Control policies for 421, a stochastic game of life

Pierre Albarède
http://pierre.albarede.free.fr

Abstract

We treat a stochastic control problem, inspired by a popular dice game, known as 421 [1]. Tools are realized, especially for combination algebra and quotient algebra modulo face permutations. Linear growth Markovian fate trees are evaluated. A Markovian utility-strategy format is introduced, along with purifying, forcing and pruning techniques. Backward induction programs are realized: - an optimal policy, mean–max, - a Markovian strategy judging program, mean–mean. The latter is also used, numerically, to evaluate probabilities, by solving Kolmogorov equation, and, symbolically, to display strategies. A constant-goal ratchet stratagem is found and most probably successful strategy final state probabilities are compiled, using dynamic self-similarity. Goal-driven policies (cheaper than mean–max) are inferred, depending on compiled probabilities and three parameters: serendipity [2], horizon and dynamism. Eight goal-driven policies are applied to seventeen utility functions and most resulting strategies, not all Markovian, are exactly judged. Empiric laws of policy utility are inferred, confirming the utility of serendipity [2]. Meta-policy is introduced. Technically, Mathematica allows choosing between the λ (anonymous), functional and procedural programming styles, depending on evaluation and naming constraints.

Key words: utility, strategy, policy, insufficient reason, backward induction, goal, ratchet, serendipity, horizon, dynamism, meta-policy.

1. Introduction

1.1. Playing against providence

Within the 421 game, which is fully described in [1], only a round is treated (when one player alone is opposed to providence). As providence is not exactly a player, a round is not exactly a game, but a stochastic control problem. A combination is a list, the order of which does not matter, or a finite set possibly with repetitions. A state is a combination of dice that have been irreversibly pushed away from the dice board; an event is a combination of dice that have just been cast. The cast number (initially zero) serves as a discrete date. After each event, a new state is chosen. Fate is an alternate sequence state-event-state… ending with some state having some utility, which may depend on final date and state but not on history. A set of possible fates, along with event probabilities and final utilities, appears as a fate tree.

A strategy is a (generally probabilistic) determination of all possible choices. A policy is a program evaluating exactly one strategy. Distinct policies can evaluate the same strategy.

The Von Neumann-Morgenstern or "mean–max" theorem [3]:
- utility exactly expresses preference, that is, choice maximizes utility,
- the utility of present state is the expected utility of next random events.

If many states are most useful (as in Buridan's donkey story), then, according to the principle of insufficient reason [1, 4], completely equiprobable choice (among all most useful states) is assumed. From this and the von Neumann-Morgenstern theorem, there exists exactly one complete most useful strategy, non-pure in general and evaluable by "backward induction" [3].

As dice are independent and, by insufficient reason, unloaded, the probabilities depending on face combinations are invariant by global face permutations.
There are $d$ dice each with $f$ faces. A player casts the dice at most $j$ times. The parameters of the control problem are thus $d$, $f$, $j$ and some utility function (a functional parameter). Normally,

\[ \{d = 3, f = 6, j = 3\} \]

although next players begin to play with smaller $j$. Moreover, smaller $d$, $f$, $j$, like

\[ \{d = 2, f = 3, j = 3\} \]

provide small-scale cases, useful for program development.

1.2. Conventions and data

« △ » introduces agenda (things to do), « △ △ » introduces some hopefully clever remarks.

In text, loose sign conventions are used, that is,
- $\geq$ is greater, $>$ is strictly greater,
- zero is both positive and negative,
- a constant function is both increasing and decreasing,
- present is both past and future,

... $\$Version

5.2 for Mac OS X (June 20, 2005)

After initializing, every numbered section of the present notebook can be evaluated independently in a meaningful way. A section usually consists of three parts: one for developing, one for defining and one for testing some function. Keep the environment clean by evaluating clearing commands as well!

Parameters will be often grouped as in

\[ \text{action}[\text{conditions}][\text{object}] \]

This grouping simplifies mapping, as in

Block[{power1, power1[n_]@x_ := Power[x, n];
{1, 8, 27}
{1, 8, 27}

make (like the unix command) is used to hide parameters that are too big to show (or that would take too much time to evaluate from a compact form). make is not supposed to evaluate anything, but to affect variables: this is procedural programming and it appears as a consequence of evaluation constraints. △ If this already has a value, it will be affected instead of this.

\[ \text{make}[\text{implicit}][\text{this}[\text{explicit}]] := \text{this}[\text{explicit}] = \text{function}[\text{implicit}, \text{explicit}] \]

Probability and strategy utility files should accompany the present notebook. Otherwise, probability files will be automatically regenerated when needed; the strategy utility files can be regenerated by evaluating section 7.4 (not so quietly because of some hard cases).

\[ << \text{AuthorTools`,}
\text{SetDirectory@NotebookFolder@InputNotebook[]};
\text{FileNames["\text{probability*", "+strategyUtility*"}]} \]

{maxStrategyUtility.txt, probability1_2_3.txt, probability2_3_3.txt,
probability2_3_4.txt, probability2_4_3.txt, probability3_6_3.txt, probability3_6_3V.txt,
randomStrategyUtility.txt, strategyUtility1.txt, strategyUtility.txt}
This beeps once for unknown reason but works all the same.

```
Utilities`Notation`AutoLoadNotationPalette = False;
<< Utilities`Notation```

2. Tools

2.1. Evaluation constraints

2.1.1. Timing

2.1.2. Spacing

2.2. Lists

2.2.1. Only element filtering

2.2.2. Elementary statistics

2.2.3. Reverse folding and history

2.2.4. Function determined by graph

2.2.5. ArgMax

2.2.6. Quotient of list modulo function

2.2.7. Filling table from coordinates and values

2.2.8. Slicing

2.3. Formats

2.3.1. Long table slicing

2.3.2. Boxes

2.3.3. Line labeling

3. Combinations

3.1. Generalizing finite set algebra

3.1.1. All combinations

3.1.2. Eulerian (occupation number) vector form

3.1.3. Element query

3.1.4. Difference

3.1.5. Intersection
3.1.6. Inclusion query

3.2. Equivalence modulo face permutations

3.2.1. Face permutation

3.2.2. Combination representative

3.2.3. Representing permutation

3.2.4. All representatives

3.2.5. Conditional representative

3.2.6. Birepresentative

3.2.7. All conditional representatives and all birepresentatives

4. Fate, utility and strategy

4.1. Linear growth Markovian fate tree

4.1.1. Branching one leaf

4.1.2. Branching many leaves and sharing next states

4.1.3. Branching deepest level

4.1.4. Branching completely

\[ \{d = 2, f = 2, j = 3\}; \text{restrictNextStates} = \text{Identity}; \]
\[ \text{branch}[d, f, \text{restrictNextStates}] [\text{fateTree} \rightarrow \{(0, (\{\}))\}] \]
\[ \{(0, \{(\{\}), (\{1, 1\}), (\{1\}, (2)), (\{1, 1\}, (4))\}), \]
\[ (\{1, 2\}), ((\{1\}, (1)), (\{1\}, (2)), (\{2\}, (3)), (\{1, 2\}, (5)))\}, \]
\[ (\{1, 2\}), ((\{1\}, (1)), (\{2\}, (3)), (\{2, 2\}, (6)))\}, \{)\}, \]
\[ (1, (\{\}), (1), (2), (1, 1), (1, 2), (2, 2))\} \]
Fold[branch[d, f, #2] & [fateTree -> #1] &, {0, (())},
Join[Table[restrictNextStates, (j - 1)], {takeLast, takeLast}]]

{{0, {{}, {{1}, {1}, 1}, {{1}, {1}, 2}, {{1}, 1, 1}, {4})}},
  {{1/2, {1, 2}, {{1}, {1}, 2}, {{1}, 2}, {{2}, 3}, {{1, 2}, 5})},
  {{1/4, {2, 2}, {{1}, {1}, 2}, {{2}, 3}, {{2, 2}, 6})}}, 1},

1, {{1}, {{1}, {1}, 2}, {{1}, 2}, {{1, 1}, 4})},

{{1/2, {1, 2}, {{1}, {1}, 2}, {{1}, 2}, {{2}, 3}, {{1, 2}, 5})},
  {{1/4, {2, 2}, {{1}, {1}, 2}, {{2}, 3}, {{2, 2}, 6})}}, 1},

{{1/2, {1, 2}, {{1}, {1}, 2}, {{1}, 2}, {{2}, 3}, {{1, 2}, 5})},
  {{1/4, {2, 2}, {{1}, {1}, 2}, {{2}, 3}, {{2, 2}, 6})}}, 1},

fateTree = dropLast@Fold[branch[d, f, #2] & [fateTree -> #1] &, {0, (())},
Join[Table[restrictNextStates, (j - 1)], {takeLast, takeLast}]]
\texttt{fateTree["p2", 1, 1, 0]}
\texttt{fateTree["p2", 0, 1, 1]}
\texttt{fateTree["p2", 1, 1, 1]}
\texttt{fateTree["p2", 1, 1, 2]}
\texttt{fateTree["p2", 2, 1, 3]}
\texttt{fateTree["p2", 2, 2, 2]}
\texttt{fateTree["p1", 2, 3, 1]}

\begin{itemize}
\item Next state sharing, on one hand, makes fate tree growing only linearly with cast number, not exponentially; on the other hand, prevents remembering fate: fate tree is Markovian.
\end{itemize}

\section*{Normal case}

Apple Powerbook G4, PPC 1 GHz

\texttt{(d = 3, f = 6, j = 3); Timing[ByteCount@fateTree["p1", d, f, j]}
\texttt{0.44 Second, 407416}

\texttt{spacing[fateTree["p1", d, f, j], Last@%]}
\texttt{691856}

\texttt{Timing[ByteCount@fateTree["p2", d, f, j]}
\texttt{0.28 Second, 340056}

\texttt{(d = ., f = ., j = .);}
4.2. Markovian utility and strategy formats

4.2.1. ...with utility

Check unicity of initial state.

```
Clear@initial;
initial@((_, {{_, {x, ___}, {}}, {}}), __) = Extract[only@{{x,}}, (1, 1)];
```

Utility precedes a strategy piece, empty for immediate utility.

