ELSEVIER

Decision Support Systems 16 (1996) 67–83

Decision Support Systems

Constraint logic programming for qualitative and quantitative constraint satisfaction problems

Ho Geun Lee [a,*], Ronald M. Lee [b], Gang Yu [c]

[a] *Information and System Management Department, Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong*

[b] *Erasmus University Research Institute for Decision and Information Systems (EURIDIS), Erasmus University Rotterdam, P.O. Box 1738, 3000 DR Rotterdam, The Netherlands*

[c] *Department of Management Science and Information Systems, Graduate School of Business, The University of Texas at Austin, Austin, TX 78712-1175, USA*

Abstract

AI and OR approaches have complementary strengths: AI in domain-specific knowledge representation and OR in efficient mathematical computation. Constraint Logic Programming (CLP), which combines these complementary strengths of the AI and OR approach, is introduced as a new tool to formalize a special class of constraint satisfaction problems that include both qualitative and quantitative constraints. The CLP approach is contrasted with the Mixed Integer Programming (MIP) method from a model-theoretic view. Three relative advantages of CLP over MIP are analyzed: (1) representational economies for domain-specific heuristics, (2) partial solutions, and (3) ease of model revision. A case example of constraint satisfaction problems is implemented by MIP and CLP for comparison of the two approaches. The results exhibit those relative advantages of CLP with computational efficiency comparable to MIP.

Keywords: Constraint logic programming; Logic modelling; Constraint solving

1. Introduction

Since most decisions are made under restrictions or constraints, Constraint Satisfaction Problems (CSP) have been widely studied in both OR and AI. In the OR approach, constraints are quantitative, and a special algorithm like simplex method optimizes single or multiple objective functions subject to numeric constraints. In contrast, AI research has mostly focused on symbolic (qualitative) constraints and employed inference-based approaches to deal with domain-specific heuristics.

Many important decision problems in organizations include not only qualitative (symbolic) constraints but also quantitative ones. Research in management science has consistently emphasized the importance of both qualitative and quantitative constraints for managerial decision supports. Those studies range from decision theories such as multiple criteria decision making [17] to applications such as strategic

* Corresponding author. E-mail: hlee@uxmail.ust.hk

planning [3], sales mix [18] and real estate investment [20]. This study addresses the class of CSP that contain both qualitative and quantitative constraints, which we call Qualitative/Quantitative Constraint Satisfaction Problems (QQCSP). The solver of the QQCSP should be capable of qualitative inference as well as mathematical computation to handle both types of constraints.

It is widely recognized that there is a remarkable parallel between logical inference and 0-1 integer programming [6,12,7]. Consider, for instance, propositions X_i joined by logical connectives and 0-1 integer variables d_i, where $d_i = 1$ means X_i is true and $d_i = 0$ means X_i is false. It can be easily seen that the constraint $X_1 \vee X_2$ is equivalent to $d_1 + d_2 \geq 1$, and that $X_1 \rightarrow X_2$ is equivalent to $d_1 - d_2 \leq 0$. Therefore, the QQCSP can be modeled using Mixed Integer Programming (MIP) in which symbolic constraints are represented by 0-1 integer variables and quantitative constraints are formed using continuous variables. In this OR-oriented approach, the QQCSP is built quantitatively through arithmetic operators and relations, and a mathematical algorithm such as branch-and-bound method may be employed as a solver.

In this study Constraint Logic Programming (CLP) is proposed as a way to structure and implement the QQCSP and is compared with MIP. CLP is an extension of logic programming where unification is replaced by constraint satisfaction [4]. In CLP qualitative constraints are expressed as symbolic relations and are logically inferred to draw conclusions. On the other hand, a mathematical solver like the simplex method deals with other quantitative constraints involving continuous variables. The motivation of this integration stems from the observation that AI and OR have complementary advantages: AI approaches are well-suited for representating symbolic and domain-specific knowledge, while OR techniques offer fast and efficient mathematical computation.

The remainder of the paper is organized as follows. In the next section the CLP approach is introduced and contrasted with MIP from a model-theoretic point of view. This is followed by the analysis of advantages of CLP over MIP. A case example of the QQCSP is then presented and implemented by MIP and CLP to illustrate those advantages. The computational efficiency of CLP is also discussed. Finally, the results are summarized and future research issues are addressed.

2. Constraint logic programming

2.1. The CLP scheme and CLP($\Re$)

The CLP scheme represents a framework for the formal foundation of a programming language class that combines logic programming and constraint solving. Like logic programming, CLP uses resolution, but syntactic unification is replaced by constraint satisfaction. There are several CLP languages such as CLP($\Re$) [13], Prolog III [5], and Constraint Handling In Prolog (CHIP) [8]. In this study we have used CLP($\Re$) in order to compare CLP with MIP.

CLP($\Re$) was an experimental implementation of the CLP paradigm. The development was motivated by the deficiency of Prolog unification. The conventional unification algorithm in logic programming uses a semantics defined within the context of the Herbrand Universe. Herbrand Universe is a semantic domain where individual names denote themselves. In this universe only those items which are syntactically equivalent can be unified. Consider, for instance, the successor function ($s(X) = X + 1$) and the factorial function ($fact(X)$) [16]. Even though the terms $s(s(0))$, $s(fact(1))$, and $s(fact(s(0)))$ are semantically equivalent, unification will fail in logic programming because they are syntactically different. Developers of CLP($\Re$) chose the real number domain $\Re$ to highlight the existence of a framework of formal semantics that is not restricted to the unification on the Herbrand Universe. As a result the semantics of arithmetic processing in CLP($\Re$) is directly inherited from the real domain $\Re$. This is significantly different from Prolog where arithmetic constraints fall outside the scope of semantics.

