אפיון סוכנים

# Black Forest (MCTS, MB)

* מבוסס **מציאת מדיניות** ובחירת פעולה אופטימלית, כמו בבעיית MDP.
* פועל בצורה שונה **במקצת** בהתאם למצב (למידה או פעולה) בו הסוכן נמצא (כמה קבועים במערכת משתנים). עם זאת, הדברים שהסוכן מבצע בזמן למידה **לא** שונים באופן מהותי מאלו הקורים בזמן ביצוע.
* אין הדבר כתוב בדוח, אך **ההערכה** של כל play-out מתרחשת לפי הדרך בה הוא **מסתיים** (הגעה למספר הצעדים המקסימלי של ה-play-out, הגעה למצב המטרה, הגעה למצב בו אין פעולות וכולי). בקוד עצמו, נבחרו כמה קבועים **שלא** ברור כיצד הושגו.
* לפרס יש פחות השפעה על צמתות רחוקות יותר מצומת הסיום.

# Jaffa (2022, reactive planner, MB)

* אינו משתמש בלמידה כלל ואינו לומד מדיניות.
* מפשט את המודלים לפני תחילת התכנון. במודלים הסתברותיים, מייצר גרסאות שונות של המודל כדי לאפשר הנחה של דטרמינסטיות במהלך התכנון.
* מבצע בכל מצב את הפעולה שניתנה על ידי ה-planner

# Brownie (Reactive Planner, MB)

* פועל על ידי שימוש **בתכנון** תגובתי (כלומר, כאשר פעולה מסוימת משפיעה על הסביבה באופן שאינו צפוי, הסוכן מתכנן מחדש)
* **משטיח** את קובץ הסביבה לקובץ **דטרמיניסטי**, תוך כדי השמטה של השפעות אפשריות של פעולות בעלות הסתברויות התרחשות נמוכות ושמירת כמה עותקים של אותה הפעולה, כאשר כל אחד בעל השפעה שונה על הסביבה, בהתאם להשפעה אחת אפשרית של הפעולה.
* **דורש** פרק זמן של למידה כדי לעבוד.
* מוצא את ההסתברויות ואת המידע הנסתר **בנפרד** (זה אחר זה).

# Crystal (Reactive Planner, MB)

* כולל שלושה מצבים:
  + מצב **תכנון**, שבו הסוכן מנסה לעקוב אחר תוכנית דטרמיניסטית, הכוללת פעולות, שהן גרסאות דטרמיניסטיות של פעולות לא דטרמיניסטיות, הכוללת השפעה דטרמיניסטית אחת שלהן. הסוכן בוחר בכל שלב בפעולה הלא דטרמיניסטית בעלת ההסתברות (בקירוב) הגדולה ביותר להניב את התוצאה הרצויה (בערך).
  + מצב **התגובה**, שבו הסוכן מגיב לסטייה מן המצב הצפוי (הנדרש להמשך התוכנית). במידה והמצב שנצפה זהה לקודם, הסוכן מנסה שוב ואחרת מתכנן מחדש.
  + מצב **החקירה**, שבו הסוכן פועל באופן אקראי (ואחיד) (כדי לנסות לגלות מידע חדש וחבוי על הסביבה. במידה ולאחר גילוי מידע חדש, הסוכן מצא תוכנית אשר יכולה להביא אותו לפתרון הבעיה, הסוכן מפסיק לחקור.
* הסוכן מתחיל במצב החקירה ואמור לסיים במצב התכנון.
* **דורש** פרק זמן של למידה כדי לעבוד.
* לא צריך למצוא את ההסתברויות של דבר, שכן הוא נוצר בזמן בו ההסתברויות היו נתונות לסוכנים.

