for isolating the impact of productivity persistence on firm decision-making, but it does not capture the full richness of productivity dynamics in real-world labor markets. In particular, the binary nature of productivity states and the exogenous structure of transition probabilities limit the model's applicability to environments where productivity evolves along more continuous or multidimensional paths. Future research could extend the model to incorporate richer stochastic processes or endogenous effort decisions by workers.

[34] Carry and Schoefer, *Conflict in Dismissals*.

The reinforcement learning implementation centered around Q-learning relies on episodic simulation of firm behavior under varied parameter settings. While this allows for flexible experimentation and policy comparison across informational regimes, it also introduces sensitivity to learning rate parameters, episode length, and convergence criteria. These elements, while manageable in a simulated environment, may differ substantially from real-world organizational learning processes. Nevertheless, this approach enables structured counterfactual analysis by comparing learned firm policies under various levels of information. The methodology thus provides a flexible foundation for exploring how imperfect information and adaptive behavior influence the persistence of conflictual separations.

Despite these limitations, the study provides valuable theoretical insights into the strategic persistence of dismissal conflicts and offers a structured approach for future empirical and policy-oriented research.

1.6 Thesis Structure

This thesis is organized into the following sections:

- **Theoretical Framework** — This section introduces a dynamic model of the employment relationship, where the firm decides whether to retain or fire a worker whose productivity follows a two-state Markov process. The model incorporates endogenous hostility costs, rigid severance payments, and uncertainty about productivity transitions.

- **Assumptions and Simplifications** — This section lays out the key modeling assumptions used to formalize the decision problem. These include the nature of the Markov process, the rigidity of wages, the structure of severance payments, and the definition and consequences of hostility. The limitations and justifications of each assumption are discussed in relation to the existing literature.

- **Computational and Experimental Approach** — This section outlines the reinforcement learning methodology and the implementation of Q-learning to simulate firm behavior under various informational regimes. It also introduces a suite of experiments involving parameter sweeps, simulations under full and partial information, and comparisons to analytically derived dynamic programming benchmarks.

- **Results** — This section presents the main findings of the computational model, focusing on the conditions under which firms retain workers in low-productivity states, the impact of unknown productivity persistence on policy learning, and the long-run effects of hostility on firing decisions. Quantitative results are visualized through plots showing firm behavior, payoff trends, and surplus outcomes across parameter variations.

- **Discussion and Future Extensions** — This section interprets the results in the context of broader economic theory and policy. It discusses the implications of learned firing policies, the gap between firm profit and social surplus, and how miscalibrated beliefs can entrench inefficient separations. Possible extensions include continuous-state productivity, endogenous wage offers, multi-agent learning, or integration with empirical data.

- **Conclusion** — This section summarizes the core insights of the thesis and reiterates its contribution to the theory of labor market frictions. It highlights how endogenous hostility, uncertain productivity, and adaptive behavior combine to produce persistent conflict, even in stylized environments where cooperation is feasible.

- **Code** — This appendix includes the Julia code used to implement the Q-learning algorithm, run parameter sweeps, and generate the data visualizations shown in the Results section.

- **References** — This section provides references for the empirical and theoretical literature used throughout the thesis.

2 Theoretical Framework

In many labor markets, a firm is forced to consider whether to keep or fire a worker whose productivity fluctuates. Workers may suffer significant personal costs upon dismissal, but it is the *firm* that must absorb any hostility cost h if severance $\bar{s}$ is insufficient to cover the worker's cost k. Consequently, the firm's decision to fire a low-productivity worker is tempered by the possibility of facing hostility.

We develop a multi-period employment model in which a firm faces a two-state productivity process, $\alpha_t \in \{H, L\}$. This process continues *unaffected by firing*: that is, regardless of whether the firm fires or retains its worker, α_t simply evolves according to its Markov law. The firm pays a fixed (rigid) wage w, assumed at least as large as the worker's outside utility u_0 (and for simplicity we may take $w = u_0$). If the firm fires the worker, it must pay severance $\bar{s}$. If $\bar{s} < k$ (where k is the worker's own firing cost), the worker becomes *hostile*, imposing an additional cost $\big(k - \bar{s}\big)$ on the firm.

A key feature here is that the firm can only employ a worker (and earn $\alpha_t - w$) when $\alpha_t = H$ or L. If it fires the worker, it earns zero while it remains *unemployed* waiting for α to return to H. Thus, if the firm fires in the low state L, it forgoes all profit until the productivity state switches back to H. Turnover costs (severance and possible hostility) can therefore lead the firm to keep the worker even in low-productivity states if the alternative is too expensive.

We present the multi-period (and infinite-horizon) profit function for the firm, as well as a social-surplus function using discount factor δ. We also outline how one would implement a Q-learning (RL) approach when parameters are partially or fully unknown.

2.1 Model Setup

2.1.1 States and Markov Productivity

We have an infinite set of discrete periods $t = 0, 1, 2, \ldots$. Each period, productivity is

$$\alpha_t \in \{H, L\},$$

and employment status is

$$e_t \in \{E, U\}$$

and evolves according to a Markov process with transition parameter $p \in [0.5, 1]$:

$$\Pr(\alpha_{t+1} = \alpha_t) = p, \quad \Pr(\alpha_{t+1} \neq \alpha_t) = 1 - p.$$

The state is then defined as:

$$s_t = \{\alpha_t, e_t\}$$

Crucially, α_t does not reset upon firing. If the firm fires a worker when $\alpha_t = L$, the process may remain at L the next period with probability p, or flip to H with probability $1 - p$. The point is that firing does not alter α_{t+1}. This Markov process parameterizes both i.i.d. shocks and shocks with persistence. When there is persistence, then when in the low state L, the firm may not wish to keep the worker while the wage $w > L$. The Markov process can be seen schematically in Figure 1.

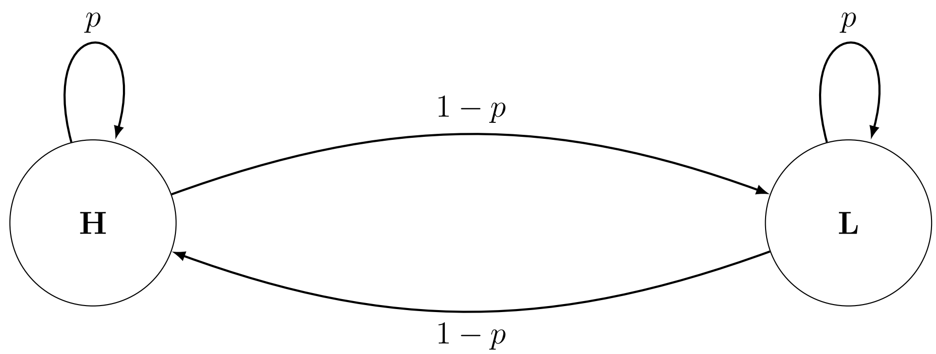

Figure 1: Markov productivity process with persistence parameter p. Productivity evolves independently of employment status, including after firing.

2.1.2 Wage, Outside Option, and Firing Cost k

- $\mathbf{w} \geq u_0$: A fixed (rigid) wage the firm must pay to an employed worker. Often we set $w = u_0$ for simplicity, mirroring a rigid or regulated wage floor.
- $\mathbf{k}$: The worker's personal cost if fired. If $\bar{s} < k$, there is a shortfall $\big(k - \bar{s}\big)$ that manifests as hostility toward the firm.

2.1.3 Severance $\bar{s}$ and Hostility Cost

If the firm fires the worker, it must pay severance $\bar{s}$. If $\bar{s} < k$, the worker's shortfall is $\big(k - \bar{s}\big)$, which we interpret as a hostility cost *on the firm*. Concretely,

$$h = \begin{cases} k - \bar{s}, & \text{if } \bar{s} < k, \\ 0, & \text{if } \bar{s} \geq k. \end{cases}$$

Thus, insufficient severance means the firm pays an extra penalty $\big(k - \bar{s}\big)$. That is, if severance meets or exceeds the worker's cost of job loss, no hostility arises. If severance is insufficient, the firm pays the price through hostility.

2.1.4 Employment vs. Unemployment (Waiting)

A crucial feature is that when the firm *fires*, it becomes *unemployed* (i.e. it does not hire another worker immediately). Instead, it must wait *until* α_t switches to H before it will hire again.[35] While waiting:

$$\text{Firm payoff} = 0.$$

Once $\alpha_t = H$ again, the firm hires a new worker at wage w.

[35] This captures the idea that the firm refuses to hire in state L because $\alpha - w$ would be negative or unprofitable if $L < w$. Hence, it earns zero payoff while it waits.

Thus, in a low state L, the firm faces a choice: keep the worker (earning $\alpha_t - w$) or fire and earn zero while unemployed (plus pay severance and hostility if applicable) until α flips to H. This employment dynamic over joint states (α_t, e_t) is illustrated in Figure 2.

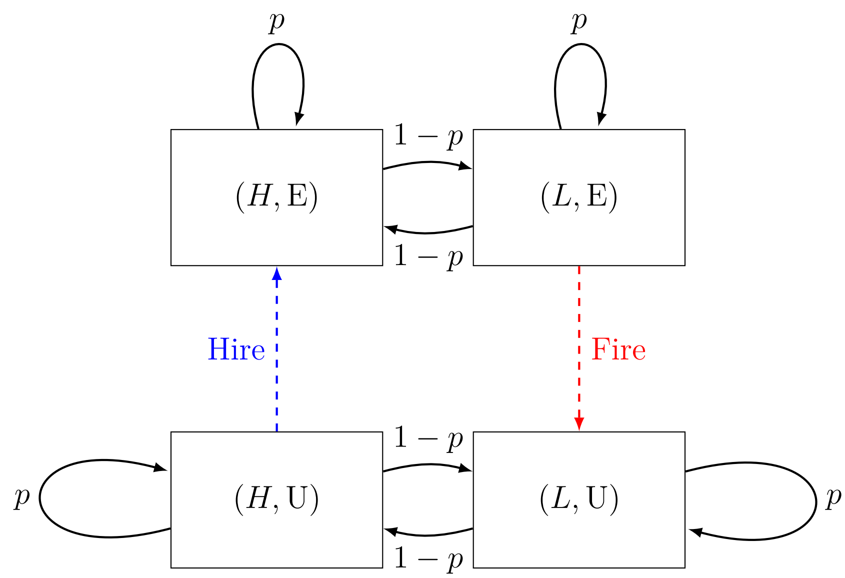

Figure 2: Joint state diagram over productivity $\alpha_t \in \{H, L\}$ and employment $e_t \in \{\mathrm{E}, \mathrm{U}\}$. Solid arrows indicate Markov productivity transitions; dashed arrows represent firm-controlled employment changes.

2.2 Per-Period Payoffs and Discounting

We let $\delta \in (0, 1)$ denote the discount factor for future payoffs. We can distinguish two payoff situations:

- **Employed**: If the firm *has* a worker in period t, its payoff is

$$\alpha_t - w.$$

  The worker's payoff is w.

- **Unemployed (waiting)**: If the firm has no worker, it earns zero. It can only hire again once $\alpha_t = H$.

When the firm decides to *fire* its current worker in period t, the immediate payoff includes the severance/hostility cost:

$$\text{Firm payoff at firing} = 0 \; - \; \max\{0, \; k - \bar{s}\} = - \; \max\{0, k - \bar{s}\}.$$

The firm then remains unemployed (earning zero) until it sees $\alpha = H$. Once $\alpha = H$, it hires and returns to "employed" status.

2.3 Multi-Period Profit Function

To write the dynamic program clearly, we define the value function $V(\alpha, e)$, where $\alpha \in \{H, L\}$ is the current productivity state and $e \in \{E, U\}$ is the employment status:

- $V(\alpha, E)$: Value when the firm is **employed** with a worker in state α.
- $V(\alpha, U)$: Value when the firm is **unemployed** in state α.

2.3.1 Employed Value $V(\alpha, E)$

If the firm is currently employing a worker and observes $\alpha_t = \alpha$, it chooses whether to keep or fire:

$$V(\alpha, E) = \max\Big\{(\alpha - w) + \delta\,\mathbb{E}[V(\alpha_{t+1}, E)],\; -\max\{0, k - \bar{s}\} + \delta\,\mathbb{E}[V(\alpha_{t+1}, U)]\Big\}.$$

Here:

- $\alpha - w$: Current period payoff if the firm keeps the worker.
- $\delta\,\mathbb{E}[V(\alpha_{t+1}, E)]$: Discounted future value if the worker is kept.
- $-\max\{0, k - \bar{s}\}$: Cost of firing, including hostility if severance is inadequate.
- $\delta\,\mathbb{E}[V(\alpha_{t+1}, U)]$: Value of being unemployed next period.

2.3.2 Unemployed Value $V(\alpha, U)$

When the firm is unemployed in state α:

$$V(\alpha, U) = \begin{cases} 0 + \delta\,\mathbb{E}[V(\alpha_{t+1}, U)], & \text{if } \alpha = L \text{ (no profitable hire)} \\ \max\Big\{\, 0 + \delta\,\mathbb{E}[V(\alpha_{t+1}, U)],\; (H - w) + \delta\,\mathbb{E}[V(\alpha_{t+1}, E)]\Big\}, & \text{if } \alpha = H \text{ (profitable hire)} \end{cases}$$

If $\alpha = L$, the firm remains unemployed and earns zero (since hiring someone at wage w with $\alpha = L$ might be strictly worse than zero). If $\alpha = H$, it can either remain unemployed (0 payoff) *or* hire a worker and earn $\big(H - w\big)$ this period plus $\delta\,\mathbb{E}[V_E(\alpha_{t+1})]$ going forward.

If $\alpha = H$, the firm can either:

- Remain unemployed and get $\delta\,\mathbb{E}[V(\alpha_{t+1}, U)]$, or
- Hire a worker and earn $(H - w) + \delta\,\mathbb{E}[V(\alpha_{t+1}, E)]$.

Typically, if $\alpha = H$, the firm hires immediately. These value functions $V(\alpha, e)$ fully characterize the firm's optimal policy over an infinite horizon.

2.4 Social Surplus: Discounted Welfare Computation

We define social surplus (or total welfare) as the discounted sum of

$$\big[\alpha_t - u_0\big]$$

whenever employed, minus any hostility cost if $\bar{s} < k$. That is, from a societal viewpoint, employing the worker in period t yields $\alpha_t - u_0$. If the firm is unemployed, we treat it as producing zero while the worker has fallback u_0 outside the firm. Additionally, if firing occurs with $\bar{s} < k$, we subtract $(k - \bar{s})$ as an efficiency loss. Formally:

$$\text{Social Surplus} = \sum_{t=0}^{\infty} \delta^t \Big[(\alpha_t - u_0) \cdot \mathbf{1}\{\text{employed}\} - H_t \Big],$$

where

$$H_t = \begin{cases} k - \bar{s}, & \text{if fired at } t \text{ and } \bar{s} < k, \\ 0, & \text{otherwise.} \end{cases}$$

Thus, a socially optimal policy might differ from the private one if the hostility cost is large.

2.5 Role of Hostility and Turnover Costs

2.5.1 Incentive to Retain a Low State

If $\alpha = L$ is below the wage w, retaining the worker may be unprofitable in the short run. In the absence of firing frictions, a profit-maximizing firm may dismiss the worker and rehire upon reversion to $\alpha = H$, particularly when p, the probability of remaining in the current productivity state, is moderate (e.g., $p = 0.5$).