2.2. Model-theoretic view for QQCSP

In general, constraints are relations and functions that should be maintained throughout the decision process. Therefore, the QQCSP can be conceptualized as an empirical relational system W that consists of the objects under investigation and relations and functions over those objects [19].

$$W = [\mathbf{O}, \mathbf{R}, \mathbf{F}].$$

Here $\mathbf{O}$ is a set of objects, $\mathbf{R}$ a set of relations and $\mathbf{F}$ a set of functions. The model-theoretic view divides the problem formalization process of the QQCSP into representational and computational level.

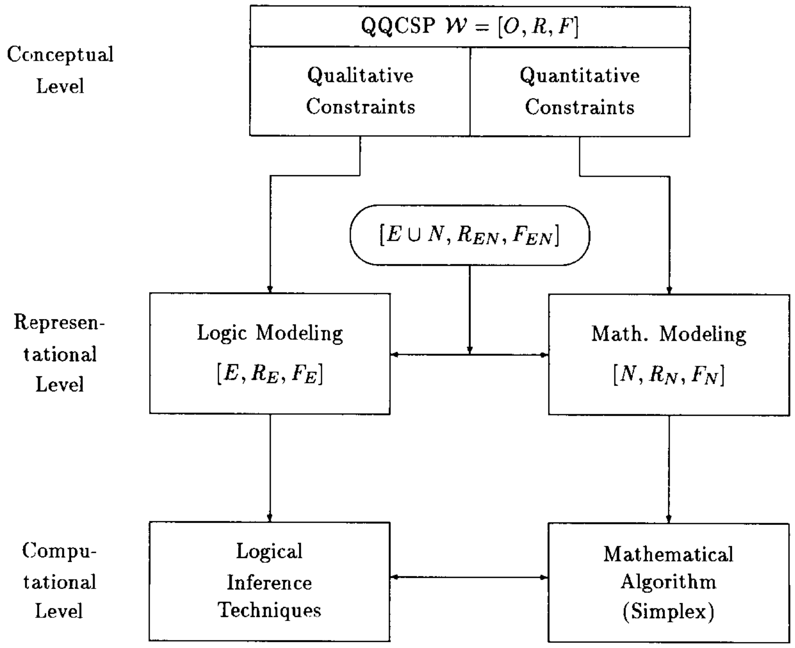
2.2.1. Representational level

This level formally represents the empirical relational system W. At the representational level the CLP approach contains two formal models: a *logical model* and a *mathematical model* (see Fig. 1). The constraint satisfaction problem is first divided into a qualitative portion and a quantitative one for the purpose of formal representation.

$$W = [E, R_E, F_E] \cup [N, R_N, F_N] \cup [E \cup N, R_{EN}, F_{EN}],$$

where E: a set of symbolic objects, R_E: a set of relations among E, F_E: a set of functions on E, N: a set of real numbers, R_N: a set of numeric relations among N, F_N: a set of numeric functions on N, R_{EN}: a set of relations among E and N, F_{EN}: a set of functions on E and N

The qualitative portion $[E, R_E, F_E]$ is represented by first-order predicate logic in CLP. The quantitative portion is expressed through mathematical modelling which can be derived from the empirical relational system W by mapping the quantitative part of W into the numeric relational system $[N, R_N, F_N]$. The motivation behind this mapping is to use an efficient mathematical computation to



Conceptual Level
QQCSP $W = [O, R, F]$
Qualitative Constraints
Quantitative Constraints
$[E \cup N, R_{EN}, F_{EN}]$
Represen-tational Level
Logic Modeling $[E, R_E, F_E]$
Math. Modeling $[N, R_N, F_N]$
Compu-tational Level
Logical Inference Techniques
Mathematical Algorithm (Simplex)

Fig. 1. Model-theoretic view of CPL for QQCSP.

reason about the problem world W [19]. Finally there exists an interdependency between symbolic constraints and numeric constraints which makes it impossible to solve those constraints separately. $[E \cup N,\ R_{EN},\ F_{EN}]$ represents this interdependency relation.

In this way the symbolic constraints are expressed in a natural way without any mapping to the numeric relational systems while the quantitative constraints are formed in terms of numeric variables, relations and functions. In addition, $[E \cup N,\ R_{EN},\ F_{EN}]$ links the logical model with the mathematical model. This is different from MIP, in which all objects, relations and functions are mapped to the numeric relational system so that numeric calculation is performed for reasoning in the whole problem world W.

2.2.2. *Computational level*

The computational level can be viewed as an implementation of the representational level with detailed computational methods for carrying out the formal procedures. Since there exist two representation schemes, the CLP approach requires not only logical inference techniques but also mathematical computation. The CLP($\Re$) interpreter consists of an *inference engine* and a *constraint solver* [14]. The inference engine employs deduction techniques to draw logical conclusions. The constraint solver uses a modified simplex method to handle quantitative constraints and find an optimal solution.