```
utilityAndStrategy1 = {{0,
    {{{}, {-7}, {((1, 1), ((1, 1), (4)))}, ((1, 2), (((1, 2), (5))))}, ((2, 2), (((2, 2), (6))))},
    {}},
    {1,
    {{{}, {-17}, {((1, 1), ((1, 1), (1)))}, {((1, 2), (((1, 2), (2))))}, ((2, 2), (((2, 2), (3))))},
    {1}, {-35/2, {((1), {((1, 1), (1))}, {((1, 2), (2))}, ((2, 2), (((2, 2), (2))))},
    {2}, {-33/2, {((1), {((1, 2), (2))}, ((2), (((2, 2), (3))))},
    {1}, {-8, {((1), {((1, 2), (2))}, ((2, 2), (6))},
    {2}, {1, {{1, 1}, {-18, {}}}, {{1, 2}, {-17, {}}}, {{2, 2}, {-16, {}}}}
}

initial@utilityAndStrategy1

-7

initial@Rest@utilityAndStrategy1

initial[{{1,
    {1}, {-17, {((1, 1), ((1, 1), (1)))}, {((1, 2), (((1, 2), (2))))}, ((2, 2), (((2, 2), (3))))},
    {1}, {-35/2, {((1), {((1, 1), (1))}, {((1, 2), (2))}, ((2, 2), (((2, 2), (2))))},
    {2}, {-33/2, {((1), {((1, 2), (2))}, ((2), (((2, 2), (3))))},
    {1}, {-8, {((1), {((1, 2), (2))}, ((2, 2), (6))},
    {2}, {1, {{1, 1}, {-18, {}}}, {{1, 2}, {-17, {}}}, {{2, 2}, {-16, {}}}}
}

utilityAndStrategy1 = .
```

4.2.2. ...without utility

```
Clear@extractStrategy; extractStrategy@utilityAndStrategy1_List :=
    MapAt[MapAt[Last, #, -1] &, #, {2}] & @ utilityAndStrategy1;

extractStrategy@{{0, {{()}, {-7},
    {((1, 1), ((1, 1), (4))), ((1, 2), (((1, 2), (5))))}, ((2, 2), (((2, 2), (6))))},
    {1, {{()}, {-17}, {((1, 1), ((1, 1), (1)))}, {((1, 2), (((1, 2), (2))))}, ((2, 2),
    {((2, 2), (3))})}, {1}, {-35/2, {((1), {((1, 1), (1))}, {((1, 2), (2))}, ((2, 2),
    (((2, 2), (2))))},
    {2}, {-33/2, {((1), {((1, 2), (2))}, ((2), (((2, 2), (3))))},
    {1}, {-8, {((1), {((1, 2), (2))}, ((2, 2), (6))},
    {2}, {1, {{1, 1}, {-18, {}}}, {{1, 2}, {-17, {}}}, {{2, 2}, {-16, {}}}}
}

((1, 1), {-8, {((1), {((1, 2), (2))}, ((2, 2), (3))}}},
    {1}, {-7, {((1, 2), (2))}, ((2, 2), (6))},
    {2, {1, {{1, 1}, {-18, {}}}, {{1, 2}, {-17, {}}}, {{2, 2}, {-16, {}}}}
} = {{0,
    {((1), {((1, 1), ((1, 1), (4)))}, {((1, 2), (((1, 2), (5))))}, ((2, 2), (((2, 2), (6))))},
    {1, {{()}, {((1, 1), ((1, 1), (1)))}, {((1, 2), (((1, 2), (2))))}, ((2, 2),
    {((2, 2), (3))})}, {1}, {-35/2, {((1), {((1, 1), (1))}, {((1, 2), (2))}, ((2, 2),
    (((2, 2), (2))))},
    {2}, {-33/2, {((1), {((1, 2), (2))}, ((2), (((2, 2), (2))))},
    {2}, {1, {{1, 1}, ((), (1, 2)}, ((2, 2), (3))}}},
    {1}, {-18, {((1), {((1, 2), (2))}, ((2, 2), (3))}}},
    {2, {}, (((1, 1), ()), ((1, 2), ()), ((2, 2), ()))}}
}

