Wojciech Legierski*, Paweł Parys**
*Politechnika Śląska, Instytut Automatyki, Zakład Automatyzacji Procesów Przemysłowych, Akademicka 16, 44-100 Gliwice, e-mail: wlegierski@ia.polsl.gliwice.pl
(autor jest stypendystą Fundacji Nauki Polskiej)
**Uniwerysytet Warszawski, Wydział Matematki, Informatyki i Mechaniki, ul. Banacha 2, 02-097 Warszawa, e-mail: pp209216@zodiac.mimuw.edu.pl

PLANOWANIE ZAJĘĆ METODAMI PROGRAMOWANIA Z OGRANICZENIAMI

Streszczenie. Programowanie z ograniczeniami ....

1. Wprowadzenie

Rzeczywiste problemy harmonogramowania posiadają wiele odmian, szczególnych przypadków, ograniczeń. Często nie pasują do standardowych algorytmów, rozwiązujących wprawdzie klasyczne przykłady problemów, ale bardzo trudne jest uwzględnienie odmian poszczególnych przypadków. Istnieje możliwość ciągłej modyfikacji istniejących algorytmów, co często nie jest efektywne. Jednym z takich problemów harmonogramowania o dużej liczbie odmian jest planowanie zajęć dla instytucji edukacyjnych, gdzie głównym problemem nie jest ich NP-zupełność [], ale zmieniające się założenia w różnych szkołach, czy uczelniach. Programowanie z ograniczeniami wydaje się być metodą, która w bardzo dobry sposób nadaje się do tego typu problemów. Jego główną zaletą jest deklaratywność – wyrażanie ograniczeń staje się częścią programu opisującego i rozwiązującego problem. Ta właściwość umożliwia łatwe dostosowanie się do różnych założeń, czy ograniczeń występujących w rzeczywistych problemach.

Programowanie z ograniczeniami ma jednak pewną wadę – jest to brak efektywnego sposobu optymalizacji. Metoda branch-and-bound, która jest naturalną metodą programowania z ograniczeniami, nie jest wystarczająca dla tak złożonych problemów jak planowanie zajęć. Dlatego stara się połączyć metody programowania z ograniczeniami z

metodami poszukiwania lokalnego, w których szukanie lepszego rozwiązania wydaje się bardzie efektywne [].

Problem planowania zajęć nie można rozpatrywać tylko w kontekście znalezienia optymalnego rozwiązania spełniającego postawione ograniczenia. Nawet najlepszy program, który daje bardzo dobre rozwiązanie w kilka sekund, nie będzie używany, jeśli nie da możliwości łatwej interakcji z użytkownikami. Użytkowników, którzy biorą udział w procesie planowania zajęć można podzielić na następujący grupy:

studenci/uczniowie – są zainteresowani łatwym dostępem do swoich planów,

nauczyciele – prócz dostępu do własnych planów zajęć, muszą mieć możliwość wprowadzenia swoich preferencji dotyczących nauczanych zajęć,

osoby układające plan – odpowiedzialne, by wprowadzić dane, narzucić ograniczenia, często w większych instytucjach edukacyjnych jest to kilka osób pracujących na wspólnych zasobach, dlatego zbudowane rozwiązanie dla jednej części planu musi uwzględniać pozostałe,

administrator – określająca części planu, za które mają być odpowiedzialni poszczególne osoby układające plan zajęć, jak i ich prawa do zasobów.

Połączenie interakcji tych wszystkich użytkowników wymaga wprowadzenia Systemu Planowania Zajęć, który pozwoli w pełni wykorzystać solver rozwiązujący poszczególne plany zajęć.

2. Metody w ramach programowania z ograniczeniami dla planowania zajęć

Programowanie z ograniczeniami swoje korzenie ma w programowaniu w logice, do której dołożono deklaratywne wyrażanie ograniczeń. Dlatego jego naturalną własnością jest zasada nawrotu (tzw. backtracking) oraz deklaratywność (postawienie problemu jest równocześnie programem rozwiązującym go). Pierwsze narzędzia, które służyły do programowania z ograniczeniami, to języki zawierające Prolog, między innymi CHIP, Eclips, Sicstus Prolog i są one wciąż popularne. Jednak ze względu na pewne ograniczenia, które wprowadzała składania Prologowo powstały dwie gałęzie w językach programowania z ograniczeniami – pierwsza z nich to biblioteki C/C++ (np. ILOG Solver), a druga to języki współbieżne (np. Mozart/Oz). Wszystkie te języki mają dwie podstawowe własności – propagację ograniczeń oraz dystrybucję połączoną z poszukiwaniem.