The key feature of CLP at the computational level is that the inference technique and the mathematical computation are interlinked so that one technique can interweave with the other during the constraint solving process. That is, if the syntactic unification fails or numeric constraints are infeasible the interpreter starts backtracking. This unique feature provides a systematic mechanism for the QQCSP where symbolic and quantitative constraints cannot be solved separately and should be satisfied simultaneously. This interaction supports the interdependency $[E \cup N,\ R_{EN},\ F_{EN}]$ at the computational level. In contrast, MIP relies only on the mathematical computation at this level. The mathematical solver employed by MIP may be general algorithms like the branch-and-bound method or specialized algorithms designed for mixed 0-1 integer programming, such as the partitioning method.

3. Advantages of CLP over MIP

The integration of inference techniques and numeric computation provides CLP with relative advantages over MIP, which relies entirely on mathematical representation and computation. These advantages include (1) representational economies for domain-specific knowledge, (2) partial solutions, and (3) ease of model revision.

3.1. *Representational economies*

In MIP domain-specific heuristic rules should be first expressed by *propositional logic* so that it can be later mapped into 0-1 logical variables. Propositional logic is the simplest type of logic where a statement is either true or false. By contrast, CLP describes domain-specific heuristic rules with *predicate logic*. Predicate logic decomposes an elementary proposition into individual variables and relations among those variables, thus having stronger expressive power than propositional logic. Consider, for instance, the following simple reasoning example.

Given that:	*All men are mortal.*
	Socrates is a man.
Conclude that:	*Socrates is mortal.*

In predicate logic, the preceding example is symbolized as follows:

Given that: $\forall X\, man(X) \rightarrow mortal(X).$
$man\ (Socrates).$
Conclude that: $mortal(Socrates).$

Propositional logic is insufficient for this in that it cannot reconcile the general statement (about all men) to the specific statement (about Socrates). The propositional logic requires that sentences for all individuals be stated explicitly. In general the MIP formulation for this kind of reasoning rules introduces a large number of 0-1 integer variables and constraints, thus enlarging its matrix size. CLP can replace the large matrix of MIP with a small number of predicate rules by decomposing the statement into predicates and individual variables and by employing universal quantifiers on those individual variables.

3.2. Partial solutions

CLP($\Re$) can generate partial solutions. If numeric variables are not bounded the CLP($\Re$) interpreter returns as an answer numeric constraints that describe the relations between unknown variables. In capital budgeting, for example, Net Present Value (NPV) is an amount at the present that is equivalent to an investment's cash flows for a particular interest rate **I**. The predicate **npv(V,L,I)**, where **V** is NPV, **L** a list of cash flows $[(CI_0, CO_0), (CI_1, CO_1),..., (CI_n, CO_n)]$ and **I** an interest rate, can be written as follows in CLP($\Re$):

```
npv(0, [], I):-!.
npv((CI - CO)+V/(1+I),[(CI,CO)|T],I):- npv(V,T,I).
```

The second line of the program uses a recursive rule to compute the NPV and the first line provides a boundary condition for the recursion. Suppose a firm is considering an investment plan whose initial investment is \$300,000, and has a target NPV of \$100,000 with 10% of a discount rate for the next four years. Then the following query will return the numeric constraints imposed on annual income **CI** and annual operating costs **CO** (dollar amounts scaled down by one thousandth):

```
query:   ?- npv(V,[(0,300),(CI,CO),(CI,CO),(CI,CO),(CI,CO)],0.1),
            V>=100.
answer:  CI - CO>=126.188.
```

The partial solution is extremely useful to decision makers since it enables them to consult various aspects of constraint satisfaction problem environments. When the input is incomplete, CLP solves the constraint satisfaction problems using only available information and yields a simplified model or answer in the form of partial solutions. This contrasts to working with a general MIP solver, in which there are only two answers: an optimal solution or no solution (infeasible solution).

3.3. Ease of model revision

Model revision entails both addition and delete of certain objects and their associated relations. The model revision is unavoidable when problem environments change. For example, the deletion of a production line or the addition of a new production process can occur frequently in manufacturing environments. CLP is superior to MIP for making revisions to a model whose underlying assumptions often change.

The 0-1 variables of MIP correspond to propositions of domain-specific knowledge. The change in this knowledge can easily undermine the mapping between propositions and integer variables since the 0-1 variables represent the knowledge in terms of truth or false of elementary statements. The addition and deletion of objects or the updating of relations among those objects requires redefinition of the 0-1 variables and their constraints. Thus when domain-specific heuristics change or evolve modellers need to rebuild the model. In contrast, CLP can accommodate changes in domain-specific knowledge with relative ease. The predicate has been widely recognized as a useful tool for changing objects and their relations in model management [9,1]. The incremental programming style of CLP, which is inherited from logic programming, also facilitates the incremental revision of domain-specific knowledge.

4. Production planning example

We consider a manufacturing company that produces several metal products. The production environment includes multiple processes, multiple machines, and multiple worker groups that have different skills and experiences. There are two objectives for the production planning: assignment and product mix.

4.1. Assignment

This refers to the assignment of appropriate labor and processes to each product. The product attributes (raw material and design) determine the necessary processes, and labor is assigned to those processes depending on its job classification. For instance, the welding process for aluminum is more difficult than for carbon steel. The surface must be cleaned and a special care should be taken because aluminum is such a good conductor of heat. Therefore, the product made of aluminum should be assigned to a welding process requiring higher skill, like the special-welding or the refined-welding process. Once a process is chosen, a worker must be allocated. Certain processes can be done only by qualified workers that have enough skill and experience.

