Universitat de Girona
Escola Politècnica Superior

Projecte fi de grau

Estudi: Grau en Enginyeria Informàtica

Títol: Ampliació i millora del “Girona Optimization System”
TFG d’enginyeria informàtica

Document: Resum

Alumne: David Pérez Sánchez

Tutor: Mateu Villaret Auselle i Jordi Coll Caballero
Departament: IMAE
Àrea: Llenguatges i sistemes informàtics

Convocatòria (mes/any): Setembre 2021

1 Introducció

Trobar la solució a un problema és la finalitat i objectiu de quasi tot procés. L'afany per descobrir aquesta solució és el que ha incentivat la creació de nombroses tècniques i paradigmes de programació. Dins aquests paradigmes es troba la programació amb restriccions o *Constraint Programming*, sovint emmarcada dins l'àmbit de la programació declarativa. Aquest paradigma redueix la cerca de la solució a un problema combinatori a identificar l'assignació que satisfà les restriccions imposades sobre un conjunt de variables o, en altres paraules, trobar la possible solució al problema combinatori que resulti vàlida, entre totes les candidates.

1.1 Motivació

La motivació pel desenvolupament d'aquest projecte radica en la necessitat de codificar les restriccions que defineixen els problemes d'una manera convenient. A la Universitat de Girona, concretament al grup de recerca de Lògica i Intel·ligència Artificial (LIA) del departament d'Informàtica, Matemàtica Aplicada i Estadística (IMAE), disposen d'un llenguatge declaratiu enfocat en el paradigma de la programació per restriccions anomenat BUP, que forma part del *Girona Optimization System* (GOS). El sistema en qüestió és el resultat d'un anterior TFG que pretén suplir aquesta necessitat. Tot i ser funcional, tal com afirma el seu autor, es tracta d'una primera iteració i es troba en una fase inicial de desenvolupament. Així doncs, el propòsit és desenvolupar i millorar el sistema GOS per tal que pugui utilitzar-se en escenaris reals, és a dir, que sigui apte tant per l'àmbit de recerca com l'educatiu i, fins i tot, el professional.

1.2 Objectius

L'objectiu principal d'aquest projecte és crear una eina que permeti modelar *Constraint Satisfaction Problems* (CSPs) d'una manera convenient, partint del sistema GOS inicial. Per assolir-lo es plantegen diverses correccions i ampliacions a implementar en el sistema. En quant a les correccions, es volen identificar i resoldre alguns errors de rendiment, gestió de memòria i correctesa d'ús del llenguatge C++. Pel que fa a les ampliacions, es centren en implementar algunes funcionalitats enfocades a la facilitat d'ús de l'usuari, com la definició de predicats i la possibilitat d'incloure alters fitxers, en completar el *Girona Optimization System* perquè suporti la optimització.

2 Conceptes previs

2.1 Compilador

Un compilador, en essència, consisteix en un programa capaç de traduir programes escrits en un llenguatge determinat a un altre llenguatge. Típicament, es tradueix a codi màquina o codi objecte per generar fitxers binaris o executables. Tot i això, el concepte de compilador pot ser molt més general, passant a considerar-se un traductor de codi.

El procés de compilació es separa en diverses etapes bàsiques. En primer lloc hi ha l'anàlisi lèxic, que identifica les cadenes de caràcters o *tokens* del codi d'entrada. Llavors hi ha l'anàlisi sintàctic, que genera típicament un arbre de *parsing* que representa l'estructura sintàctica del programa. A continuació es realitza l'anàlisi semàntic mitjançant l'exploració d'aquest arbre i s'encarrega de dotar de significat les diferents parts del codi font. En totes les etapes anteriors es realitza el control d'errors pertinent. Finalment, en l'etapa de generació de codi, s'escriu el codi objectiu o el fitxer binari corresponent.

2.2 *Constraint Programming*

La programació amb restriccions o *Constraint Programming* (CP) és un paradigma de programació que s'utilitza sobretot per la cerca de solucions a problemes combinatoris. Es tracta d'un tipus de programació declarativa i, per tant, consisteix en descriure les característiques del problema a resoldre d'una manera expressiva però sense donar detalls de com trobar la solució.

Un *constraint* consisteix en una relació lògica sobre un conjunt de variables. Cada variable té un domini o conjunt dels possibles valors que pot prendre. Les restriccions limiten el domini de les variables i determinen quins valors poden tenir a la vegada. El propòsit de la programació amb restriccions és resoldre problemes descrits amb *constraints*. Una modalitat de problemes que contempla aquest paradigma són els problemes de satisfacció de restriccions o *Constraint Satisfaction Problems* (CSPs). La solució a un CSP és l'assignació de variables amb els valors del seu domini que satisfà totes les restriccions.