However, if the firm anticipates that firing will trigger hostility—i.e., if $\bar{s} < k$ and thus the separation cost is higher than severance—it internalizes the full turnover cost of dismissal:

$$\max\{0,\, k - \bar{s}\} \quad + \quad \text{lost output until re-hiring.}$$

This forward-looking calculation may induce the firm to keep the worker even in the low state $\alpha = L$, waiting for a return to $\alpha = H$, particularly when productivity is not persistent. In such cases, the threat of hostility deters premature separation and improves surplus outcomes, even if hostility never actually occurs.

In contrast, when the firm lacks prior knowledge of the worker's firing cost k, it must learn through repeated interaction. Early in the reinforcement process, the firm may mistakenly fire in L without anticipating the hostility cost, leading to inefficient separations that would not occur under full information. This gap between anticipatory rational behavior and trial-and-error learning underscores the dual role of hostility: it is both a frictional cost and a potential enforcement mechanism that, when understood, steers behavior toward socially preferable, more efficient outcomes.

2.5.2 Worker's Firing Cost vs. Actual Implementation

While the worker's personal cost is k, in this model, the cost is *passed on to the firm* if severance is insufficient. That is, $\bar{s} < k \implies$ the firm pays $(k - \bar{s})$ in intangible or direct losses from a hostile departure. This arrangement can be interpreted as:

- Legal or reputational damage to the firm if it fires at a severance below the worker's legitimate cost k.

Hence hostility is a penalty on the firm for failing to meet the worker's firing cost in severance.

2.5.3 Large p: Persistent L

If $\alpha = L$ tends to stay L with high probability ($p \approx 1$), the firm may choose to fire and pay hostility if it believes staying in a prolonged low state is worse than waiting for the next H. Conversely, if p is more moderate, the firm might keep the worker in L in the face of potential hostility, hoping α flips to H soon, thereby avoiding a firing cost and lost time in unemployment.

2.5.4 The Disciplinary Role of Anticipated Hostility

One important conceptual point of this model is that *anticipated* hostility can prevent inefficient dismissals. Suppose productivity is currently low ($\alpha = L$), but the persistence parameter is $p = 0.5$—meaning productivity is equally likely to revert to H next period. In such an environment, a firm would favor firing in a frictionless environment, since the firm can regain high productivity before hiring again rather than dealing with a potential hostility cost.

If the firm knows that firing will incur a hostility penalty of $k - \bar{s}$, it has strong incentives to avoid dismissal in the low state, effectively internalizing the turnover externality. Thus, when $\bar{s} < k$, hostility operates as a forward-looking enforcement device: its threat deters premature separation and promotes efficient waiting for state recovery. In this sense, hostility can be welfare-improving, not by being incurred, but by being avoided due to its credible threat.

2.5.5 Learning and Inefficient Conflict

However, if the firm does not initially know the worker's cost k or the productivity transition structure, it must learn these parameters through experience. In early Q-learning episodes, which will be discussed shortly, the firm may misestimate the long-run value of retention and fire workers in L even when doing so is socially inefficient. When $\bar{s} < k$, such mistakes incur deadweight losses that could have been avoided with full information.

This learning dynamic creates a tradeoff. Hostility, though costly when triggered, can serve as an efficient deterrent if properly anticipated. However, in settings where the firm must learn hostility's cost through trial and error, the process of discovering this deterrent is itself inefficient. As a result, early-stage behavior in the model may reflect welfare losses that stem not from hostility per se, but from the firm's incomplete understanding of its consequences.

Therefore, while hostility introduces an ex post efficiency loss, it can also drive ex ante efficient behavior in the long run once firms have learned its strategic implications.

2.5.6 Parameter Sweeps in Hostility Context

- If $\bar{s}$ is always above k, no hostility arises. The firm can separate at will, making the model akin to frictionless firing.
- If $\bar{s} < k$, hostility triggers whenever the firm dismisses the worker.

2.6 Q-Learning Framework

2.6.1 Q-Learning Representation for Adaptive Beliefs

To model adaptation, one can employ a stylized Q-learning approach in which each player maintains an *action-value function* $Q(\text{action})$ representing the expected long-run payoff from taking a particular action in a given state. In order to understand why Q-learning can serve as an optimal method of adaptation, there is some important foundation to understand.

2.6.2 A Foundation of Q-Learning

While so far the focus has been primarily on theory, it can only take us so far, so utilizing computational tools can allow for problem-solving that escapes analysis. A powerful tool for learning to play games is *reinforcement learning* (RL), a form of machine learning where an algorithm participates in many iterations of the game and then the outcome of that game is used to update the RL agent's strategy. This process can be thought as trial-and-error to find out which actions within the state space are most valuable in various scenarios for the RL agent[36].

2.6.3 The Q-matrix

A particularly useful RL algorithm is Q-Learning, a model-free approach to learning. The core feature of the Q-learning algorithm is the *Q-matrix*: a table that stores the expected reward from taking each possible action in a given situation.

Formally, Let $\mathcal{Q}$ be a function that takes in world state $\mathcal{S}$ and the action space $\mathcal{A}$ and finds the "quality" (hence Q) of a given action. So

$$\mathcal{Q} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$$

Then, to select the optimal action, select the optimal action A^* that maximizes quality:

$$A^* = \arg\max_{\mathcal{A}} \ \mathcal{Q}(\mathcal{S}, \mathcal{A}) \tag{1}$$

We now illustrate how one can apply Q-learning to the model in which the firm can be either *employed* or *unemployed* and the productivity state $\alpha_t \in \{H, L\}$ evolves continuously,

[36] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. Second. The MIT Press, 2018. URL: http://incompleteideas.net/book/the-book-2nd.html.

unaffected by firing. The idea is that Q-learning can discover an optimal keep/fire (or hire/wait) policy without requiring explicit knowledge of transition probabilities or cost distributions.

2.6.4 Firm Learning Across Many Worker Types

We assume the firm is a single Q-agent that interacts with an environment over many episodes. Each episode might correspond to employing one worker (and potentially firing and rehiring later), with parameters (k, u_0) or a distribution thereof.

State Space A convenient representation of the firm's state at time t can be:

$$(\alpha_t,\ e_t)\ \in\ \{\,H, L\,\} \times \{\,E, U\,\},$$

where

- $\alpha_t \in \{H, L\}$ is the current productivity level.
- $e_t \in \{E, U\}$ is the employment status and indicates whether the firm is *employed* (has a worker) or *unemployed* (fired the last worker and has not yet rehired).

Actions

- If $e_t = E$ (employed) and $\alpha_t \in \{H, L\}$, the firm chooses either:

$$\text{keep} \quad \text{or} \quad \text{fire}.$$

- If $e_t = U$ (unemployed):
  - If $\alpha_t = L$, the firm typically remains unemployed (no profitable hire).
  - If $\alpha_t = H$, the firm chooses either hire or stay unemployed. (In most parameter settings, $\alpha = H$ leads to immediate hire.)

Rewards Following the model's per-period payoffs:

- **Employed in state** α: The reward is $\alpha - w$.
- **Fire action**: The reward is $-\max\{0,\ k - \bar{s}\}$ in the firing period. Afterwards, the firm transitions to $e = U$ for next period.
- **Unemployed (no worker)**: The reward is 0.
- **Hire action** (allowed if $\alpha = H$ and unemployed): The immediate reward is $\alpha - w$ in the hiring period, since the firm becomes employed that same period.

Dynamics The Markov chain for α continues with transition probability p. Thus:

$$\Pr(\alpha_{t+1} = \alpha_t) = p, \quad \Pr(\alpha_{t+1} \neq \alpha_t) = 1 - p.$$

The e evolves based on the firm's action: fire leads to U, hire leads to E, etc.

Q-Learning Update Define $Q(\alpha, e, a)$ as the Q-value for taking action a in state (α, e). The update after an observed reward r_t and next state (α_{t+1}, e_{t+1}) is:

$$Q_{\text{new}}(\alpha, e, a) \;=\; (1 - \alpha_Q)\, Q_{\text{old}}(\alpha, e, a) \;+\; \alpha_Q \left[r_t + \delta \max_{a'} Q_{\text{old}}(\alpha_{t+1}, e_{t+1}, a') \right]$$

Here, α_Q is the learning rate. By repeating episodes (and possibly randomizing (k, u_0) or other parameters across episodes), the firm learns an approximate policy for keep/fire/hire that maximizes its expected discounted payoff-even without explicit knowledge of p or distribution of k.

2.6.5 Parameter Sweeps

Once the Q-learning structure is in place, one can systematically vary parameters:

- $\mathbf{k}$: The worker's firing cost (from 0 to some upper bound).
- $\mathbf{q}$: The fraction of workers with high k.
- $\mathbf{p}$: The Markov parameter for α.
- $\mathbf{u_0}$: The worker's outside option (and possibly the wage).
- $\bar{\mathbf{s}}$: The severance level. Hostility arises if $\bar{s} < k$.

In each case, run Q-learning for many episodes:

- *Track how often* the firm fires while in L.
- *Measure how frequently* hostility costs arise.
- *Compute approximate profits* (like $V_E(H)$ or total returns).

This shows how hostility and waiting costs shape the firm's keep/fire policy under different parameter regimes.

2.7 Conclusion

In summary, Q-learning can be applied to the model where $\alpha \in \{H, L\}$ evolves independently of firing, and the firm endures a zero-payoff "unemployed" phase if it does fire. The critical elements—hostility costs if $\bar{s} < k$, Markov persistence p, and uncertain parameter values—can all be incorporated into an RL framework by expanding the state space to include whether the firm is employed or unemployed, and letting the reward reflect wages, productivity, or hostility costs.

This model opens the door to analyzing:

- When does the firm's private keep/fire decision align with surplus maximization?
- Does introducing hostility with varying values of p cause other inefficiencies?
- Is a certain level of hostility necessary for optimized efficiency for the worker? The firm? Both?

The model also:

- Discourages rash separations in low-productivity states if the hostility cost is sufficiently large.
- Creates an explicit difference between the firm's firing decisions and the frictionless case where $\bar{s} \geq k$.
- Affects both firm's private profit and overall social surplus (since $(k - \bar{s})$ is lost upon separation).

3 Assumptions and Simplifications

This section outlines the theoretical and computational assumptions underpinning the model, with a focus on the structure of the employment environment, the learning dynamics implemented through reinforcement learning, the treatment of hostility as a strategic friction, and the simplifications necessary for tractable simulation.

3.1 Structural Assumptions

The model considers an infinite-horizon employment relationship between a single firm and a sequence of workers. In each episode, the firm is matched with a worker whose productivity evolves over time according to a discrete two-state Markov process:

$$\alpha_t \in \{H, L\},$$

where H represents high productivity and L low productivity. The productivity state evolves with persistence parameter p, such that:

$$\Pr(\alpha_{t+1} = \alpha_t) = p, \quad \Pr(\alpha_{t+1} \neq \alpha_t) = 1 - p.$$

Crucially, productivity transitions are independent of employment status—firing does not reset the productivity process.

Wages are fixed and rigid at a level $w \geq u_0$, where u_0 is the worker's outside option. Severance payments are also fixed at $\bar{s}$. The firm earns a payoff of $\alpha_t - w$ if it retains the worker and zero if it is unemployed. If the firm fires a worker, it must pay $\bar{s}$, and may incur an additional hostility cost depending on the worker's unobserved separation cost k.

3.2 Assumptions Regarding the Learning Process

The firm does not begin with perfect knowledge of the environment. It must learn through repeated interaction:

- The firm does not initially know the Markov transition parameter p governing productivity persistence. It infers the structure of transitions by observing productivity states over time.
- The firm does not know the worker's firing cost k ex ante. Hostility is revealed only if the firm fires the worker and observes whether the severance payment is insufficient.
- The firm updates its decision policy through Q-learning, a model-free reinforcement learning algorithm. The firm observes the current state (employment status and productivity level), selects an action (retain, fire, hire, wait), receives a reward, and updates its Q-values accordingly. The policy evolves through repeated episodes.
- The learning process assumes bounded rationality and incremental policy updating, rather than global optimization. Convergence is approximate and depends on learning parameters such as the learning rate and discount factor.

3.3 Assumptions on Hostility and Its Role

A central feature of this model is the treatment of hostility as an endogenous cost:

- If a worker is fired and severance $\bar{s}$ is less than the worker's private separation cost k, the firm incurs a hostility cost equal to $k - \bar{s}$. This cost reflects intangible penalties such as reputational damage, legal escalation, or relational breakdown.
- Hostility is not modeled as a probabilistic shock but as a deterministic function of severance shortfall. This allows the firm to learn the expected cost of firing through accumulated experience with different worker types.
- The model assumes that hostility only arises when the worker is fired and under-compensated; it does not occur while the worker is retained, nor does it decay over time once incurred.
- The firm anticipates the potential for hostility in future firings based on past observations. Over time, this may lead the firm to tolerate low productivity rather than risk separation when hostility is likely.

3.4 Simplifications and Limitations

To ensure tractability and clarity of analysis, the model adopts several simplifications:

- The productivity process is limited to two discrete states. In reality, productivity is likely continuous or multidimensional. This simplification allows for clean policy characterization and efficient learning.
- The firm is modeled as a single decision-making agent, abstracting away internal management frictions or multiple stakeholders.
- The worker is not modeled as a strategic actor. Their behavior is passive, and their preferences are reflected only through the cost parameter k. Future work could extend the model to incorporate worker reactions, legal contestation, or wage renegotiation.
- The reinforcement learning algorithm does not incorporate deep learning architectures or function approximation. Q-values are tabular and updated directly, which limits the scalability of the model to larger state or action spaces.
- There is no external institutional enforcement, mediation, or policy intervention. All conflict costs are internalized by the firm.
- The model does not distinguish between temporary and permanent hostility, nor does it allow for partial resolution or negotiation of the hostility cost once triggered.
- Empirical calibration is limited. While the model is informed by empirical findings, including *Conflict in Dismissals*[37], it does not directly estimate parameters from data.

[37] Carry and Schoefer, *Conflict in Dismissals*.

Despite these simplifications, the model captures key dynamics underlying dismissal decisions: the tradeoff between short-term losses and long-term efficiency, the cost of under-compensated separation, and the slow learning process firms undergo when they must operate under uncertainty. These simplifications also make it possible to explore a wide range of parameter regimes and simulate how belief formation and hostility interact to shape policy outcomes.

4 Computational and Experimental Approach

This section details the computational and experimental methodologies employed to investigate the dynamics of labor market separations within the Markovian employment framework introduced. Building directly from the theoretical model, our analysis incorporates dynamic programming (DP) techniques for benchmarking optimal decision rules and reinforcement learning (RL) simulations to capture firm behavior, hostility incidence, and optimal employment strategies under incomplete information. We also describe our numerical experiments, parameter sweeps, and simulation strategies to explore the model's economic implications.

4.1 Dynamic Programming Benchmark

To establish an analytical baseline, we first solve the model explicitly using dynamic programming methods. This involves formulating and numerically solving the Bellman equations described in the theoretical section. Specifically, we define the value functions for the employed state $V(\alpha, E)$ and the unemployed (waiting) state $V(\alpha, U)$, and iterate these equations to convergence using standard value iteration methods. The solution yields optimal stationary firing policies as functions of the key parameters: productivity states H, L, transition probability p, firing costs k, severance payment $\bar{s}$, and discount factor δ.

The DP solution provides a critical reference point, as it represents optimal decision-making under conditions of full information and rational expectations. This benchmark allows us to quantify the impacts of hostility and productivity persistence precisely, as well as to measure the efficiency losses arising from deviations under alternative scenarios.