There is another type of constraint called *preference* in the assignment. It is assumed that some products are considered more important than others from a marketing perspective. This is because products sold in different regions have varying levels of competition. The ordering of product importance may be total or partial. The heuristic rules for the preference specify that, whenever possible, more skilled labor should be assigned to more important products.

4.2. Product mix

Product mix is a Linear Programming (LP) problem. The objective is to determine the production quantity of products in order to maximize total profit. There are three types of constraints: minimum production requirements, labor resource constraints and facility resource constraints. All of these constraints are quantitative and are formed by numeric relations and operators.

It should be pointed out that the formulation of the LP problem is influenced by the assignment. This corresponds to the interdependency $[E \cup N, R_{EN}, F_{EN}]$ in the model-theoretic view. Labor groups work for hourly wages, and skilled workers receive higher wages than unskilled ones. Because of different experiences and skills, however, it takes less time for skilled workers to do a certain process than for unskilled ones. Therefore, the coefficients of the objective function and constraints in the LP formulation vary depending on the assignment. On the other hand, the assignment is also highly dependent on the LP computation. The assignment that satisfies the qualitative constraints may result in an infeasible or non-optimal product mix. The overall goal is to find the assignment and the optimal product mix which satisfy both qualitative and quantitative constraints.

5. MIP formulation

The MIP formulation is presented by first defining all subscripts, sets, variables and coefficients, followed by the objective function and the constraints. Note that artificial variables z_{ijk} are introduced to transform the initial nonlinear formulation into a linear one.

5.1. Sets, variables and coefficients

5.1.1. Subscripts

$i = 1,2,...,m$ labor
$j = 1,2,...n$ products
$k = 1,2,...l$ processes
$e = 1,2,...,h$ machines
$v = 1,2,...,p$ raw materials
$w = 1,2,...,q$ designs

5.1.2. Description of sets

$\mathscr{M} = \{(j, v)$: a raw material of a product j is $v\}$
$\mathscr{S} = \{(j, w)$: a design of a product j is $w\}$
$\mathscr{R} = \{(v, K_v)$: a material v requires one of processes in $K_v\}$ where $K_v \subseteq \{1,2,...,l\}$
$\mathscr{D} = \{(w, K_w)$: a design w requires one of processes in $K_w\}$ where $K_w \subseteq \{1,2,...,l\}$
$\mathscr{T} = \{(i, K_i)$: laborer i can do processes in $K_i\}$ where $K_i \subseteq \{1, 2,...,l\}$
$\mathscr{F} = \{(k, I_k)$: a process k can be done by one of labors in $I_k\}$ where $I_k \subseteq \{1,2,...,m\}$
$\mathscr{E} = \{(e, K_e)$: a machine e is used for processes in $K_e\}$ where $K_e \subseteq \{1,2,...,l\}$
$\mathscr{L} = \{(i \succ \bar{i})$: laborer i is more skilled than laborer $\bar{i}\}$
$\mathscr{P} = \{(j \succ \bar{j})$: a product j is more important than a product $\bar{j}\}$
$\mathscr{G} = \{(k \succ \bar{k})$: a process k is preferred to a process $\bar{k}\}$

5.1.3. Variables

$$x_{ijk} = \begin{cases} 1 & \text{if laborer } i \text{ is assigned to product } j \text{ for process } k \\ 0 & \text{otherwise} \end{cases}$$

for all i, j, $k \in K_i$ where $(i, K_i) \in \mathscr{T}$
$y_j =$ production quantity of product j for all j
z_{ijk} =artificial variables replacing $x_{ijk} y_j$ for all i, j, $k \in K_i$ where $(i, K_i) \in \mathscr{T}$

5.1.4. Coefficients

$a_{ik} =$ number of hours required for laborer i to do process k, for all i, j, $k \in K_i$ where $(i, K_i) \in \mathscr{T}$
$l_i =$ total available time for laborer i
$u_i =$ unit labor cost (wage/hour) of laborer i
$f_e =$ total available facility capacity (hours) for machine e
$g_j =$ minimum production requirement for product j
$p_j =$ price of product j
$mc_j =$ unit material cost of product j
$sc_j =$ unit sales cost of product j
$f_c =$ fixed cost (both production and sales)

5.2. Objective function

$$\max \sum_{j=1}^{n} \left(p_j - mc_j - sc_j - \sum_{i} \sum_{k \in K_i} a_{ik} u_i x_{ijk} \right) y_j - fc \text{ where } (i,K_i) \in \mathscr{T}.$$

Here, $\Sigma_i \Sigma_{k \in K_i} a_{ik} u_i x_{ijk}$ is the unit labor cost of product j. A linear objective function is obtained by replacing $x_{ijk} y_j$ by z_{ijk} together with associated constraints:

$$\max \sum_{j=1}^{n} \left((p_j - mc_j - sc_j) y_j - \sum_{i} \sum_{k \in K_i} a_{ik} u_i z_{ijk} \right) - fc \text{ where } (i,K_i) \in \mathscr{T}.$$

5.3. Constraints

5.3.1. Possible assignment

$$\sum_{i \in I_k} \sum_{k \in K_v} x_{ijk} = 1 \text{ where } (j,v) \in \mathscr{M}, (v,K_v) \in \mathscr{R} \text{ and } (k,I_k) \in \mathscr{F}.$$

$$\sum_{i \in I_k} \sum_{k \in K_w} x_{ijk} = 1 \text{ where } (j,w) \in \mathscr{S}, (w,K_w) \in \mathscr{D} \text{ and } (k,I_k) \in \mathscr{F}.$$