2.1. Propagacja ograniczeń

Propagacja ograniczeń jest jedną z najsilniejszych zalet programowania z ograniczeniami, która sprawiła, że ta technika stała się najlepsza metodą dla wielu problemów kombinatorycznych. Zasadą działania propagacji jest usuwanie wartości nie spełniających ograniczeń z domen zmiennych. Domeny zmiennych to skończone zbiory liczb, które wyrażają przestrzeń obliczeniową. Języki do programowania z ograniczeniami mają możliwość wyrażania zmiennych z zakresu domen liczb naturalnych (najczęściej stosowane), przedziałów liczb rzeczywistych, zbiorów i innych. Aby zobrazować propagacje ograniczeń wprowadzony zostanie prosty przykłada:

$$x \in \{1..5\},\ y \in \{1..6\}$$

Gdy na powyższe dwie zmienne wprowadzimy ograniczenie x>y+1, wtedy propagacja ograniczeń zredukuje powyższe domeny do następujących wartości:

$$x \in \{3, 4, 5\},\ y \in \{1, 2, 3\}$$

ponieważ wartości $\{1, 2\}$ z domeny x nie spełniają ograniczenia x>y+1 dla żadnej z wartości z domeny y. Podobnie można rozpatrywać wartości $\{4, 5, 6\}$ z domeny y. Jednak możemy wprowadzić takie ograniczenie jak x+y=6, które nie usuwa żadnych wartości z domen.

Ograniczenia nie są zazwyczaj tak proste ja to przedstawiono, często łączą ze sobą wiele zmiennych, a metody usuwania poszczególnych wartości, zwane algorytmami filtracyjnymi są bardzo złożone. Sama propagacja z ograniczeniami rzadko daje rozwiązanie. Dlatego jest ona zawsze łączona z dystrybucją i poszukiwaniem.

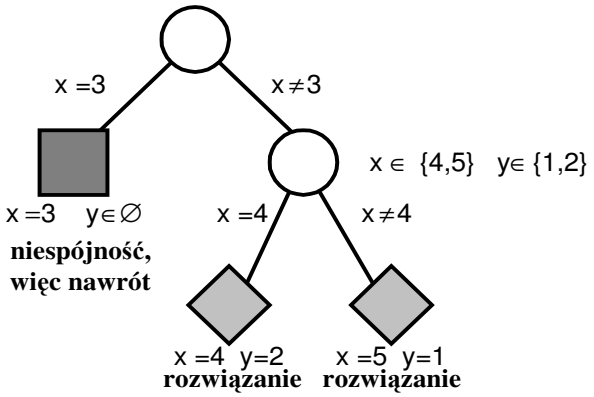
Dystrybucja i poszukiwanie

W większości przypadków propagacja ograniczeń nie prowadzi do rozwiązania (jak to zostało przedstawione w powyższym przykładzie). Dlatego programowaniu z ograniczeniami jest ściśle związane z dystrybucją połączoną z poszukiwaniem. Dystrybucja polega na wprowadzeniu dodatkowego ograniczenia (często jest to przyporządkowanie jednej wartości do zmiennej, a zadaniem dystrybucji jest odpowiednie wybranie zmiennej i wartości). Kiedy to nastąpi sprawdzana jest spójność poprzez propagację ograniczeń i istnieją trzy możliwości:

- znalezione zostanie rozwiązanie (wszystkie zmienne mają po jednej wartości w swojej domenie),

- domeny niektórych zmiennych zostaną zawężone, ale jednoznaczne rozwiązanie nie jest wciąż wyznaczone, więc dystrybucja jest dokonywana na kolejnej zmiennej,
- dodatkowe ograniczenie jest niespójne z pozostałymi ograniczeniami, więc proces nawrotu jest dokonywany, a wybrana wartość z domeny wybranej zmiennej jest usuwana.

Ten proces jest dokonywany iteracyjnie i jest nazywany poszukiwaniem. Poszukiwanie jest odpowiedzialne za zatrzymanie; po znalezieniu pierwszego rozwiązania lub pewnej liczby rozwiązań lub wszystkich rozwiązań. Poszukiwanie tworzy tzw. drzewo poszukiwań, gdzie każdy węzeł jest stanem zmiennej. Na Rys.1. przedstawiono drzewo poszukiwań dla przykładu podane w poprzednim podrozdziale.



Rys. 1. Przykład drzewa poszukiwań

3. Zastosowane metody w ramach programowania z ograniczeniami.

3.1. Ograniczenia miękkie i oznaczeni wartości

Planowanie zajęć jest problemem o dużej liczbie zmiennych oraz wielu ograniczeniach, które w części przypadków są ze sobą sprzeczne, stąd istnieje potrzeba ich porządkowania i selekcji. Wprowadza się pojęcie ograniczeń twardych, które muszą zostać spełnione oraz ograniczeń miękkich, których spełnienie jest zalecane z pewną wagą. W ramach prezentowanej pracy starano się odpowiednio opisać i wyrazić ograniczenia w ramach programowania z ograniczeniami. Zastosowano sposób podejście do ograniczeń miękkich

zgodnie z formalizmem Weighted CSP [], gdzie najlepsze rozwiązanie traktuje się jako te, które posiada minimalną suma niespełnionych ograniczeń. Aby efektywnie wprowadzić ograniczenia miękkie, użyto koncepcji oceny wartości nadawanych dla zmiennych związanych z czasem rozpoczęcia zajęcia [] []. Gdy pewne czasokresy naruszały ograniczenia miękkie, ich wartości były oceniane z wysokim kosztem.