True
```
4.2.3. Utility function samples

4.3. Fate tree or strategy transformations

4.3.1. Purifying strategy

4.3.2. Forcing fate tree by some strategy

4.3.3. Pruning fate tree according to some utility function

5. Backward induction programs

5.1. Mean-max policy

5.1.1. Fructifying

5.1.2. Folding one state branch

5.1.3. Folding deepest level

5.1.4. Folding completely

Time scheme.

\[
\{d = 2, f = 3, j = 3\};
\]

With \( \{\text{fructifiedFateTree} = \text{Range}[0, j]\} \),

\[
\text{reverseFoldList}[\text{meanMaxStep}@\{1, 2\} \&,
\text{dropLast}@\text{fructifiedFateTree}, \text{Last}@\text{fructifiedFateTree}]
\]

\[
\{\text{meanMaxStep}[0, \text{meanMaxStep}[1, \text{meanMaxStep}[2, 3]]],
\text{meanMaxStep}[1, \text{meanMaxStep}[2, 3]], \text{meanMaxStep}[2, 3], 3\}
\]

utility = -10 \{1 + Total@2 \&; fructifiedFateTree = fructify[utility]@fateTree["pl", d, f, j];

\[
\text{reverseFoldList}[\text{meanMaxStep}@\{1, 2\} \&, \text{dropLast}@\text{fructifiedFateTree}, \text{Last}@\text{fructifiedFateTree}]
\]

\[
\{0, \{\{1\}, -6, \{\{1, 1\}, \{\{1, 1\}, 5\}\}\}, \{\{1, 2\}, \{\{1, 2\}, 6\}\}, \{1, 3\}, \{\{1, 3\}, 7\}\},
\{\{1, 2\}, \{\{2, 2\}, 9\}\}, \{\{2, 3\}, \{\{2, 3\}, 10\}\}\},
\{\{1, 2\}, \{\{2, 2\}, 8\}\}, \{\{2, 3\}, \{\{2, 3\}, 10\}\}\},
\{\{1, 2\}, \{\{1, 2\}, 6\}\}, \{\{1, 3\}, \{\{1, 3\}, 9\}\}\},
\{\{1, 2\}, \{\{2, 2\}, 8\}\}, \{\{2, 3\}, \{\{2, 3\}, 10\}\}\},
\{\{1, 2\}, \{\{1, 2\}, 6\}\}, \{\{1, 3\}, \{\{1, 3\}, 9\}\}\},
\{\{1, 2\}, \{\{2, 2\}, 8\}\}, \{\{2, 3\}, \{\{2, 3\}, 10\}\}\}
\]

\[
\{0, \{\{1\}, -6, \{\{1, 1\}, \{\{1, 1\}, 5\}\}\}, \{\{1, 2\}, \{\{1, 2\}, 6\}\}, \{1, 3\}, \{\{1, 3\}, 7\}\},
\{\{1, 2\}, \{\{2, 2\}, 9\}\}, \{\{2, 3\}, \{\{2, 3\}, 10\}\}\},
\{\{1, 2\}, \{\{2, 2\}, 8\}\}, \{\{2, 3\}, \{\{2, 3\}, 10\}\}\},
\{\{1, 2\}, \{\{1, 2\}, 6\}\}, \{\{1, 3\}, \{\{1, 3\}, 9\}\}\},
\{\{1, 2\}, \{\{2, 2\}, 8\}\}, \{\{2, 3\}, \{\{2, 3\}, 10\}\}\},
\{\{1, 2\}, \{\{1, 2\}, 6\}\}, \{\{1, 3\}, \{\{1, 3\}, 9\}\}\},
\{\{1, 2\}, \{\{2, 2\}, 8\}\}, \{\{2, 3\}, \{\{2, 3\}, 10\}\}\}
\]
5.2. Mean-mean Markovian strategy judging program

5.2.1. Folding one strategy state branch

5.2.2. Folding last step

5.2.3. Folding completely

Time scheme. \( \text{meanMeanStep} \) must format utilities as in fate tree last level.
meanMeanStep[t[0], meanMeanStep[t[1], meanMeanStep[t[2], t[3]], s[2]], s[1]], s[0]]

meanMeanStep[t[1], meanMeanStep[t[2], t[3]], s[2]], s[1]]

meanMeanStep[t[2], t[3]], s[2]]

t[3]

j = 2;
strategy1 = extractStrategy@maxUtilityAndStrategy[fateTree["p1", d, f, j], δ[{(1, 2), #2} &];
fructifiedFateTree = fructify[δ[{(3, 4), #2} &]@fateTree["p1", d, f, j];

{d =., f =., j =.};

reverseFoldList[meanMeanStep[First@1, #2], Last@1, withStrategy -> True] &,
Transpose[dropLast /@ (fructifiedFateTree, strategy1), Last@fructifiedFateTree]

{{0,
  {[(0, [1/32, [([[1, 1], [([1], [2])]), [(1, 2)], [[1, 2], [7]]), [(1, 3)], [[(1), [2])]], [[1, 4]], [[(1), [2])]], [[2, 2], [[(2), [3])]], [[2, 3], [[(2), [3])]], [[2, 4], [[(2), [3])]], [[3, 3], [[(1), [1])], [[3, 4], [[(1), [1])], [[4, 4], [[(1), [1])]]])},
  {1, {[(1, [1/8, [([1, 1], [([1], [1])]), ([1], [1])], [[1, 2], [[1, 2], [2])]]],
    [[1, 3], [[(1), [3])], [[1, 4], [[(1), [4])], [[2, 2], [[(2), [2])], [5])],
    [[2, 3], [[(2), [3])], [[2, 4], [[(2), [4])], [[3, 3], [[(3), [3])], [[3, 4], [[(3), [4])], [[4, 4], [[(4), [4])], [[10])]]],
    [[1], [[(0), [1]), [[(1), [1])], [[(2), [1]), [[(2), [2])], [[(3), [[(1, 3), [3])], [[(4), [[(1), [4])], [[(2), [2])], [6])],
    [[(2), [2], [5])], [[(3), [[(2), [3])], [6])], [[(4), [[(2), [4])], [7])]]],
    [[(3), [1/4], [[(1), [[(1), [3])], [[(2), [[(2), [3])], [6])], [[(3), [[(3), [3])], [[(4), [8])],
    [[(4), [[(3), [4])], [9])], [[4], [[(1), [4])], [[(2), [4])], [7])]],
    [[(2), [[(2), [4])], [7])], [[(3), [[(3), [4])], [9])], [[4], [[(1), [4])], [[(2), [4])], [10])]]],
  }},
  {[(0, [0]), [[(1), [1])], [[(2), [1]), [[(2), [2])], [[(3), [0])], [[(4), [0])],
    [[(2), [2]], [[(2), [3])], [[(2), [4])], [0])],
    [[(3), [0])], [[(3), [4])], [1])], [[(4), [0])], [0])]},
  {[(2, [0]), [[(1), [1])], [[(2), [1])], [[(2), [2])], [[(3), [0])], [[(4), [0])],
    [[(2), [2]], [[(2), [3])], [[(2), [4])], [0])],
    [[(3), [0])], [[(3), [4])], [1])], [[(4), [0])], [0])]}]

meanMeanStep[fructifiedFateTree[[2]], fructifiedFateTree[[3]],
strategy1[[2]], withStrategy -> True]

{1, {[(1, [1/8), [[(1), [1])], [[(1), [1])]],
  [[(1), [2]], [[(2), [2])], [[(1), [3]), [[(3), [3])], [[(1), [4])], [[(4), [4])],
    [[(2), [2]], [[(2), [5])], [[(2), [3])], [[(2), [6])], [[(2), [4])], [[(2), [7])],
    [[(3), [3], [[(3), [8])], [[(3), [4]), [9])], [[(4), [4])], [[(4), [4])], [10])]]],
  [[(1), [0]), [[(1), [1])], [[(2), [1]), [[(2), [2])], [[(3), [[(1, 3), [3])],
    [[(4), [[(1), [4])], [[(2), [2])], [4])], [[(2), [2], [5])], [[(3), [[(2), [3])], [6])], [[(4), [[(2), [4])], [7])]],
    [[(3), [1/4), [[(1), [[(1), [3])], [[(2), [[(2), [3])], [6])], [[(3), [[(3), [3])], [[(4), [8])],
    [[(4), [[(3), [4])], [9])], [[4], [[(1), [4])], [[(2), [4])], [7])]],
    [[(2), [[(2), [4])], [7])], [[(3), [[(3), [4])], [9])], [[4], [[(1), [4])], [[(2), [4])], [10])]]],
  }},
  {[(0, [0]), [[(1), [1])], [[(2), [1])], [[(2), [2])], [[(3), [0])], [[(4), [0])],
    [[(2), [2]], [[(2), [3])], [[(2), [4])], [0])],
    [[(3), [0])], [[(3), [4])], [1])], [[(4), [0])], [0])]},
  {[(2, [0]), [[(1), [1])], [[(2), [1])], [[(2), [2])], [[(3), [0])], [[(4), [0])],
    [[(2), [2]], [[(2), [3])], [[(2), [4])], [0])],
    [[(3), [0])], [[(3), [4])], [1])], [[(4), [0])], [0])]}]
meanMeanStep[{fructifiedFateTree[[1]], _}, strategy1[[1]], withStrategy -> True]

{0,

\[\{\{\{\}, \{\{1\}, \{\}, \{\}, \{3\}, \{\}, \{\}, \{3\}\}\}, \{\{1, 2\}, \{\}, \{\}, \{\}, \{\}, \{\}, \{\}\}\}, \{\{1, 3\}, \{\}, \{\}, \{\}, \{\}, \{\}, \{\}\}\}, \{\{1, 4\},

\{\{\}, \{\}, \{\}, \{\}, \{\}, \{\}, \{\}\}\}\}, \{\{\}, \{\}, \{\}, \{\}, \{\}, \{\}, \{\}\}\}\}, \{\}\}

Check: when aiming at \{1, 2\}, reaching actually \{3, 4\} (a complete failure) implies raising at last cast \{3, 4\} after obtaining at first cast \{3, 4\} (prob. 1/8) or \{3, 3\} (prob. 1/16) or \{4, 4\}.

\[\frac{1}{8} \times \frac{1}{8} + \frac{2}{16} \times \frac{1}{8} = \frac{1}{32}\]

True

\{fructifiedFateTree, strategy1\} = ;

\[\text{\textbackslash \hspace{0pt}}\text{Remember section 5.2.1: shared state positions in fate tree and strategy must be consistent.}\]

Clear@utilityAndStrategy;
utilityAndStrategy[fateTree_, utility_, options___]@strategy1_List :=
  With[{fructifiedFateTree = fructify[utility]@fateTree,
    reverseFoldList[meanMeanStep[[First@1, #2], Last@1, options] &,
    Transpose[dropLast /@ {fructifiedFateTree, strategy1}], Last@fructifiedFateTree]}
]

Check strategy echo.

With[{d = 2, f = 2, j = 3},
  With[{fateTree1 = fateTree["pl", d, f, j]},
    With[{strategy1 = extractStrategy@
      maxUtilityAndStrategy[fateTree1, \delta (Last@allRepresentatives[d, f, #2] &)],
      Equal @@ {extractStrategy@utilityAndStrategy[fateTree1, -10 #1 + Total@#2 &,
        withStrategy -> True]@strategy1, strategy1}]}
  True

With[{d = 2, f = 2, j = 3},
  With[{fateTree1 = fateTree["pl", d, f, j]},
    With[{strategy1 = extractStrategy@
      maxUtilityAndStrategy[fateTree1, \delta (Last@allRepresentatives[d, f, #2] &)],
      utilityAndStrategy[fateTree1, -10 #1 + Total@#2 &@strategy1]}
    {{0, {{}, {-\frac{29}{2}, {}}}, {}},
      {1, {{}, {-22, {}}}, {1, {-\frac{89}{4}, {}}}, {2, {-\frac{87}{4}, {}}}},
      {\{\{1, 1\}, {-8, {}}\}, \{1, 2\}, {-7, {}}}, {\{2, 2\}, {-6, {}}}},
      {2, {{}, {-27, {}}}, {1, {-\frac{55}{2}, {}}}, {2, {-\frac{53}{2}, {}}}},
      {\{\{1, 1\}, {-18, {}}\}, {1, 2}, {-17, {}}}, {\{2, 2\}, {-16, {}}}},
      {3, {{}, {{1, 1}, {-28, {}}, \{1, 2\}, {-27, {}}, \{2, 2\}, {-26, {}}} }}
    }
  With[{d = 2, f = 2, j = 3},
    With[{fateTree1 = fateTree["pl", d, f, j]},
      initial@utilityAndStrategy[fateTree1, -10 #1 + Total@#2 &@
        extractStrategy@maxUtilityAndStrategy[fateTree1, \delta (Last@allRepresentatives[d, f, #2] &]
      ]]
    ]
  ]
\[\frac{-29}{2}\]

5.3. Consistency between mean-max and mean-mean

5.3.1. Checking mean-max utility from strategy, with mean-mean
5.3.2. Mean-max strategy is more useful than other strategies

5.3.3. Checking pure strategy utility with forcing and mean-max

5.4. Displaying strategies with mean-mean and symbolic utility

5.4.1. Displaying one strategy state branch

5.4.2. Displaying last step

5.4.3. Displaying completely

6. Most probably successful constant-goal strategies

6.1. A stratagem of mean-max: the ratchet policy

6.1.1. First player

6.1.2. Next players' dilemmas

6.2. Evaluating probabilities

6.2.1. All-time success probabilities

6.2.2. Cast number probabilities

6.2.3. Formula for brelan constant-goal cast number probabilities

6.3. Compiling final state probabilities

6.3.1. Definition and evaluation

6.3.2. Dynamic self-similarity

6.3.3. Constant final date and state

6.3.4. Variable final date and state

6.3.5. Variable goal with domain restrictions

6.3.6. Saving and retrieving

6.4. Final state probability charts

6.4.1. Success probabilities

All-time success probabilities, meaningful for first player, are also given.
\(d = 2, f = 3, j = 3\); Block[({probability}),

TableForm[
With[{goal = Table[1, {d}]},
{
  (\#, smallBox@Total@\#) & [probability["pl", f, j, goal, \#, goal] & @ Range@j, 
    probability["p2", f, \#, goal, \#, goal] & @ Range@j}
}
],
TableDirections -> {Row, Row, Column},
TableHeadings -> {"First player", "Next players"}]/. probability \[Rule] \[Pi]

First player | Next players
---|---
\[\pi[pl, 3, 3, \{1, 1\}, 1, \{1, 1\}]\] | \[\pi[pl, 3, 3, \{1, 1\}, 1, \{1, 1\}] + \pi[pl, 3, 3, \{1, 1\}, 1, \{1, 1\}]\]
\[\pi[pl, 3, 3, \{1, 1\}, 2, \{1, 1\}]\] | \[\pi[pl, 3, 3, \{1, 1\}, 2, \{1, 1\}] + \pi[pl, 3, 3, \{1, 1\}, 2, \{1, 1\}]\]
\[\pi[pl, 3, 3, \{1, 1\}, 3, \{1, 1\}]\] | \[\pi[pl, 3, 3, \{1, 1\}, 3, \{1, 1\}] + \pi[pl, 3, 3, \{1, 1\}, 3, \{1, 1\}]\]

TableForm[
With[{goal = \#, 
  With[{player = \#}, 
    With[{probabilities = probability[player, f, \#, goal, \#, goal] & @ Range@j}, 
      Switch[player, 
        "pl", \{\#, Total /@ Rest@history@\#\} \[\&] probabilities, 
        "p2", \{\#\} \[\&] probabilities
      ] & @ {"pl", "p2"}] & @ allRepresentatives[d, f],
TableDirection -> 4,
TableDirections -> {Column, Row, Row, Column},
TableHeadings -> {allRepresentatives[d, f], {"First player", "Next players"}}]

First player | Next players
---|---
\[\{1, 1\}\] | \[\{1, 1\}\]
\[\{1, 2\}\] | \[\{1, 2\}\]

With[{dl = First@\#, j1 = Last@\#},
TableForm[
With[{goal = \#}, 
  With[{player = \#}, 
    With[{probabilities = probability[player, f, \#, goal, \#, goal] & @ Range@j1}, 
      Switch[player, 
        "pl", \{\#, Total /@ Rest@history@\#\} \[\&] probabilities, 
        "p2", \{\#\} \[\&] probabilities
      ] & @ {"pl", "p2"}] & @ allRepresentatives[dl, f],
TableDirection -> 4,
TableDirections -> {Column, Row, Row, Column},
TableHeadings -> {allRepresentatives[dl, f], {"First player", "Next players"}}]
] & @ finalDynamicParameters[d, j] \[\&\] scanPrint

First player | Next player
---|---
\[\{1\}\] | \[\{1\}\]
\[\{1, 1\}\] | \[\{1, 1\}\]
\[\{1, 2\}\] | \[\{1, 2\}\]
TableForm[Flatten[
With[{dl = First@#, j1 = Last@#},
With[{{goal = #},
  With[{{player = #},
    With[{{probabilities = probability[player, f, #, goal, #, goal] & /@ Range@j1},
      Switch[player,
      "pl", ReplacePart[Map[numberForm1, 
        (#, Total /@ Rest@history@##) &@probabilities, {2}], "", {-1, 1}],
      "p2", Map[numberForm1, (#) &@probabilities, {2}]}
    ] & /@ {"pl", "p2"}
  ] & /@ allRepresentatives[dl, f]
} & finalDynamicParameters[d, j, 1],
TableDirections -> {Column, Row, Row, Column},
TableHeadings -> {combinationBox /@ Flatten[With[{{dl = First@#, j1 = Last@#}, allRepresentatives[dl, f]] & /@
  finalDynamicParameters[d, j, 1], {"First player", "Next players"}]}
]

First player       Next players
numberForm1[\(\frac{1}{2}\)]       numberForm1[\(\frac{1}{2}\)]
1
numberForm1[\(\frac{2}{9}\)] numberForm1[\(\frac{5}{9}\)]       numberForm1[\(\frac{1}{2}\)]
numberForm1[\(\frac{1}{2}\)]       numberForm1[\(\frac{1}{2}\)]
1 1
numberForm1[\(\frac{4}{27}\)] numberForm1[\(\frac{5}{27}\)]       numberForm1[\(\frac{1}{2}\)]
numberForm1[\(\frac{5}{27}\)] numberForm1[\(\frac{6}{27}\)]       numberForm1[\(\frac{2}{9}\)]
numberForm1[\(\frac{4}{27}\)] numberForm1[\(\frac{3}{9}\)]       numberForm1[\(\frac{4}{9}\)]
2 1
numberForm1[\(\frac{10}{81}\)] numberForm1[\(\frac{13}{81}\)]       numberForm1[\(\frac{8}{27}\)]
numberForm1[\(\frac{9}{27}\)] numberForm1[\(\frac{12}{27}\)]       numberForm1[\(\frac{9}{27}\)]
(d = ., f = ., j = .);
Clear@successProbabilityChart;
Protect@numberForm1;
Options@successProbabilityChart = {numberForm1 -> Identity};
successProbabilityChart[d_Integer, f_Integer, j_Integer, probability, options___] :=
(Print@"Success probabilities"; Print@TableForm[Flatten[With[{{dl = First@#, j1 = Last@#},
  With[{{goal = #},
    With[{{player = #},
      With[{{probabilities = probability[player, f, #, goal, #, goal] & /@ Range@j1},
        Switch[player,
        "pl", ReplacePart[Map[numberForm1, 
          (#, Total /@ Rest@history@##) &@probabilities, {2}], "", {-1, 1}],
        "p2", Map[numberForm1, (#) &@probabilities, {2}]}
      ] & /@ {"pl", "p2"}
    ] & /@ allRepresentatives[dl, f]
} & finalDynamicParameters[d, j, 1],
TableDirections -> {Column, Row, Row, Column},
TableHeadings -> {combinationBox /@ Flatten[With[{{dl = First@#, j1 = Last@#}, allRepresentatives[dl, f]] & /@
  finalDynamicParameters[d, j, 1], {"First player", "Next players"}]}
])
successProbabilityChart[2, 3, 3, probability]

Success probabilities

<table>
<thead>
<tr>
<th>First player</th>
<th>Next players</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\frac{1}{3})</td>
<td>(\frac{1}{3})</td>
</tr>
<tr>
<td>(\frac{2}{9})</td>
<td>(\frac{2}{9})</td>
</tr>
<tr>
<td>(\frac{5}{9})</td>
<td>(\frac{5}{9})</td>
</tr>
<tr>
<td>(\frac{1}{9})</td>
<td>(\frac{1}{9})</td>
</tr>
<tr>
<td>(\frac{729}{729})</td>
<td>(\frac{211}{729})</td>
</tr>
<tr>
<td>(\frac{81}{81})</td>
<td>(\frac{81}{81})</td>
</tr>
<tr>
<td>(\frac{361}{729})</td>
<td></td>
</tr>
</tbody>
</table>

successProbabilityChart[3, 6, 3, probability, numberForm1 → approximationBox]

Success probabilities

<table>
<thead>
<tr>
<th>First player</th>
<th>Next players</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\frac{1}{6})</td>
<td>(\frac{1}{6})</td>
</tr>
<tr>
<td>(\frac{5}{36})</td>
<td>(\frac{0.028}{0.028})</td>
</tr>
<tr>
<td>(\frac{1296}{1296})</td>
<td>(\frac{0.093}{0.070})</td>
</tr>
<tr>
<td>(\frac{1}{18})</td>
<td>(\frac{0.056}{0.056})</td>
</tr>
<tr>
<td>(\frac{324}{324})</td>
<td>(\frac{0.164}{0.117})</td>
</tr>
<tr>
<td>(\frac{216}{216})</td>
<td>(\frac{0.005}{0.005})</td>
</tr>
<tr>
<td>(\frac{1115}{46656})</td>
<td>(\frac{0.029}{0.025})</td>
</tr>
<tr>
<td>(\frac{46656}{10077696})</td>
<td>(\frac{0.075}{0.051})</td>
</tr>
<tr>
<td>(\frac{1}{12})</td>
<td>(\frac{0.014}{0.014})</td>
</tr>
<tr>
<td>(\frac{12992}{25992})</td>
<td>(\frac{0.069}{0.057})</td>
</tr>
<tr>
<td>(\frac{279936}{279936})</td>
<td>(\frac{0.154}{0.096})</td>
</tr>
<tr>
<td>(\frac{1}{36})</td>
<td>(\frac{0.028}{0.028})</td>
</tr>
<tr>
<td>(\frac{2592}{2592})</td>
<td>(\frac{0.115}{0.092})</td>
</tr>
<tr>
<td>(\frac{186624}{186624})</td>
<td>(\frac{0.228}{0.132})</td>
</tr>
</tbody>
</table>

6.4.2. Failure probabilities

Gross classing, depending on the representatives of the birepresentative components.

\[d = 2, f = 3\]; Sort@TableDeleteCases[allBirepresentatives[d, f], {x_, x_}]

\[
\begin{align*}
\text{listQuotient} & = \text{Sort@TableDeleteCases[allBirepresentatives[d, f], \{x_, x_\}, \{First\#, representativeLast\#\} \&, withValues → True]\} \\
\text{MapAt} & = \{\#, function1[\#] \& \@ \# \& \#, -1 \& /\text{listQuotient}[ \\
\text{Sort@TableDeleteCases[allBirepresentatives[d, f], \{x_, x_\}, \{First\#, representativeLast\#\} \&, withValues → True]\}}
\end{align*}
\]
Slice the table so that each piece holds in one page.