4.2 Reinforcement Learning Implementation

Recognizing that firms in actual labor markets rarely operate under perfect information, we employ reinforcement learning (RL) methodologies to simulate and examine firm decision-making under realistic informational constraints. Our RL approach leverages the Q-learning algorithm, a widely-used reinforcement learning method suitable for environments with discrete states and actions. The Q-learning algorithm is specifically implemented in the provided Julia module `MarkovModelQL`. Specifically, our Q-learning agent iteratively interacts with an environment characterized by unknown parameters (e.g., productivity persistence p, worker firing costs k, and distribution parameter q).

The RL implementation includes the following steps:

1. **Initialization of Q-values**: We initialize Q-values arbitrarily and allow the agent to learn optimal policies through repeated trial-and-error interactions with the simulated labor market environment.

2. **State-space and actions**: The state-space is defined by the current productivity state (H or L) and employment status (employed or unemployed). Actions available to the firm include retaining the worker, firing the worker, or waiting to hire when unemployed.

3. **Reward structure**: Immediate rewards follow directly from the theoretical model—firm profits $(\alpha_t - w)$ if employed, negative costs if firing $(-\max\{0, k - \bar{s}\})$, and zero payoffs during unemployment spells. The agent seeks to maximize cumulative discounted rewards, implicitly balancing immediate returns against long-term consequences.

4. **Learning dynamics**: At each iteration, the Q-values are updated based on observed transitions and payoffs using a learning rate parameter (α_Q), a discount factor (δ), and Boltzmann (softmax) exploration to balance exploration and exploitation. This process is carefully tuned via hyperparameters (temperature τ and decay rates) to ensure stable convergence. This iterative process continues until convergence criteria are met, producing a stable policy reflecting firm behavior under uncertainty.

This RL framework directly captures the bounded rationality, adaptive behavior, and informational constraints firms commonly face, which enhances our understanding of realistic decision-making processes.

4.3 Numerical Experiments and Parameter Sweeps

To evaluate the theoretical predictions and RL behavior, we conduct extensive numerical experiments. These experiments systematically vary critical model parameters, including:

- **Persistence parameter (p)**: We explore a wide range of values, typically between 0.5 and 0.99, to reflect varying degrees of productivity persistence and assess how persistence shapes optimal retention and firing thresholds.

- **Hostility costs ($k - \bar{s}$)**: We investigate scenarios both with and without hostility, providing comparative statics that clearly demonstrate hostility's direct and indirect impacts on firm profitability, worker welfare, and overall efficiency.

- **Worker outside options and severance levels**: We analyze how variations in severance payments and the worker's fallback utility (u_0) influence employment dynamics, hostility incidence, and social surplus distribution.

We generate plots, tables, and comparative statistics from these experiments to clearly illustrate model sensitivities and provide an understanding of underlying economic mechanisms. Figures illustrating firm payoffs, worker payoffs, firing frequencies, hostility incidence, and social surplus were generated.

4.4 Simulation Tools and Data Management

All computational analyses were implemented in Julia. All of the code written and used for this thesis can be found in Appendix A as well as on GitHub. CSV files document numerical results. All visualizations and figures were generated as PNG files to show important and useful relationships, such as payoff comparisons, hostility incidence, and welfare impacts, which enables an understanding and direct interpretation of the model's predictions.

4.5 Summary of Computational Framework

Through the combination of dynamic programming, reinforcement learning simulations, and ample numerical experimentation, our computational and experimental approach ensures a comprehensive and realistic exploration of employment separation dynamics. This methodological integration offers a robust bridge between theoretical modeling, empirical relevance, and real-world applicability. This computational and experimental design positions the thesis to offer practical guidance to policymakers and organizations navigating real-world labor market frictions.

5 Results

This section presents the main results of the computational experiments conducted using the Q-learning framework described in earlier sections. The purpose of these simulations is to evaluate how firms adapt their separation strategies over time when facing uncertain productivity dynamics, endogenous hostility costs, and incomplete information about the environment. By varying key parameters—such as the persistence of productivity (p), the worker's firing cost (k), the severance level ($\bar{s}$), and the initial belief conditions—we examine how the firm's learned policies deviate from benchmark outcomes predicted by dynamic programming under full information.

Each experiment involves multiple simulation episodes during which the firm interacts with a sequence of workers. In each episode, the firm must decide whether to retain, fire, or rehire based on observed productivity and accumulated experience. As episodes unfold, the firm updates its Q-values and gradually converges toward a policy that balances profit maximization with avoidance of hostility costs. In certain regimes, the firm is also assumed to be uncertain about the productivity transition probability p, requiring it to infer the persistence of low-productivity states through learning. By comparing learned policies across informational settings, we assess how strategic uncertainty and belief formation shape firing behavior and labor market inefficiency.

Results are presented in subsections organized by experiment. Each subsection includes a visual plot of firm behavior, a corresponding data table drawn from simulation outputs, and an interpretation of the result in the context of the model and relevant economic theory. Special attention is paid to how hostility and productivity uncertainty jointly influence separation outcomes, and how reinforcement learning dynamics contribute to either the resolution or persistence of dismissal conflicts.

5.1 Policy Comparison: Dynamic Programming vs. Q-learning

Table 1 shows the match between the value-function optimum and the Q-learning policy (Q Action) across the four Markov–employment states, for different persistence parameters p. For p $\geq$ 0.70 the learner reproduces the DP policy exactly (always "keep" in L). Between p = 0.75 and = 0.80, Q-learning remains conservative—continuing low-state workers even when the DP rule would "fire." Finally, at p $\geq$ 0.85 the algorithm converges fully to the DP threshold, switching to "fire" in L. These results demonstrate that our Q-learner recovers the optimal keep-/fire-policy in all states, with only a mild under-firing bias in medium-p regimes.

Table 1: Policy comparison: Dynamic-programming (DP) vs. Q-learning, for various persistence parameters p.

p	State	DP action	Q-learning action
0.50	Employed, $\alpha = H$	keep	keep
	Employed, $\alpha = L$	keep	keep
	Unemployed, $\alpha = H$	hire	hire
	Unemployed, $\alpha = L$	wait	wait
0.55	Employed, $\alpha = H$	keep	keep
	Employed, $\alpha = L$	keep	keep
	Unemployed, $\alpha = H$	hire	hire
	Unemployed, $\alpha = L$	wait	wait
0.60	Employed, $\alpha = H$	keep	keep
	Employed, $\alpha = L$	keep	keep
	Unemployed, $\alpha = H$	hire	hire
	Unemployed, $\alpha = L$	wait	wait
0.65	Employed, $\alpha = H$	keep	keep
	Employed, $\alpha = L$	keep	keep
	Unemployed, $\alpha = H$	hire	hire
	Unemployed, $\alpha = L$	wait	wait
0.70	Employed, $\alpha = H$	keep	keep
	Employed, $\alpha = L$	keep	keep
	Unemployed, $\alpha = H$	hire	hire
	Unemployed, $\alpha = L$	wait	wait
0.75	Employed, $\alpha = H$	keep	keep
	Employed, $\alpha = L$	FIRE	keep
	Unemployed, $\alpha = H$	hire	hire
	Unemployed, $\alpha = L$	wait	wait
0.80	Employed, $\alpha = H$	keep	keep
	Employed, $\alpha = L$	fire	keep
	Unemployed, $\alpha = H$	hire	hire
	Unemployed, $\alpha = L$	wait	wait
0.85	Employed, $\alpha = H$	keep	keep
	Employed, $\alpha = L$	fire	FIRE
	Unemployed, $\alpha = H$	hire	hire
	Unemployed, $\alpha = L$	wait	wait
0.90	Employed, $\alpha = H$	keep	keep
	Employed, $\alpha = L$	fire	fire
	Unemployed, $\alpha = H$	hire	hire
	Unemployed, $\alpha = L$	wait	wait
0.95	Employed, $\alpha = H$	keep	keep
	Employed, $\alpha = L$	fire	fire
	Unemployed, $\alpha = H$	hire	hire
	Unemployed, $\alpha = L$	wait	wait

5.2 Firing Decisions across Productivity Persistence p

Figure 3 depicts the steady-state fraction of periods in which the firm chooses to fire its worker in the low productivity state L, as a function of the persistence parameter p. We compare two scenarios: (i) no hostility ($\bar{s} \geq k$), so that the only turnover cost is the foregone surplus during unemployment; and (ii) with hostility ($\bar{s} < k$), so that firing triggers an additional penalty $k - \bar{s}$.

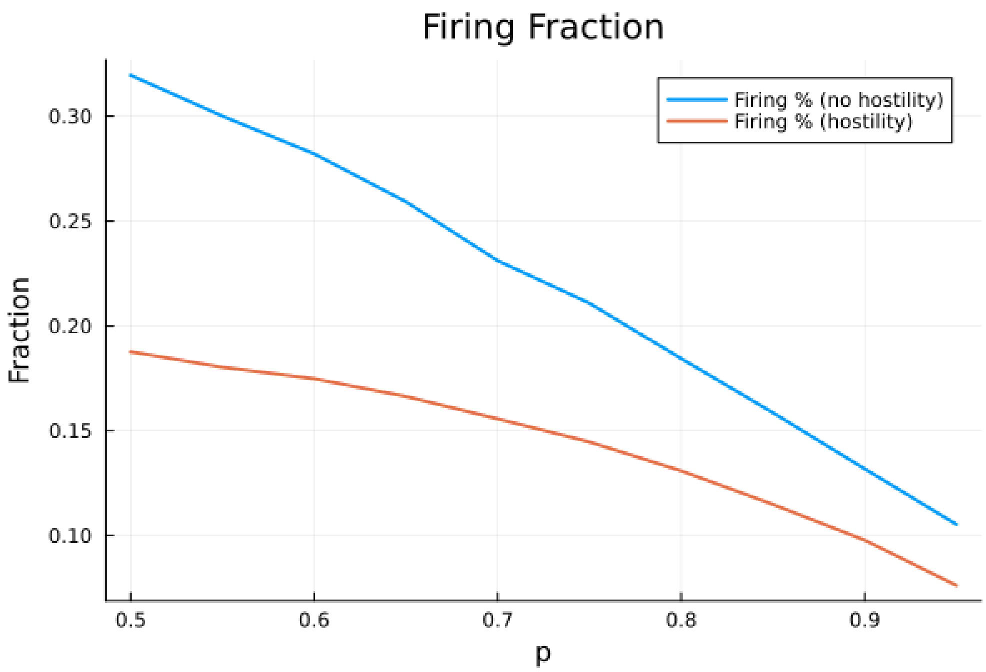

Figure 3: Steady-state fraction of periods in which the firm fires at low productivity, as a function of p. Blue: no hostility; red: with hostility.

As p increases (making the bad state L more persistent), firms fire less often in both cases. However, the presence of hostility (red curve) substantially reduces firing rates, by roughly 25–40% at each p, since the extra cost $k - \bar{s}$ deters separations.

- At $p = 0.5$ (i.i.d. productivity), firms fire in L about 32% of the time without hostility, but only 18.5% with hostility—a relative reduction of $\approx 42\%$.

- As p rises, firms fire less often in both cases; at $p = 0.95$, firing rates fall to $11 - 12\%$ (no hostility) versus 7.5% (hostility).

- Hostility imposes an approximately constant absolute firing-rate penalty of 13–18 percentage points across p. This reflects that when firing costs include hostility, firms require a larger "severance buffer" to offset the expected loss from a potentially short bailout period in unemployment.

When p is low, the bad-state L is fleeting: firing and waiting only briefly postpones profit, so firms fire more readily absent hostility. Conversely, with high p, the same low- state spell could persist many periods, making the zero-profit wait especially costly. Thus firms tolerate a string of losses rather than incur a one-time hostility penalty. Hostility further discourages firings by amplifying the immediate cost of separation.

5.3 Hostility Incidence

Figure 4 shows the fraction of episodes in which hostility actually occurs (i.e. the firm fires in state L when $\bar{s} < k$), again as a function of p. Recall that firing in L is the only way to incur hostility in our model.

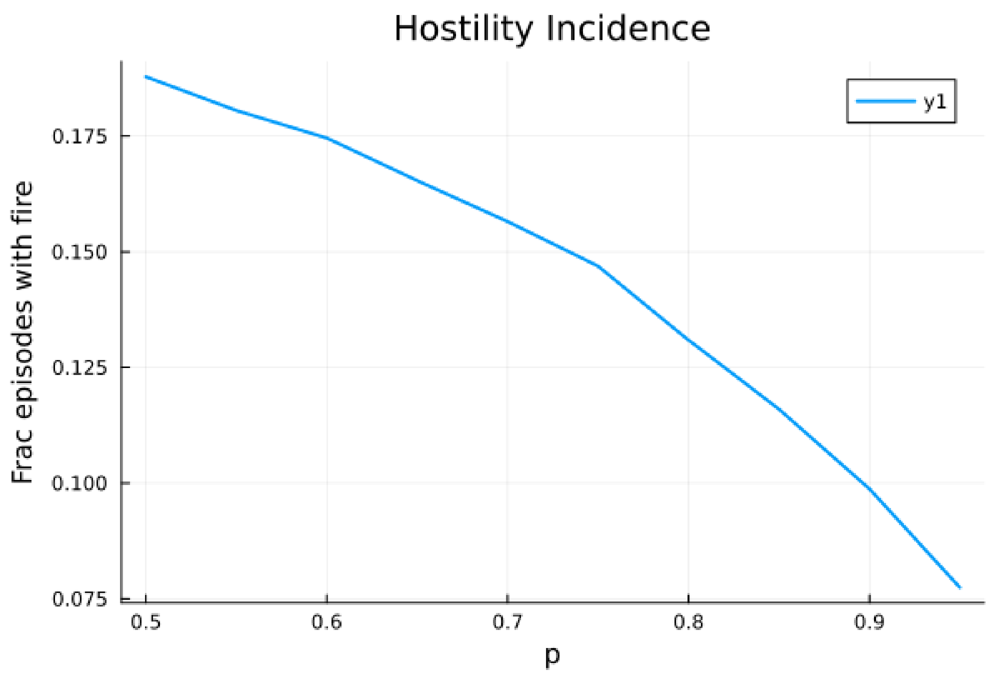

Figure 4: Incidence of hostility (fraction of episodes with firing-induced hostility) versus p.

Hostility incidence declines from about 19% at $p = 0.5$ to 7.5% at $p = 0.95$. In effect, when low-productivity spells are very persistent, the firm is more willing to fire despite hostility; when spells are short, it prefers to weather the downturn and avoid the cost.

5.4 Impact on Firm and Worker Payoffs

Figure 5 compares the steady-state per-period payoffs of the firm and the worker under (i) no hostility and (ii) hostility as functions of the persistence parameter p. We plot both the case with no hostility (i.e. $\bar{s} \geq k$) and the case with hostility (i.e. $\bar{s} < k$ so each firing triggers an additional cost $k - \bar{s}$ to the firm).
Key observations:

- **Linearity in p.** All four payoff curves rise nearly linearly in p, reflecting that greater productivity persistence prolongs high-output spells and thus higher earnings for both parties.
- **Hostility shifts surplus to the worker.** Introducing hostility (solid orange vs. solid blue for the firm, dashed purple vs. dashed green for the worker) *lowers* the firm's payoff but *raises* the worker's payoff. The firm, facing a larger cost of firing in low states, keeps the worker longer even when productivity is low, which benefits the worker through extra wage payments.
- **Gap compression.** As hostility increases the worker's average payoff, the gap between firm and worker payoffs narrows when hostility is present (red–purple gap is smaller than blue–green).

