# Risk-Aware Project Scheduling

## SimpleUCT

Karol Walędzik

# Definicja Zagadnienia

# Resource-Constrained Project Scheduling (RCPS)

Table 1
Symbols and definitions

| | | |
|---|---|---|
| $J$ | : | Number of activites. |
| $M_j$ | : | Number of modes activity $j$ can be performed in. |
| $d_{jm}$ | : | Duration of activity $j$ being performed in mode $m$. |
| $R(N)$ | : | Set of renewable (nonrenewable) resources. |
| $\bar{T}$ | : | Upper bound on the project's makespan. |
| $K_r^\rho \geq 0$ | : | Number of units of renewable resource $r$, $r \in R$, available in period $t$, $t = 1, \ldots, \bar{T}$. |
| $K_r^\nu \geq 0$ | : | Total number of units available of nonrenewable resource $r$, $r \in N$. |
| $k_{jmr}^\rho \geq 0$ | : | Number of units of renewable resource $r$, $r \in R$, used by activity $j$ being performed in mode $m$ each period the activity is in process. |
| $k_{jmr}^\nu \geq 0$ | : | Number of units of nonrenewable resource $r$, $r \in N$, consumed by acitivity $j$ being performed in mode $m$. |
| $\mathscr{P}_j$ ($\mathscr{S}_j$) | : | Set of immediate predecessors (successors) of activity $j$. |
| $ES_j$ ($EF_j$) | : | Earliest start time (finish time) of activity $j$, calculated by using minimal activity durations and neglecting resource usage (consumption). |
| $LS_j$ ($LF_j$) | : | Latest start time (finish time) of activity $j$, calculated by using minimal activity durations, neglecting resource usage (consumption) and taking into account the upper bound $\bar{T}$ on the project's duration. |

# Risk-Aware Project Scheduling (RAPS)

- 1 tryb wykonywania działań
- Czas trwania zadań jako zmienna probabilistyczna o znanym rozkładzie
- Funkcja celu: minimalizacja czasu trwania projektu
- Budżet: zamodelowany przez zasób nieodnawialny

# Risk-Aware Project Scheduling (RAPS)

- Ryzyko:
  - rozkład prawdopodobieństwa wystąpienia (w jednostce czasu)
  - definicja efektów wystąpienia ryzyka:
    - czasowa zmiana dostępnej ilości zasobu odnawialnego
      - (w przyszłości: zmiana ilości zasobu nieodnawialnego, zmiana czasu trwania zadania / zadań, ...)
  - ryzyko może zrealizować się tylko raz per projekt (ale może być wiele identycznych ryzyk)
  - efekt nie wpływa na zadania w trakcie realizacji w momencie realizacji ryzyka
    - (w przyszłości potencjalnie warunki wystąpienia: minimalny / maksymalny dotychczasowy czas trwania projektu, wykonanie / niewykonanie dotychczas danego zadania, ..)

# Risk-Aware Project Scheduling (RAPS)

- Działanie:
  - specjalny rodzaj zadania:
    - nie jest niezbędne dla realizacji projektu
    - może (ale nie musi) mieć zerowy czas trwania
    - wymaga zasobów, może mieć poprzedniki (zadania i/lub działania)
  - lista efektów:
    - czasowa zmiana ilości zasobu (odnawialnego lub nie)
      - (w przyszłości potencjalnie: zmiana rozkładu prawdopodobieństwa wystąpienia ryzyka (liniowe przeskalowanie), zmiana wartości nasilenia już występującego ryzyka)
    - efekty nie wpływają na zadania właśnie wykonywane
  - opcjonalnie: opóźnienie wystąpienia efektów
    - (w przyszłości potencjalnie warunki wykonalności: minimalne nasilenie występowania danego ryzyka, minimalny/maksymalny czas od początku projektu, wykonanie poprzedników)

# Risk-Aware Project Scheduling (RAPS)

- Efekt:
  - rozkład prawdopodobieństwa intensywności
  - rozkład prawdopodobieństwa czasu trwania
  - czasowa zmiana ilości dostępnych zasobów:
    - typ zasobu
    - identyfikator zasobu
      - (w przyszłości inne rodzaje efektów)

# Instancje Problemu

# Baza instancji RCPS

- PSPLib
- http://www.om-db.wi.tum.de/psplib/main.html

# Konwersja do instancji RAPS (wersja uproszczona)

- Czas trwania zadań – rozkład Beta(3,5)
  - Moda == szacunek punktowy instancji RCPS
  - Minimum ≈ 75% mody
  - Maksimum ≈ 150% mody
- Dodatkowy zasób nieodnawialny (budżet)
  - Ilość == ½ liczby rodzajów odnawialnych

# Konwersja do instancji RAPS (wersja uproszczona)

- Ryzyka
  - czasowa niedostępność zasobu
    - prawdopodobieństwo wystąpienia == odwrotność sumy czasu trwania wszystkich zadań
    - natężenie == -1 / -2 z równym prawdopodobieństwem
    - czas trwania == rozkład jednostajny z przedziału od 10% do 20% sumy czasu trwania wszystkich zadań

