Пример решения задачи оптимизации портфеля через динамическое программирование в Julia

Динамическое программирование (ДП) — метод решения задач с оптимальной подструктурой через разбиение на подзадачи. Основная идея:

- Принцип оптимальности Беллмана: оптимальное решение всей задачи содержит оптимальные решения подзадач
- Мемоизация: сохранение результатов вычислений для повторного использования

В инвестиционной задаче:

- Состояние: текущий капитал и распределение активов
- Действие: ребалансировка портфеля
- Награда: прирост капитала за период

Постановка задачи

Инвестор хочет распределить $10,000 между 3 активами:

Актив	Ожидаемая доходность	Риск (σ)	Минимальная доля	Максимальная доля
A	8%	10%	10%	50%
B	12%	15%	10%	50%
C	6%	5%	10%	50%

Ковариационная матрица: - Cov(A,B) = 0.5% - Cov(A,C) = -0.2% - Cov(B,C) = 0.3%

Особенности модели:

- Риск активов растёт на 20% каждый период
- Ребалансировка возможна с шагом 10%
- Цель: максимизация общего капитала

[ ]:
```
INITIAL_CAPITAL = 10000.0
ASSETS = [:A, :B, :C]
EXPECTED_RETURNS = [0.08, 0.12, 0.06]
RISKS = [0.10, 0.15, 0.05]
N_PERIODS = 3
ALLOCATION_OPTIONS = 0.1:0.1:0.5  # Шаг 10%
MIN_SHARE = 0.10
MAX_SHARE = 0.50
RISK_GROWTH_RATE = 0.20  # Рост риска за период
ROUND_PRECISION = 3      # Точность округления долей
```

[ ]: 3

Инициализация окружения

Подключим необходимые пакеты:

[ ]:
```
# import Pkg
# Pkg.add(["Distributions", "Random", "LinearAlgebra"])
```

[ ]:
```
using Distributions, Random, LinearAlgebra
```

Генерация доходностей

Модель шоков с растущим риском. Для генерации используем многомерное нормальное распределение

[ ]:
```
function generate_returns(period)
    shocks = rand(MvNormal(EXPECTED_RETURNS, diagm(RISKS)))
    return shocks .* (1 + RISK_GROWTH_RATE * (period-1))  # Риск␣
  ↪растёт со временем
end
```

[ ]:
```
generate_returns (generic function with 1 method)
```

Мемоизация

Словарь для хранения уже вычисленных значений функции Беллмана. Ключ — кортеж (период, доли активов)

[ ]:
```
const memo = Dict{Tuple{Int, Float64, Float64, Float64}, Float64}()
```

```
WARNING: redefinition of constant Main.memo. This may fail, cause␣
  ↪incorrect
answers, or produce other errors.
```

[ ]:
```
Dict{Tuple{Int64, Float64, Float64, Float64}, Float64}()
```

Функция Беллмана

Рекурсивная функция для вычисления максимального ожидаемого капитала. На каждом шаге:

1. Проверяем терминальное условие
2. Используем кэш при наличии
3. Перебираем все допустимые ребалансировки

4. Вычисляем текущий прирост и рекурсивно следующий период

[ ]:
```
function bellman(period, xA, xB, xC)
    # Нормировка и округление
    xC = round(1 - xA - xB, digits=ROUND_PRECISION)
    (xC < MIN_SHARE || xC > MAX_SHARE) && return -Inf

    # Терминальное условие
    (period > N_PERIODS) && return 0.0

    # Проверка кэша
    key = (period, xA, xB, xC)
    haskey(memo, key) && return memo[key]

    # Генерация доходностей для периода
    returns = generate_returns(period)

    max_value = -Inf
    for new_xA in ALLOCATION_OPTIONS
        (new_xA < MIN_SHARE || new_xA > MAX_SHARE) && continue

        for new_xB in ALLOCATION_OPTIONS
            new_xC = round(1 - new_xA - new_xB,␣
  ↪digits=ROUND_PRECISION)
            (new_xC < MIN_SHARE || new_xC > MAX_SHARE) && continue

            # Расчет прироста капитала
            growth = new_xA*returns[1] + new_xB*returns[2] +␣
  ↪new_xC*returns[3]
            current_value = (1 + growth) * INITIAL_CAPITAL

            # Рекурсивный вызов
            future_value = bellman(period+1, new_xA, new_xB, new_xC)
            total_value = current_value + future_value

            # Обновление максимума
            max_value = max(max_value, total_value)
        end
    end

    memo[key] = max_value  # Сохраняем в кэш
    return max_value
end
```

[ ]:
```
bellman (generic function with 1 method)
```

Оптимизация

Запуск вычислений с начальной точки. Для воспроизводимости фиксируем seed

[ ]:
```
Random.seed!(42)
optimal_value = bellman(1, 0.3, 0.5, 0.2)  # Начальная стратегия
```

[ ]:
```
39890.61203656556
```

Визуализация результатов

Извлекаем оптимальные стратегии из кэша. Для каждого периода показываем допустимые распределения