# Ostkaka (Reactive RL, MF)

* בזמן למידה, משתמש **בלמידת Q** עם פונקצית פרס כללית ובסיסית יחסית (מלבד העובדה שהיא עם הזמן מתגמלת מצבים המזוהים עם הצלחה).
* הלמידה משמשת את הסוכן להכנת "גרף ידע", שבו הוא משתמש לחישוב הדרך המהירה ביותר לפתרון הבעיה, **כמעין planner**. הגרף מורכב אך ורק ממצבים שהסוכן כבר ביקר בהם.
* **אין** שום ניסיון **ללמוד מידע חבוי** והסוכן מסתמך על גרף הידע שלו בלבד למיטב הבנתי.
* בזמן ביצוע, הסוכן משתמש בגרף ידע יחד עם אלגוריתם למציאת הדרך הקצרה ביותר לקודקוד מסוים ופועל לפי הפעולות בגרף. סטייה מהדרך הרגילה תגרום **לחישוב מסלול מחדש** במידה והמצב החדש מופיע בגרף, או בחירת פעולה **אקראית**. הסוכן בפועל משתמש **במדיניות** המשתנה בהתאם לדרך שהסוכן מצא.

# Funnel (Q-learning, MF)

* הסוכן מממש את אלגוריתם **למידת ה-Q**, עם שימוש ב-**Epsilon Greedy** למען בחירת הפעולות. הסוכן לכן **דורש** זמן למידה מסוים, כפי שניתן לנחש.
* הסוכן **לא** מנסה ללמוד מידע חבוי כלל, אך משתמש בפונקצית תגמול כללית, בעלת היוריסטיקות.
* הסוכן משתמש בלמידת Q **טהורה**, ללא שינויים.
* הסוכן משתמש **בתכנון** עבור תחומים **דטרמיניסטיים**.

# Dobos (Planner-policy, MB)

* משתמש באלגוריתם PRP, ראשי תיבות של Planner for relevant policies, כדי ליצור **מדיניות**, שבמסגרתה מאותרים כל זוג של מצב-פעולה אשר ממנו ניתן להגיע **בוודאות** למצב המטרה (או מצבי המטרה).
* האלגוריתם גם מייצר **היוריסטיקה** של **המרחק** מכל זוג פעולה-מצב למטרה. היוריסטיקה זו משמשת אותו בתהליך בחירת הפעולה עבור כל צומת החלטה. הסוכן יבחר בפעולה בעלת המרחק **הגדול** **ביותר** למטרה.
* הסוכן **דורש** זמן למידה לפיכך.

# Mango (MCTS (MB)+BQL (MF), MF)

* מורכב משני סוכנים שונים (סוכן מבוסס MCTS וסוכן מבוסס BQL), יחד עם מערכת תכנון דטרמיניסטית.
* **בוחר** בסוכן המתאים ביותר לכל סיטואציה (תכנון עבור תחומים דטרמיניסטיים) ו-BQL או MCTS **בהתאם** **לביצועים** של שניהם (או ב-BQL **במידה ואין זמן למידה**). באופן כללי, MCTS פועל בצורה טובה יותר בתחומים בעלי מרחב מצבים גדול יותר.
* גם סוכן ה-MCTS וגם סוכן ה-BQL משתמשים בפונקצית פרס, מבוססת היוריסטיקות בסיסיות, **ללא** שיקולי avoid the past (בערך, לא לגמרי), שאמורה להתאים למגוון בעיות שונות.
* **אין** ניסיון ללמוד מידע חבוי והסוכן **דורש** זמן למידה מסוים.

# Carrot (HFF Policy, MB)

* משתמש במתכנן עבור תחומים דטרמיניסטיים או בדירוג כל פעולה על ידי **מטה-היוריסטיקה** מבוססת HFF (עם פעולות **ללא** **dellist**, למען **הרגעת** הבעיה), יחד עם ממוצע משוקלל בתחומים שאינם דטרמיניסטיים (כלומר, בכל תחום רלוונטי).
* יוצר מעין **מדיניות** מבוססות מטריקה של "קרבה" לפתרון בפועל ולא "מתכנן" באמת עבור בעיות לא דטרמיניסטיות. המטריקה לרוב תנוצל לחישוב תוחלת כפי שנכתב לעיל.