```
scanPrint[
  tableFormSlice[1 & /@ allRepresentatives[d, f]][First@#, TableDepth -> 4,
  TableHeadings -> Last@#] & @ fill[MapAt[(#, function1 @@ #) & /@ # & &, #, -1] & @ listQuotient[
    Sort@DeleteCases[allBirepresentatives[d, f], {x_, x_}],
    {First@#, representative@Last@#} & , withValues -> True] / . function1 -> "f1"
]
```

```
player = "pl"; j = 3;
```

TableForm bug: the line entry is missing in first table.
scanPrint@Flatten[
  With[{dl = First@#, j1 = Last@#},
    tableFormSlice[1 & @ allRepresentatives[dl, f]][First@#,
      TableHeadings -> Last@#,
      TableSpacing -> {1, 1, 3, 1},
      TableAlignments -> {Center, Center, Center, Center}
    ] &@fill[MapAt[
      ColumnForm[Insert[#, "\[Downarrow]", 2], Center],
      With[{goal = First@#, finalState = Last@#},
        TableForm[
          Map[numberForm1, probability[player, f, #, goal, #, finalState] & @ Range@j1]
        ]
      ] & /@ # & #, -1] & @ listQuotient[
    Sort@DeleteCases[allBirepresentatives[dl, f], {x_, x_}],
    {First@#, representative[Last@#]} &, withValues -> True]
  ] & @ finalDynamicParameters[d, j],
] 1]

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(\text{numberForm}1\left[\frac{1}{9}\right])</td>
<td>(\text{numberForm}1\left[\frac{1}{2}\right])</td>
</tr>
<tr>
<td>2</td>
<td>(\text{numberForm}1\left[\frac{1}{2}\right])</td>
<td>(\text{numberForm}1\left[\frac{1}{9}\right])</td>
</tr>
</tbody>
</table>

Patch.
When `TableForm` bug is repaired, replace patch by identity.
Clear[failureProbabilityChart];
Options[failureProbabilityChart] = {numberForm1 -> Identity};
failureProbabilityChart[player : "p1" | "p2",
  d_Integer, f_Integer, j_Integer, probability, options___] :=
  With[{patch = MapAt[TableForm[{{grayCombinationBox@{1}, #}, TableDirections -> Row] &, #, 1] &},
    Switch[player,
      "p1", Print["First player’s failure probabilities"],
      "p2", Print["Next players’ failure probabilities"]]
    ];
  scanPrint[patch@Flatten[
      With[{d1 = First@#, j1 = Last@#},
        tableFormSlice[1 & @ allRepresentatives[[d1, f]] [First@#,
              TableHeadings -> Map[grayCombinationBox, Last@#, {2}],
              TableSpacing -> {1, 1, 3, 1}]
          ] & @ fill[MapAt[
              ColumnForm[Insert[combinationBox/@#, ",", 2], Center],
              With[{goal = First@#, finalState = Last@#},
                TableForm[
                  With[{probabilities =
                        probability[player, f, #, goal, #, finalState] & @ Range@j1},
                    Map[numberForm1 /. (options) /. Options[failureProbabilityChart, probabilities]]
                  ]
              ]
            ] & @ fill[] & @ listQuotient[
                Sort[DeleteCases[allBirepresentatives[[d1, f]], {x_, x_}],
                  {First@#, representative@Last@#} & , withValues -> True]]
      ] & @ finalDynamicParameters[[d, j]], 1]]
With\(\{d = 2, f = 3, j = 3\}\),
\[
\text{failureProbabilityChart["p1", d, f, j, probability, numberForm1 \rightarrow \text{approximationBox}];}
\]
\[
\text{failureProbabilityChart["p2", d, f, j, probability, numberForm1 \rightarrow \text{approximationBox}];}
\]

First player's failure probabilities
\[
\begin{align*}
\frac{1}{3} &\approx 0.333 \\
\frac{2}{3} &\approx 0.222
\end{align*}
\]
\[
\begin{array}{c}
\frac{1}{3} \\
\frac{2}{3}
\end{array}
\]
\[
\begin{array}{c}
\frac{2}{3} \\
\frac{1}{3}
\end{array}
\]
\[
\begin{array}{c}
\frac{2}{3} \\
\frac{1}{3}
\end{array}
\]
\[
\begin{array}{c}
\frac{2}{3} \\
\frac{1}{3}
\end{array}
\]
\[
\begin{array}{c}
\frac{2}{3} \\
\frac{1}{3}
\end{array}
\]
\[
\begin{array}{c}
\frac{2}{3} \\
\frac{1}{3}
\end{array}
\]

Next players' failure probabilities
\[
\begin{align*}
\frac{1}{3} &\approx 0.333 \\
\frac{2}{3} &\approx 0.333
\end{align*}
\]
\[
\begin{array}{c}
\frac{1}{3} \\
\frac{2}{3}
\end{array}
\]
\[
\begin{array}{c}
\frac{2}{3} \\
\frac{1}{3}
\end{array}
\]
\[
\begin{array}{c}
\frac{2}{3} \\
\frac{1}{3}
\end{array}
\]
\[
\begin{array}{c}
\frac{2}{3} \\
\frac{1}{3}
\end{array}
\]
\[
\begin{array}{c}
\frac{2}{3} \\
\frac{1}{3}
\end{array}
\]
With\{d = 3, f = 6, j = 3\},

failureProbabilityChart["p1", d, f, j, probability, numberForm1 -> approximationBox];
failureProbabilityChart["p2", d, f, j, probability, numberForm1 -> approximationBox]

First player's failure probabilities
Next players' failure probabilities

\[
\begin{align*}
\frac{1}{6} & \approx 0.167 \\
\frac{1}{4} & \approx 0.250 \\
\frac{1}{3} & \approx 0.333 \\
\frac{1}{2} & \approx 0.500 \\
\end{align*}
\]
\[
\begin{align*}
\frac{1}{2} &= 0.056 \\
\frac{29}{324} &= 0.090 \\
\frac{25}{1296} &= 0.019 \\
\frac{1}{18} &= 0.056 \\
\frac{25}{648} &= 0.039 \\
\frac{19}{36} &= 0.028 \\
\frac{21}{18} &= 0.056 \\
\frac{19}{324} &= 0.059 \\
\frac{1}{18} &= 0.056 \\
\frac{1}{81} &= 0.012 \\
\frac{1}{81} &= 0.025 \\
\frac{72}{36} &= 0.028 \\
\frac{275}{7776} &= 0.018 \\
\frac{125}{15625} &= 0.008 \\
\frac{322}{3359232} &= 0.005
\end{align*}
\]
### 6.5. Consistency of probabilities

#### 6.5.1. Dynamic domain

#### 6.5.2. Invariance modulo face permutations

#### 6.5.3. Symmetry breaking due to purifying

#### 6.5.4. Unit probability sum over all final dates and states
6.5.5. All-time success probabilities as time sums

6.5.6. Inequalities on success probabilities

6.5.7. First player success probabilities do not depend on greatest cast number

6.5.8. Failure probabilities are the same for all players if goal and final state differ by at least 2 dice

6.5.9. One-cast probabilities

6.5.10. Formula for complete failure probabilities \( f = 2d \)

6.6. Statistics (Monte Carlo)

6.6.1. Simulating ratchet policy

6.6.2. All-time success statistics

6.6.3. Comparison with probabilities

7. Goal-driven policies and goal-driven strategy judging

7.1. Goal-driven utility functions

7.1.1. Relative goal utility, with or without serendipity

Clear@goalUtility;

\[
\text{goalUtility}[\_\_\_\_, \_\_\_, \_\_\_, \text{state1\_List}, \text{utility\_}, \_\_\_, \_\_] \text{@emptyGoal : } \{\} := \text{utility}[\_\_\_, \text{state1}] \\
\text{goalUtility}[\text{player\_}, \_\_\_, \_\_\_, \_\_\_, \text{state1\_List}, \text{utility\_}, \_\_\_, \text{probability}, \_\_\_, \text{serendipity\_}] \text{@goal\_List := Switch[\text{player},}
\]


\[
\sum_{j=1}^{\text{allCombinations}[d, f]} \text{probability}[p1, 3, j, \{1, 2\}, j2, \{1, 2\}] \text{utility}[j2, \{1, 2\}] + \\
\text{probability}[p1, 3, j, \{1, 2\}, j, \{1, 2\}] \text{utility}[j, \{1, 2\}]
\]

\(j = 3; \text{utility} = \text{only}\_\text{Cases}[\text{utilityList}[d, f, j], \text{transfer}[\_]@\_ \&]
\)

\(\text{transfer[2, 3][\#2] \&}
\)

\(\text{allCombinations}[d, f]
\)

\(\text{utility[@, \#] \&/@}
\)

\(\{\{1, 1\}, \{1, 2\}, \{1, 3\}, \{2, 2\}, \{2, 3\}, \{3, 3\}\}
\)

\(\{4, 6, 3, 2, 1, 3\}
\)

\(\text{goalUtility["p2", f, j, 0, {}, utility, probability, False] @ allCombinations[d, f]}
\)

1. %

\(\{844, 484, 242, 422, 242, 211, 729, 729, 729, 243, 729, 729, 243\}
\)

\(\{1.15775, 1.99177, 0.995885, 0.578875, 0.331962, 0.868313\}
\)

Goal utility depends on serendipity.

\(\text{Ordering[goalUtility["p2", f, j, 0, {}, utility, probability, False] @ allCombinations[d, f]]}
\)

\(\{5, 4, 6, 3, 1, 2\}
\)

\(\{6, 5, 4, 3, 2, 1\}
\)

Check: for a stationary utility function, the utility of a goal without serendipity is the all-time success probability multiplied by the utility of reaching the goal.

\(\text{utility = utility1[@2] \&}
\)

\(\text{And}@\text{Flatten[}
\)

\(\text{With[\{player = \#\},}
\)

\(\text{With[\{d1 = \text{First}@\#, j1 = \text{Last}@\#\},}
\)

\(\text{With[\{\text{fateTree1 = fateTree[player, d1, f, j1]\},}
\)

\(\text{With[\{\text{goal = \#}\},}
\)

\(\text{anyTimeSuccessProbability[fateTree1, goal] utility[@, goal] =}
\)

\(\text{goalUtility[player, f, j1, 0, {}, utility, probability, False] @ goal}
\)

\(\text{] @ allRepresentatives[d1, f]\}}
\)

\(\text{] \& @} \text{dynamicParameters[d, j]}
\)

\(\text{] \& @} \text{"p1", "p2"}]
\)

True

Check: with serendipity, goal utility also is the constant-goal strategy utility (for the same goal).

\(\text{utility = ;}
\)

\(\text{And}@\text{Simplify @}\text{Flatten[}
\)

\(\text{With[\{player = \#\},}
\)

\(\text{With[\{d1 = \text{First}@\#, j1 = \text{Last}@\#\},}
\)

\(\text{With[\{\text{fateTree1 = fateTree[player, d1, f, j1]\},}
\)

\(\text{With[\{\text{goal = \#}\},}
\)

\(\text{goalUtility[player, f, j1, 0, {}, utility, probability, True] @ goal =}
\)

\(\text{initial@utilityAndStrategy[fateTree1, utility] @}
\)

\(\text{extractStrategy@maxUtilityAndStrategy[fateTree1, \delta[goal, \#2] \&]
\)

\(\text{] @ allRepresentatives[d1, f]\}}
\)

\(\text{] \& @} \text{dynamicParameters[d, j]}
\)

\(\text{] \& @} \text{"p1", "p2"}]
\)

True

\(\&\) Once probabilities are known, constant-goal strategies can be judged with serendipitous goal utility function (relying on compiled probabilities) instead of and much faster than mean-mean.
Dynamic check.

And @With[{player = "pl"}, With[{fateTree = fateTree[player, d, f, j]},
Simplify@With[{goal = #}, goalUtility[player, f, j, 0, {}, utility, probability, True]@goal =
    initial@utilityAndStrategy[fateTree, utility]@extractStrategy@
    maxUtilityAndStrategy[fateTree1, δ[goal, #2] &] & @ allCombinations[d, f]]]
True

With[{player = "pl", utility = only@Cases[utilityList[d, f, j], transfer[___]@ &]},
With[{fateTree1 = fateTree[player, d, f]}, zoomTiming[
    With[{goal = #}, goalUtility[player, f, j, 0, {}, utility, probability, True]@goal] & @
    allCombinations[d, f], With[{goal = #},
    initial@utilityAndStrategy[fateTree1, utility]@extractStrategy@maxUtilityAndStrategy[    
    fateTree1, δ[goal, #2] &] & @ allCombinations[d, f], 30, compare]]]

{{0.08 Second, 2.67 Second}, {same, \{\frac{2924}{729}, \frac{3467}{729}, \frac{2315}{729}, \frac{1994}{729}, \frac{1379}{729}, \frac{1979}{729}\}}} 

(d = ., f = ., j = .);

7.1.2. Goal-driven utility

7.1.3. Driving goals

7.2. Restricted horizon policies evaluating non-Markovian strategies

7.3. Some dynamic policies

7.3.1. Definition and restriction

7.3.2. Unit horizon

7.3.3. Null horizon: goal explosion

7.4. Compiling strategy utilities

7.4.1. Variable goal-driven parameters and utility function

7.4.2. Hard cases

7.4.3. Completely random strategy and complete most useful strategy

7.5. Consistency of strategy utilities