5.3.2. Preference constraints

The heuristic rules for the preference constraints can be summarized as:

If laborer i is assigned to a product j for a process k,
and a product $\bar{j}$ requires the same process k,
and the product j is more important than the product $\bar{j}$,
and laborer i is more skilled than laborer $\bar{i}$
then assign laborer $\bar{i}$ to the product $\bar{j}$ for the process k

The following constraint equivalently says that if laborer i is assigned to a product j for the process k ($x_{ijk} = 1$), then laborer i or a more skilled laborer than i ($\hat{i}$) cannot be assigned to the product $\bar{j}$ for the process k ($\Sigma_{\hat{i} \in \mathscr{I}} x_{\hat{i}\bar{j}k} = 0$).

$$x_{ijk} + \sum_{\hat{i} \in \mathscr{I}} x_{\hat{i}\bar{j}k} \leq 1,$$

where $(j, \bar{j}) \in \mathscr{P}$, $\mathscr{I} = \{(\hat{i} \succ i) \in \mathscr{L} \text{ or } \hat{i} = i\}$ and $i, \hat{i} \in \Gamma_v, \Gamma_w$

$$\Gamma_v = \{ (j,v) \in \mathscr{M}, (\bar{j},\bar{v}) \in \mathscr{M}, (v,K_v) \in \mathscr{R}, (\bar{v},K_{\bar{v}}) \in \mathscr{R},$$
$$k \in K_v, k \in K_{\bar{v}}, i \in I_k, \hat{i} \in I_k, (k,I_k) \in \mathscr{F} \}$$

$$\Gamma_w = \{ (j,w) \in \mathscr{S}, (\bar{j},\bar{w}) \in \mathscr{S}, (w,K_w) \in \mathscr{D}, (\bar{w},K_{\bar{w}}) \in \mathscr{D},$$
$$k \in K_w, k \in K_{\bar{w}}, i \in I_k, \hat{i} \in I_k, (k,I_k) \in \mathscr{F} \}.$$

5.3.3. *Labor and facility constraints*

$$\sum_{j=1}^{n} \sum_{k \in K_i} a_{ik} z_{ijk} \le l_i \text{ for all } i \text{ where } (i, K_i) \in \mathcal{T}.$$

$$\sum_{k \in K_e} \sum_{j=1}^{n} \sum_{i \in I_k} a_{ik} z_{ijk} \le f_e \text{ for all } e \text{ where } (k, I_k) \in \mathcal{T} \text{ and} (e, K_e) \in \mathcal{E}.$$

5.3.4. *Minimum production constraints*

$$y_j \ge g_j \text{ for all } j.$$

5.3.5. *Constraints for replacing $x_{ijk} y_j$ with z_{ijk}*

$$z_{ijk} - M_j x_{ijk} \le 0,$$

$$-y_j + z_{ijk} \le 0,$$

$$y_j - z_{ijk} + M_j x_{ijk} \le M_j$$

for all i, j, $k \in K_i$ where $(i, K_i) \in \mathcal{T}$ and M_j is an upper bound of y_j.

The MIP formulation process can be considered as building a matrix that contains coefficients of the objective function and constraints. The MIP formulation requires a large number of variables and constraints to represent the production planning problem at a propositional level. Modellers may build a large matrix by using an automatic matrix generator such as GAMS [2]. However, the benefits of using such an automatic tool are restricted to certain constraints whose expressions share common algebraic structures, such as labor or facility constraints. In general the constraints for domain-specific heuristic rules lack these shared common structures. In preference constraints, for example, modellers should define the complex set like $\mathcal{S}$, Γ_v and Γ_w for each pair of $(j, \hat{j})$. This propositional level description requires a large amount of cognitive and coding efforts of modellers.

6. Implementation by CLP($\Re$)

The CLP($\Re$) formulation consists of qualitative modelling for the assignment and quantitative modelling for the product mix. The implementation uses databases assertions called **assign** as an interface between the two. Once a candidate assignment is determined it is stored in the database and used for the LP formulation in quantitative modelling.

6.1. *Qualitative model: Assignment*

The following rules describe the heuristic rules for the possible **assignment**. The predicate **assignment** specifies the assignment order of products. The predicate **possible** first obtains the attribute values of the product and determines the list of assignable processes based on those attribute values. The predicate **find** takes the list of assignable processes, and returns one candidate process together with an assignable laborer to that process. For this purpose the **find** employs a conventional Prolog predicate **member**. Note that the **find** also obtains a laborer's required work time from the database **labor_hour** to provide useful information for the LP formulation process. After a valid assignment is returned, the predicate **possible** stores the output to the database **assign** by assertion. The update of the asserted database is performed by the predicate **update**.

```
assignment:-        possible(p1),possible(p2), ...,possible(pn).
possible(P):-       product(P,M,D,_,_,_,_),
                    material(M,MList), find(MList,MP,ML,MT),
                    design(D,DList), find(DList,DP,DL,DT),
                    update(P,MP,ML,MT,DP,DL,DT).
find(List,P,L,T):-  member(P,List), pro_labor(P,PList),
                    member(L,PList), labor_hour(L,P,T).
```