# Konwersja do instancji RAPS (wersja uproszczona)

- Działania naprawcze
  - czasowa wynajęcie dodatkowego zasobu
    - koszt == 1 jednostka budżetu
    - czas trwania == 20% sumy czasu trwania wszystkich zadań
    - efekt:
      - czasowa zmiana dostępnej ilości zasobu odnawialnego
      - natężenie == 1
      - czas trwania == 20% sumy czasu trwania wszystkich zadań

- Konsekwencja: na pewno opłaca się w pierwszej fazie projektu podjąć jak najwięcej działań – trzeba tylko wiedzieć, które

# SIMPLEUCT W RAPS

# UCT jako planer

- Gra jednoosobowa z niedeterminizmem
- Możliwe akcje:
  - rozpoczęcie wykonanie działania
  - rozpoczęcie wykonanie zadania
  - *noop*
- Tylko *noop* powoduje przejście do kolejnej jednostki czasowej
- W przyszłości:
  - możliwe zastosowanie heurystyk w celu ograniczenia liczby rozpatrywanych akcji
    - np.: zawsze rozpoczynaj zadania, jeśli tylko są wolne zasoby

# Faza UCT

```
private int Uct(OngoingProject project)
{
    if (project.ToDo.Count == 0) return project.Time;

    var node = GetOrCreateUctNode(project);
    var decision = GetPossibleDecisions(project, node)
                                        .Greatest(d => d.Priority);
    ExecuteDecision(project, decision, true);

    int result = (node.Visits == 0) ?
                               MonteCarlo(project) : Uct(project);

    decision.AddScore(result);

    return result;
}
```

# Faza Monte-Carlo

```
private int MonteCarlo(OngoingProject project)
{
    if (project.ToDo.Count == 0) return project.Time;


    var actions = project.GetLegalActions();
    var activities = project.GetLegalActivities();

    //If there are any legal activities consider starting them
    if (activities.Count > 0 && _random.NextDouble() <= .85)
    {
        project.StartActivity(activities.Random());
    }
    //Consider starting action
    else if (actions.Count > 0 && _random.NextDouble() < .1)
    {
        project.StartAction(actions.Random());
    }
    //Noop; move ahead in time
    else
    {
        project.Go();
    }
    return MonteCarlo(project);
}
```

# Reprezentacja stanu w drzewie UCT

- Uproszczona w celu umożliwienia jakiejkolwiek nauki
- Składowe:
  - Zbiór identyfikatorów zrealizowanych ryzyk
  - Zbiór identyfikatorów zrealizowanych akcji
  - Zbiór uproszczonych opisów działań w trakcie realizacji
    - Działanie: (identyfikator, pozostały czas trwania)
  - Zbiór uproszczonych opisów zadań w trakcie realizacji
    - Zadanie: (identyfikator, pozostały czas trwania)
  - Zbiór identyfikatorów zrealizowanych zadań
  - Zbiór uproszczonych opisów aktualnie aktywnych efektów
    - Efekt: (identyfikator, przybliżona siła, pozostały czas trwania, probabilistyczne parametry specyficzne dla typu efektu)
  - Ilości dostępnych zasobów odnawialnych i nieodnawialnych

# SIMPLEUCT – PIERWSZE WYNIKI

# Benchmarki

- Greedy-Random
  - Algorytm zachłanny
  - W każdym kroku:
    - jak długo istnieje możliwość rozpoczęcia jakiegokolwiek zadanie, wybieraj losowe zadania i rozpoczynaj je
    - jak długo istnieje możliwość rozpoczęcia jakiegokolwiek działania, wybieraj losowe działania i rozpoczynaj je

# Benchmarki

- Greedy-Longest
  - Algorytm zachłanny
  - W każdym kroku:
    - jak długo istnieje możliwość rozpoczęcia jakiegokolwiek zadanie, rozpoczynaj zadania poczynając od najdłuższych
    - jak długo istnieje możliwość rozpoczęcia jakiegokolwiek działania, wybieraj losowe działania i rozpoczynaj je

# Eksperyment

- Zbiór testowy:
  - pula instancji wygenerowanych na bazie 30-zadaniowych instancji PSPLib
- Liczba powtórzeń każdego eksperymentu:
  - 128
- Uwaga:
  - algorytmy NIE operują na tych samych realizacjach zmiennych losowych
    - wprowadza to dodatkowy element niedeterminizmu, ale nie uniemożliwia porównania