[ ]:
```
println("Максимальный ожидаемый капитал: \$", round(optimal_value,␣
  ↪digits=2))
println("\nОптимальные стратегии по периодам:")

for period in 1:N_PERIODS
    println("\nЭтап $period:")
    for (key, value) in memo
        key[1] == period || continue
        println("  Доли: A=$(key[2]), B=$(key[3]), C=$(key[4]) → ",␣
  ↪round(value, digits=2))
    end
end
```

```
Максимальный ожидаемый капитал: $39890.61

Оптимальные стратегии по периодам:

Этап 1:
  Доли: A=0.3, B=0.5, C=0.2 → 39890.61

Этап 2:
  Доли: A=0.3, B=0.3, C=0.4 → 22927.59
  Доли: A=0.1, B=0.5, C=0.4 → 21892.18
  Доли: A=0.4, B=0.1, C=0.5 → 26756.93
  Доли: A=0.1, B=0.4, C=0.5 → 24236.6
  Доли: A=0.4, B=0.3, C=0.3 → 26463.4
  Доли: A=0.4, B=0.4, C=0.2 → 27193.87
  Доли: A=0.4, B=0.5, C=0.1 → 28790.92
  Доли: A=0.5, B=0.3, C=0.2 → 27408.74
  Доли: A=0.5, B=0.4, C=0.1 → 21051.43
  Доли: A=0.2, B=0.5, C=0.3 → 28268.35
  Доли: A=0.3, B=0.2, C=0.5 → 26400.81
  Доли: A=0.2, B=0.4, C=0.4 → 26545.59
```

```
  Доли: A=0.5, B=0.2, C=0.3 → 29091.81
  Доли: A=0.4, B=0.2, C=0.4 → 22415.74
  Доли: A=0.2, B=0.3, C=0.5 → 24893.27
  Доли: A=0.3, B=0.4, C=0.3 → 28843.29
  Доли: A=0.3, B=0.5, C=0.2 → 27077.92
  Доли: A=0.5, B=0.1, C=0.4 → 26982.42

Этап 3:
  Доли: A=0.5, B=0.4, C=0.1 → 12227.0
  Доли: A=0.2, B=0.5, C=0.3 → 14524.8
  Доли: A=0.3, B=0.2, C=0.5 → 13975.28
  Доли: A=0.2, B=0.4, C=0.4 → 10845.37
  Доли: A=0.5, B=0.2, C=0.3 → 12374.47
  Доли: A=0.4, B=0.2, C=0.4 → 15120.2
  Доли: A=0.2, B=0.3, C=0.5 → 13382.5
  Доли: A=0.3, B=0.4, C=0.3 → 12513.98
  Доли: A=0.3, B=0.5, C=0.2 → 12388.51
  Доли: A=0.5, B=0.1, C=0.4 → 11402.38
  Доли: A=0.3, B=0.3, C=0.4 → 12108.25
  Доли: A=0.1, B=0.5, C=0.4 → 12535.23
  Доли: A=0.4, B=0.1, C=0.5 → 11171.89
  Доли: A=0.1, B=0.4, C=0.5 → 14421.63
  Доли: A=0.4, B=0.3, C=0.3 → 14950.75
  Доли: A=0.4, B=0.4, C=0.2 → 14242.78
  Доли: A=0.4, B=0.5, C=0.1 → 13926.13
  Доли: A=0.5, B=0.3, C=0.2 → 9764.97
```

Анализ оптимального пути

Поиск последовательности стратегий с максимальной общей стоимостью

[ ]:
```julia
struct Strategy
    period::Int
    allocation::Tuple{Float64, Float64, Float64}
    value::Float64
end

# Преобразование словаря в массив структур
function parse_memo(memo_dict)
    strategies = Strategy[]
    for (key, val) in memo_dict
        push!(strategies, Strategy(
            key[1],
            (key[2], key[3], key[4]),
            val
        ))
    end
```

```
    return strategies
end

# Фильтрация по периоду
function filter_by_period(strategies, period)
    filter(s -> s.period == period, strategies)
end

# Топ-N стратегий
function top_strategies(strategies, n=5)
    sort(strategies, by=s->s.value, rev=true)[1:min(n,␣
  ↪length(strategies))]
end
```

[ ]:
```
top_strategies (generic function with 2 methods)
```

[ ]:
```
function trace_optimal_path(strategies)
    n_periods = maximum(s.period for s in strategies)
    path = Vector{Strategy}(undef, n_periods)

    # Начинаем с последнего периода
    current = top_strategies(filter_by_period(strategies, n_periods),␣
  ↪1)[1]
    path[end] = current

    # Идем в обратном порядке
    for p in (n_periods-1):-1:1
        prev_strategies = filter_by_period(strategies, p)
        current = argmax(s -> s.value + current.value,␣
  ↪prev_strategies)
        path[p] = current
    end

    return path
end
```

[ ]:
```
trace_optimal_path (generic function with 1 method)
```

[ ]:
```
strategies = parse_memo(memo)
optimal_path = trace_optimal_path(strategies)

# Вывод результатов
println("\nОптимальная последовательность стратегий:")
for (i, s) in enumerate(optimal_path)
    println("Период $i: ",
            "A=$(s.allocation[1]), ",
            "B=$(s.allocation[2]), ",
```

```
            "C=$(s.allocation[3]) → ",
            "Капитал: \$", round(s.value, digits=2))
end
```

```
Оптимальная последовательность стратегий:
Период 1: A=0.3, B=0.5, C=0.2 → Капитал: $39890.61
Период 2: A=0.5, B=0.2, C=0.3 → Капитал: $29091.81
Период 3: A=0.4, B=0.2, C=0.4 → Капитал: $15120.2
```

Интерпретация графика и вывода

Ключевые выводы:

1. На ранних этапах преобладают агрессивные стратегии (высокая доля актива B)
2. К 2-му периоду происходит перераспределение в консервативные активы (A и C)
3. Оптимальный путь демонстрирует постепенное снижение риска:

- Период 1: 30% A, 50% B, 20% C
- Период 2: 50% A, 20% B, 30% C
- Период 3: 40% A, 20% B, 40% C