The next step is to test whether the current assignment satisfies the preference constraints. The predicate **prefer_check** assures that the preference constraints are imposed on all product pairs of **P1** and **P2** for which **P1** is more important than **P2**. The predicate **check** takes the current assignment values for products **P1** and **P2** and tests those values with rules described by **check_labor**. The predicate **check_labor** specifies that if **P1** and **P2** require the same process, more skilled labor should be assigned to **P1** than to **P2**.

```
prefer_check:-   forall(important(P1,P2), check(P1,P2)).
check(P1,P2):-   assign(P1,MP1,ML1,_,DP1,DL1,_),
                 assign(P2,MP2,ML2,_,DP2,DL2,_),
                 check_labor(MP1,ML1,MP2,ML2),
                 check_labor(DP1,DL1,DP2,DL2).
check_labor(AP1,AL1,AP2,AL2):- not (AP1==AP2),!.
check_labor(AP1,AL1,AP1,AL2):- skilled(AL1,AL2).
```

In this qualitative modelling, the query `?- assignment, prefer_check` yields a valid assignment that is possible and satisfies the preference constraints.

6.2. *Quantitative model: Product mix*

Quantitative modelling constructs the LP formulation by exploiting the partial solutions of CLP ($\Re$). The objective function and constraints are formed by using database and by leaving the product mix variables Q_1, Q_2,...,Q_n unbound. The linear programming for the product mix can be built as follows:

```
product_mix:-  minpro_con(Q1,Q2,...,Qn),
               labor_con(labor1,LR1,Q1,Q2,...,Qn),
               labor_con(labor2,LR2,Q1,Q2,...,Qn),
               :
               machine_con(machine1,MR1,Q1,Q2,...,Qn),
               machine_con(machine2,MR2,Q1,Q2,...,Qn),
               :
               objective(OBJ,Q1,Q2,...,Qn).
```

The minimum production constraints are generated by the predicate **minpro_con**. In our example with 5 products, for instance, the query `?- minpro_con(Q1,Q2,Q3,Q4,Q5)` returns constraints `Q1>=250, Q2>=170, Q3>=190, Q4>=180, Q5>=200`. The constraints on labor resources are formed through the predicate **labor_con** while the machine resource constraints are generated by the predicate **machine_con**. If we assume an assignment stored in the assertion **assign** the labor constraint for the *skilled mechanic* is obtained by `?- labor_con(skilled_mechanic, LR,Q1,Q2,Q3,Q4, Q5)`. The resulting constraint is:

$$Q1 + 2.66667 * Q2 + 1.66667 * Q3 + 3.25Q5 \leq 2133.33.$$

Assuming the same assertion of **assign**, we obtain the following constraint for the *welding machine* through the query `?- machine_con(weld_machine,MR,Q1,Q2,Q3,Q4,Q5)`.

$$1.25 * Q1 + Q2 + 1.25 * Q3 + 0.75 * Q4 + Q5 \leq 1530.$$

The predicate **objective** forms the objective function of the product mix to get an optimal solution. After all constraints and an objective function are formed the modified simplex method is applied to the entire constraint set to test for feasibility and optimality. If it is infeasible or if its objective value is less than the current optimal value, backtracking is initiated and the interpreter moves to the qualitative model to generate another feasible assignment. Otherwise the new incumbent value is stored to replace the current optimal one. This procedure continues until all feasible assignments are explored.

Finally the qualitative model is combined with the quantitative model through the query `?- assignment,prefer_check,linear_program`. The CLP($\Re$) interpreter then returns the assignment that is feasible and satisfies the preference constraints, together with the product mix which maximizes the total profit.

7. Comparison of MIP and CLP

7.1. Representational economies

For the purpose of comparison at the problem representation level, the following notation is used:
N: number of products
B: number of laborers
L: sum of number of processes that each laborer can do; $L = \sum_i n(K_i) \text{where} (i,K_i) \in \mathcal{T}$.
E: number of machinesHere L is defined using the set $\mathcal{T}$ described in the MIP formulation.

It can be easily seen that the MIP formulation requires $2N\,L + N$ numeric variables for x_{ijk}, z_{ijk} and y_j. In CLP formulation there are N numeric variables for the production quantity of products. MIP

Table 1
Comparison of representation by MIP and CLP

Representation			MIP	CLP	
				Numeric	Rules
Variables	0-1 variables (x_{ijk})		N L		
	prod. quantity (y_j)		N	N	
	artif. variables (z_{ijk})		N L		
	total number		2N L+N	N	20
Constraints	assignment	possible	2N		3
		preference	C		3
	product mix	labor	B	B	
		facility	E	E	
		min. prod.	N	N	
		linear trans.	3N L		
	total number		3N+C+B+E+3N L	B+E+N	6

Table 2
Representations of three case examples by MIP and CLP

Case		I			II			III		
Data	N	5			15			30		
	B	5			8			12		
	L	14			23			34		
	C	31			68			125		
	E	3			6			10		
Method		MIP	CLP		MIP	CLP		MIP	CLP	
			Num.	Rul.		Num.	Rul.		Num.	Rul.
Variable	0-1 integer	70			345			1,020		
	total	145	5	20	705	15	20	2,070	30	20
Constraint	assignment	41		6	98		6	185		6
	prod. mix	223	13		1,064	29		3,112	52	
	total	264	13	6	1,162	29	6	3,297	52	6

needs $2N$ constraints for the possible assignment since each product attribute constitutes N constraints. The number of constraints C for preference constraints in MIP can be defined as:

$$C = \sum_{(j,\bar{j}) \in \mathscr{P}} \sum_{k \in H_{(j,\bar{j})}} n(H_{(j,\bar{j})}) \times (n(I_k) - 1),$$

where $(k, I_k) \in \mathscr{F}$ and $H_{(j,\bar{j})}$ is the process required by both product j and $\bar{j}$. The set $\mathscr{P}$ and $\mathscr{F}$ are described in the MIP formulation. In contrast, the CLP formulation for the assignment includes six predicate rules containing 20 symbolic variables (10 for the possible assignment and another 10 for the preference). Both MIP and CLP include the same number of quantitative constraints for minimum production, labor and machine resources. The MIP formulation requires additional $3N\,L$ constraints to convert the nonlinear formulation into linear one (see Table 1).