# Rezultaty

| Greedy-L | +/- | Greedy-R | +/- | SimpleUCT | +/- |
|---|---|---|---|---|---|
| 49,555 | 2,459 | 55,063 | 3,610 | 48,789 | 2,416 |
| 47,047 | 2,494 | 50,109 | 3,333 | 46,516 | 2,130 |
| 64,430 | 3,184 | 66,422 | 4,411 | 65,211 | 2,714 |
| 73,641 | 4,270 | 73,578 | 3,902 | 73,133 | 3,546 |
| 68,648 | 3,891 | 70,984 | 4,187 | 66,813 | 3,124 |
| 48,969 | 2,662 | 50,406 | 3,085 | 48,242 | 2,670 |
| 53,289 | 2,959 | 55,211 | 4,144 | 54,727 | 3,019 |
| 55,164 | 2,861 | 55,406 | 2,976 | 54,000 | 2,411 |
| 61,414 | 2,995 | 65,078 | 4,171 | 61,055 | 3,316 |
| 52,484 | 3,066 | 56,172 | 3,650 | 51,805 | 2,848 |
| 65,359 | 5,105 | 63,305 | 4,100 | 68,672 | 2,994 |
| 43,570 | 2,637 | 44,531 | 2,948 | 42,680 | 2,562 |
| 64,398 | 3,215 | 65,234 | 4,843 | 62,867 | 3,640 |
| 85,930 | 2,999 | 87,680 | 3,038 | 84,992 | 3,073 |
| 67,820 | 3,171 | 67,945 | 3,153 | 67,195 | 3,731 |
| 57,711 | 2,418 | 60,336 | 4,053 | 55,789 | 2,258 |
| 49,688 | 2,015 | 50,938 | 3,164 | 48,789 | 2,237 |
| 41,141 | 1,586 | 40,984 | 1,761 | 40,172 | 1,622 |
| 66,352 | 3,250 | 70,305 | 5,050 | 65,313 | 3,094 |
| 71,891 | 3,259 | 73,266 | 3,535 | 71,352 | 3,634 |
| 50,461 | 2,144 | 50,789 | 2,464 | 50,031 | 2,118 |
| 59,734 | 2,879 | 59,945 | 2,684 | 58,414 | 2,445 |
| 50,344 | 1,966 | 50,086 | 2,008 | 49,398 | 1,974 |
| 39,141 | 1,847 | 39,258 | 1,929 | 38,250 | 1,874 |
| 66,383 | 2,624 | 66,305 | 2,376 | 65,320 | 2,531 |

# Proactive Scheduling with UCT

# ProUCT

- Planowanie:
  - na początku działania algorytmu
  - po wystąpieniu *wyzwalacza planowania*
- Dostępny jest *solver* rozwiązujący problem RCPS bez ryzyk
  - w wersji z deterministycznym trwania zadań
  - potencjalnie w przyszłości:
    - w wersji uwzględniającej niedeterministyczny czas trwania zadań

# ProUCT

- Algorytm planowania:
  - UCT w zastosowaniu do niedeterministycznej gry jednoosobowej
    - możliwe ruchy:
      - wykonanie działania
      - *noop*
    - *Noop* powoduje uruchomienie solvera dla problemu uwzględniającego efekty działań i realizację projektu do momentu wystąpienia *wyzwalacza planowania*
    - Reprezentacja stanu jak w SimpleUCT

# ProUCT

- Wyzwalacze etapu planowania:
  - realizacja ryzyka
    - zakończenie efektu ryzyka?
  - zmiana spodziewanego czasu trwania całego projektu (daty zakończenia) o co najmniej 5%
  - czas, który upłynął od ostatniego planowania:
    - ½ oczekiwanego czasu trwania najkrótszego efektu działania
    - stała wartość: 10 jednostek?
  - …

# ProUCT

- Algorytm planowania:
  - efekt działania UCT:
    - lista działań do podjęcia w danej chwili
    - uaktualnione przy okazji symulacji statystyki czasów i kosztów zadań
  - faktyczny harmonogram tworzony przez *solver* na bazie danych z UCT
  - projekt realizowany do czasu kolejnej rundy planowania

# Benchmarki

- Planowanie statyczne:
  - proste wykorzystanie *solvera*
  - wykorzystanie *solvera* z czasami trwania zadań uzyskanymi za pomocą czystych symulacji Monte-Carlo (bez uwzględnienia działań)
  - wykorzystanie *solvera* z uwzględnieniem działań wybranych przez UCT, ale bez zebranych przez niego statystyk dotyczących zadań
- Planowanie reaktywne:
  - planowanie ponawiane w momencie wystąpienia *wyzwalacza planowania*

# Solver

- model deterministyczny
- podejście heurystyczne:
  - w wersji multi-pass z jedną lub wieloma funkcjami heurystycznymi, na bazie Single i Parallel SGS
    - (proste wielokrotne wykonanie algorytmu i wybór najlepszego wyniku)
    - możliwe zastosowanie ważonych kombinacji różnych funkcji heurystycznych
    - możliwy reżim czasowy: zaprzestanie kolejnych obliczeń po przekroczeniu czasu obliczeń
- Implementacja na bazie PSPSolvera: http://sourceforge.net/projects/pspsolver/

# Solver

- W przyszłości potencjalnie:
  - solver na bazie CI (zwykle skuteczniejszy)
  - niedeterministyczne czasy trwania zadań (rozkłady z UCT)