# Flan (Q-learning, MF)

* משתמש בלמידת Q, עם פונקצית פרס שמנסה **להתחמק מן העבר ומלולאות**. הפונקציה עם זאת **לא** מתייחסת **לשהייה ללא השגת מטרה** למשל.
* **אין** שום ניסיון מפורש ללמוד מידע חבוי.
* הסוכן בעת למידה לפחות, בוחר בין **חקירה** ולבין **ניצול** ידע קיים.
* הסוכן כפי שניתן לנחש **דורש** זמן למידה מסוים.

# Lemon (2021, Planning, Q-Learning, MB)

* משטח את הדומיין לדטרמיניסטי ולומד מדיניות (הלמידה עצמה מתרחשת רק כשהסוכן קרוב למצוא מדיניות אופטימלית).
* בשלב הלמידה, משתמש באלגוריתם exploration על מנת לבדוק דטרמיניסטיות וכדי למצוא מידע חבוי, אם קיים. אלגוריתם ה-exploration עוקב אחרי מספר הפעמים שבוצעה כל פעולה באמצעות טבלת מונים.
* במקרה של סטייה מהתכנון, יווצר קובץ דומיין חדש שהמצב ההתחלתי בו הוא המצב הנוכחי. בזמן ריצה במודלים הדטרמיניסטים נלמדת טבלת מעברי מצבים.
* בשלב הריצה, אם לא בוצעה למידה על דומיין זה, יבצע פעולות רנדומליות בהתאם לטבלת מונים חדשה. הפלאנר ישתמש בקובץ דומיין דטרמיניסטי שבו האפקט של כל פעולה הוא האפקט בעל ההסתברות הגבוהה ביותר בפעולה המקורית.

# Chocolate (2021, Varies)

* לסוכן העל מנהל הרצה שמחלק את הבעיות השונות לשלושה תת-סוכנים אפשריים:
* Dynamic Q-Learning - למידת Q שמוסיפה בכל פעם רק את הפעולות והמצבים הרלוונטיים האפשריים
* BehaviorBase - בונה גרף מצבים ומבצע בו חיפוש רוחבי
* Lookahead - בודק כמה צעדים קדימה ובוחר באופציה המועדפת
* סוכן העל ממיין את הבעיות לפי תכונות שונות שלהן.

# Lolly (2021, Planning (fast-downward planner), MB)

* מפשט את הדומיין לדטרמיניסטי על ידי בחירת האפקט בעל ההסתברות הגבוהה ביותר לכל פעולה. על מנת לחשוף מידע חבוי מתבצע תכנון שהמטרה בו היא התנאי לחשיפת המידע החבוי.
* fast-downward planner מקבל כפרמטר איזה אלגוריתם חיפוש לבצע בגרף המצבים (a-star, lazy-greedy ועוד). בעת הלמידה של דומיין מסוים, מריץ אלגוריתם חיפוש רנדומלי שעוד לא הורץ על דומיין זה, ומייצא את המסקנות לקובץ json.
* בעת הריצה משתמש באלגוריתם הטוב שהניב את התוצאות המוצלחות ביותר לבעיה.

# Suncake (2021, Q-learning + state table, MF)

* משתמש בלמידת Q ולומד טבלת מצבים, כאשר הלמידה מגיעה למבוי סתום או ללולאה אינסופית משתמש בטבלת המצבים.
* לומד את הסתברויות המעבר ומשתמש בהן בזמן עדכון ערכי ה-Q (כסוג של שקלול הפרס המקסימלי האפשרי בהתאם להסתברות שפרס זה יתקבל).

# Pandan (2021, Planning, MB)

* מפשט את הדומיין לדטרמיניסטי מתכנן תוכנית באמצעות planner, ועוקב אחריה עד שהדבר לא מתאפשר.
* מתכנן תוכנית חדשה על הדומיין המפושט הנוכחי, עוקב אחריה וחוזר חלילה.