There is no significant difference between MIP and CLP in representational economies for the quantitative constraints. Even if the MIP formulation requires a large number of constraints for linear conversion, this part can be easily constructed by an automatic matrix generator. Therefore, the emphasis for the comparison should be on the representation of the domain-specific heuristic rules for the assignment. In MIP the knowledge representation process for the assignment is establishing a $(2N + C + 1) \times (N\,L + 1)$ matrix. The CLP formulation replaces this matrix by six predicate rules that include 20 symbolic variables.

The comparison of representation is applied to three cases that have different number of products, processes and laborers. The results are given in Table 2. The representational economies for domain-specific knowledge of CLP becomes more evident as the number of products, processes and laborers increases. Consider, for instance, Case III in which there are 30 products, 12 labor groups, 10 machines, etc. The MIP formulation process in Case III requires building a $186 \times 1{,}021$ matrix for the assignment. If CLP is employed, modellers can replace this large matrix with six predicate rules involving 20 individual variables. Note that as the number of products, processes and labors increases the MIP matrix size for the assignment grows exponentially, whereas the number of predicate rules in CLP remains unchanged.

7.2. Partial solutions

If the input is incomplete, the CLP($\mathfrak{R}$) interpreter simplifies the model using available domain information and returns partial solutions where final judgements are left to decision makers. This allows users to consult constraint satisfaction environments with various queries. For instance, users can query "how many man-hours of *skilled welder* time are necessary to achieve profit between \$7,300 and \$7,800?" as follows:

```
?-  objective(OBJ,Q1,Q2,Q3,Q4,Q5),
    labor_con(skilled_welder,LR,Q1,Q2,Q3,Q4,Q5),OBJ>=7300,OBJ <=7800.
```

CLP($\Re$) returns the hour range of the *skilled welder* time necessary for obtaining the target profit, that is, `LR>=2470, LR <=3307`. In a similar way, decision makers can have the CLP($\Re$) interpreter generate assignments which guarantee a certain profit by using the following query:

```
?-  assignment,prefer_check,objective(OBJ,Q1,Q2,Q3,Q4,Q5),OBJ>=7300.
```

The answer of this query is useful especially under multiple criteria decision situations in which the profit maximization is just one objective. Once a list of assignments that result in certain profits is obtained, a decision maker can select the best one from the list by considering other decision criteria. In this way, CLP($\Re$) users can get useful information about any local parts of the problem by combining associated predicate rules. By contrast, the output of a general MIP solver is an *all-or-nothing* type. It returns an optimal solution only when the problem is feasible and the input data is complete. If users want solutions with partial information, they need to modify the MIP solver to work with subset of total data sets.

7.3. Ease of model revision

It is easier to make revisions to a plan in CLP than in MIP when underlying assumptions of constraint satisfaction problems often change. The incremental programming style of CLP matches well the incremental revision of a model. To illustrate the ease of model revision in CLP, it is assumed that some processes are preferred over others and assigned to more important products. Thus the following heuristic rule should be added to the existing model.

If a product j is more important than a product $\bar{j}$,
and a process k is preferred to a process $\bar{k}$,
and both product j and $\bar{j}$ have a same product attribute,
and either process k or $\bar{k}$ can be used for the product attribute,
then assign the process k to the product j and the process $\bar{k}$ to the product $\bar{j}$

In CLP this incremental revision to the preference constraints can be easily achieved by adding the following rule **check_process** to the predicate **check**:

```
check_process(A1,AP1,A2,AP2):- not (A1==A2),!.
check_process(A1,AP1,A2,AP2):- AP1==AP2,!.
check_process(A1,AP1,A2,AP2):- prefer(AP1,AP2).
```

In contrast, the following set of constraints needs to be built into MIP to make the same revision. Note that modellers should define complex sets such as $\mathscr{K}$, Θ_v and Θ_w again.

$$x_{ijk} + \sum_{\hat{k} \in \mathscr{K}} x_{ij\hat{k}} \leq = 1,$$

where$(j,\bar{j}) \in \mathscr{P}, \mathscr{K} = \{(\hat{k} \succ k) \in \mathscr{G}\ or\ \hat{k} = k\}$ and $k,\hat{k} \in \Theta_v,\Theta_w$,

$$\Theta_v = \left\{(j,v) \in \mathscr{M},(\bar{j},v) \in \mathscr{M}, k \in K_v, \hat{k} \in K_v, (i,K_v) \in \mathscr{T}\right\},$$

$$\Theta_w = \left\{(j,w) \in \mathscr{S},(\bar{j},w) \in \mathscr{S}, k \in K_w, \hat{k} \in K_w, (i,K_w) \in \mathscr{T}\right\}.$$

Another aspect of revision is that certain changes actually require that changes be made in the whole set of constraints on domain-specific knowledge. For example, the importance ordering of products may be reversed because of market evolution. If this is the case, CLP can accommodate the changes with relative ease because individual objects are represented as variables in predicates. Modellers can revise the model by just changing the assertions **important** in CLP. There is no need to revise the predicate rules. This contrasts with MIP, in which modellers should rebuild the whole matrix of preference constraints since 0-1 variables correspond to propositional logic that describes the knowledge at the individual object level.