Existeixen diverses tècniques per modelar CSP i resoldre'ls. En aquest projecte s'exploren les metodologies de codificació a SAT i MaxSAT. La primera consisteix en reduir un CSP a un problema de satisfactibilitat booleana (SAT), format per un conjunt finit de variables booleanes (literals) i un conjunt de restriccions que es codifiquen en fórmules proposicionals. Les fórmules acostumen a expressar-se en *Conjunctive Normal Form* (CNF), és a dir, una conjunció de clàusules. Una clàusula és una disjunció de literals positius o negatius. La solució a un problema SAT consisteix en trobar una assignació de valors a les

variables de la fórmula que satisfaci totes les clàusules.

La codificació a MaxSAT és una variant de SAT on l'objectiu és trobar una assignació que satisfaci el màxim nombre de clàusules. Existeixen diverses extensions de problemes MaxSAT, però la contemplada en aquest projecte és la *Weigthed Partial* MaxSAT. Aquesta consisteix en assignar un pes a algunes clàusules. El pes representa el cost que comporta no satisfer-la. Per tant, hi ha clàusules sense pes (*hard clauses*) i amb pes (*soft clauses*).

3 Correccions, millores i ampliacions

3.1 Escalabilitat

Aquestes correccions pretenien corregir els errors d'estructura i d'utilització del llenguatge C++ perquè el projecte GOS fos més escalable i flexible.

El primer problema identificat i corregit va ser la incorrecta utilització dels espais de noms o *namespaces*. Es van eliminar les sentències `using namespace` i es van substituir tots els identificadors pel *fully qualified name*[1]. També es va englobar tot el codi relatiu a GOS en un espai de noms adequat. Aquests canvis evitarien potencials problemes d'ambigüitat d'identificadors en futures ampliacions del sistema o en cas que es volgués incloure GOS en un altre projecte.

3.2 Rendiment

El sistema GOS inicialment patia dos tipus de problemes de rendiment: pèrdues de memòria o *memory leaks* i errors de programació que provocaven que en certes instàncies el sistema trigués una quantitat de temps no raonable en executar-se. Els *memory leaks* es van solucionar substituint tots els apuntadors o *raw pointers* per *shared pointers*. El problema del temps de càlcul va requerir un estudi i anàlisi en profunditat del codi. Es van utilitzar *profilers* i eines avançades per obtenir estadístiques de l'execució per tractar d'identificar el problema però, finalment, es va trobar mitjançant l'anàlisi de la traça d'execució de manera manual.

3.3 Predicats

Amb aquesta ampliació del llenguatge BUP l'usuari podria definir i utilitzar predicats personalitzats en el model, concretament en un nou bloc de definició de predicats. Addicionalment podria incloure fitxers externs on hi hagués definicions de predicats i també inclusions d'altres fitxers. Els predicats podrien rebre

[1] Identificador precedit pel nom del *namespace* on es troba definit. Per exemple std::string.

qualsevol tipus de dades suportat per BUP. El cos dels predicats estaria format per un conjunt de restriccions o *constraints* i podria contenir definicions de variables locals auxiliars. Aquests predicats es podrien cridar en el mateix lloc que es podria expressar un *constraint*. Per tant, haurien de suportar recursivitat i referències creuades. La crida a un predicat avaluaria els *constraints* del cos del predicat i generaria, segons els paràmetres indicats, les clàusules corresponents.

3.4 *Soft constraints*

El *Girona Optimization System* en un principi estava pensat per ser un sistema d'optimització però, en la seva primera versió, només permetia trobar solucions satisfactibles, no pas òptimes. Per aquesta raó es va implementar la possibilitat de modelar els problemes a MaxSAT, mitjançant un sistema d'anotació de pesos a clàusules. Així doncs, GOS suportaria *soft clauses* i seria capaç de codificar-la en format DIMACS.

3.5 *Solvers*

Abans de la realització d'aquest projecte, GOS només permetia resoldre fórmules amb MiniSat. Arrel de la millora dels *soft constraints* i per tal d'oferir més flexibilitat a l'usuari, es va desacoblar el procés de *solving* del sistema GOS. Aquest passaria a realitzar-se en un *solver* executant-se en un procés extern a GOS. Addicionalment, es mantindria la possibilitat de resoldre la fórmula amb l'API dels *solvers* i s'afegiria una opció per utilitzar un *solver* personalitzat. A la figura 1 es mostra el funcionament general de l'aplicació.

