Универзитет Св. Кирил и Методиј - Скопје

Факултет за електротехника и информациски технологии

Институт за компјутерска техника и информатика

Семинарска работа по Експертни системи

Hybrid Heuristics for Solving the Constraints Modeled

решавање на проблем со распоред за средните училишта

Ментор

Доц. д-р Трајковиќ Владимир

Изработил

Перица Николовски 979/2007 ИНФО

Скопје, декември 2011

Содржина

Апстракт

Оваа семинарска работа го опишува проблемот со решавање на распоредот во средните училишта. Хибридно Хеуристичките алгоритми се применуваат за решавање на ограничени моделарни проблеми. Constraint Satisfaction се користи за модел на проблемот од стана на класи на Constraint Solve Engine (CSE) и Constraint Programming Library (CPL) кои предходно се развиени. Главниот хибриден алгоритам е изграден врз основа на Simulated Annealing, Greedy Search, Guided Search и меморија од Tabu Search.

Вовед

Constraint Programming е методологија за проблеми која му овозможува на корисникот да ги опише податоците и ограничувањата на проблемот без експлицитно решавање на декларативна фаза. Constraint Satisfaction Problems (CSP) едноставно може да се дефинира како збир на променливи и сет од ограничувања меѓу вредностите и променливите. Типичен метод за решавање на CSP моделот е изградба на решение со помош на откажан пристап(backtracking approach) во кое делумно се доделуваат променливи и постепено се проширува, но и истовремено се чува фазибилити на досегашното решение. Ограничувањата се чуваат во текот на решавање на целиот процес. Многу оптимизациски проблеми од практично и теоретско значење се стреми за наоѓање на најдобрата конфигурација на вредности како сет од променливи. Таквите проблеми каде решението е моделирано со дискретни променливи припаѓаат на комбинаторна оптимизација (CO). Проблемите од комбинаторната оптимизација се состојат од сет на променливи, нивните домени, ограничувања меѓу променливите и главна функција која треба да се минимизира(максимизира). Школскиот распоред е типичен пример на CO проблем. Средношколскиот распоред генерално вклучува и временско и просторно планирање. Тоа е сложена пресметка и обично бара покомплексна задача. Исто така е проблем кој е тежок за оптимизација кој бара хеуристички пристап за решавање. Constraint Satisfaction не е единствениот метод на избор за моделирање пробреми со распоред, поради нивната висока сложеност. Само кончниот распоред ги задоволува сите наметнати ограничувања. Во текот на генералниот распоред, повеќето ограничувања ќе бидат незадоволени во одреден момент. Системот, каде што стенпенот на constraint satisfaction се мери и споредува, така CSP може да успешно да се користи во распоредувањето. Кога во constraint satisfaction е вклучено мерењето, тогаш системот станува Constraint Optimization Problem (COP). Многу општ пристап кон решавање на проблемите со комбинаторна оптимизација како распоред е да се генерира почетна состојба, неоптимално решение и потоа се применува хеуристичкиот модел за да се подобри решението. Овој метод е малку согласен со стандардот за откажан метод (backtracking method) во ограниченото програмирање. Во потешките проблеми, градењето на решение од откажан (backtracking) пристап во кои делумно задачата на променливите е постепено да ја проширува, е практично невозможен. На пример, кога бројот на кралици во прблемот N-Queens надминува одреден број (на пример 50), GAC-CBJ алгоритмот не даде резултат во разумен рок, времетраењето на пребарувањето бро расте надвор од предвиденото време.

Целта на моето истражување е да се постигне успешна симбиоза на Constraint Programming и Хеуристичките алгоритми. Исто така, хеуристичките алгоритми користат хибридна комбинација за различен хеуристички пристапи. Идејата е да се создаде хибрид хеуристички алгоритам кој ќе ги користи предностите на многу предходно предложените алгоритми. Мојот пристап започнува со почетно неоптимално решение проследено со хеуристичка поправка за да се постигнат точни конечни решенија Идеите од алгоритмите како Simulated Annealing,Tabu Search, Guided Search се приклучени за да се постигне побрзо и попрецизно решение. Хеуристичките алгоритми бараат constraint satisfaction систем кој може да го измери нивото на constraint satisfaction и да се обезбеди хеуристичко за подобро решение на проблемот. Повеќето од хеуристичките одлуки се донесуваат во процесот на поправање на првото неоптимално решение. Постојата механизми за да се избегне deadlock и да се постигне ковергенција преку поинаква имплементација. Функциите кои генерираат соодветно решение врз основа на предходното се клучни за успешно, брзо и точно решение. Тие користат база на знаење за проблемот и повторно ги употребуваат информациите за одредени ограничувања за да се генерираат подобри решенија од тековното. Точниот редослед на решенијата врз основа на предходните недостатоци се користи во моментот кога ќе се случи deadlock. Стохастичките компоненти во процесот на решавање на deadlock се покажуваат како добри и водат кон конечно решение.