7.5.1. Invariance modulo face permutations

7.5.2. Formulae for strategy utilities

8. Conclusion

8.1. Policy utility

8.1.1. Definition
8.1.2. Chart

Clear@policyUtilityChart;
policyUtilityChart[player : "p1" | "p2", d_Integer, f_Integer, j_Integer] := With[{parameters = {
utility -> # & /@ utilityList[d, f, j],
serendipity -> # & /@ {False, True},
horizon -> # & /@ {0, 1},
dynamism -> # & /@ {False, True}}),
Print@Switch[player, "p1", "First player", "p2", "Next players"];
Print@TableForm[Outer[
Unevaluated@noRational@

noSymbol@policyUtilityChart[player, d, f, j, utility, serendipity, horizon, dynamism]
/.(#1 & $ #2 & $ #2 & parameters, TableHeadings -> MapAt[Composition[smallBox, Last] /@ #1 &, parameters, 1] /.(horizon -> "h", dynamism -> "dyn")]]

"?" stands for undetermined utilities. Δ Misalignment: TableForm bug.

Block[{strategyUtility, randomStrategyUtility, maxStrategyUtility},
<< strategyUtility.txt; << randomStrategyUtility.txt;
<< maxStrategyUtility.txt; With[{d = 2, f = 3, j = 3},
policyUtilityChart["p1", d, f, j];
policyUtilityChart["p2", d, f, j]];
With[{d = 3, f = 6, j = 3},
policyUtilityChart["p1", d, f, j];
policyUtilityChart["p2", d, f, j]]

First player

<table>
<thead>
<tr>
<th>serendipity → False</th>
<th>serendipity → True</th>
</tr>
</thead>
<tbody>
<tr>
<td>dyn → False</td>
<td>dyn → True</td>
</tr>
<tr>
<td>h → 0</td>
<td>h → 0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0.705</td>
<td>0.705</td>
</tr>
<tr>
<td>0.807</td>
<td>0.857</td>
</tr>
<tr>
<td>0.686</td>
<td>0.686</td>
</tr>
<tr>
<td>0.970</td>
<td>0.970</td>
</tr>
<tr>
<td>0.836</td>
<td>0.836</td>
</tr>
<tr>
<td>0.888</td>
<td>0.914</td>
</tr>
<tr>
<td>0.952</td>
<td>0.952</td>
</tr>
<tr>
<td>0.813</td>
<td>0.980</td>
</tr>
<tr>
<td>0.755</td>
<td>1</td>
</tr>
<tr>
<td>0.355</td>
<td>1</td>
</tr>
<tr>
<td>0.656</td>
<td>1</td>
</tr>
<tr>
<td>0.256</td>
<td>1</td>
</tr>
<tr>
<td>Times @@ (#1 - 1 &amp;) /@ #2 &amp;</td>
<td></td>
</tr>
<tr>
<td>h → 0</td>
<td>h → 1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0.764</td>
<td>0.680</td>
</tr>
<tr>
<td>0.439</td>
<td>0.439</td>
</tr>
<tr>
<td>0.885</td>
<td>0.900</td>
</tr>
<tr>
<td>0.505</td>
<td>0.505</td>
</tr>
<tr>
<td>0.647</td>
<td>0.739</td>
</tr>
<tr>
<td>0.394</td>
<td>0.394</td>
</tr>
<tr>
<td>0.614</td>
<td>0.699</td>
</tr>
<tr>
<td>0.213</td>
<td>0.291</td>
</tr>
<tr>
<td>0.505</td>
<td>0.525</td>
</tr>
<tr>
<td>0.599</td>
<td>0.670</td>
</tr>
<tr>
<td>0.383</td>
<td>0.451</td>
</tr>
<tr>
<td>0.502</td>
<td>0.902</td>
</tr>
</tbody>
</table>

Next players
| δ([1, 2], #2) & |
|-----------------|-----------------|-----------------|-----------------|
| h → 0 | 0.308 | 0.308 | 1 | 1 |
| h → 1 | 0 | 0 | 1 | 1 |

| δ([1, 2], #2) + δ([2, 3], #2) & |
|-----------------|-----------------|-----------------|-----------------|
| h → 0 | 0.813 | 0.813 | 1 | 1 |
| h → 1 | 0.963 | 1 | 1 | 1 |

| δ([1, 1], #2) + δ([2, 2], #2) & |
|-----------------|-----------------|-----------------|-----------------|
| h → 0 | 0.615 | 0.615 | 1 | 1 |
| h → 1 | 0.615 | 0.615 | 1 | 1 |

| transfer[2, 3][#2] & |
|-----------------|-----------------|-----------------|-----------------|
| h → 0 | 0.492 | 0.492 | 1 | 1 |
| h → 1 | 0.492 | 0.492 | 1 | 1 |

| Max[#2] & |
|-----------------|-----------------|-----------------|-----------------|
| h → 0 | 0.308 | 0.308 | 1 | 1 |
| h → 1 | 0 | 0 | 1 | 1 |

| Total[#2] & |
|-----------------|-----------------|-----------------|-----------------|
| h → 0 | 0.754 | 0.754 | 1 | 1 |
| h → 1 | 0.938 | 1 | 1 | 1 |

| Times @@ (#1 - 1 &) / @#2 & |
|-----------------|-----------------|-----------------|-----------------|
| h → 0 | 0 | 1 | 1 | 1 |
| h → 1 | 1 | 1 | 1 | 1 |

| randomUtility[2, 3, 280865] |
|-----------------|-----------------|-----------------|-----------------|
| h → 0 | 0.871 | 0.989 | 0.965 | 0.965 |
| h → 1 | 0.989 | 0.992 | 0.989 | 0.996 |

| randomUtility[2, 3, 30868] |
|-----------------|-----------------|-----------------|-----------------|
| h → 0 | 0.777 | 0.777 | 1 | 1 |
| h → 1 | 0.777 | 0.775 | 1 | 1 |

| randomUtility[2, 3, 3088] |
|-----------------|-----------------|-----------------|-----------------|
| h → 0 | 0.704 | 0.704 | 0.968 | 0.968 |
| h → 1 | 0.704 | 0.686 | 0.968 | 1 |

| randomUtility[2, 3, 380] |
|-----------------|-----------------|-----------------|-----------------|
| h → 0 | 0.905 | 0.905 | 0.905 | 0.976 |
| h → 1 | 0.911 | 0.913 | 0.976 | 1 |

| randomUtility[2, 3, 28] |
|-----------------|-----------------|-----------------|-----------------|
| h → 0 | 0.181 | 0.181 | 1 | 1 |
| h → 1 | 0.122 | 0.122 | 1 | 1 |

| randomUtility[2, 3, 2] |
|-----------------|-----------------|-----------------|-----------------|
| h → 0 | 0.962 | 0.953 | 0.962 | 0.962 |
| h → 1 | 0.823 | 0.835 | 0.962 | 1 |

### First player

| δ([1, 2, 3], #2) & |
|-----------------|-----------------|-----------------|-----------------|
| h → 0 | 0.684 | 0.684 | 0.684 | 0.684 |
| h → 1 | 0.980 | 0.980 | 0.980 | 0.980 |

| δ([1, 2, 3], #2) + δ([4, 5, 6], #2) & |
|-----------------|-----------------|-----------------|-----------------|
| h → 0 | 0.806 | 0.806 | 0.806 | 0.806 |
| h → 1 | 0.911 | 0.980 | 0.911 | 0.980 |

| δ([1, 1], #2) + δ([2, 2], #2) & |
|-----------------|-----------------|-----------------|-----------------|
| h → 0 | 0.605 | 0.605 | 0.605 | 0.605 |
| h → 1 | 0.946 | 1 | 0.946 | 1 |

| δ([1, 2, 3], #2) + δ([3, 4, 5], #2) + δ([1, 5, 6, #2) & |
|-----------------|-----------------|-----------------|-----------------|
| h → 0 | 0.599 | 0.599 | 0.599 | 0.599 |
| h → 1 | 0.946 | 0.960 | 0.950 | 0.974 |

| δ([1, 1], #2) + δ([2, 2], #2) + δ([2, 2, 3], #2) & |
|-----------------|-----------------|-----------------|-----------------|
| h → 0 | 0.587 | 0.587 | 0.587 | 0.587 |
| h → 1 | 0.865 | 1 | 0.865 | 1 |

| Max[#2] & |
|-----------------|-----------------|-----------------|-----------------|
| h → 0 | 0.684 | ? | 1 | ? |
| h → 1 | 0.050 | 0.050 | 1 | 1 |

| Total[#2] & |
|-----------------|-----------------|-----------------|-----------------|
| h → 0 | 0.768 | 0.829 | 0.857 | 0.893 |
| h → 1 | 0.004 | 0.004 | 0.915 | 0.996 |

| Times @@ (#1 - 1 &) / @#2 & |
|-----------------|-----------------|-----------------|-----------------|
| h → 0 | 0.778 | 0.815 | 0.778 | 0.819 |
| h → 1 | 0.739 | 0.706 | 0.857 | 0.992 |

| Times @@ #2 & |
|-----------------|-----------------|-----------------|-----------------|
| h → 0 | 0.798 | 0.835 | 0.798 | 0.838 |
| h → 1 | 0.507 | 0.433 | 0.871 | 0.993 |

| randomUtility[3, 6, 3, 280865] |
|-----------------|-----------------|-----------------|-----------------|
| h → 0 | 0.411 | 0.507 | 0.697 | 0.858 |
| h → 1 | 0.336 | 0.348 | 0.437 | 0.461 |

| randomUtility[3, 6, 3, 2808] |
|-----------------|-----------------|-----------------|-----------------|
| h → 0 | 0.620 | 0.561 | 0.877 | 0.974 |
| h → 1 | 0.442 | 0.440 | 0.479 | 0.504 |

| randomUtility[3, 6, 3, 280] |
|-----------------|-----------------|-----------------|-----------------|
| h → 0 | 0.436 | 0.465 | 0.761 | 0.944 |
| h → 1 | 0.395 | 0.395 | 0.444 | 0.452 |

| randomUtility[3, 6, 3, 28] |
|-----------------|-----------------|-----------------|-----------------|
| h → 0 | 0.400 | 0.339 | 0.400 | 0.406 |
| h → 1 | 0.436 | 0.465 | 0.761 | 0.944 |

| randomUtility[3, 6, 3, 2] |
|-----------------|-----------------|-----------------|-----------------|
| h → 0 | 0.325 | 0.325 | 0.338 | 0.340 |
| h → 1 | 0.520 | 0.516 | 0.846 | 0.938 |

### Next players
8.2. Empiric laws of policy utility

8.2.1. Definition: fuzzy utility function

8.2.2. Serendipity is almost always useful (though not strictly)

<< Graphics`GraphicsUtilityChart;
serendipityUtilityChart[d_, f_, j_] := With[{serendipityUtilityList =
  Distribute[{"p1", "p2"}, utilityList[d, f, j], {0, 1}, {False, True}], List, List, List, With[{player = #1, utility = #2, horizon = #3, dynamism = #4},
  (policyUtility[player, d, f, j, utility, True, horizon, dynamism] =
   policyUtility[player, d, f, j, utility, False, horizon, dynamism],
   player, smallBoxAt[utility, horizon, dynamism, noSymbol, fuzzyQ[player, d, f, j, utility]] &}],
  Print@TableForm[MapAt[Composition[noRational, noSymbol], 
    {1, 1} & /@
    Sort@serendipityUtilityList, TableHeadings ->
    (None, {"", "", "", horizon, dynamism, fuzzyQ}), TableSpacing -> {1, 1}];
  Histogram[DeleteCases[noSymbol @@ First @@ serendipityUtilityList, 
    ""]]}]
Block[{strategyUtility, randomStrategyUtility, maxStrategyUtility},
  << strategyUtility.txt; << randomStrategyUtility.txt; << maxStrategyUtility.txt]
Block[{strategyUtility, randomStrategyUtility, maxStrategyUtility},
  << strategyUtility.txt; << randomStrategyUtility.txt; << maxStrategyUtility.txt;
  serendipityUtilityChart[2, 3, 3];
  serendipityUtilityChart[3, 6, 3]]
0 p1 δ[[1, 2], #2] & 1 False False
0 p1 δ[[1, 2], #2] & 1 True False
0 p1 δ[[1, 1], #2] + δ[[1, 2], #2] & 0 False False
0 p1 δ[[1, 1], #2] + δ[[1, 2], #2] & 0 False True
0 p1 δ[[1, 1], #2] + δ[[2, 2], #2] & 0 False False
0 p1 δ[[1, 1], #2] + δ[[2, 2], #2] & 0 True False
0 p1 δ[[1, 1], #2] + δ[[2, 2], #2] & 0 False False
0 p1 δ[[1, 1], #2] + δ[[2, 2], #2] & 1 False False
0 p1 δ[[1, 1], #2] + δ[[2, 2], #2] & 1 True False
0 p1 δ[[1, 2], #2] + δ[[2, 3], #2] & 0 False False
0 p1 δ[[1, 2], #2] + δ[[2, 3], #2] & 0 True False
0 p1 transfer[2, 3][#2] & 0 False False
0 p1 transfer[2, 3][#2] & 0 True False
0 p1 randomUtility[2, 3, 28] & 0 False False
0 p1 randomUtility[2, 3, 28] & 0 True False
0 p1 randomUtility[2, 3, 2808] & 0 False False
0 p1 randomUtility[2, 3, 2808] & 0 True False
0 p1 randomUtility[2, 3, 28086] & 0 False False
0 p1 randomUtility[2, 3, 28086] & 0 True False
0 p2 Times[] & 0 False False
0 p2 Times[] & 0 True False
0 p2 Times[] & 1 False False
0 p2 Times[] & 0 False False
0 p2 Times[] & 0 True False
0 p2 Times[] & 1 False False
0 p2 Times[] & 1 True False
0 p2 Times[] & 1 True False
0 p2 Times[] & 1 True False
0 p2 Times[] & 1 True False
0 p2 δ[[1, 2], #2] & 0 False False
0 p2 δ[[1, 2], #2] & 0 True False
0 p2 δ[[1, 2], #2] & 0 False False
0 p2 δ[[1, 2], #2] & 1 False False
0 p2 δ[[1, 2], #2] & 1 True False
0 p2 δ[[1, 2], #2] & 1 True False
0 p2 δ[[1, 1], #2] + δ[[2, 2], #2] & 1 False False
0 p2 δ[[1, 1], #2] + δ[[2, 2], #2] & 1 True False
0 p2 δ[[1, 1], #2] + δ[[2, 2], #2] & 1 True False
0 p2 δ[[1, 1], #2] + δ[[2, 2], #2] & 1 False False