3.2. Alokacja sal

Planowanie zajęć jest również problemem dwuwymiarowym, ponieważ każdemu zajęciu jest przydzielany zarówno czasu rozpoczęcia, jak i sala. Często alokację sal wykonuje się po przyporządkowaniu czasów rozpoczęcia, jednak jest to możliwe tylko w szkołach, które dysponują dużym zapleczem sal. Aby dostosować metody do trudniejszych przypadków, wprowadzono specjalne metody poszukiwania, które w trakcie przyporządkowywania startów, dobierają sale dla poszczególnych zajęć oraz pilnują, by zajęcia nie nakładały się w salach. Łączy się to również ze specjalnym sposobem wprowadzania ograniczeń.

4. Strategia poszukiwań rozwiązań optymalnych

Programowanie z ograniczeniami jest bardzo dobrym paradygmatem do opisywania ograniczeń oraz szukania rozwiązania dopuszczalnego. Niestety nie zachowuje się dobrze przy poszukiwaniu rozwiązania optymalnego. W swej strukturze zakłada przede wszystkim algorytm Branch-and-Bound, który poprzez swoją systematyczność i konieczność przeszukania całego drzewa, jedynie obcinając gorsze niż znalezione do tej pory rozwiązania, często nie nadaje się do dużych problemów. Znacznie lepszą wydajność pod tym względem mają metody poszukiwań lokalnych, które mogą być kierowanie w odpowiednie obszary rozwiązań. Dlatego starano się połączyć ze sobą dwa podejścia: programowanie z ograniczeniami i poszukiwanie lokalne, wykorzystując zalety każdego z nich.

4.1. Poszukiwanie pierwszego rozwiązanie

Pierwsze rozwiązanie jest szukane metodą DFS (Depth First Search), gdzie w pojedynczym kroku stara się znaleźć zarówno czas rozpoczęcia, jak i salę dla wybranego zajęcia. W razie naruszenia ograniczeń przez dane przyporządkowanie, generowany jest nawrót i dany czas rozpoczęcia jest traktowany jako niemożliwy. Wybór zajęć do przyporządkowania oraz wybór wartości czasu rozpoczęcia i sali dla danego zajęcia wydają się być najistotniejszą sprawą na tym etapie. Dlatego zrezygnowano z popularnej metody first-fail (wybiera się zmienną o najmniejszej domenie i przyporządkowuje się jest najmniejszą wartość) i wprowadzono własną, mocno złożoną metodę wyboru. Bazuje ona na znajdowaniu zajęcia, które najtrudniej jest ułożyć (np. zajęcia nauczycieli o dużym obciążniu dydaktycznym są wybierane w pierwszej kolejności) i przyporządkowaniu jemu takiego startu, który odpowiada najmniejszej ocenie wartości, zgodnej z wprowadzonymi ograniczeniami miękkimi.

Problem planowania zajęć jest na tyle złożony, że często znalezienie pierwszego, dopuszczalnego rozwiązania jest bardzo trudne lub wręcz niemożliwe. Dlatego dopuszcza się wynik, w którym nie wszystkie zajęcia są ułożone, przyporządkowując im wysoki koszt, mając nadzieje, że zostaną one ułożone przez metody optymalizacyjne.

4.2. Metody optymalizacyjne

Wiele prac zostało poświęconych na połączenie programowania z ograniczeniami z poszukiwaniem lokalnym. Przykładem takiego zastosowania do problemów układania zajęć jest przedstawiona przez ...[] metoda generowania pierwszego rozwiązania przez programowanie z ograniczeniami, które to rozwiązanie staje się startowe dla poszukiwania tabu. Jednak poszukiwanie tabu jest już zupełnie oddzielone od typowych metod programowania z ograniczeniami jak propagacja ograniczeń i nawroty. Zastosowana metoda jest znacznie bardziej złożona, gdyż łączy ona ściśle te dwie metody, wykorzystując propagację ograniczeń, w trakcie znajdowania kolejnych rozwiązań przez poszukiwanie lokalne.