Constraint модел за проблемот со распоред во средните училишта

Распоредот опфаќа широка област на проблеми со временска и просторна распределба на ресурсите. Постојат повеќе фамилии на проблеми на распоред тие можат да се разликуваат во зависност од степенот на слобода, во позиционирање на понудените ресурси и побарувачката на ресурсни интервали во времето, чист распоред на проблеми, чиста алокација на ресурси и конечно заедничката планирање и алокација на ресурсните проблеми. Средношколскиот распоред е сложен проблем. Во случајот на средношколскиот распоред, моделот вклучува средства за временска и просторна распределба на ресурсите. Треба да се спроведат приоритети меѓу ограничувањата, да се обезбедат методи за задоволување првенствено на поважните правила проследени со помалку значајните. Главниот интерес на порграмите за ограничување лежи во активното користење на ограничувањата за да се намалат компјутерските потреби за да се реши еден проблем во исто време и за постигање добра декларативна проблем формулација. Ограничувањата се користат не само за тестирање на валидноста на решенијата, се користат и на конструктивен начин да заклучат нови ограничувања и за откривање на нови недоследности. Овој процес е наречен Ограничување на размножување (constraint propagation). Овој проблем домен спаѓа во категорија на Ограничување оптимизација проблеми (COP), каде constraint satisfaction треба да се оценува. Примена на алгоритми како Simulated Annealing (SA) овој алгоритам има тенденција да најде решение кое ќе ја намали цената на функцијата. Затоа, моделот за ограничување е способен за производство на нумерички мерења на нивното задоволство. Распоред генерација е формализиран како проблем Constraint Relaxation Problem – CRP од Yoshikawa et. al. Тие се фокусираат за користење на мин-конфликт хеуристички модел тие генерираат првични решенија за истовремено решавање на училишни и универзитетски распоред проблеми. Прилично добар квалитет даваат првичните решенија генерирани од Arc-Consistency алгоритмот, нивниот предлог се потпира на хеуристички билјард потези на операторот за поправка на сегашните и завршните задачи на распоредот за училиштето / факултетот. Минималниот хеуристички конфликт (MCH) е да се испита секоја променлива и да му ја додели вредноста со минимален број на ограничувања. Комбинација од минимални-конфликти и lookforward хеуристичко се користат во локално пребарување методи за ефикасно решавање на општ Универзитетски распоред проблеми. Сите овие алгоритми се претходно развиени во Constraint Programming Library (CPL). Софтверската библиотека се состои од :

а) збир на часови – генерички ограничувања и топови за моделирање на друг вид на проблеми и б) механизам за селекција на оптимален алгоритам за даден проблем. CPL содржи различни алгоритми, вклучувајќи Simulated Annealing, Tabu Search, Arc Consistency итн. Овој пристап е потребен бидејќи Constraint Solving Engine (CSE) е развиен за да се овозможи решавање на проблеми со различна природа. Моторот е модуларен, и им дозволува на специфичните хеуристички модели за одредени проблеми да бидат имплементирани со прескокнување на некои функции. Секој чекор за решавање на проблемот може да се реализира или да биде имплементиран со нови додадени модули кои веќе се содржани во основниот објектно-ориентиран систем. CSE се базира на концептот на променливи и нивните домени. Домените се граничат со постојните ограничувања во моментот на нивното создавање, што го прави пребарувачкиот просто помал. Подоцна во текот на процесот на предлагање на нови решенија, го оценува степенот на подобрувања и се мери степенот на подобрувања кон најдоброто финално решение. Во овој модел, цената за решението потекнува од нивото на задоволства на секое ограничување. Секое ограничување спроведува функција за пресметка на износот на незадоволство. Постојат дополнителни множители на незадоволство, да се зголеми влијанието на одредени ограничување во однос на другите во вкупните трошоци. Абрамсон во неговиот модел ги дели вкупните трошоци на три дела професор , клас и училница тие го сочинуваат резултатот од судирите во текот на решението. Исто така на одредена компонента може да и се додаде повисоко значење во вкупниот број. Проблемот е моделиран и следува дека постојат: G*D*N (G-групи, D-денови, N-број на часови во текот на денот) променливи кои ја дефинираат задачата на часови на групи и на училници.Променливите се групирани во блокови од D * N променливи. Секој блок одговара на распоред за една група.