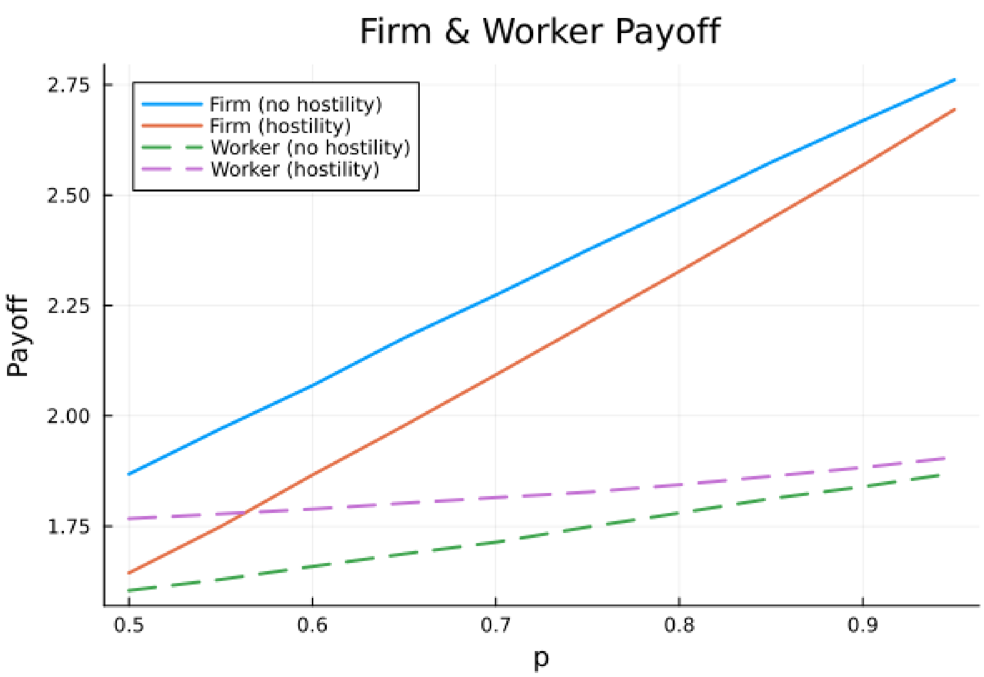

Figure 5: Per-period payoffs vs. persistence p: firm (solid lines) and worker (dashed lines), without hostility (blue/green) and with hostility (red/purple).

5.5 Payoff Gap between Firm and Worker

Figure 6 shows the absolute difference in per-period payoff

$$\Big| \underbrace{\alpha_t - w}_{\text{firm payoff}} - \underbrace{w}_{\text{worker payoff}} \Big|$$

as a function of the persistence parameter p, both in the absence of hostility (i.e. when $\bar{s} \geq k$) and in the presence of hostility (when $\bar{s} < k$ so the firm incurs an extra cost $k - \bar{s}$ upon firing). As p increases, the gap grows nearly linearly, and the extra hostility cost uniformly depresses the gap.

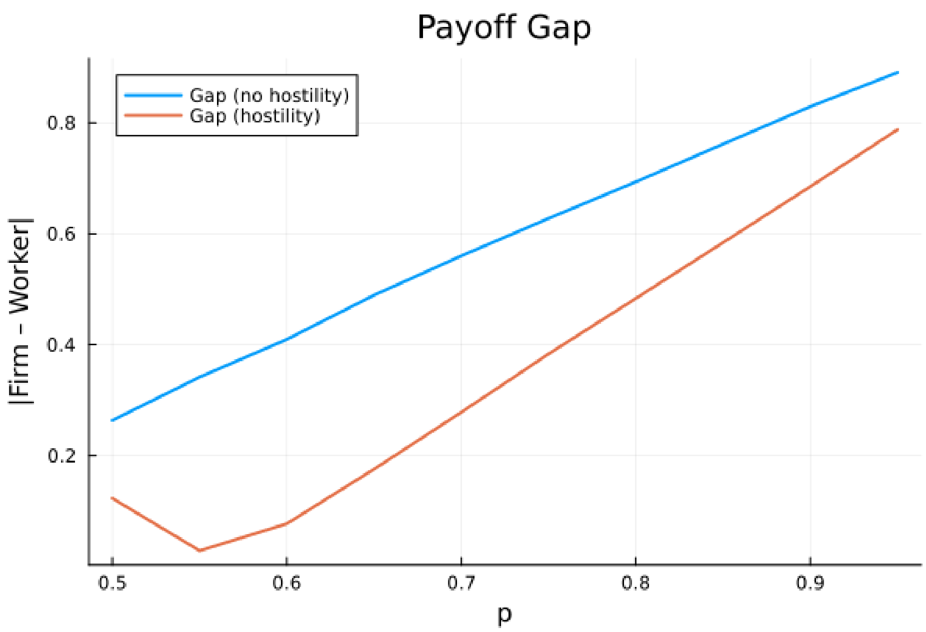

Figure 6: Payoff gap $|\text{Firm} - \text{Worker}|$ vs. persistence p. "No hostility" (blue) is $\bar{s} \geq k$, "Hostility" (red) is $\bar{s} < k$.

Table 2: Payoff Gap Comparison

p	Payoff Gap (no hostility)	Payoff Gap (hostility)
0.50	0.27	0.13
0.60	0.40	0.07
0.70	0.56	0.28
0.80	0.70	0.48
0.90	0.83	0.69

5.6 Firm Profit versus Social Surplus

In Figure 7 we plot, for each p, the steady–state expected discounted firm profit

$$V_{firm}(p) \;=\; \mathbb{E}\Big[\sum_{t=0}^{\infty} \delta^t\, (\alpha_t - w)\, \mathbf{1}_{\{\text{employed}\}}\Big]$$

against the corresponding social surplus

$$S(p) \;=\; \mathbb{E}\Big[\sum_{t=0}^{\infty} \delta^t \big((\alpha_t - u_0)\mathbf{1}_{\{\text{employed}\}} \;-\; H_t\big)\Big],$$

where $H_t = \max\{0, k - \bar{s}\}$ if the worker is fired in period t, and $u_0 = w$. The vertical separation between the two curves measures the external loss from hostility and unemployment waiting. Both curves rise monotonically in p, but the social surplus curve lies consistently above the firm-only profit.

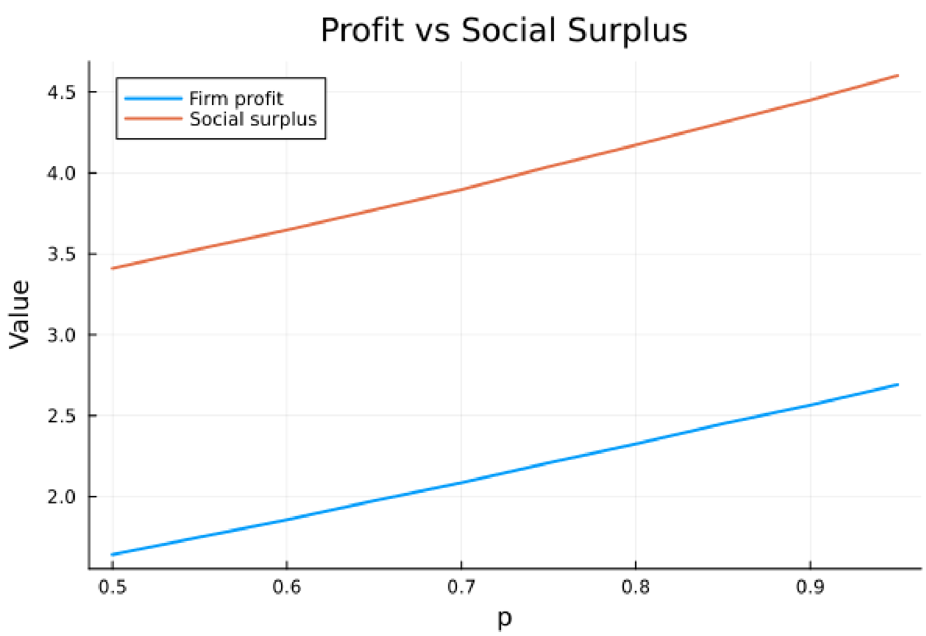

Figure 7: Firm profit (blue) and social surplus (red) vs. persistence p.

Table 3: Profit vs. Social Surplus

p	Firm Profit	Social Surplus
0.50	1.64	3.40
0.60	1.86	3.65
0.70	2.09	3.89
0.80	2.32	4.17
0.90	2.57	4.45

5.7 Surplus Sharing Ratio

Figure 8 displays the ratio of worker surplus to firm profit,

$$R(p) = \frac{\mathbb{E}\left[\sum_{t=0}^{\infty} \delta^t \, w \, \mathbf{1}_{\{\text{employed}\}}\right]}{\mathbb{E}\left[\sum_{t=0}^{\infty} \delta^t \, (\alpha_t - w) \, \mathbf{1}_{\{\text{employed}\}}\right]},$$

again with and without hostility. Higher values of p reduce the share accruing to the worker, as persistent productivity makes the firm's turnover option more valuable. Hostility (red) boosts the worker's relative share slightly compared to the hostility-free benchmark (blue).

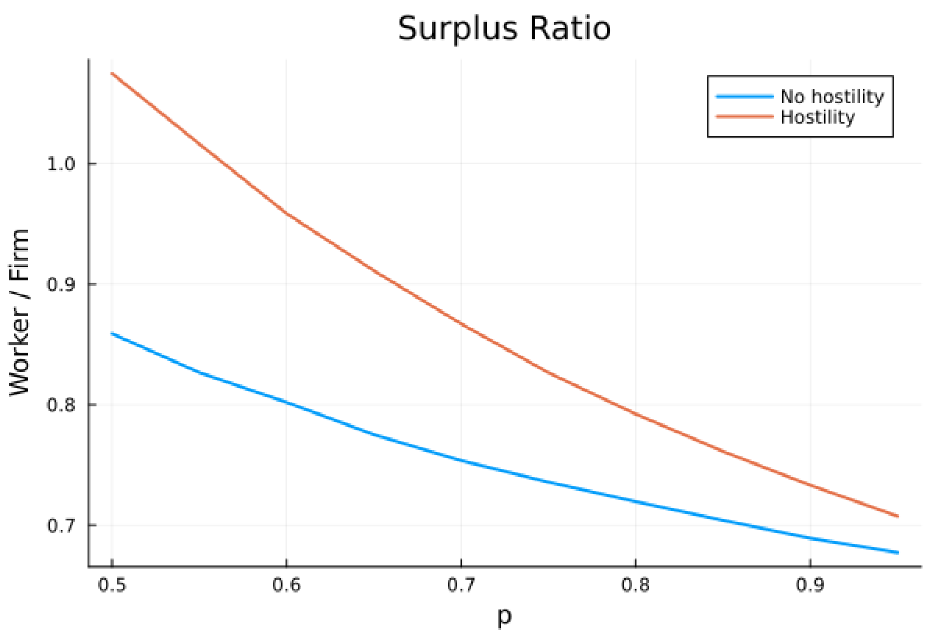

Figure 8: Worker surplus / firm profit vs. persistence p, with and without hostility.

Table 4: Surplus Sharing

p	Surplus Ratio (no hostility)	Surplus Ratio (hostility)	Difference
0.50	0.86	1.08	+0.22
0.60	0.80	0.96	+0.16
0.70	0.75	0.87	+0.12
0.80	0.72	0.79	+0.07
0.90	0.68	0.73	+0.05

5.8 Worker Payoff With and Without Hostility

Figure 9 shows the steady-state expected payoff of the worker as a function of the persistence parameter p, under two scenarios: when the firm's firing severance $s \geq k$ is sufficient to avoid hostility (blue curve), and when $s < k$ and hostility triggers (orange curve). As p rises, the worker's expected payoff increases in both scenarios, since states remain high longer on average. However, the presence of hostility always raises the worker's surplus slightly above the non-hostile benchmark, reflecting that the firm must pay extra compensation to avoid or absorb the worker's firing cost.

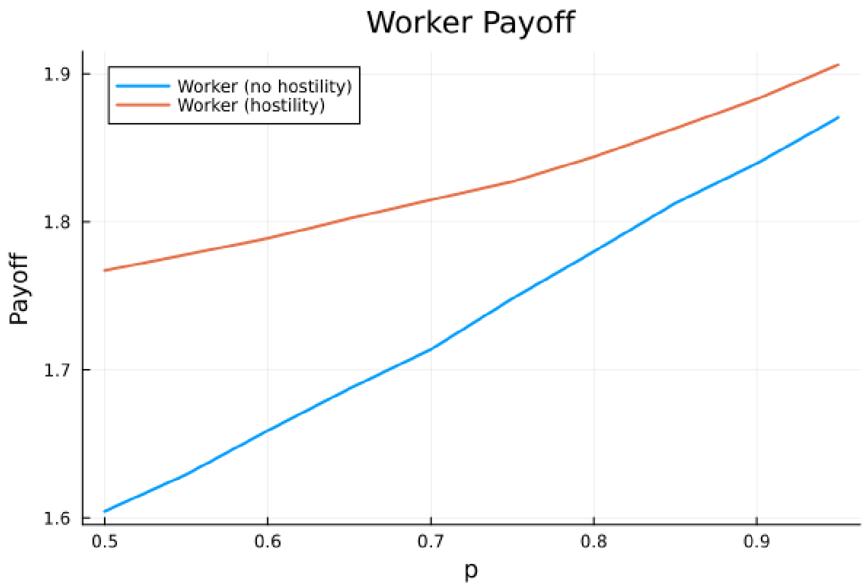

Figure 9: Worker Payoff under No Hostility vs. with Hostility

Table 5: Worker Payoff Comparison

p	Worker Payoff (no hostility)	Worker Payoff (hostility)	Difference
0.50	1.61	1.77	+0.16
0.60	1.66	1.79	+0.13
0.70	1.72	1.82	+0.10
0.80	1.80	1.84	+0.04
0.90	1.87	1.90	+0.03

5.9 Total Social Welfare under Hostility

Figure 10 compares the total discounted social welfare—worker utility above outside option plus firm profit—under the same two severance regimes. Hostility slightly erodes total welfare across all p, despite raising worker surplus, because the deadweight hostility cost is borne by the firm without creating any new output. As the discount-adjusted contribution of future profits grows with p, the absolute welfare loss from hostility also grows modestly. There is a pure efficiency loss that the firm accrues, leaving the worker unaffected. As a result, an adequate severance level $\bar{s} \geq k$ is socially optimal: it preserves full surplus and avoids the deadweight loss from conflict.