3.6 *Debug*

Per facilitar la comprensió de la fórmula generada i oferir a l'usuari un mètode de *debug* convenient, es va implementar la possibilitat de fer anotacions en forma de comentaris en la codificació DIMACS de la fórmula i de mostra-hi el valors de les variables i paràmetres declarats al model. Aquesta funcionalitat seria de gran utilitat també en l'àmbit docent.

3.7 Altres correccions i millores

Durant el desenvolupament del projecte es van detectar alguns problemes que es van corregir i es van implementar algunes funcionalitats secundàries. Alguns dels punts a destacar són la implementació de missatges *warning*, la correció de la prioritat dels operadors lògics o la incorporació dels operadors *and* i *or* aplicats a llistes.

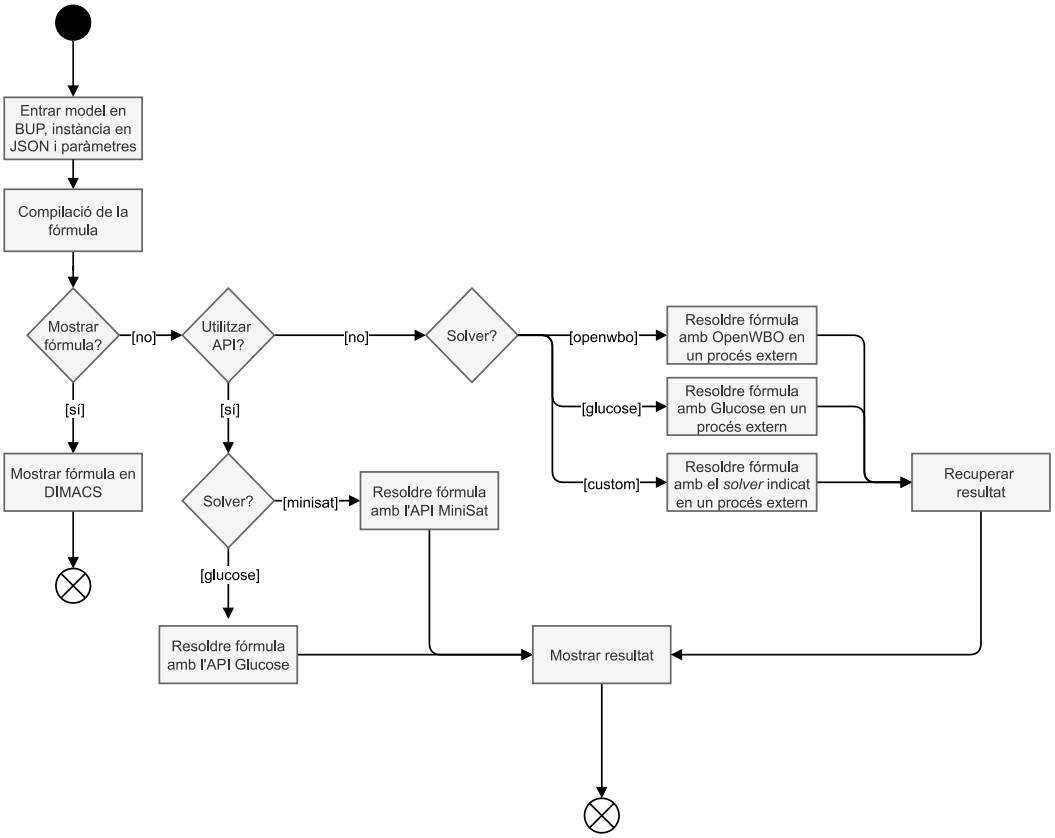

Figura 1: Diagrama de flux simplificat del funcionament de GOS

4 Conclusions

Aquest projecte deixava com a resultat un GOS completament funcional que incorporava característiques noves i millores en el rendiment i estructura. Això el convertia en un sistema més flexible, eficient i aplicable tant en entorns reals com en l'àmbit de la docència i la recerca. Cal destacar que GOS es requeria per ser utilitzat com a eina docent en assignatures com PDA i, gràcies a les noves millores, això seria possible. Algunes de les característiques noves més rellevants eren la possibilitat de definir predicats personalitzats, inclusió de fitxers externs, obtenció d'informació de *debug*, flexibilitat alhora d'escollir el *solver* o possibilitat de modelar problemes MaxSAT.

Tot i això, el sistema encara tenia marge de millora. Per això es proposen algunes característiques que s'hi podrien implementar com a treball futur. Addicionalment, es va deixar el projecte millor estructurat i ben comentat en aquesta memòria per facilitar-ne la col·laboració.