T={t0,t1,....,t t-1} е збир на сите часови во распоредот.

T=|T|=G*D*N е бројот на часови во распоредот.

Часовите се групирани во блокови на N лекции кои се во истиот ден. Часовите се подредени во растечки редослет во однос на тоа како се зголемуваат деновите.

Нотација

Следна фаза, пред да биде напишен програмскиот код за ограничувањата, е создаден математички модел. За таа цел потребна е точна нотација, дел од која е дефинирана на следниот начин:

$\Pi = \{\pi 0, \pi 1, \ldots, \pi P\text{-}1\}$ сет од професори

$B = \{\beta 0, \beta 1, \ldots, \beta B\text{-}1\}$ сет на предмети

$\Psi = \{\psi 0, \psi 1, \ldots, \psi Y\text{-}1\}$ сет од школски години

$\Delta = \{\delta 0{=}0, \delta 1{=}1, \ldots, \delta D{=}D\text{-}1\}$ сет од работни денови

$\Gamma = \{\gamma 0, \gamma 1, \ldots, \gamma G\text{-}1\}$ сет на групи

$X = \{\chi 0, \chi 1, \ldots, \chi C\text{-}1\}$ сет од училници

$I = \{Imin, Imin{+}1, \ldots, N\}$ сет од бројот на часови во еден ден

$Imin$ минимален број на часови во еден ден

N максимален број на часови во еден ден

База на знаење

Алгоритмите работат користејќи постоечки податоци. Податоците претходно треба да бидат внесени во програмата, и подготвени во одреден формат. Наведените низи се задолжителни за подготовка на овој проблем модел. Овој модел се состои од осум претходно внесени податочни структури. Некои од нив се:

1) Број на неделни часови x по предмет β по група γ е дефиниран со следниот сет на подредени торки: ВГ ={(βi, γj, x) | i = 0, …, B-1; j = 0, …, G-1}.

Функцијата x = Xβγ(β, γ): B×Г → Z+ враќа број на часови по предмет β за група γ.

Функцијата x = wγ(γ): Г → Z+ враќа број од неделни часови за група γ.

2) Професорите π кои ги предаваат предметите β за група γ се дефинирани со следниот сет од подредени торки: ПВГ ={(πk, βi, γl) = | k = 0, …, P-1; i = 0, …, B-1; l = 0, …, G-1}.

Функцијата π = Pβγ(β, γ): B×Г → П ги враќа професорите π за предмет β за група γ.

3) Предметите β можат да се предаваат во училницата χ е дефинирано со следниот сет од подредени парови:

BX = {(βi, χj) | i = 0,…, B-1; j = 0, …, C-1}.

Функцијата Tβχ(β, χ): B×X → {0, 1} враќа 1 ако предметот β може да се предаде во училница χ, инаку враќа 0.

4) Максималниот број m од групите кои можат да се распоредат во училницата χ е дефинирана со следниот сет од подредени парови:

X M = {(χi, m) | i = 0, …, C-1}.

Функцијата Mχ(χ): X → Z+ го враќа максималниот број од групи кои можат да се распоредат во училница χ во било кое време.

