Nr rejestracyjny: 2016/21/D/ST6/02402; Kierownik projektu: dr Łukasz Jeż

Szeregowanie zadań w celu maksymalizacji liczby ukończonych zadań

(*wersja popularnonaukowa)*

Problemy szeregowania są obecne wszędzie: w planowaniu produkcji i/lub dystrybucji, układaniu terminarzy, etc. Dlatego zostały precyzyjnie sformułowane już we wczesnych latach rozwoju informatyki, i do dziś są jednymi z ważniejszych fundamentalnych problemów optymalizacji. Opisane w terminach zadań i maszyn, w większości mają za cel minimalizację określonej funkcji zależnej od czasów ukończeń zadań. Wśród tych funkcji dwie dominujące klasy to maksimum (np. *maksymalny czas ukończenia*) oraz suma (np. *sumaryczny „czas w systemie"*, tj. od zwolnienia do ukończenia). W tego typu problemach wszystkie zadania muszą zostać wykonane, a celem jest uszeregowanie ich w sposób minimalizujący funkcję celu. Przez uszeregowanie rozumiemy przydział zadań do maszyn (gdy jest więcej niż jedna) oraz porządek wykonania w obrębie każdej z maszyn.

Zamierzamy badać inną klasę problemów, gdzie celem jest maksymalizacja zysku, określonego jako liczba ukończonych (na czas) zadań, lub ogólniej suma wag tych zadań. Powodem, dla którego nie wszystkie zadania mogą zostać ukończone, jest fakt, że każde z zadań ma ustalony deadline; pozostałe charakterystyki zadań to czas zwolnienia/przybycia, czas wykonania (rozmiar), oraz waga. Zatem rozważane przez nas problemy są tzw. problemami pakowania. Stosunkowo dobrze znanym (i dość ograniczonym) przykładem jest *problem plecakowy*, tj. maksymalizacji wartości (wag) przedmiotów zapakowanych do plecaka o ograniczonej pojemności: plecak odpowiada jednej maszynie, zaś przedmioty zadaniom, które mają wspólny czas zwolnienia oraz deadline; różnica tych ostatnich równa jest pojemności plecaka. Zamierzamy badać kilka podobnych, ale ogólniejszych problemów: skoro problem plecakowy jest dogłębnie zbadany, rozważamy najprostsze uogólnienia, które nie są.

Jednym z przykładów, gdzie zadania mają jednostkowe rozmiary, jest „szeregowanie pakietów". Zauważmy jednak, że zadania (pakiety) mają dowolne czasy zwolnienia i deadline'y. Okazuje się, że ten prosty problem dobrze modeluje zadanie routerów sieciowych, które muszą decydować, który pakiet powinien zostać w danej chwili przesłany dalej. Oczywiście router pracuje w innym trybie niż „zwykły" algorytm (bądź człowiek) usiłujący spakować plecak. Pomijając drobne szczegóły, zasadniczą różnicą jest to, że router musi działać „w czasie rzeczywistym" lub „na bieżąco". W szczególności nie może dokładnie zbadać wszystkich pakietów, i to nie tylko dlatego, że byłoby to niezwykle nieefektywne: wśród wszystkich pakietów są bowiem i takie, które jeszcze do routera w ogóle nie dotarły! By uchwycić to, co wiadomo w chwili podejmowania decyzji, rozważamy algorytmy on-line. Przetwarzają one sekwencję wejściową kawałkami, gdzie każdy kawałek może być „zwykłą informacją" (np. o jednym zadaniu/pakiecie) lub też żądaniem działania ze strony algorytmu. To działanie polega na podjęciu ostatecznej decyzji, która opierać się może jedynie na dotychczas widzianym ciągu wejścia. W rozważanych przez nas problemach szeregowania ciągu wejściowego jest uporządkowana względem czasu: dane zadań wyjawiane są w chwili ich zwolnienia, zaś algorytmowi wolno podjąć decyzję w chwilach takich zdarzeń jak pojawienie się lub ukończenie zadania (oraz innych, które tu pomijamy.)

Podajemy ścisłe gwarancje dotyczące jakości algorytmów poprzez analizę najgorszego przypadku, typową dla algorytmów. W przypadku problemów on-line oznacza to użycie analizy konkurencyjności, tj. dowód, że zysk algorytmu stanowi co najmniej określony ułamek optymalnego zysku na danej instancji poprzez analizę zamortyzowaną; ten ułamek nazywany jest współczynnikiem konkurencyjności. Szczegóły pomińmy, jednak zauważmy, że analizowany „najgorszy przypadek" obejmuje wszystkie możliwe ciągi przyszłych(!) wydarzeń następujące po rozważanej decyzji algorytmu. Zatem dowodzone gwarancje są niezwykle silne: możemy myśleć, że instancja tworzona jest przez złośliwego adwersarza, który sprawia, że każda podjęta przez algorytm decyzja ostatecznie okazuje się zła.

Myślenie o ciągu wejściowym w ten sposób jest powszechne w przypadku algorytmów on-line. Stanowi wręcz podstawę dowodów *ograniczeń dolnych*. Są one pewnym typem „dowodów niemożliwości" i stanowią uzupełnienie analizy algorytmów: ich celem jest wykazanie, że żaden algorytm nie może zagwarantować współczynnika konkurencyjności lepszego niż owo ograniczenie. Tego typu wyniki pozwalają stwierdzić, czy i na ile znane algorytmy można poprawić. W szczególności, bez odpowiednich ograniczeń dolnych nie możemy stwierdzić optymalności żadnego algorytmu.