# Gingerbread (2021, Dyna Q (inactive), Q learning, MB)

* משלב מודלים עם למידה על ידי תהליך של למידה ולאחר מכן למידת הסביבה.
* *(בשלב האחרון של כל איטרציה נבחרים פעולה רנדומלית ומצב רנדומלית מספר מוגדר של פעמים, ועל ידי ביצוע פעולות אלה במצבים אלה הסוכן מעדכן את טבלת ה-Q.)* באופן דיפולטיבי אין בתכונה זאת שימוש.
* הסוכן שומר גם את מספר הפעמים שפעולה בוצעה בנוסף לתגמול שהרוויחה, על מנת ללמוד את ההסתברויות בדומיינים לא דטרמיניסטים.
* במידה ולא בוצעה למידה על בעיה מסוימת:
* אם אין בה מידע חבוי - מפעיל planner באופן חזרתי.
* אם יש בה מידע חבוי - מבצע פעולות באופן רנדומלי.

# Mooncake (Q-learning, MF)

* משתמש בלמידת Q למען יצירת סוכן. סוכן זה, כמו סוכני למידת Q רבים אחרים, **לא** מנסה למצוא את **המידע החבוי** באופן מפורש. עם זאת, האלגוריתם **מונע מעגלים**, (הן בזמן ריצה והן בזמן למידה), על ידי שימוש במזהה מעגלים, המחזיר את הסוכן למצב שבו תפקד **בתחילת תהליך הלמידה** (דבר הקורה גם בזמן ההרצה (דבר שאני משער כי ודאי יצר בעיות רבות)).
* בזמן הלמידה, האלגוריתם **חוקר את הסביבה** עם אלגוריתם ה-Epsilon Greedy (וגם בזמן ריצה במידת הצורך).
* הסוכן משתמש בפרמטרים **הממזערים** את כמות הצעדים הנדרשת עבור בעיה **שרירותית** לחלוטין שיוצר הסוכן התמקד בה ולא את כמות הצעדים בבעיות אחרות בממוצע (או בחציון). עם זאת, הפרמטרים לא מוזערו על ידי שימוש בחיפוש רשת.

# Apple (Reactive Planner, MB)

* הסוכן מבצע **חיפוש** על ידי שימוש **בהיוריסטיקת** **המידע** **החדש** (כלומר, ישנה העדפה למציאת פרדיקטים חדשים) ובתור דרך להתמודדות עם שוויונות, הסוכן **יעדיף** פעולות שיכולות להוביל למצבים בהם ניתן לבצע פעולות שעזרו לסוכן לגלות **מידע** **חבוי** **בעבר**.
* לפי כותב הסוכן, היוריסטיקה זו הושפעה על ידי האופן שבו **בני אדם** חוקרים את העולם סביבם. במהלך חיפוש זה, הסוכן ינסה למצוא, כפי שניתן לנחש, מידע חבוי.
* כאשר הסוכן יכול להשתמש בידע שלו אודות העולם כדי להכין תוכנית, הוא יפסיק ללמוד על העולם סביבו ויתחיל בשלב הריצה.
* בזמן ריצה, הסוכן משתמש בתכנון תגובתי. הסוכן מפשט את התחום לתחום דטרמיניסטי, שבו כל פעולה בעלת כמה השפעות אפשריות על העולם **הופכת** לפעולה חדשה אחרת, שבה רק השפעה אחת יכולה להתרחש. ביצירת התוכנית שלו, הסוכן **ראשית** מסתכל רק על פעולות שההשפעה שלהן **סבירה** (בערך, הוא מביא לכל פעולה דירוג יותר מורכב ובוחר בפעולה עם הדירוג הרב ביותר בפועל)ובמידה והדבר לא יצליח, יסתכל הסוכן על כל הפעולות האפשריות.
* כאשר הסוכן סוטה משביל התכנון, הסוכן ינסה לחזור למצב **הנוכחי** בתוכנית על ידי שימוש ביצירת "תוכנית" באורך של K (היפר-פרמטר). במידה והוא נכשל בכך, הוא יחשב תוכנית מחדש.
* במידה והתחום דטרמיניסטי, ישתמש הסוכן במתכנן המובנה של הסימולטור ו**לא** ילמד כלל.