0 p2 δ[[1, 1], #2] + δ[[2, 2], #2] & 1 True False
0 p2 δ[[1, 1], #2] + δ[[2, 2], #2] & 1 True False
0.004 p2 randomUtility[2, 3, 28086] & 1 False False
0.009 p2 randomUtility[2, 3, 28086] & 1 True False
0.014 p1 randomUtility[2, 3, 28086] & 1 False False
0.019 p1 randomUtility[2, 3, 28086] & 1 False True
0.026 p1 δ[[1, 1], #2] + δ[[1, 2], #2] & 1 False False
0.038 p2 δ[[1, 1], #2] + δ[[2, 2], #2] & 1 False False
0.044 p1 randomUtility[2, 3, 280] & 1 True True
0.051 p1 δ[[1, 2], #2] + δ[[2, 3], #2] & 1 False False
0.052 p1 δ[[1, 1], #2] + δ[[1, 2], #2] & 1 True False
0.059 p1 Times[] & 0 False False
0.062 p2 Total[#2] & 0 False False
0.066 p2 randomUtility[2, 3, 280] & 1 False False
0.068 p1 randomUtility[2, 3, 280] & 0 False True
0.071 p1 randomUtility[2, 3, 280] & 0 True False
0.072 p2 randomUtility[2, 3, 280] & 0 True False
0.077 p1 randomUtility[2, 3, 280] & 0 False True
0.085 p1 randomUtility[2, 3, 280] & 1 False False
0.087 p2 randomUtility[2, 3, 280] & 1 True False
0.092 p1 randomUtility[2, 3, 280] & 1 False False
0.094 p2 randomUtility[2, 3, 280] & 0 False False
0.109 p1 randomUtility[2, 3, 280] & 1 True False
0.139 p2 randomUtility[2, 3, 280] & 1 False False
0.162 p1 randomUtility[2, 3, 28] & 1 False False
0.165 p1 δ[[1, 2], #2] + δ[[2, 3], #2] & 1 True False
0.165 p2 randomUtility[2, 3, 28] & 1 True False
0.167 p1 transfer[2, 3][#2] & 1 False False
0.171 p1 randomUtility[2, 3, 280] & 0 True True
0.188 p2 δ[[1, 1], #2] + δ[[2, 2], #2] & 0 False False
0.188 p2 δ[[1, 1], #2] + δ[[2, 2], #2] & 0 True False
0.208 p1 randomUtility[2, 3, 280] & 1 False False
0.236 p1 Times[] & 1 False False
0.244 p1 Total[#2] & 0 True True
0.245 p1 Max[#2] & 0 False True
0.245 p1 Max[#2] & 0 True True
0.246 p2 Total[#2] & 1 False False
0.246 p2 Total[#2] & 1 True False
0.246 p2 randomUtility[2, 3, 280] & 0 False False
| m    | randomUtility[2, 3, 3, 2808] | q | t | r | o | T | Z | W | V | U | T | S | R | Q | P | O | N | M | L | K | J | I | H | G | F | E | D | C | B | A |
| 0.264| p2 randomUtility[2, 3, 3, 2808] | 0 | True | True |
| 0.264| p2 randomUtility[2, 3, 3, 2808] | 1 | False | True |
| 0.286| p1 randomUtility[2, 3, 3, 2] | 0 | True | True |
| 0.298| p1 transfer[2, 3] [\#2] & | 1 | True | False |
| 0.314| p2 randomUtility[2, 3, 3, 2808] | 1 | True | False |
| 0.320| p1 Times \@1 \#1 & | 1 | True | False |
| 0.344| p1 Total[\#2] & | 0 | False | True |
| 0.385| p2 \[1, 1], \#2 + \[1, 2], \#2 & | 0 | False | True |
| 0.385| p2 \[1, 1], \#2 + \[1, 2], \#2 & | 0 | True | True |
| 0.385| p2 \[1, 1], \#2 + \[1, 2], \#2 & | 1 | False | True |
| 0.385| p2 \[1, 1], \#2 + \[1, 2], \#2 & | 1 | True | True |
| 0.400| p1 randomUtility[2, 3, 3, 2] | 1 | True | False |
| 0.490| p1 randomUtility[2, 3, 3, 2] | 1 | True | True |
| 0.508| p2 transfer[2, 3] [\#2] & | 0 | False | True |
| 0.508| p2 transfer[2, 3] [\#2] & | 0 | False | True |
| 0.508| p2 transfer[2, 3] [\#2] & | 1 | True | False |
| 0.508| p2 transfer[2, 3] [\#2] & | 1 | True | True |
| 0.645| p1 Max[\#2] & | 1 | False | True |
| 0.685| p1 Max[\#2] & | 1 | True | True |
| 0.692| p2 \[1, 1], \#2 + \[2, 3], \#2 & | 0 | False | True |
| 0.692| p2 \[1, 1], \#2 + \[2, 3], \#2 & | 0 | True | True |
| 0.692| p2 Max[\#2] & | 0 | False | True |
| 0.692| p2 Max[\#2] & | 0 | True | True |
| 0.744| p1 Total[\#2] & | 1 | False | True |
| 0.800| p1 Total[\#2] & | 1 | True | True |
| 0.819| p2 randomUtility[2, 3, 3, 28] | 0 | False | True |
| 0.819| p2 randomUtility[2, 3, 3, 28] | 0 | True | True |
| 0.878| p2 randomUtility[2, 3, 288] | 1 | False | True |
| 0.878| p2 randomUtility[2, 3, 288] | 1 | True | True |
| 0.923| p2 randomUtility[2, 3, 3, 28086] | 0 | False | True |
| 0.923| p2 randomUtility[2, 3, 3, 28086] | 0 | True | True |
| 0.925| p2 randomUtility[2, 3, 3, 28086] | 1 | True | True |
| 1    | p2 \[1, 2], \#2 + \[2, 3], \#2 & | 1 | False | True |
| 1    | p2 \[1, 2], \#2 + \[2, 3], \#2 & | 1 | True | True |
| 1    | p2 Max[\#2] & | 1 | False | True |
| 1    | p2 Max[\#2] & | 1 | True | True |

---

**Graph:**

![Graph](image_url)

**Table:**

<table>
<thead>
<tr>
<th>horizon</th>
<th>dynamism</th>
<th>fuzzyQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>0</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>0</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>0</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>0</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>0</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>0</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>0</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>0</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>0</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>0</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>0</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>0</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>0</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>0</td>
<td>False</td>
<td>False</td>
</tr>
</tbody>
</table>

---

**Diagram:**

![Diagram](image_url)
0.118 p2 Times [#1 - 1 &] / @ #2 & 1 True False
0.126 p2 δ[(1, 2, 3), #2] + δ[(2, 3, 4), #2] & 1 False False
0.142 p2 Times #2 & 1 True False
0.151 p2 transfer[3, 6] #2 & 0 True True
0.165 p1 transfer[3, 6] #2 & 1 True False
0.199 p2 δ[(1, 2, 3), #2] + δ[(2, 3, 4), #2] & 1 False False
0.207 p2 randomUtility[3, 6, 3, 280] 0 True True
0.218 p2 transfer[3, 6] #2 & 1 False True
0.257 p1 randomUtility[3, 6, 3, 28086] 1 False True
0.263 p2 transfer[3, 6] #2 & 1 True True
0.285 p1 Times [#1 - 1 &] / @ #2 & 1 True False
0.286 p1 randomUtility[3, 6, 3, 28086] 1 False True
0.302 p1 randomUtility[3, 6, 3, 2808] 1 False True
0.316 p1 Max[#2] & 0 False ?
0.325 p1 randomUtility[3, 6, 3, 28] 1 False False
0.326 p1 randomUtility[3, 6, 3, 2] 1 False True
0.345 p1 randomUtility[3, 6, 3, 280] 1 False False
0.347 p1 randomUtility[3, 6, 3, 28086] 0 True True
0.351 p1 randomUtility[3, 6, 3, 28086] 1 True True
0.365 p1 Times #2 & 1 False False
0.372 p1 randomUtility[3, 6, 3, 28086] 0 False True
0.390 p2 Max[#2] & 0 False ?
0.413 p1 randomUtility[3, 6, 3, 28086] 1 True True
0.422 p1 randomUtility[3, 6, 3, 2] 1 True True
0.430 p2 randomUtility[3, 6, 3, 28086] 1 False False
0.433 p2 randomUtility[3, 6, 3, 2808] 0 True True
0.433 p1 randomUtility[3, 6, 3, 2808] 1 True True
0.467 p2 Total[#2] & 1 False True
0.479 p1 randomUtility[3, 6, 3, 28] 1 True False
0.484 p2 randomUtility[3, 6, 3, 28086] 1 True True
0.496 p2 randomUtility[3, 6, 3, 28086] 1 False True
0.497 p2 randomUtility[3, 6, 3, 2808] 0 False True
0.513 p2 Total[#2] & 1 True True
0.539 p1 randomUtility[3, 6, 3, 280] 1 True True
0.540 p2 randomUtility[3, 6, 3, 2] 0 False True
0.549 p2 randomUtility[3, 6, 3, 28086] 1 True True
0.560 p1 Times #2 & 1 True False
0.577 p2 randomUtility[3, 6, 3, 2] 0 True True
0.586 p2 randomUtility[3, 6, 3, 2808] 1 False True
0.586 p2 randomUtility[3, 6, 3, 28] 1 False True
0.606 p2 randomUtility[3, 6, 3, 280] 1 False True
0.614 p2 randomUtility[3, 6, 3, 28] 1 True True
0.641 p2 randomUtility[3, 6, 3, 2808] 1 True True
0.655 p2 randomUtility[3, 6, 3, 280] 1 True True
0.690 p2 randomUtility[3, 6, 3, 2] 1 False True
0.742 p2 randomUtility[3, 6, 3, 28] 0 False True
0.763 p2 randomUtility[3, 6, 3, 2] 1 True True
0.797 p2 randomUtility[3, 6, 3, 28] 0 True True
0.824 p2 randomUtility[3, 6, 3, 28086] 0 False True
0.850 p2 randomUtility[3, 6, 3, 28086] 0 True True
0.911 p1 Total[#2] & 1 False True
0.950 p1 Max[#2] & 1 False ?
0.950 p1 Max[#2] & 1 True ?
0.992 p1 Total[#2] & 1 True True
1.032 p2 Max[#2] & 1 False ?
1.032 p2 Max[#2] & 1 True ?
? p1 Max[#2] & 0 True ?
? p2 Max[#2] & 0 True ?
8.2.3. Cases of harmful serendipity

<< strategyUtility.txt; << randomStrategyUtility.txt; << maxStrategyUtility.txt;

Check:

With[{player = "p1", d = 2, f = 3, j = 3}, With[{fateTree1 = fateTree[player, d, f, j],
    utility = randomUtility[d, f, j, 280865], horizon = 1, dynamism = True},
  Block[{strategyUtility, randomStrategyUtility, maxStrategyUtility, no, yes, zero, unit},
    make[fateTree1, probability]@strategyUtility[_. _. _, _. _. _, utility, False, horizon, dynamism];
    Print[1. (no = strategyUtility[player, d, f, j, utility, False, horizon, dynamism])]; make[
      fateTree1, probability]@strategyUtility[_. _. _, _. _. _, utility, dynamism, horizon, dynamism];
    Print[1. (yes = strategyUtility[player, d, f, j, utility, True, horizon, dynamism])]; make[
      fateTree1, utility];
    Print[1. (zero = randomStrategyUtility[player, d, f, j, utility])];
    make@maxStrategyUtility[fateTree1, utility];
    Print[1. (unit = maxStrategyUtility[player, d, f, j, utility])];
    1. Last@# - First@# &@{no - zero, yes - zero, unit - zero})
0.635986
0.635171
0.426232
0.643238
-0.00375516

With[{player = "p2", d = 2, f = 3, j = 3}, With[{fateTree1 = fateTree[player, d, f, j],
    utility = randomUtility[d, f, j, 280865], horizon = 0, dynamism = True},
  Block[{strategyUtility, randomStrategyUtility, maxStrategyUtility, no, yes, zero, unit},
    make[fateTree1, probability]@strategyUtility[_. _. _, _. _. _, utility, False, horizon, dynamism];
    Print[1. (no = strategyUtility[player, d, f, j, utility, False, horizon, dynamism])]; make[
      fateTree1, probability]@strategyUtility[_. _. _, _. _. _, utility, dynamism, horizon, dynamism];
    Print[1. (yes = strategyUtility[player, d, f, j, utility, True, horizon, dynamism])]; make[
      randomStrategyUtility[fateTree1, utility];
    Print[1. (zero = randomStrategyUtility[player, d, f, j, utility])];
    make@maxStrategyUtility[fateTree1, utility];
    Print[1. (unit = maxStrategyUtility[player, d, f, j, utility])];
    1. Last@# - First@# &@{no - zero, yes - zero, unit - zero})
0.430143
0.427943
0.339934
0.431121
-0.0241287

What is going on can be seen easily, as parameters are small and strategy is Markovian.
player = "p1"; (d = 2, f = 3, j = 3); utility = randomUtility[d, f, j, 280865];

fateTree1 = fateTree["p1", d, f, j, utility];

strategy1 = strategy[fateTree1, utility, probability, True, 1, True]

1. initial@utilityAndStrategy[fateTree1, utility]@strategy1

0.635171

Show[strategyGraphics[fateTree1, withProbabilities → False]@strategy1, ImageSize → 872];

Compare with complete most useful strategy.
Strategies occur to differ by the choice of 11 after first cast! \( \nabla \) Missing: diff tool (see AuthorTools).