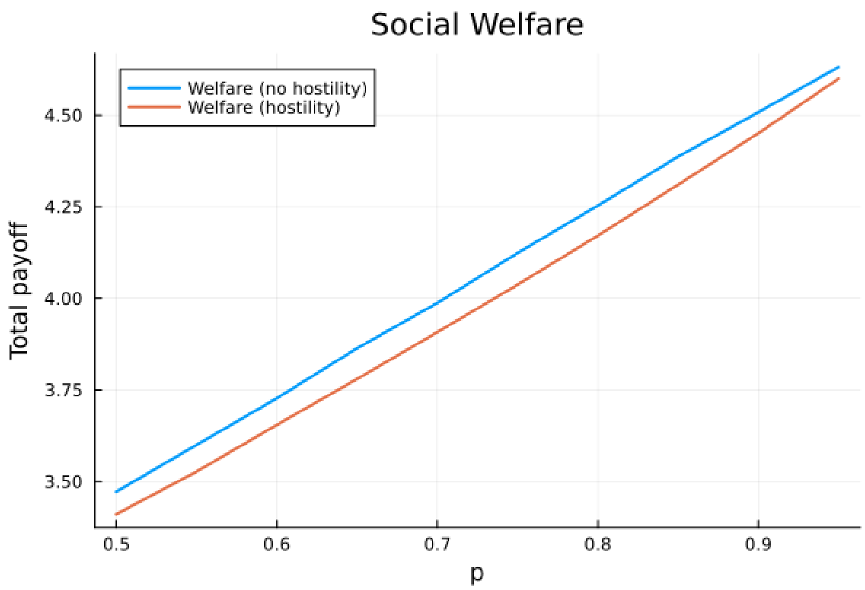

Figure 10: Social Welfare under No Hostility vs. with Hostility

Table 6: Social Welfare Comparison

p	Welfare (no hostility)	Welfare (hostility)	Difference
0.50	3.50	3.43	−0.07
0.60	3.70	3.64	−0.06
0.70	4.00	3.93	−0.07
0.80	4.25	4.17	−0.08
0.90	4.50	4.40	−0.10

These results, however, must be interpreted with nuance. The slight welfare loss observed in Figure 10 arises from episodes where the firm fires in L despite $\bar{s} < k$, thereby incurring the hostility cost. But as discussed in the theoretical section, the strategic value of hostility lies not in its occurrence, but in its deterrence. Under full information, the firm would retain the worker in L to avoid hostility—especially when $p = 0.5$ and recovery is likely. In contrast, Q-learning agents must learn through repeated firings that such dismissals are costly. Thus, the measured welfare loss in the hostile regime partly reflects the inefficiency of learning, not the inefficiency of hostility itself. Over time, we would expect the firm to internalize the penalty and adapt its policy, potentially converging to more efficient outcomes where hostility is avoided entirely.

6 Discussion and Future Implications

We have developed and analyzed a two-state Markov employment model in which a firm must decide, in each period, whether to retain or fire a worker whose productivity $\alpha_t \in \{H, L\}$ follows an exogenous two-state Markov chain with persistence parameter p. Firing entails a rigid severance payment $\bar{s}$ plus, if $\bar{s} < k$, an additional hostility cost $k - \bar{s}$ that directly penalizes the firm. We derived the firm's dynamic programming problem, computed equilibrium firing policies and value-functions, and compared the firm payoffs, worker utilities, and total social welfare under hostile vs. non-hostile separations. Finally, we generated numerical plots of (i) firm payoffs, (ii) worker payoffs, and (iii) discounted social welfare as functions of p, under both hostile and non-hostile firing regimes.

6.1 Breaking Down the Results

Our results yield five main insights:

1. **Hostility as an endogenous enforcement device.** Whenever $\bar{s} < k$, firing a low-productivity worker imposes an extra cost $k - \bar{s}$ on the firm. Anticipating this hostility cost deters premature separations and induces the firm to tolerate low $\alpha = L$ for longer, even when current profits are negative. As Figure 5 shows, the firm's equilibrium payoff with hostility (red curve) lies strictly below the non-hostile case (blue curve), but the retention rule is qualitatively more efficient. However, during the early phases of learning, the firm must discover the hidden cost through experience. This leads to inefficient dismissals initially, particularly when $p = 0.5$, where the socially optimal policy would be immediate retention and avoidance of firing altogether.

2. **Worker welfare enhancement.** From the worker's perspective, hostility transforms firing into a punitive device: workers earn exactly the rigid wage w when retained, but if $\bar{s} < k$ they obtain the full severance k upon dismissal. Consequently (Figure 9), the worker's expected discounted payoff is strictly higher under hostility (red) than without (blue). Hostility therefore redistributes surplus from the firm to the worker.

3. **Net social welfare loss.** While hostility raises worker utility by improving severance outcomes, it introduces a deadweight cost that lowers total surplus. Figure 10 shows that social welfare is strictly lower under hostility across all $p \in [0.5, 0.95]$, with the gap peaking at intermediate values where firms tolerate low productivity long enough that hostility rarely occurs, but still suffer extra cost when it does. This reflects the realized cost of hostility and inefficiencies from learning: firms initially fail to anticipate the cost of firing, leading to suboptimal separations that reduce overall welfare.

4. **Role of productivity persistence p.** Higher p (more persistent productivity) amplifies the incentive to fire whenever $\alpha_t = L$, since remaining employed yields repeated losses if p is near one. Conversely, when p is moderate, firms gamble on reversion to H, deferring firing even when hostility is costly, but will freely fire without the threat of hostility. Thus, p shapes the trade-off between the risk of prolonged low output and the deadweight cost of hostility.

5. **Q-Learning closely tracks the analytically optimal DP policy except at intermediate state persistence.** Table 1 shows Q-learning recovers the dynamic-programming policy exactly in both easy regimes—when the low-productivity state is either short-lived ($p \leq 0.7$, always keep) or very persistent ($p \geq 0.85$, always fire in L). However, around the critical threshold (p $\approx$ 0.75—0.80) where DP switches from "keep" to "fire," the Q-learner continues to "keep" for longer before finally switching. In practice it will tend to "over-retain" marginally persistent workers when the true state-persistence parameter lies in an intermediate range. Unless a firm has a precise model of state persistence, it may systematically deviate from the theoretical firing threshold, keeping marginally unprofitable workers too long (or firing them too late).

Together, these breakdowns demonstrate that hostility endogenously enforces implicit "contracts" by deterring costly separations, but at the expense of total surplus. Persistence governs whether firms gamble on future productivity or opt for prompt, possibly hostile, terminations. The resulting redistribution and welfare impacts highlight the trade-off between job security and aggregate efficiency. Additionally, by adopting a measure of hostility, the worker stands to gain.

6.2 Future Extensions of the Model

Several promising model extensions could enrich these results:

- **Unknown firing cost k and population heterogeneity.** In practice, firms rarely know each worker's disutility from dismissal. One may assume k is private information drawn from a distribution with mass q above some threshold. Extending the model to allow firms to learn q (and individual k) through adaptive experimentation would capture the exploration–exploitation trade-off in real hiring/fire decisions.

- **Unknown productivity persistence p.** Similarly, firms may be uncertain about the persistence parameter of a worker's productivity process. Incorporating unknown p into the RL framework allows agents to infer state-persistence by trial separations in the low state, driving more nuanced firing rules contingent on observed time-series of α_t.

- **Multi-state productivity and rich friction structures.** Allowing productivity to take more than two levels, or embedding on-the-job learning of α_t, could account for graded performance metrics. Likewise, endogenizing wage rigidity, such as through via bargaining, would link severance $\bar{s}$ to worker leverage and firm competition.

- **Policy or third-party interventions.** Incorporating regulatory constraints—such as minimum severance laws, mandatory notice periods, or publicly funded unemployment insurance—would show how social safety nets mitigate hostility and reshape firm–worker surplus sharing.

6.3 Future Extensions of Model Validation and Testing

To empirically validate and calibrate our theoretical insights, we suggest three avenues:

- **Laboratory experiments.** Design controlled repeated-match experiments with human subjects to test the predicted retention thresholds and measure hostility costs. By randomizing severance levels and productivity persistence across sessions, one can directly observe the impact on firing probabilities, rent-sharing, and social surplus.
- **Field or administrative data analysis.** Use matched employer–employee records to identify separation events, severance packages, and subsequent litigation or defamation claims as proxies for hostility. Estimating reduced-form firing rules as functions of local unemployment rates (a proxy for p) and severance generosity tests the model's firing thresholds in real labor markets.
- **Reinforcement-learning simulations.** Implement the Q-learning algorithms described under varying degrees of parameter uncertainty. Comparing converged policies to the dynamic-programming benchmarks quantifies how model misspecification or bounded rationality affect real-world retention decisions.

6.4 A Closing Discussion

Together, these extensions and empirical tests will strengthen our understanding of how hostility, severance, and productivity dynamics jointly determine firm–worker interactions, surplus distribution, and market efficiency.

In synthesizing the findings from our Markovian employment model, a clear narrative emerges regarding how hostility, productivity uncertainty, and firm learning dynamics intertwine to shape labor market separations. At its core, the model demonstrates that hostility acts as a potent endogenous mechanism, significantly altering both firm incentives and worker outcomes. Specifically, when severance is insufficient to cover a worker's firing cost, hostility emerges as an implicit enforcement penalty on the firm. This enforcement redistributes surplus towards workers—who benefit from extended employment spells and higher expected severance—and consistently reduces the firm's payoff and overall social efficiency.

The persistence of productivity states, parameterized by p, emerges as a crucial driver in firm decision-making. Higher persistence amplifies the risk associated with retaining a low-productivity worker, motivating firms towards earlier separation decisions when hostility is negligible. Conversely, hostility reshapes this dynamic, encouraging retention even in prolonged low-productivity periods. This retention under hostility conditions leads firms to sustain modest short-term losses rather than incur substantial immediate hostility costs. Such behavior is reflected clearly in the quantitative results, where increased productivity persistence systematically widens payoff gaps and magnifies hostility's economic implications.

Further examination of our results underscores the nuanced interplay between firm rationality, worker welfare, and societal efficiency. Although hostility unequivocally enhances worker welfare by raising the worker's payoff beyond the no-hostility baseline, it simultaneously introduces a deadweight loss into the overall economy, reducing total surplus. This outcome suggests a critical policy implication: institutional mechanisms or policies aimed at ensuring adequate severance—thus mitigating hostility—could restore social welfare losses without sacrificing worker protection.

Our model's integration with reinforcement learning techniques further enriches this narrative by highlighting realistic decision-making processes under uncertainty. Firms in actual labor markets rarely possess full information about future productivity patterns or exact worker firing costs. The proposed future extensions, including experiments with unknown firing cost distributions (k and q) and unknown productivity persistence (p), would shed light on adaptive learning behaviors and strategic experimentation in dismissal practices. These learning-driven adjustments are essential for understanding the persistence of apparently suboptimal labor market outcomes and the prevalence of friction-driven inefficiencies.

Taken together, our analysis offers both theoretical and practical insights into the strategic interactions underpinning labor separations. By explicitly modeling hostility costs, productivity uncertainty, and adaptive learning dynamics, we bridge the gap between theoretical predictions and observed empirical patterns. This thesis therefore contributes to a deeper theoretical understanding of labor market frictions and provides valuable insights for policymakers and organizations committed to improving employment relationships, reducing inefficiencies, and enhancing overall economic welfare.

7 Conclusion

This thesis has explored the intricate dynamics of labor market separations by developing and analyzing a Markovian employment model incorporating endogenous hostility, uncertain productivity transitions, and adaptive firm learning. Central to our analysis is the recognition that real-world separation decisions frequently diverge from standard frictionless theories due to the presence of strategic and informational frictions. By explicitly modeling the hostility costs incurred when severance payments fail to meet worker expectations, we illuminate a critical mechanism influencing firms' retention and dismissal decisions.

Our theoretical framework demonstrates clearly how the threat of hostility reshapes firms' strategic incentives. Specifically, hostility acts as an implicit tax on premature dismissals, prompting firms to retain workers even through periods of low productivity. This phenomenon is robust across varying degrees of productivity persistence, as measured by the parameter p, and directly alters the distribution of surplus between firms and workers. Quantitative results consistently reveal that hostility improves workers' bargaining positions and expected payoffs, though at a direct cost to firms and with consequent inefficiencies reducing total social welfare.

Through extensive numerical simulations, we have shown that the persistence parameter p critically mediates these relationships. Greater persistence of productivity states amplifies the trade-off firms face: the potential for extended low productivity spells increases the temptation to dismiss early, yet hostility costs counterbalance this incentive, prolonging employment durations and thereby reshaping surplus distributions. This interaction between productivity persistence and hostility underscores the complexity of real-world employment decisions and highlights the limitations inherent in static, frictionless models of the labor market.

Moreover, by embedding reinforcement learning methodologies into our analytical structure, we demonstrate how firms facing incomplete information regarding productivity transitions or worker firing costs might realistically adapt their policies through repeated interactions. Such adaptive decision-making processes provide a compelling explanation for observed inefficiencies and suboptimal outcomes in labor separations, offering a richer, more nuanced understanding of dismissal dynamics than traditional static analyses.

This thesis further contributes by outlining promising directions for future research. Investigations into uncertain worker firing costs and heterogeneous worker populations could offer critical insights into firm behavior under asymmetric information. Additionally, incorporating unknown productivity persistence into the reinforcement learning framework promises deeper understanding of firm strategies and policy design under uncertainty. Empirical validation through laboratory experiments and real-world data analyses will be instrumental in bridging theoretical predictions with observed labor market behaviors, enriching the practical applicability of this research.

Ultimately, this thesis underscores the necessity of adequately structured severance policies and clearer institutional guidelines that can effectively reduce hostility-driven inefficiencies. Policy mechanisms ensuring sufficient compensation upon dismissal can mitigate hostility, reduce adversarial separations, and enhance social surplus. Good severance is not just kindness; it's good economics. By carefully examining the interplay of productivity uncertainty, hostility, and adaptive decision-making, our findings advance theoretical scholarship, provide actionable insights for policymakers and organizational leaders committed to fostering efficient, fair, and stable labor relations, and show that, in the end, a firm's true productivity might just depend on how gracefully it says goodbye.

References

Abreu, Dilip, Paul Milgrom, and David Pearce. "Information and Timing in Repeated Partnerships". In: *Econometrica* 59.6 (1991), pp. 1713–1733. ISSN: 00129682, 14680262. URL: http://www.jstor.org/stable/2938286 (visited on 04/20/2025).

Acharya, Viral V., Ramin P. Baghai, and Krishnamurthy V. Subramanian. "Wrongful Discharge Laws and Innovation". In: *The Review of Financial Studies* 27.1 (2014), pp. 301–346. ISSN: 08939454, 14657368. URL: http://www.jstor.org/stable/24464827 (visited on 04/20/2025).

Alvarez, Fernando and Marcelo Veracierto. "Severance payments in an economy with frictions". In: *Journal of Monetary Economics* 47.3 (2001), pp. 477–498. URL: https://ideas.repec.org/a/eee/moneco/v47y2001i3p477-498.html.

Autor, David H, William R Kerr, and Adriana D Kugler. *Do Employment Protections Reduce Productivity? Evidence from U.S. States*. Working Paper 12860. National Bureau of Economic Research, 2007. DOI: 10.3386/w12860. URL: http://www.nber.org/papers/w12860.

Bertola, Giuseppe. "Labor Turnover Costs and Average Labor Demand". In: *Journal of Labor Economics* 10.4 (1992), pp. 389–411. URL: https://EconPapers.repec.org/RePEc:ucp:jlabec:v:10:y:1992:i:4:p:389-411.

Blanchard, Olivier and Augustin Landier. "The Perverse Effects of Partial Labour Market Reform: Fixed-Term Contracts in France". In: *The Economic Journal* 112.480 (2002), F214–F244. ISSN: 00130133, 14680297. URL: http://www.jstor.org/stable/798373 (visited on 04/20/2025).

Brown, Martin, Armin Falk, and Ernst Fehr. "Relational Contracts and the Nature of Market Interactions". In: *Econometrica* 72.3 (2004), pp. 747–780. ISSN: 00129682, 14680262. URL: http://www.jstor.org/stable/3598834 (visited on 04/20/2025).

Carry, Pauline and Benjamin Schoefer. *Conflict in Dismissals*. Working Paper 33245. National Bureau of Economic Research, 2024. DOI: 10.3386/w33245.