Ограничувања

Проблемот е моделиран и е представен преку 16 ограничувања. Еве некои од нив:

$$1.\quad \sum_{k=i}^{i+D\cdot N-1}(b_\tau(\tau_k)=\beta_j) \;=\; X_{\beta\gamma}(\beta_j,\ \gamma_i) \text{ for } i = 0,\ D\cdot N,$$

2·D·N, …, (G-1)·D·N and j = 1, 2, …, B, каде сумата е број од неделни часови за предметот βj во група γi. Бројот од неделни часови по предмет во групата се дефинирани со сет во внесените податоци. Кога ова ограничување беше кодирано со дадени алатки на Constraint Solving Engine, во имплементација, се користеше класата CsetCover. Оваа класа за ограничување како и сите други во CSL, наследува од основната класа за ограничување. Нејзината задача е да го провери бројот на појавувања на одредена вредност за дадена променлива координирана во неза од променливи. Кога бројот на појавувања се соодветни, ограничувањата се ,,покриени“. Во сетот да бидат опфатени како еднаков збир на предмети B. Секој сет на вредности треба да бидат препокриени точно Xβγ (β, γ) пати. Од ова може да се заклучи дека секој сет може да покриени и да бидат различни за различни групи, посебни истанци од класата CSetCover е потребно да се создадат за секоја група.

$$2.\quad \sum_{k=i}^{i+D-1}(l_\nu(\nu_k)-f_\nu(\nu_k)+1)= w_\gamma(\gamma_m) \text{ for } i = 0,\ D,$$

2·D, …, (G-1)·D, m = i/D,каде сумата е број од неделни часови за групата γm. Бројот од неделни часови по група се дефинирани со сет во внесените податоци.

$$3.\ T_{\beta\chi}(b_\tau(\tau_i),\ r_\tau(\tau_i)) = 1 \text{ for } i = 0,\ 1,\ 2,\ \ldots,\ G\cdot D\cdot N\text{-}1$$

Предметите се секогаш предавани во соодветни училници дефинирани од старана на влезните податоци.

$$4.\ \sum_{i=1}^{G}\begin{cases}0 & if \quad r_\tau(\tau_{(i-1)\cdot D\cdot N+l})\neq\chi_j\\ 1 & if \quad r_\tau(\tau_{(i-1)\cdot D\cdot N+l})=\chi_j\end{cases} \leq M_\chi(\chi_j) \text{ for } j = 1,\ 2,\ \ldots,\ C \text{ and } l = 1,\ 2,\ 3,\ \ldots,\ D\cdot N$$

Во секое време l, училницата χj може да има најмногу Mχ(χj) групите се дефинирани со сет X M во влезот на внесените податоци. Како што беше споменато во предходните објавени материјали класите на ограничувањата имплементирани во Constraint Solving Library (CSL) беа создадени за да бидат универзални и да важат за различни проблеми. Затоа во ова ограничување на класата CSetCover (генеричко Ограничување) се користи повторно. Сет од вредности да бидат препокриени беше X M. Во ограничувачки-променлива мрежа практично моделот на проблемот имаше D·N од ова ограничување. Имаше еден примерок за секој временски слот во работната недела. Меѓутоа секој примерок е всушност истиот пример на генерички CsetCover Constraint, со оглед на множество на вредностите да бидат покриени секогаш беше X M. Во секое различно копирање училниците координатите на променливите за различни групи за таа часови беше земена во предвид

5. $b_\tau(\tau_i) \in \mathbf{A} \wedge i \neq l_v(\nu_k) \wedge (b_\tau(\tau_i) \neq b_\tau(\tau_{i-1}) \vee i = f_v(\nu_k)) \Rightarrow b_\tau(\tau_{i+1}) = b_\tau(\tau_i)$ for $i = 0, 1, \ldots, G \cdot D \cdot N\text{-}1$, and $k = i$ div N.

Ако предметот bт(ті) можат да бидат предавани во блок од 2 часови, и ті не е последен час во денот k, и предметот bт(ті) е различен од предходниот предмет bт(ті-1) или ті е прв предмет во денот, тогаш следниот час треба да биде на истиот предмет bт(ті+1).

6. bт(ті) ≠ bт(тj) за k = 0, 1, …, G-1 и l = 0, 1, …, D-2, каде i = lv(vn) и j = fv(vn+1), n = k·D+l. n е индекс од променливата vn
тогаш е дефиниран индекс i за последниот час ті во ден l за група γk.