# Banana (Reactive Planner + Q-learning, MB)

* הסוכן מחולק לשני תת-סוכנים (וכמובן, המתכנן המובנה של הסימולטור, הידוע לשמצה):
  + סוכן מבוסס למידת Q המעדכן את ערכי ה-Q של זוגות (מצב, פעולה) **כמה** צעדים אחורה, הרץ במקרים בהם המתכנן המובנה של הסימולטור לא מצא תוכנית, אך התחום בו הסוכן רץ הוא **דטרמיניסטי**
  + סוכן מבוסס תכנון ריאקטיבי, לשאר המצבים האפשריים (כלומר, הפועל בתחומים שאינם דטרמיניסטיים)
* סוכן למידת ה-Q משתמש באלגוריתם ה-Epsilon Greedy ללא ניסיון למניעת לולאות.
* סוכן התכנון הריאקטיבי **מפשט** את התחום לתחום הכולל פעולות בעלות השפעה אחת בלבד על העולם, כאשר ההשפעה הנבחרת לכל פעולה היא ההשפעה **הסבירה** **ביותר** (דבר שיכול לגרום לכשל בפתרון בעיות מסוימות). כמו כן, הסוכן יוצר בעיה חדשה, בה כל המידע החבוי **ידוע** **מראש** (דבר שהוא, לפחות לדעתי, ללא ספק כלל, **רמאות**). כאשר יש סטייה מן התוצאה הצפויה לפי התוכנית, שהסוכן יוצר עם הבעיה והתחום החדשים, בעזרת המתכנן המובנה, הסוכן יתכנן מחדש.

# Batik (Q-learning +BM Exploration, MF)

* סוכן זה, כסוכנים אחרים, מורכב מכמה סוכנים שונים:
  + המתכנן המובנה של הסימולטור משומש כאשר התחום הוא **דטרמיניסטי**.
  + הסוכן פועל באופן **אקראי**, ללא חזרה על הפעולות שלו, כאשר הוא במצב **ריצה** ו**אינו** מכיר את הבעיה (ואו הסביבה) עליה הוא רץ.
  + הסוכן משתמש בלמידת Q, עם חקר על ידי פילוג בולצמן, כאשר הוא במצב למידה. במקרה זה, הסוכן משתמש בסוכן החקר 30 פעמים ברצף ולאחר מכן מסיים ללמוד. כאשר הוא כמובן מוצא את המדיניות המתאימה, הוא ישתמש בה בזמן ריצה.
* פילוג הבולצמן שומר על **איזון** **טוב** **יותר** בין ניצול לחקר, בצורה המאפשרת לסוכן לפעול בצורה **דומה** **לאופטימלית** בסביבה שאינו מכיר. כלומר, הוא מסוגל לנצל את המיטב מהמצב העגום בו נמצא.
* במידה והסוכן ביצע פעולה "טובה מאוד" או "רעה מאוד", כלומר אם איבד או השיג מטרה נוספת, הוא יריץ עדכון ערכי Q על **כל** המצבים (והפעולות) אשר הביאו לתוצאה הזו.
* הסוכן משתמש בהיפר-פרמטרים שנבחרו כדי למזער את מספר הצעדים הנדרשים כדי לפתור בעיה **שרירותית** שכותבת הסוכן בחרה.
* פונקצית הפרס של הסוכן מעודדת פתרונות קצרים ומשתמשת בהיוריסטיקת ה-"avoid the past".

# Spit (-learning, MF)

* הסוכן מורכב מכמה "סוכנים" שונים:
  + המתכנן המובנה משומש במקרים בהם התחום הוא דטרמיניסטי.
  + סוכן **אקראי**, עם היוריסטיקת ה-"avoid the past" משומש כאשר לסוכן אין מדיניות בגין צירוף בעיה ותחום (וכאשר **אין** לו זמן **למידה**, כמובן).
  + סוכן מבוסס למידת , עם חקר מבוסס התפלגות בולצמן ופונקצית פרס מבוססת היוריסטיקות ה-"avoid the past" ואובדן (או השגת) המטרה, מופעל ליצירת מדיניות כאשר הסוכן במצב **למידה**.
* הסוכן **לא** מנסה באופן מפורש ללמוד מידע חבוי ויתר על כן, מריץ את המתכנן המובנה גם כאשר **ישנה** אפשרות למידע חבוי!
* הסוכן מריץ בעת למידה מספר סוכנים שונים **במקביל**, עם היפר-פרמטר טמפרטורה שונה, למשך זמן רב. בסיום ההליך תוצאות הטבלאות של כל סוכן נאגרות והמדיניות הסופית מוחלטת על ידי לקיחת **ממוצע** של ערכי הטבלאות.