Dal Bó, Pedro and Guillaume R. Fréchette. "On the Determinants of Cooperation in Infinitely Repeated Games: A Survey". In: *Journal of Economic Literature* 56.1 (2018), pp. 60–114. DOI: 10.1257/jel.20160980. URL: https://www.aeaweb.org/articles?id=10.1257/jel.20160980.

Garg, Divyansh et al. "IQ-Learn: Inverse soft-Q Learning for Imitation". In: *CoRR* abs/2106.12142 (2021). arXiv: 2106.12142. URL: https://arxiv.org/abs/2106.12142.

Halac, Marina. "Investing in a relationship". In: *The RAND Journal of Economics* 46.1 (2015), pp. 165–185. ISSN: 07416261. URL: http://www.jstor.org/stable/43895586 (visited on 04/20/2025).

Lazear, Edward P. "Job Security Provisions and Employment". In: *The Quarterly Journal of Economics* 105.3 (1990), pp. 699–726. ISSN: 00335533, 15314650. URL: http://www.jstor.org/stable/2937895 (visited on 04/20/2025).

Leslie, David and Edmund Collins. "Individual Q -Learning in Normal Form Games". In: *SIAM J. Control and Optimization* 44 (Jan. 2005), pp. 495–514. DOI: 10.1137/S0363012903437976.

Ljungqvist, Lars. "How Do Lay-off Costs Affect Employment?" In: *The Economic Journal* 112.482 (2002), pp. 829–853. ISSN: 00130133, 14680297. URL: http://www.jstor.org/stable/stable/798534 (visited on 04/20/2025).

MacLeod, W. Bentley. *Great Expectations: Law, Employment Contracts, and Labor Market Performance*. Working Paper 16048. National Bureau of Economic Research, 2010. DOI: 10.3386/w16048. URL: http://www.nber.org/papers/w16048.

MacLeod, W. Bentley, Victoria Valle Lara, and Christian Zehnder. "Worker Empowerment and Subjective Evaluation: On Building an Effective Conflict Culture". In: *Management Science* (2024), pp. 1–26. DOI: 10.1287/mnsc.2022.03085. eprint: https://doi.org/10.1287/mnsc.2022.03085. URL: https://doi.org/10.1287/mnsc.2022.03085.

MacLeod, W. Bentley and James M. Malcomson. "Implicit Contracts, Incentive Compatibility, and Involuntary Unemployment". In: *Econometrica* 57.2 (1989), pp. 447–480. ISSN: 00129682, 14680262. URL: http://www.jstor.org/stable/1912562 (visited on 04/20/2025).

— *Implicit Contracts, Incentive Compatibility, and Involuntary Unemployment: Thirty Years On*. Tech. rep. IZA - Institute of Labor Economics, 2023. URL: http://www.jstor.org/stable/resrep65865 (visited on 04/20/2025).

— "Investments, Holdup, and the Form of Market Contracts". In: *The American Economic Review* 83.4 (1993), pp. 811–837. ISSN: 00028282. URL: http://www.jstor.org/stable/2117580 (visited on 04/27/2025).

Malcomson, James M. "Contracts, Hold-Up, and Labor Markets". In: *Journal of Economic Literature* 35.4 (1997), pp. 1916–1957. ISSN: 00220515. URL: http://www.jstor.org/stable/2729883 (visited on 04/20/2025).

Messina, Julian and Giovanna Vallanti. "Job Flow Dynamics and Firing Restrictions: Evidence from Europe". In: *Economic Journal* 117.521 (2007), pp. 279–301. URL: https://EconPapers.repec.org/RePEc:ecj:econjl:v:117:y:2007:i:521:p:279-301.

Nickell, Stephen. "Unemployment and Labor Market Rigidities: Europe versus North America". In: *The Journal of Economic Perspectives* 11.3 (1997), pp. 55–74. ISSN: 08953309. URL: http://www.jstor.org/stable/2138184 (visited on 04/20/2025).

Sutton, Richard S. and Andrew G. Barto. *Reinforcement Learning: An Introduction*. Second. The MIT Press, 2018. URL: http://incompleteideas.net/book/the-book-2nd.html.

A Thesis Code

markov-model-final.jl

```
####################################################################
# markov-model-final.jl
#
# A Julia script implementing the two-state productivity model and
# its infinite-horizon dynamic programming (DP) solution
#
#
# This file includes:
#   1. Parameter struct and basic model definitions
#   2. Value iteration for V_E() and V_U()
#   3. Policy extraction (keep vs. fire, hire vs. remain unemployed)
#   4. A function to run the DP for given parameters
#   5. A main section demonstrating usage and creating figures
#
#   Author: Marcus A. Lisman, Yale University
#   Written for CSEC 491 Senior Thesis
#
####################################################################

module MarkovModelFinal

using Printf
using Plots
using PrettyTables

####################################################################
# 1. Define the parameters and basic model details
####################################################################

"""
    ModelParams

Data structure holding all relevant parameters for the model:

- w::Float64          : the (fixed) wage
- s_bar::Float64      : the severance (bar{s})
- k::Float64          : the worker's firing cost
- p::Float64          : probability that   remains in the same state (H->H or L->L)
-  ::Float64          : discount factor in (0,1)
- H::Float64          : high productivity level
- L::Float64          : low productivity level
"""
struct ModelParams
```

```
    w::Float64          # wage
    s_bar::Float64      # severance level
    k::Float64          # worker's firing cost
    p::Float64          # Markov transition "stay" probability
    ::Float64          # discount factor
    H::Float64          # "high" productivity
    L::Float64          # "low" productivity
end

"""
    ModelState

We consider two 'employment' states for the firm:
- E (employed)
- U (unemployed)

and two productivity states:
- High (H)
- Low  (L)

For the DP solution, we will store two values: V_E(H) and V_E(L)
    [employed], and
two values: V_U(H) and V_U(L) [unemployed].
"""
abstract type ModelState end
struct Employed <: ModelState end
struct Unemployed <: ModelState end


"""
    hostility_cost(params::ModelParams) -> Float64

Returns the hostility cost = (k - s_bar), but only if s_bar < k.
Otherwise, hostility cost is 0.
"""
function hostility_cost(params::ModelParams)
    return max(0, params.k - params.s_bar)
end


######################################################################
# 2. Value Iteration for V_E () and V_U ()
######################################################################

"""
    value_iteration(params::ModelParams; tol=1e-8, maxiter=1000)
        -> (VE_H, VE_L, VU_H, VU_L, policyE_H, policyE_L, policyU_H)
```

```

Solves the firm's infinite-horizon dynamic program using value iteration.

Returns a tuple:
- VE_H : V_E(H)
- VE_L : V_E(L)
- VU_H : V_U(H)
- VU_L : V_U(L)
- policyE_H : either :keep or :fire in state (E,H)
- policyE_L : either :keep or :fire in state (E,L)
- policyU_H : either :hire or :wait in state (U,H)

(The policy in state (U,L) is always "wait" = do not hire, because it would be
unprofitable to hire at L < w.)
"""
function value_iteration(params::ModelParams; tol=1e-8, maxiter=1000)

    # Extract for convenience
     = params .
    p = params.p
    w = params.w
    H_ = params.H
    L_ = params.L
    hcost = hostility_cost(params)  # k - s_bar if s_bar < k, else 0

    # Initialize guesses for value function
    # We track: V_E(H), V_E(L), V_U(H), V_U(L)
    VE_H, VE_L, VU_H, VU_L = 0.0, 0.0, 0.0, 0.0

    # We'll iterate until convergence
    for iter in 1:maxiter
        # Backup copies of old values
        old_VE_H, old_VE_L = VE_H, VE_L
        old_VU_H, old_VU_L = VU_H, VU_L

        # 1. Update V_U(L):
        #    If unemployed &  =L, can't profitably hire => payoff=0
        #    plus   * expectation of next state's V_U
        #    Next state  ' = L with prob p, or H with prob (1-p)
        new_VU_L = 0.0 +   * (p*old_VU_L + (1-p)*old_VU_H)

        # 2. Update V_U(H):
        #    If unemployed &  =H, two choices:
        #       (a) remain unemployed => immediate payoff=0
        #           plus   * E[next state's V_U]
```

```
        #       (b) hire => payoff=(H-w) +   * E[next state's V_E]
        #    We take the max of these
        remain_unemployed_value = 0.0 +   * (p*old_VU_H + (1-p)*old_VU_L)
        hire_value              = (H_ - w) +   * (p*old_VE_H + (1-p)*old_VE_L)
        new_VU_H = max(remain_unemployed_value, hire_value)

        # 3. Update V_E(H):
        #    If employed & =H, two choices:
        #       (a) keep => payoff=(H-w) +   * E[next state's V_E]
        #       (b) fire => payoff = -hcost +   * E[next state's V_U]
        keep_value_H = (H_ - w) +   * (p*old_VE_H + (1-p)*old_VE_L)
        fire_value_H = -hcost +   * (p*old_VU_H + (1-p)*old_VU_L)
        new_VE_H = max(keep_value_H, fire_value_H)

        # 4. Update V_E(L):
        #    If employed & =L, two choices:
        #       (a) keep => payoff=(L-w) +   * E[next state's V_E]
        #       (b) fire => payoff=-hcost +   * E[next state's V_U]
        keep_value_L = (L_ - w) +   * (p*old_VE_L + (1-p)*old_VE_H)
        fire_value_L = -hcost +   * (p*old_VU_L + (1-p)*old_VU_H)
        new_VE_L = max(keep_value_L, fire_value_L)

        # Check convergence
        diff = maximum(abs.([
            new_VE_H - VE_H,
            new_VE_L - VE_L,
            new_VU_H - VU_H,
            new_VU_L - VU_L
        ]))

        # Update
        VE_H, VE_L, VU_H, VU_L = new_VE_H, new_VE_L, new_VU_H, new_VU_L

        if diff < tol
            # We consider convergence
            # println("Value iteration converged in $(iter) iterations.")
            break
        end
    end

    # After convergence, extract the policy by seeing which choice is better
    # for each employed/unemployed state.
```

```

    # For (U,L) the policy is always "wait" because new_VU_L = 0 + E[VU],
    # and hiring is (L-w) + E[VE], which is typically negative if L < w.
    policyU_L = :wait

    # For (U,H):
    # Compare remain_unemployed_value vs. hire_value
    remain_unemployed_value = 0.0 +   * (p*VU_H + (1-p)*VU_L)
    hire_value              = (H_ - w) +   * (p*VE_H + (1-p)*VE_L)
    policyU_H = remain_unemployed_value >= hire_value ? :wait : :hire

    # For (E,H):
    keep_value_H = (H_ - w) +   * (p*VE_H + (1-p)*VE_L)
    fire_value_H = -hcost +   * (p*VU_H + (1-p)*VU_L)
    policyE_H = keep_value_H >= fire_value_H ? :keep : :fire

    # For (E,L):
    keep_value_L = (L_ - w) +   * (p*VE_L + (1-p)*VE_H)
    fire_value_L = -hcost +   * (p*VU_L + (1-p)*VU_H)
    policyE_L = keep_value_L >= fire_value_L ? :keep : :fire

    return (VE_H, VE_L, VU_H, VU_L,
            policyE_H, policyE_L, policyU_H)
end

######################################################################
# 3. Helper function to pretty-print the results
######################################################################

"""
    print_solution(params::ModelParams, solution)

Prints the value function results and corresponding policies in a neat format.
"""
function print_solution(params::ModelParams, solution)
    (VE_H, VE_L, VU_H, VU_L, policyE_H, policyE_L, policyU_H) = solution

    println("--------------------------------------------------------------")
    println(" Model Parameters:")
    println("    w       = ", params.w)
    println("    s_bar   = ", params.s_bar)
    println("    k       = ", params.k)
    println("    p       = ", params.p)
```

```
    println("           = ", params .)
    println("   H        = ", params.H)
    println("   L        = ", params.L)
    println("   hostility cost = max(0, k - s_bar) = ", hostility_cost(params))
    println()
    println(" DP Solution (Value Function Results):")
    println("   V_E(H) = ", VE_H)
    println("   V_E(L) = ", VE_L)
    println("   V_U(H) = ", VU_H)
    println("   V_U(L) = ", VU_L)
    println()
    println(" Optimal Policy:")
    println("   (E, H) -> ", policyE_H, "  (keep or fire when employed &  =H)")
    println("   (E, L) -> ", policyE_L, "  (keep or fire when employed &  =L)")
    println("   (U, H) -> ", policyU_H, "  (hire or wait when unemployed &  =H)")
    println("   (U, L) -> wait (no profitable hire at L)")
    println("--------------------------------------------------------------")
end


######################################################################
# 4. Run the DP for a range of p-values and plot results
######################################################################

"""
    run_param_sweep_and_plot(; s_bar, k, w, H, L,  , p_list)

Runs the dynamic program across a list of p-values, storing the resulting
optimal values of V_E(H) and V_E(L). Then produces a line plot to visualize
how the value function changes with p.
"""
function run_param_sweep_and_plot(;
    s_bar::Float64=1.0,
    k::Float64=2.0,
    w::Float64=1.0,
    H::Float64=2.5,
    L::Float64=0.5,
    ::Float64=0.95,
    p_list::Vector{Float64} = 0.5:0.05:0.99
)
    VEH_vals = Float64[]
```

```
    VEL_vals = Float64[]

    for p in p_list
        # Create model parameters for each p
        params = ModelParams(w, s_bar, k, p,  , H, L)
        sol = value_iteration(params)
        (VE_H, VE_L, VU_H, VU_L, policyE_H, policyE_L, policyU_H) = sol

        push!(VEH_vals, VE_H)
        push!(VEL_vals, VE_L)
    end

    # Now plot V_E(H) and V_E(L) vs. p
    plt = plot(
        p_list, VEH_vals,
        xlabel="p (prob of staying in the same state)",
        ylabel="Value Function",
        label="V_E(H)",
        title="V_E(H) and V_E(L) as functions of p",
        legend=true
    )
    plot!(p_list, VEL_vals, label="V_E(L)")

    return plt
end

######################################################################
# 5. Main
######################################################################

if abspath(PROGRAM_FILE) == @__FILE__
    # Example usage: run a single DP solution

    """
    ModelParams

    - w::Float64          : the (fixed) wage
    - s_bar::Float64      : the severance (bar{s})
    - k::Float64          : the worker's firing cost
    - p::Float64          : probability that   remains in the same state (H->H or L->L)
    -  ::Float64          : discount factor in (0,1)
    - H::Float64          : high productivity level
    - L::Float64          : low productivity level
    """
    myparams = ModelParams(
```

```
        1.0,    # w
        0.5,    # s_bar
        2.0,    # k
        0.5,    # p --> 0.5 = i.i.d. case
        0.95,   #
        2.5,    # H
        0.5     # L
    )

    solution = value_iteration(myparams)
    print_solution(myparams, solution)
    (VE_H, VE_L, VU_H, VU_L, policyE_H, policyE_L, policyU_H) = value_iteration(myparams)


    # Parameter sweep over p
    p_list = 0.5:0.05:1.0
    plt = run_param_sweep_and_plot(
        s_bar=0.5, k=2.0, w=1.0, H=2.5, L=0.5,  =0.95,
        p_list=collect(p_list)
    )

    p_list = 0.5:0.05:1.0

    # Example plot
    savefig(plt, "ValueFunction_vs_p.png")
end

end # module MarkovModelFinal
```