Pierwszym etapem tej metody jest próba ułożenia zajęć, które nie zostały ułożone przez DFS, oczywiście jeśli takie zajęcia istnieją. Polega to na przegrupowaniu tak zajęć,

które używają tych samych zasobów jak dane nie ułożone zajęcie, tak by powstał przedział czasowy, w którym to zajęcia może zostać ułożone. Kolejnym etapem jest szukanie lepszego rozwiązania. Poszukiwanie lokalne dokonywane jest na sąsiedztwie, gdzie zajęcia używające ten sam zasób (będące z tej samej sali, bądź przyporządkowane do tej samej grupy, bądź prowadzone przez tego samego nauczyciela) są stawione. Dlatego losowo wybierane jest zajęcie, które próbuje się wymienić z innymi zajęciami zależnymi od niego. Jeżeli przez tę wymianę uzyska się lepsze rozwiązanie, jest ono zapamiętywane (podobnie jak w metodach zachłannych).

5. System Planowania Zajęć

6. Wyniki badań

Przedstawione metody zostały przetestowane na rzeczywistych danych pochodzących z wydziału Automatyki Politechniki Śląskiej układając

LITERATURA:

1. Adams J., Balas E., Zawack D., The shifting bottleneck procedure for job shop scheduling. *Management science*, 34(3), 1988, pp.391-401.
2. Boufflet J.P., Negre S., Three Methods Used to Solve an Examination Timetable Problem, Practice and Theory of Automated Timetabling, Springer-Verlag, Lecture Notes in Computer Science, vol. 1153, 1996, pp. 327-344.
3. Burke E., Ross P. (Eds.), Practice and Theory of Automated Timetabling, First International Conference, Edinburgh 1995, Selected Papers, Springer-Verlag, Lecture Notes in Computer Science, vol. 1153.
4. Burke E., Carter M. (Eds.), Practice and Theory of Automated Timetabling II, Second International Conference, PATAT'97, Toronto 1997, Selected Papers, Springer-Verlag, Lecture Notes in Computer Science, vol. 1408.

5. Burke E.K., Newall J.P., Weare R.F., A Memetic Algorithm for University Exam Timetabling, Practice and Theory of Automated Timetabling, Springer-Verlag, Lecture Notes in Computer Science, vol. 1153, 1996, pp. 241-250.
6. Carlier J., Pinson E., An algorithm for solving the job-shop problem. *Management Science*, 35(2), 1989,pp.164-176.
7. Cooper T.M., Kingston J.H., The Complexity of Timetable Construction Problems, Springer-Verlag, Lecture Notes in Computer Science, vol. 1153, 1996, pp. 283-295.
8. Elmohamed S., Coddington P., Fox G., A Comparison of Annealing Techniques for Academic Course Scheduling, Practice and Theory of Automated Timetabling II, Springer-Verlag, Lecture Notes in Computer Science, vol. 1408, 1998, pp. 92-112.
9. Guéret Ch., Jussien N., Boizumault P., Prins Ch., Building University Modular Timetabling Using Constraint Logic Programming, Practice and Theory of Automated Timetabling, Springer-Verlag, Lecture Notes in Computer Science, vol. 1153, 1996, pp. 130-145.
10. Henz M., Würtz J., Using Oz for College Timetabling, Practice and Theory of Automated Timetabling, Springer-Verlag, Lecture Notes in Computer Science, vol. 1153, 1996, pp. 162-177.
11. Kaneko K., Yoshikawa M., Nakakuki Y., Improving a Heuristic Repair Method for Large-Scale School Timetabling Problems, Springer-Verlag Principles and Practice of Constraint Programming – CP'99, 2000, pp. 275-288.
12. Lajos G., Complete University Modular Timetabling Using Constraint Logic, Practice and Theory of Automated Timetabling, Springer-Verlag, Lecture Notes in Computer Science, vol. 1153, 1996, pp. 146-161.
13. Legierski W., Using Mozart for Timetabling Problems, Proceedings of the CPDC'01 Workshop on Constraint Programming for Decision and Control, Gliwce, 2001
14. Mozart Consortium. The Mozart Programming System. http://www.mozart-oz.org, 1999.
15. Muth J. F., Thompson G. L., *Industrial Scheduling*. Prentice Hall, Englewood Cliffs, NJ, USA, 1963.
16. Rich D.C., A Smart Genetic Algorithm for University Timetabling, Practice and Theory of Automated Timetabling, Springer-Verlag, Lecture Notes in Computer Science, vol. 1153, 1996, pp. 181-197.
17. Tsang E., Foundation of Constraint Satisfaction, Academic Press, 1993.
18. Würtz J., Oz Scheduler: A workbench for scheduling problems, *Eighth International Conference on Tools with Artificial Intelligence*, Toulouse, France, 1996. IEEE, pp. 149-156.

Recenzent:

CONSTRAINT PROGRAMMING METHODS FOR TIMETABLING

Abstract:

[kliknij i wpisz rozszerzone streszczenie referatu w j. angielskim - ok. 2/3 str.]