```math
\text{strategy2} = \text{extractStrategy@maxUtilityAndStrategy[fateTree1, utility]};
\text{Show[strategyGraphics[fateTree1, withProbabilities -> False]@strategy2, ImageSize -> 8 72];}
```

8.2.4. **Serendipity makes horizon and dynamism useful**

8.2.5. **No serendipity makes horizon or dynamism often strictly harmful, especially for fuzzy utility functions**

8.2.6. **For non-fuzzy utility functions, horizon and dynamism are often more useful than serendipity**

8.3. **Meta-policy**

Def.: a *meta-policy* is a program evaluating exactly one policy.

A typical meta-policy consists in choosing among a finite set of policies, according to some utility function.
Def.: the meta-timing (resp. spacing) of a policy is the timing (resp. spacing) of its judging for some utility function.

The timing of the typical meta-policy is the sum of meta-timings on the policy set:
"timing of meta-policy = sum of policy meta-timings".
(We neglect the timing of utility maximizing.)

As mean-max judges itself, remarkably, its meta-timing (resp. spacing) identifies with its timing (resp. spacing). Indeed, among all "reasonable" policies mean-max maximizes timing but minimizes meta-timing [1]; we will checked this numerically:

```plaintext
player = "p1"; {d = 3, f = 6, j = 3}; utility = onlyCases[utilityList[d, f, j], transfer[___]@ _ &];
fateTree1 = fateTree[player, d, f, j]; fateTree2 = fateTree[player, d, f, j, utility];
Block[{maxStrategyUtility}, Timing@make@maxStrategyUtility[fateTree1, utility]]
timing1 = First@%;
{0.34 Second, 349621/93312}