markov-model-qlV2.jl

```
# ############################################################################
# markov-model-qlV2.jl
#
# A file implementing Q-learning for the two-state
# Markov productivity environment. We assume that "ModelParams" and
# basic environment definitions are accessible (from
   markov-model-final.jl).
#
# Also generates plots showing the results.
#
# Boltzmann (softmax) exploration is used for action selection.
# We demonstrate how to:
#   1) Run Q-learning for "no hostility" (s_bar >= k => 0 hostility),
#   2) Run Q-learning for "with hostility" (s_bar < k => hostility
   cost > 0),
#
#   To run this file in Julia REPL:
#   1) include("markov-model-final.jl")
#   2) include("markov-model-qlV2.jl")
#   3) MarkovModelQL.demo_run()
#
#   Author: Marcus A. Lisman, Yale University
#   Written for CSEC 491 Senior Thesis
#
# ############################################################################

module MarkovModelQL

using Random, Printf, Statistics, Plots, Distributions
using CSV, DataFrames
using ..MarkovModelFinal: ModelParams, hostility_cost, print_solution
using Interpolations
using PrettyTables

# ############################################################################
# 0. - Housekeeping constants & helpers                                      #
# ############################################################################

# Figures I want to keep
const KEEP_FIGS = Set([
    "firing_comparison.png",
    "flip_threshold.png",
    "hostility_incidence.png",
    "payoff_comparison.png",
    "payoff_gap.png",
    "profit_surplus.png",
```

```
    "reward_variance.png",
    "surplus_ratio.png",
    "total_welfare.png",
    "worker_payoff.png",
    "profit_heat_unknown_kq.png",
    "fire_heat_unknown_kq.png",
    "profit_unknown_p.png",
    "fire_unknown_p.png",
])
const N_AVG = 1000

# Ensure output folders exist
mkpath("plots")
mkpath("data")

############# CSV helper #################
#=
    save_to_csv(fname; kw...)

Append vectors stored in keyword arguments to `fname` as a DataFrame.
If the file is absent it is created with headers.
=#
function save_to_csv(fname; kw...)
    df = DataFrame(; kw...)
    CSV.write(fname, df; append=isfile(fname))
end

############## Plot gate helper #################
#=
    maybe_plot(fname::AbstractString, plot_func, args...; kwargs...)

Run `plot_func(args...; filename="plots/$fname", kwargs...)`
*only* if `fname   KEEP_FIGS`.
=#

function maybe_plot(fname, plot_func, args...; kwargs...)
    if fname in KEEP_FIGS
        kwargs = merge(Dict(:filename=>"plots/$fname"), kwargs)
        plot_func(args...; kwargs...)
    end
    return nothing
end

function avg_qmetric(f::Function, params; N=15)
    tmp = [f(run_q_learning(params; rng=MersenneTwister(i),
                            num_episodes=num_ep, max_steps=ms)...)
           for i in 1:N]
```

```
    return mean(tmp)
end

# ###################################################################
# 1. Environment Setup for Q-Learning
# ###################################################################

"""
We define the firm 'environment' states as:

  (empStatus, alpha)   { (E, H), (E, L), (U, H), (U, L) }

Possible actions:
  - If (E, H): keep or fire
  - If (E, L): keep or fire
  - If (U, H): hire or wait
  - If (U, L): wait  (only 1 feasible action: do nothing / remain unemployed)

We'll index states and actions for Q-table as integers for convenience:

We can define:
   state_index(E,H) = 1
   state_index(E,L) = 2
   state_index(U,H) = 3
   state_index(U,L) = 4

   action_index(keep) = 1
   action_index(fire) = 2
   action_index(hire) = 1
   action_index(wait) = 2

We'll store Q[state, action].
"""

# define enumerations
@enum EmpStatus begin
    E   # employed
    U   # unemployed
end

@enum ProdState begin
    High   # H
    Low    # L
end

```

```
const ALL_STATES = [(E, High), (E, Low), (U, High), (U, Low)]


# Ensure the output folder exists:
mkpath("plots")

state_index(emp::EmpStatus,  ::ProdState)::Int =
    emp==E &&  ==High ? 1 :
    emp==E &&  ==Low  ? 2 :
    emp==U &&  ==High ? 3 : 4

action_space(emp::EmpStatus,  ::ProdState)::Vector{Symbol} = emp==E ? [:keep, :fire] :
    (==High ? [:hire, :wait] : [:wait])

action_index(a::Symbol)::Int = a===:keep  ? 1 :
                             a===:fire  ? 2 :
                             a===:hire  ? 1 :
                             a===:wait  ? 2 : error("Unknown action $a")

# ######################################################################
# 2. Softmax & Sampling
# ######################################################################

"""
    softmax(vals,  )

Boltzmann probabilities over `vals` at temperature  .
"""
function softmax(vals::AbstractVector{T},  ::T) where T<:Real
    if    1e-9
        probs = zero.(vals)
        probs[argmax(vals)] = one(T)
        return probs
    else
        exps = exp.(vals ./  )
        return exps ./ sum(exps)
    end
end

"""
    sample_from_probs(probs, rng)

Sample an index i with probability = probs[i].
"""
function sample_from_probs(probs::AbstractVector{T},
```

```
    rng::AbstractRNG=Random.GLOBAL_RNG) where T<:Real
    u = rand(rng)
    cum = cumsum(probs)
    return searchsortedfirst(cum, u)
end


# ######################################################################
# 3. Environment Step
# ######################################################################

function env_step(params::ModelParams, emp::EmpStatus,  ::ProdState, a::Symbol)
    p, w,¯s, k,  , H, L = params.p, params.w, params.s_bar, params.k, params., params.H, params.L
    h = max(0, k -¯s)

    # reward for the firm
    r = emp==E ? (a==:keep ? (==High ? H-w : L-w) : -h) :
        (emp==U && ==High && a==:hire ? (H-w) : 0.0)

    # next employment status
    next_emp = emp==E ? (a==:fire ? U : E) : (==High && a==:hire ? E : U)

    # alpha transition
    next_ = rand() < p ?   : (==High ? Low : High)

    return (r, next_emp, next_)
end

# ######################################################################
# 4. -QLearning
# ######################################################################

"""
    run_q_learning(params; ...)

Returns:
  Q, rewards, surplus, q_diff, fired_flag, fire_frac, temps, entropy
"""
function run_q_learning(
    params::ModelParams;
    num_episodes::Int=100,
    max_steps::Int=2,
    alpha_Q::Float64=0.1,
    gamma::Float64=0.95,
```

```
    temperature::Float64=1.0,
    temp_decay::Float64=0.999,
    rng::AbstractRNG=Random.GLOBAL_RNG
)
    Q = zeros(4,2)
    rewards   = zeros(num_episodes)
    surplus   = zeros(num_episodes)
    q_diff    = zeros(num_episodes)
    fired     = falses(num_episodes)
    fire_frac= zeros(num_episodes)
    temps     = zeros(num_episodes)
    entropy   = zeros(num_episodes)

    for ep in 1:num_episodes
        Q_old = copy(Q)
        total_r = 0.0
        total_w = 0.0
        did_fire = false
        fire_count = 0

         = temperature * (temp_decay^(ep-1))
        temps[ep] = 

        # entropy at (E,High)
        acts0 = action_space(E, High)
        vals0 = [Q[state_index(E,High), action_index(a)] for a in acts0]
        ps0   = softmax(vals0, )
        entropy[ep] = -sum(p>0 ? p*log(p) : 0.0 for p in ps0)

        emp,   = E, High

        for step in 1:max_steps
            sidx = state_index(emp, )
            acts = action_space(emp, )
            vals = [Q[sidx, action_index(a)] for a in acts]
            ps   = softmax(vals, )

            ai = sample_from_probs(ps, rng)
            a  = acts[ai]

            (r, emp2, 2) = env_step(params, emp, , a)

            # track worker payout
            if a in (:keep, :hire)
                total_w += params.w
            end
```

```julia

            # track firing events
            if a==:fire
                did_fire = true
                fire_count += 1
            end

            total_r += r

            # -Qupdate
            next_idx = state_index(emp2, 2)
            next_actions = action_space(emp2, 2)
            maxQ_next = maximum([Q[next_idx, action_index(a2)] for a2 in next_actions])
            td = r + gamma*maxQ_next - Q[sidx, action_index(a)]
            Q[sidx, action_index(a)] += alpha_Q * td

            emp,   = emp2, 2
        end

        rewards[ep]   = total_r
        surplus[ep]   = total_r + total_w
        q_diff[ep]    = maximum(abs.(Q .- Q_old))
        fired[ep]     = did_fire
        fire_frac[ep] = fire_count / max_steps
    end

    return (Q, rewards, surplus, q_diff, fired, fire_frac, temps, entropy)
end

# ###################################################################
# 5. Policy Extraction
# ###################################################################

function derive_policy(Q)
    pol = Dict{Tuple{EmpStatus,ProdState},Symbol}()
    for (i,st) in enumerate(ALL_STATES)
        acts = action_space(st...)
        qv   = [Q[i, action_index(a)] for a in acts]
        pol[st] = acts[argmax(qv)]
    end
    return pol
end

###################################################################
#    Print Summary for -Qlearning
```

```
####################################################################
"""
    print_qlearning_summary(params, Q, rewards, surplus, fired)

Console report analogous to 'DPs `print_solution`.
"""
function print_qlearning_summary(
    params::ModelParams,
    Q::AbstractMatrix,
    rewards::AbstractVector,
    surplus::AbstractVector,
    fired::AbstractVector
)
    pol = derive_policy(Q)
    println("--------------------------------------------------------------")
    println(" -QLEARNING SUMMARY")
    @printf("   Episodes              : %d\n", length(rewards))
    @printf("   Avg firm reward       : %.4f\n", mean(rewards))
    @printf("   Avg social surplus    : %.4f\n", mean(surplus))
    @printf("   Firing incidence      : %.2f %%\n", 100*mean(fired))
    println("   Learned greedy policy:")
    for st in ALL_STATES
        println("     ", st, " → ", pol[st])
    end
    println("--------------------------------------------------------------")
end

"""
    compare_dp_rl(params; num_runs=10, num_ep=200, max_steps=2)

Prints the DP solution and an averaged -Qlearning summary
for the same parameters.
"""
function compare_dp_rl(params::ModelParams;
                       num_runs::Int=10,
                       num_ep::Int=200,
                       max_steps::Int=2)

    # ---------------- DP ----------------------------------------------------
    println("\n====  DP (exact) ==============================================")
    dp_sol = value_iteration(params)
    print_solution(params, dp_sol)

    # ---------------- -Qlearning (averaged over seeds) -----------------------
```

```
    println("\n==== - QLEARNING  (average of $num_runs runs)
       ===================")
    rewards = Float64[]; surplus = Float64[]; fired = Float64[]
    Q_accum = zeros(4,2)

    for seed in 1:num_runs
        rng = MersenneTwister(seed)
        (Q, r, s, _, f, _, _, _) = run_q_learning(
            params;
            num_episodes = num_ep,
            max_steps    = max_steps,
            rng          = rng                # deterministic seed
        )
        Q_accum .+= Q
        append!(rewards, r)
        append!(surplus, s)
        append!(fired,   f)
    end

    Q_mean = Q_accum ./ num_runs
    print_qlearning_summary(params, Q_mean, rewards, surplus, fired)
end


# #
   ###################################################################
# 3.  Plotting
                                                                 #
# #
   ###################################################################

function plot_profit_surplus(pvals, profits, surpluses; filename)
    plt = plot(pvals, profits, label="Firm profit", lw=2,
               xlabel="p", ylabel="Value", title="Profit vs Social
                  Surplus")
    plot!(plt, pvals, surpluses, label="Social surplus", lw=2)
    savefig(plt, filename); println("saved → $filename"); plt
end

function plot_hostility_incidence(incid, pvals; filename)
    plt = plot(pvals, incid, lw=2,
               xlabel="p", ylabel="Frac episodes with fire",
               title="Hostility Incidence")
    savefig(plt, filename); println("saved → $filename"); plt
end

function plot_payoff_comparison(
```

```
    pvals, firm_noh, worker_noh, firm_h, worker_h; filename)
    plt = plot(pvals, firm_noh, label="Firm (no hostility)",
               xlabel="p", ylabel="Payoff",
               title="Firm & Worker Payoff", lw=2)
    plot!(plt, pvals, firm_h,   label="Firm (hostility)", lw=2)
    plot!(plt, pvals, worker_noh, linestyle=:dash,
          label="Worker (no hostility)", lw=2)
    plot!(plt, pvals, worker_h,  linestyle=:dash,
          label="Worker (hostility)", lw=2)
    savefig(plt, filename); println("saved → $filename"); plt
end

function plot_firing_comparison(pvals, fire_noh, fire_h; filename)
    plt = plot(pvals, fire_noh, label="Firing % (no hostility)",
               xlabel="p", ylabel="Fraction",
               title="Firing Fraction", lw=2)
    plot!(plt, pvals, fire_h, label="Firing % (hostility)", lw=2)
    savefig(plt, filename); println("saved → $filename"); plt
end

function plot_surplus_ratio(pvals, ratio_noh, ratio_h; filename)
    plt = plot(pvals, ratio_noh, label="No hostility", lw=2,
               xlabel="p", ylabel="Worker / Firm",
               title="Surplus Ratio")
    plot!(plt, pvals, ratio_h, label="Hostility", lw=2)
    savefig(plt, filename); println("saved → $filename"); plt
end

function plot_reward_variance(pvals, var_noh, var_h; filename)
    plt = plot(pvals, var_noh, label="No hostility", lw=2,
               xlabel="p", ylabel="Var(reward)",
               title="Reward Variance")
    plot!(plt, pvals, var_h, label="Hostility", lw=2)
    savefig(plt, filename); println("saved → $filename"); plt
end

function plot_worker_payoff(pvals, wpay_noh, wpay_h; filename)
    plt = plot(pvals, wpay_noh, label="Worker (no hostility)",
               xlabel="p", ylabel="Payoff",
               title="Worker Payoff", lw=2)
    plot!(plt, pvals, wpay_h, label="Worker (hostility)", lw=2)
    savefig(plt, filename); println("saved → $filename"); plt
end

function plot_payoff_gap(pvals, gap_noh, gap_h; filename)
    plt = plot(pvals, gap_noh, label="Gap (no hostility)",
               xlabel="p", ylabel="|-Firm  Worker|",
```

```
                 title="Payoff Gap", lw=2)
    plot!(plt, pvals, gap_h, label="Gap (hostility)", lw=2)
    savefig(plt, filename); println("saved → $filename"); plt
end

function plot_total_welfare(pvals, tot_noh, tot_h; filename)
    plt = plot(pvals, tot_noh, label="Welfare (no hostility)",
                 xlabel="p", ylabel="Total payoff",
                 title="Social Welfare", lw=2)
    plot!(plt, pvals, tot_h, label="Welfare (hostility)", lw=2)
    savefig(plt, filename); println("saved → $filename"); plt
end

function plot_flip_threshold(p_noh, p_h; filename)
    plt = scatter([p_noh, p_h], [0, 0],
                    label=["No hostility" "Hostility"],
                    xlabel="p", yticks=false,
                    title="Critical p for policy flip", markersize=8)
    savefig(plt, filename); println("saved → $filename"); plt
end


# ######################################################################
#    Value Iteration for V_E () and V_U ()
# ######################################################################

"""
    value_iteration(params::ModelParams; tol=1e-8, maxiter=1000)
        -> (VE_H, VE_L, VU_H, VU_L, policyE_H, policyE_L, policyU_H)

Solves the firm's infinite-horizon dynamic program using value iteration.

Returns a tuple:
- VE_H : V_E(H)
- VE_L : V_E(L)
- VU_H : V_U(H)
- VU_L : V_U(L)
- policyE_H : either :keep or :fire in state (E,H)
- policyE_L : either :keep or :fire in state (E,L)
- policyU_H : either :hire or :wait in state (U,H)

(The policy in state (U,L) is always "wait" = do not hire, because it would be
unprofitable to hire at L < w.)
"""
function value_iteration(params::ModelParams; tol=1e-8, maxiter=1000)
```

```

    # Extract for convenience
     = params .
    p = params.p
    w = params.w
    H_ = params.H
    L_ = params.L
    hcost = hostility_cost(params)  # k - s_bar if s_bar < k, else 0

    # Initialize guesses for value function
    # Track: V_E(H), V_E(L), V_U(H), V_U(L)
    VE_H, VE_L, VU_H, VU_L = 0.0, 0.0, 0.0, 0.0

    # We'll iterate until convergence
    for iter in 1:maxiter
        # Backup copies of old values
        old_VE_H, old_VE_L = VE_H, VE_L
        old_VU_H, old_VU_L = VU_H, VU_L

        # 1. Update V_U(L):
        #    If unemployed &  =L, can't profitably hire => payoff=0
        #    plus   * expectation of next state's V_U
        #    Next state  ' = L with prob p, or H with prob (1-p)
        new_VU_L = 0.0 +   * (p*old_VU_L + (1-p)*old_VU_H)

        # 2. Update V_U(H):
        #    If unemployed &  =H, two choices:
        #        (a) remain unemployed => immediate payoff=0
        #             plus   * E[next state's V_U]
        #        (b) hire => payoff=(H-w) +   * E[next state's V_E]
        #    We take the max of these
        remain_unemployed_value = 0.0 +   * (p*old_VU_H +
           (1-p)*old_VU_L)
        hire_value                 = (H_ - w) +   * (p*old_VE_H +
           (1-p)*old_VE_L)
        new_VU_H = max(remain_unemployed_value, hire_value)

        # 3. Update V_E(H):
        #    If employed &  =H, two choices:
        #        (a) keep => payoff=(H-w) +   * E[next state's V_E]
        #        (b) fire => payoff = -hcost +   * E[next state's V_U]
        keep_value_H = (H_ - w) +   * (p*old_VE_H + (1-p)*old_VE_L)
        fire_value_H = -hcost +   * (p*old_VU_H + (1-p)*old_VU_L)
        new_VE_H = max(keep_value_H, fire_value_H)

        # 4. Update V_E(L):
        #    If employed &  =L, two choices:
```

```
        #       (a) keep => payoff=(L-w) +   * E[next state's V_E]
        #       (b) fire => payoff=-hcost +   * E[next state's V_U]
        keep_value_L = (L_ - w) +   * (p*old_VE_L + (1-p)*old_VE_H)
        fire_value_L = -hcost +   * (p*old_VU_L + (1-p)*old_VU_H)
        new_VE_L = max(keep_value_L, fire_value_L)

        # Check convergence
        diff = maximum(abs.([
            new_VE_H - VE_H,
            new_VE_L - VE_L,
            new_VU_H - VU_H,
            new_VU_L - VU_L
        ]))

        # Update
        VE_H, VE_L, VU_H, VU_L = new_VE_H, new_VE_L, new_VU_H, new_VU_L

        if diff < tol
            # We consider convergence
            # println("Value iteration converged in $(iter) iterations.")
            break
        end
    end

    # After convergence, extract the policy by seeing which choice is better
    # for each employed/unemployed state.

    # For (U,L) the policy is always "wait" because new_VU_L = 0 + E[VU],
    # and hiring is (L-w) + E[VE], which is typically negative if L < w.
    policyU_L = :wait

    # For (U,H):
    # Compare remain_unemployed_value vs. hire_value
    remain_unemployed_value = 0.0 +   * (p*VU_H + (1-p)*VU_L)
    hire_value              = (H_ - w) +   * (p*VE_H + (1-p)*VE_L)
    policyU_H = remain_unemployed_value >= hire_value ? :wait : :hire

    # For (E,H):
    keep_value_H = (H_ - w) +   * (p*VE_H + (1-p)*VE_L)
    fire_value_H = -hcost +   * (p*VU_H + (1-p)*VU_L)
    policyE_H = keep_value_H >= fire_value_H ? :keep : :fire