# Blackout (Heuristic Learning, MF)

* הסוכן **לא** משתמש בלמידת Q, אלא לומד את המדיניות שלו על סמך חישוב היוריסטיקה של **מרחק** **מהמטרה** (המצב הסופג הרצוי).
* כאשר הסוכן, במצב ריצה, מעוניין לבחור פעולה, הוא משתמש במידע המאוחסן במדיניות שלו, שהוא המרחק הקצר ביותר, מכל מצב, למטרה. הוא יבחר בפעולה שתוחלת המרחק הקצר ביותר שלה למטרה הוא הנמוך ביותר (ערך המכונה באופן מבלבל ביותר "פרס" בדוח).
* הסוכן חוקר את העולם בצורה **אקראית**, עם העדפה לפעולות **שלא** **ניסה** **בעבר**. הסוכן **לא** מנסה באופן מיוחד למצוא מידע חבוי ו**לא** מנסה, מעבר לכתוב, להתמודד עם לולאות (למרות שהוא יבחר בין הפעולות הטובות ביותר בצורה המנסה לחקור פעולות חדשות).
* במידה והסוכן במצב ריצה ואין לו מדיניות, הוא יפעל בניסיון לחקור את סביבתו, כמתואר לעיל (קרי, באופן **אקראי**…).

# Cornbread (Q-learning, MF)

* הסוכן משתמש בלמידת Q, כאשר התחום **אינו** דטרמיניסטי (או כולל מידע חבוי) ובמתכנן המובנה של הסימולטור בכל מקרה אחר.
* **לפי הדוח**, הסוכן, הן במצב ריצה והן במצב למידה, בעל סיכוי נמוך מאוד לפעול בצורה אקראית, כדי למנוע לולאות.
* כאשר הסוכן לומד, **כל** מצב נחשב "שלילי", למעט מצבים בהם הסוכן השיג או איבד מטרה. במצבים כאלה, כל המצבים שהסוכן **עבר** כדי להגיע למצב זה מקבלים פרס (חיובי או שלילי, בהתאם לאובדן, או השגת המטרה).
* הסוכן יפעל באופן אקראי במידה ואין לו מדיניות לעקוב אחרייה (אך כמובן יעדיף זוגות של (מצב, פעולה) חדשים.

# Kabuni (Q-learning, MF)

* סוכן זה מורכב משלושה סוכנים שונים:
  + סוכן המשתמש במתכנן המובנה של הסימולטור, אשר מסוגל לרוץ **אך ורק** בתחומים דטרמיניסטיים.
  + סוכן המבצע למידת Q, אך הינו **off-policy** (אין פירוט נוסף בדוח). ניתן להניח עם זאת כי הוא מבצע מימוש בסיסי ופשטני של האלגוריתם.
  + סוכן המממש את אלגוריתם ה-SARSA. כלומר, הוא מממש מעין גרסה **on-policy**-ית של אלגוריתם למידת ה-Q.
* לסוכן ישנו בקר, האחראי על ניהול תת-הסוכנים השונים. בזמן ריצה, יבחר הסוכן בתת-הסוכן היעיל ביותר לבעיה (ואם אין אחד כזה, שכן הסוכן עוד לא למד, הוא ידרוש לבצע למידה ראשית, ויקרוס.
* בזמן הלמידה, הסוכן יריץ כל תת-סוכן על הבעיה ויאסוף אודותיהם מידע שיאפשר לו להחליט מהו הסוכן הטוב ביותר לבעיה בה הוא נתקל. הסוכן יריץ **ראשית** את סוכן התכנון ולאחר מכן, את סוכני הלמידה **לסירוגין (לא במקביל)**.