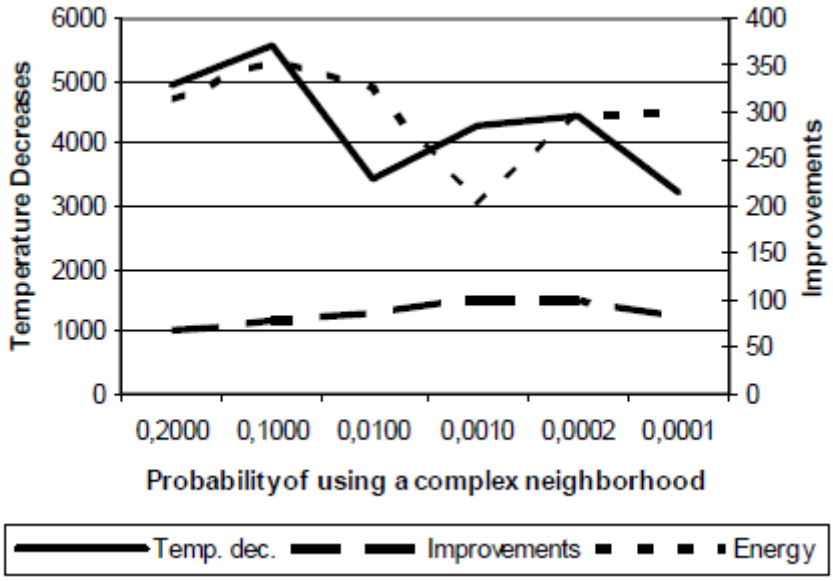
Предметот bт(ті) за последниот час во денот освен за последниот ден во неделата (секогаш Петок) и предметот bт(тj) за првиот час во предходниот ден за многу групи може да биде различен.

Имплементација на Hybrid Simulated Annenaling

Simulated Annenaling е опширно истражуван и покажа задоволителни резултати во решавање на проблемите со комбинаторна оптимизација и временски и просторни распореди. Во софтверската библиотека се споредува во комбинирана верзија на алгоритмот SA. Имаше елементи на меморијата од Tabu Search како и комплексни слични функции за локално пребарување слични како во Guided Search алгоритмот. Детално објаснување на алгоритмот следува во следниот псевдо код:

```
initialSol ← ConstructAsCorrectInitSolutAsPossible();
SolutionNeighborhood.SetSolution( initialSol );
{listOfVarsToChange, currentEnergy} ← CalculateEnergy(initialSol);
temp = InitialTemperature;
do
   do
         SolutionNeighborhood.GenerateCandidateSol
                              ( currentSol, listOfVarsToChange);
         FindAffectedConstraints();
         {listOfVarsToChange,newEnergy}          ←
                    CalcEnergy(SolNeighborhood.Candidate);

         if  ( Metropoliten(newEnergy - currentEnergy,temp))
                    SolutionNeighborhood.AcceptCandidate();
         else
                    SolutionNeighborhood.RefuseCandidate();
         endif

   while ( !stopSearch and (trials < saMaxTrials )
          and successfulTrials < saMaxSuccessfulTrials
          and smallestEnergy > lowerEnergyBound );

   temp = TempSchedule.GetNewTemperature();

   while ( !stopSearch
       and numberOfTempDecreases < MaxTempDecreases
       and SolutionCount < MaxNumberOfSolutionsToFind
       and smallestEnergy > lowerEnergyBound
       and consecutiveNoSuccess < MaxConsecNoSuccess )

   return SolutionNeighborhood.ReturnBestFoundSolution();
```

Алгоритмот првично конструира решение на начин колку што може повеќе да ги задоволи ограничувањата, и цената и енергијата да се сведат на минимум. Особено во распоредот за средното училиште за секоја група во достапните временски слотови за соодветен број на часови по предмет е исполнет. Исто така за секој предмет е доделен професор итн. Сите часови се внесуваат постојано во временските слотови, се додека не се постигне соодвет број на часови по предмет по група. Останатата задача за алгоритмот е да ги движи предметите(група, предмет,професор) во други термини во текот на неделата, така што ќе бидат задоволени останатите ограничувања (да не биде повторена истата тема два пати во истиот ден и др). Во меѓувреме, е генериран објект од класата CNeighborhood.