```

```
    # For (E,L):
    keep_value_L = (L_ - w) +   * (p*VE_L + (1-p)*VE_H)
    fire_value_L = -hcost +   * (p*VU_L + (1-p)*VU_H)
    policyE_L = keep_value_L >= fire_value_L ? :keep : :fire

    return (VE_H, VE_L, VU_H, VU_L,
            policyE_H, policyE_L, policyU_H)
end

# # ####################################################################
#    DP-vs-Q-learning policy comparison for p = 0.50 : 0.05 : 0.99
# # ####################################################################
function compare_dp_q_policies(params::ModelParams;
                               ps=collect(0.5:0.05:0.99),
                               num_ep::Int=2000,
                               max_steps::Int=20,
                               num_runs::Int=30)

    comparison_tables = Dict{Float64, DataFrame}()
    state_labels = Dict(
        (E, High) => "Employed,   = H",
        (E, Low)  => "Employed,   = L",
        (U, High) => "Unemployed,   = H",
        (U, Low)  => "Unemployed,   = L"
    )

    for p in ps
        # Update model parameters
        par = ModelParams(params.w, params.s_bar, params.k, p, params., params.H, params.L)

        # ---------------- DP policy ----------------
        _, _, _, _, pol_E_H, pol_E_L, pol_U_H = value_iteration(par)
        dp_policy = Dict(
            (E, High) => pol_E_H,
            (E, Low)  => pol_E_L,
            (U, High) => pol_U_H,
            (U, Low)  => :wait
        )

        # ---------------- Q-learning policy (averaged) ----------------
        Q_accum = zeros(4,2)
        for seed in 1:num_runs
            rng = MersenneTwister(seed)
```

```
            Q, _, _, _, _, _, _, _ = run_q_learning(
                par; rng=rng,
                num_episodes=num_ep, max_steps=max_steps
            )
            Q_accum .+= Q
        end
        Q_avg = Q_accum ./ num_runs
        q_policy = derive_policy(Q_avg)

        # ---------------- Assemble comparison table ----------------
        rows = Vector{NamedTuple}()
        for st in ALL_STATES
            push!(rows, (
                State     = state_labels[st],
                DP_Action = dp_policy[st],
                Q_Action  = q_policy[st]
            ))
        end
        df = DataFrame(rows)
        comparison_tables[p] = df

        println("\n Policy comparison for p = $(round(p, digits=2))")
        pretty_table(df; tf=tf_unicode, alignment=:l)
    end

    return comparison_tables
end

function demo_run()
    # baseline parameters
    w,¯s, k,  , H, L = 1.0, 0.5, 2.0, 0.95, 2.5, 0.5  # baseline 0.5 for severance
    num_ep, ms = 100, 2
    Q,  , 0, decay = 0.1, 0.95, 1.0, 0.999

        """
            avg_metric(f, params; episodes=num_ep, steps=ms)

        Returns the average of `f(run_q_learning(...))` across `N_AVG` seeds.
        `f` is a function that extracts the statistic you want from the-
        8tuple returned by `run_q_learning`.
        """
        function avg_metric(f::Function, params::ModelParams;
                            episodes::Int=num_ep, steps::Int=ms)
            vals = Float64[]
```

```
            for seed in 1:N_AVG
                rng = MersenneTwister(seed)
                stats = run_q_learning(params;
                            rng=rng, num_episodes=episodes, max_steps=steps,
                            alpha_Q =Q, gamma =, temperature =0, temp_decay=decay)
                push!(vals, f(stats))
            end
            return mean(vals)
        end

    pvals = collect(0.5:0.05:0.99)

    # ---------------------------------------------------------------#
    # 1)  Profit & Surplus sweep                                     #
    # ---------------------------------------------------------------#
    profits    = [avg_metric(x -> mean(x[2]),     # 2 = rewards
                            ModelParams(w,¯s, k, p,  , H, L))
                  for p in pvals]

    surpluses  = [avg_metric(x -> mean(x[3]),     # 3 = surplus
                            ModelParams(w,¯s, k, p,  , H, L))
                  for p in pvals]

    maybe_plot("profit_surplus.png", plot_profit_surplus, pvals,
               profits, surpluses)
    save_to_csv("data/profit_surplus.csv";
        p = pvals, firm_profit = profits, social_surplus = surpluses)

    # ---------------------------------------------------------------#
    # 2)  Hostility incidence                                        #
    # ---------------------------------------------------------------#
    incid = [avg_metric(x -> mean(x[5]),          # 5 = fired (Bool vector)
                        ModelParams(w,¯s-1e-6, k, p,  , H, L))
             for p in pvals]

    maybe_plot("hostility_incidence.png", plot_hostility_incidence,
               incid, pvals)
    save_to_csv("data/hostility_incidence.csv"; p = pvals, fired_frac = incid)

    # ---------------------------------------------------------------#
    # 3)  Hostility vs -Nohostility comparisons                      #
    # ---------------------------------------------------------------#
    n = length(pvals)
```

```
    firm_noh = zeros(n); worker_noh = zeros(n); fire_noh = zeros(n)
    firm_h   = zeros(n); worker_h   = zeros(n); fire_h   = zeros(n)

    for (i,p) in enumerate(pvals)
        params_noh = ModelParams(w, max(k̄,s), k, p,  , H, L)
        firm_noh[i] = avg_metric(
            x -> mean(x[2]),                     # rewards
            params_noh)

         # Worker payoff = surplus - reward
        worker_noh[i] = avg_metric(
            x -> mean(x[3]) - mean(x[2]),        # 3-  2
            params_noh)

        # Firing fraction
        fire_noh[i] = avg_metric(
            x -> mean(x[5]),                     # fired Bool
            params_noh)

        params_h = ModelParams(w, min(k̄,s-1e-6), k, p,  , H, L)

        firm_h[i] = avg_metric(
            x -> mean(x[2]),                     # rewards
            params_h)

         # Worker payoff = surplus - reward
        worker_h[i] = avg_metric(
            x -> mean(x[3]) - mean(x[2]),        # 3-  2
            params_h)

        # Firing fraction
        fire_h[i] = avg_metric(
            x -> mean(x[5]),                     # fired Bool
            params_h)
    end

    maybe_plot("payoff_comparison.png", plot_payoff_comparison,
               pvals, firm_noh, worker_noh, firm_h, worker_h)
    maybe_plot("firing_comparison.png", plot_firing_comparison,
               pvals, fire_noh, fire_h)

    save_to_csv("data/payoff_comparison.csv";
        p = pvals,
        firm_noh = firm_noh, worker_noh = worker_noh,
        firm_h   = firm_h,   worker_h   = worker_h)

    save_to_csv("data/firing_comparison.csv";
```

```
    p = pvals, fire_noh = fire_noh, fire_h = fire_h)

# ---------------------------------------------------------------#
# 4)  Additional line plots                                      #
# ---------------------------------------------------------------#
ratio_noh = worker_noh ./ firm_noh
ratio_h   = worker_h   ./ firm_h
var_noh   = fire_noh
var_h     = fire_h

# variance in episode reward (need separate sweep)
var_noh .= 0;  var_h .= 0
for (i,p) in enumerate(pvals)
    params_noh = ModelParams(w, max(k̄,s), k, p,  , H, L)
    (_, r_noh, _, _, _, _, _, _) = run_q_learning(params_noh;
        num_episodes=num_ep, max_steps=ms,
        alpha_Q =Q, gamma =, temperature =0, temp_decay=decay)
    params_h = ModelParams(w, min(k̄,s-1e-6), k, p,  , H, L)
    (_, r_h, _, _, _, _, _, _) = run_q_learning(params_h;
        num_episodes=num_ep, max_steps=ms,
        alpha_Q =Q, gamma =, temperature =0, temp_decay=decay)
    var_noh[i] = var(r_noh)
    var_h[i]   = var(r_h)
end

maybe_plot("surplus_ratio.png",   plot_surplus_ratio,
           pvals, ratio_noh, ratio_h)
maybe_plot("reward_variance.png", plot_reward_variance,
           pvals, var_noh, var_h)

save_to_csv("data/surplus_ratio.csv";
    p = pvals, ratio_noh = ratio_noh, ratio_h = ratio_h)
save_to_csv("data/reward_variance.csv";
    p = pvals, var_noh = var_noh, var_h = var_h)

# ---------------------------------------------------------------#
# 5)  Worker payoff & welfare & gap                              #
# ---------------------------------------------------------------#
gap_noh = abs.(firm_noh .- worker_noh)
gap_h   = abs.(firm_h   .- worker_h)
tot_noh = firm_noh .+ worker_noh
tot_h   = firm_h   .+ worker_h

maybe_plot("worker_payoff.png",   plot_worker_payoff,
           pvals, worker_noh, worker_h)
maybe_plot("payoff_gap.png",      plot_payoff_gap,
           pvals, gap_noh, gap_h)
```

```
    maybe_plot("total_welfare.png",   plot_total_welfare,
               pvals, tot_noh, tot_h)

    save_to_csv("data/worker_payoff.csv";
        p = pvals, worker_noh = worker_noh, worker_h = worker_h)
    save_to_csv("data/payoff_gap.csv";
        p = pvals, gap_noh = gap_noh, gap_h = gap_h)
    save_to_csv("data/total_welfare.csv";
        p = pvals, welfare_noh = tot_noh, welfare_h = tot_h)

    # ---------------------------------------------------------------#
    # 6) - Flipthreshold (critical p where policy switches in (E,L)) #
    # ---------------------------------------------------------------#
    function first_fire(params_template)
        for p in pvals
            par = params_template(p)
            Q, = run_q_learning(par; num_episodes=100, max_steps=ms)
            pol = derive_policy(Q)
            pol[(E,Low)] == :fire && return p
        end
        return NaN
    end
    p_noh = first_fire(p -> ModelParams(w, max(k,̄s), k, p,  , H, L))
    p_h   = first_fire(p -> ModelParams(w, min(k,̄s-1e-6), k, p,  , H,
       L))

    maybe_plot("flip_threshold.png", plot_flip_threshold, p_noh, p_h)
    save_to_csv("data/flip_threshold.csv"; p_noh = [p_noh], p_h =
       [p_h])

    # --------------------------------------------------------------#
    # 7)  Print a representative -Qlearning run                     #
    # --------------------------------------------------------------#
    params_demo = ModelParams(w,̄s, k, 0.7,  , H, L)
    (Q_demo, r_demo, s_demo, _, fired_demo, _, _, _) =
        run_q_learning(params_demo; num_episodes=100, max_steps=ms,
                       alpha_Q =Q, gamma =, temperature =0,
                          temp_decay=decay)
    print_qlearning_summary(params_demo, Q_demo, r_demo, s_demo,
       fired_demo)

    # Print sweep over p
    println("\n================ SUMMARY SWEEP (DP vs RL)
       =======================")
    for p in pvals
        println("\n----------------  p = $(round(p, digits=2))
           -------------------")
```

```
        params_sweep = ModelParams(w,¯s, k, p,  , H, L)
        compare_dp_rl(params_sweep; num_runs=15, num_ep=100,
           max_steps=ms)
    end

    println("demo_run complete - figures saved in plots/, data in CSV
       files.")


    params = ModelParams(1.0, 0.5, 2.0, 0.95, 0.95, 2.5, 0.5)
    policy_tables = compare_dp_q_policies(params)

    println("DP vs Q-learning tables finished.")
end

end # module MarkovModelQL
```