# Butter (UCS (MB)+IDDFS (MB), MB)

* סוכן זה אינו משתמש במתכנן המובנה של הסימולטור באופן כללי ובמקום זאת, מחלק את פעולתו לשני מצבים שונים:
  + מצב למידה **התחלתי**, מבוסס IDDFS, שבמסגרתו הסוכן לומד את רמת הוודאות של כל פעולה (עד כמה סביר כי הפעולה תתבצע באופן רגיל ולא תיכשל). צירופי פעולה-מצב בעלי **וודאות** גבוהה, יקבלו משקל נמוך יותר. כמו כן, בתחילת שלב זה, כל צירוף יקבל משקל של 1, בתור ברירת מחדל. לאחר והסוכן מגיע למצב בו ישנו צירוף אשר נכשל יותר משלוש פעמים, הסוכן **ייצא** ממצב זה ויעבור למצב UCS. ו**ישמור** את המשקלים אשר חישב בקובץ, על מערכת הקבצים.
  + מצב למידה **עוקב** ומצב הרצה מבוסס UCS, במסגרתו הסוכן משתמש במשקלים הקיימים של כל צירוף ומנסה למצוא שביל עד להגעה למצב המטרה (**או** אחד ממצבי המטרה, בהנחה והמטרות המושגות ימשיכו להתקיים, מהרגע והושגו), **או** פשוט שביל בעל משקל נמוך ככול האפשר, אם אין שביל שכזה. כמו כן, גם בשלב זה הסוכן ימשיך ולעדכן את המשקלים של כל צירוף וישמוריהם במקומם הרגיל, כלומר בקובץ. הסוכן יעקוב אחר השביל, עד והוא מסתיים או עד הרגע בו ישנה **סטייה** מן השביל (במקרה זה, סטייה הינה פשוט כישלון של פעולה מסוימת). במצב כזה, הסוכן מתכנן את מסלולו **מחדש**.
* עקב זאת, הסוכן **דורש** זמן למידה מסוים.
* הסוכן נחשב בכל זאת לסוכן **מבוסס** **מודלים**, שכן הוא משתמש במודלים של הסביבה כדי לייצר את הגרפים שלו וכמו כן משתמש במתכנן המובנה של הסימולטור, עבור חישוב העומק **המינימלי** של הרצת אלגוריתם ה-IDDFS.

# Pancake (2022, State Graph, MF)

* במהלך הלמידה, אחרי כל ביצוע פעולה שומר את המצב הקודם ואת כל האפקטים שנגרמו מהפעולה, על מנת ללמוד מידע חבוי.
* משתמש ב-finite state machine, שומר את המצבים השונים בגרף AND/OR המאפשר לבצע חיפוש עומק רקורסיבי. לכל מצב מושמת label שלפיה ניתן לדעת האם הוא יכול להגיע למטרה או לא (מצבים של מבוי סתום או כאלה שמובילים אליהם נסגרים, וכדומה).
* ישנו גם מנגנון המונע לולאות במהלך הרקורסיה.
* על החיפוש הרקורסיבי מוטל timeout על מנת למנוע בזבוז זמן בדומיינים גדולים.
* אם הדומיין לא נלמד טרם הריצה, במהלך הריצה הסוכן יפעל באופן רנדומלי תוך הימנעות מפעולות שכבר בוצעו או שידוע שהן מסוכנות.

# Marble (2021, Planning, Q-Learning, MB)

* בעת הלמידה, מחלק את הבעיות לדטרמיניסטיות/סטוכסטיות וקשות/קלות לפתירה, ומסווג אותן לסוכנים שונים בצורה הבאה:
* דומיין דטרמיניסטי - נפתר על ידי planner
* דומיין סטוכסטי פשוט - למידת Q כפולה; מייצא שתי טבלאות Q
* דומיין סטוכסטי מורכב - למידת R-Max; מייצא את הסתברויות המעבר
* פעולת החלוקה מתבצעת על פי מספר המטרות הקיימות בבעיה, ועל פי מכפלת מספר העצמים במספר הפעולות האפשריות.
* בזמן הריצה, אם לא נלמדה הבעיה הנתונה, הסוכן ימנע מהעבר וימנע מלולאות.