Во овој објект се сместува досегашното решение, и кога од него ќе се побара решение тој нуди нов предлог решение за тековната состојба. Ова е место каде локалното пребарување се врши во простор. Потоа neighborhood функцијата генерира нови решенија, алгоритмот повикува функција FindAffectedConstraints (). Таа открива ограничувања чие задоволство е променето во текот на претходното ужасно решение. Оваа информација фукнцијата CalculateEnergy () го пресметува само учеството на засегнатите ограничувања во рамките на целокупната енергија, што е спротивно на повторно пресметаната цена. Дополнително фунцијата CalculateEnergy () генерира нова листа listOfVarsToChange. Листата на состојби со променливите врши промени во следните решени пертурбации така што е изведено подобро решение. Metropoliten () функцијата ја спроведува Metropoliten веројатноста дистрибутивна функција. Со оглед на разликата на енергии на претходното и вистинското решение како и параметрот температура, тој одлучува дали да прифати или да одбие ново решение.

$$P_T(\Delta E(X)) = \begin{cases} 1, & \Delta E(X) <= 0 \\ \exp(-\frac{\Delta E(X)}{T}), & else \end{cases}$$

Циклусот продолжуваат со извршување на задачата се доека еден од дадените услови е исполнет. Извршувањето запира кога ќе биде постигнат дефинираниот максимален број на повторување на температурата, кога ќе бидат постигнат максималниот број на прифатливи решенија или се очекуваната минимална енергија се оценува. По завршување на циклусот, се повикува функцијата myTempSchedule.GetNewTemperature (). Во зависност од избраната распоред температура добиена е новата вредност за параметрот температура. По ново добиената температура следи нов циклус на повторувања. Ако условите за завршување на целиоту услов се исполнети, алгоритмот ги враќа најдобрите најдени решенија. Завршените услови се состојат од постигнување на предефиниран број на повторувања на температурата, постигнување на минималната енергија или постигнување на предефиниран број на повторувања, каде што metropolis функцијата не го прифаќа секое предлог решение. Презентираните спроведување на SA алгоритмот, освен компоненти на оригиналниот алгоритам, вклучува функционалности од други metaheuristic алгоритми. Вклучувајќи ја и листата на претходно погодени променливи вклучувајќи меморија, елементи од Tabu Search. Листата помага во водењето на пребарување и избегнување на циклуси.

TABLE I.
DEPENDENCES OF SOLVING FROM THE PROBABILITY OF USING A COMPLEX NEIGHBORHOOD

Probabil.	Min. Cost	Number Improve.	Nr.Temp Decreases	Duration (msec)
1/5	315	68	2926	335857
1/10	355	80	5558	306890
1/100	326	86	3434	137983
1/1000	201	99	4273	133300
1/5000	297	99	4440	238560
1/10000	300	83	3214	259314

Заклучок

Секој договор со опис на хеуристички идеи додека ние ги искористивме за создавање на совтвер за изградба на училишен распоред. Софтверот е базиран на Constraint Solving Engine (CSE) и Constraint Programming Library (CPL) кои се претходно развиени. Различните симулации и тестирања за решавање на процесот подразбира потребни корекции на моделот и на алгоритмите. При изградба на решение универзалните алгоритми и функции така имплементирани успеваат да најдат решение за проблемот. Одредени прилагодувања и интеграции на малите хеуристички методи драстично го забрзуваат решавањето на проблемот. Додавањата водени во потрага по просторно решение се покажа дека значително се подобрува решението наспроти едноставното предлог решение. Сепак, стохастичките компоненти се корисни за да се избегне deadlock и замка во локалната оптима. Со ограничување на вклучување на соодветна хеуристика за дадениот тип на проблем на модуларни компоненти и се одржува универзалноста на библиотеката. Во исто време претставата е значително зголемена. Наоѓањето на баланс меѓу универзално оптимизирање на функции и проблемот зависи од хеуристиката, се подобрува за уште повеќе или слични задачи.

Референци

-Solving the High School Scheduling Problem Modelled with Constraints Satisfaction using Hybrid Heuristic Algorithms

Ivan Chorbev, Suzana Loskovska, Ivica Dimitrovski and Dragan Mihajlov

Faculty of Electrical Engineering and Information Technologies - Republic of Macedonia