Block[{maxStrategyUtility}, Timing@make@maxStrategyUtility[fateTree2, utility]]
{0.26 Second, 349621/93312}
```

Some meta-timings are lower than mean-max timing, because of a goal utility function short cut (using compiled probabilities).

```plaintext
scanPrint[
  timing2 = Block[{strategyUtility}, Distribute[({False, True}, {0, 1}, {False, True}), List, List, List, {##, Timing@make[fateTree2, probability]@strategyUtility[___, __, __, __, utility, ##]} &])]
{False, 0, False, {0.03 Second, 952727/279936}}
{False, 0, True, {6.94 Second, 952727/279936}}
{False, 1, False, {0.26 Second, 478337/139968}}
{False, 1, True, {0.41 Second, 52757/15552}}
{True, 0, False, {0.46 Second, 952727/279936}}
{True, 0, True, {6.24 Second, 956723/279936}}
{True, 1, False, {0.75 Second, 167123/46656}}
{True, 1, True, {1.45 Second, 58057/15552}}

Apply[And, #〚[-1, 1]〛 ≟ timing1 /. Second → 1 & /@ timing2]
False
```

Without short cut:
scanPrint[
  timing3 = Block[{strategyUtility}, Distribute[{{False, True}, {0, 1}, {False, True}}, List, List, List, {##, Timing@make[fateTree2, probability, meanMaxShortCut -> False]@strategyUtility[_, _, _, _, utility, ##]}]]

{False, 0, False, {0.88 Second, 952727 279936 }}
{False, 0, True, {7.06 Second, 952727 279936 }}
{False, 1, False, {1.44 Second, 478337 139968 }}
{False, 1, True, {0.42 Second, 52757 15552 }}
{True, 0, False, {1.3 Second, 952727 279936 }}
{True, 0, True, {6.23 Second, 956723 279936 }}
{True, 1, False, {1.75 Second, 167123 46656 }}
{True, 1, True, {1.48 Second, 58057 15552 }}

Q. E. D. & Try more utility functions.

Apply[And, #[-1, 1] \[DoubleDash] timing1 /. Second \[DoubleDash] 1 & @ timing3]

True

player =.; {d, f, j} =.; utility =.; fateTree1 =.;
fateTree2 =.; timing1 =.; timing2 =.; timing3 =.;

In general, policy utility should take into account not only strategy utility but also policy timing and spacing (negatively). However, timing and spacing are not yet available, as goal-driven policies were judged but not realized.

9. References