8. Computational efficiency issues

The OR approach has focused on elegant algorithms and provided efficient methods for solving problems. Hooker [12] proposed integer programming methods for logical inference (propositional logic) since mathematical computation is more efficient than manipulation of symbols. The motivation to represent qualitative constraints using 0-1 integer variables is to use this efficient mathematical computation to reason about the domain world. In this section we compare the computational efficiency of CLP with that of MIP.

8.1. Matrix size

The branch-and-bound method has proved most successful in general on practical MIP problems. In the branch-and-bound method, an MIP problem is first solved as an LP problem by relaxing the integrality conditions. If the resulting solution satisfies the integrality conditions, the problem is solved. Otherwise, we perform a tree search by branching the variables that have fractional values. The efficiency of the method highly depends on the search strategy, that is, the selection of branching variables and search nodes.

The MIP formulation of the QQCSP contains a relatively large matrix. It is already shown that the domain-specific knowledge representation requires a large number of integer variables and constraints. There is another aspect of the QQCSP that augments the matrix size of MIP: the interdependency between qualitative and quantitative constraints. In the example of production planning, the initial MIP formulation becomes nonlinear because of the interdependency between the assignment and the product mix. MIP formulation should introduce a great number of artificial variables and constraints to convert the nonlinear formulation into a linear one. This transformation, which is unavoidable if we want to employ an efficient MIP solver like the branch-and-bound algorithm, increases the matrix size of the MIP. Therefore, the MIP in general begins with solving LP with a large matrix for the QQCSP.

In contrast, the LP formulation of CLP includes variables and constraints associated only with the product mix. It is clear that the LP matrix of CLP is much smaller than that of MIP. In the Case I of Table 2, for example, the MIP formulation contains a 264×145 matrix whereas the LP matrix of the CLP is 13×5. The difference is more significant in the Case III where the MIP matrix is $3{,}297 \times 2{,}070$ and the CLP includes a 52×30 LP matrix.

8.2. Rounds of simplex computation

We also need to consider how many rounds of LP computation are necessary in CLP and MIP. The number of LP computation in MIP is determined by the search strategy and problem structures. The branch-and-bound method prunes the search tree using the resultant LP solution at each search node,

```
go :- assignment, prefer_check.
assignment :- possible(p1), possible(p2), ..., possible(p5).
prefer_check :- forall(important(P1,P2),check(P1,P2)).
```

(a) without using domain specific information

```
go :- possible(p2),
      possible(p3), check(p2,p3),
      possible(p4), check(p2,p4), check(p3,p4),
      possible(p1), check(p2,p1), check(p3,p1), check(p4,p1),
      possible(p5), check(p2,p5), check(p3,p5), check(p4,p5).
```

(b) with using domain specific information

Fig. 2. Pruning search space with domain specific information.

thus reducing the number of LP computations. In contrast, the number of LP computations in CLP varies depending on the set size of feasible solutions of qualitative constraints because of the generate-and-solve mechanism (CLP first generates an assignment and formulates a LP problem based on that assignment). Therefore, there is a trade-off between MIP and CLP regarding the number of LP computations. If the qualitative constraints yield a large number of feasible solutions, the branch-and-bound method may be more efficient than CLP. However, if complex domain-specific heuristics result in a small number of feasible solutions, we can use CLP to solve the QQCSP with a small number of LP computations whose matrix size is much smaller than that of MIP.

8.3. Pruning search space with domain specific information

MIP and CLP employ different search mechanisms. The prominent feature of the branch-and-bound method is that it can dramatically curtail the search space using the LP solution at search nodes. The search process of MIP in some sense is mixed with mathematical computation. In contrast, the generate-and-solve mechanism of CLP separates the search process from mathematical computation. This separation provides us with a somewhat different opportunity to improve the search process: pruning the search tree with domain-specific information.

Consider two programs in Fig. 2 where the program (a) represents the original CLP formulation for the assignment in Case I. In the program (a), the predicate **assignment** explores all possible nodes and the predicate **prefer_check** returns only valid assignments. We can exploit the preference predicate **check** to trim the search tree in advance as in the program (b). This improvement of the search process using domain-specific information is possible because the logical inference is separated from the mathematical computation. We are here not arguing that CLP is computationally more efficient than MIP. Instead, it is argued that the computational efficiency of CLP is comparable to that of MIP.

9. Conclusions

The emphasis of the AI research has been on domain-specific knowledge representation. The basic assumption of the AI approach is that the mental processes under discussion can be represented in the form of symbolic manipulation. On the other hand, OR has been considered as fundamental instruments when fast and efficient algorithms are essential for problem solving. Unlike AI techniques, the OR approach ultimately reduces the problem situation to something that can be formulated in terms of real numbers. It has been well recognized that AI and OR approaches have complementary advantages: AI for domain-specific knowledge representation and OR for efficient mathematical computation.

This study has introduced Constraint Logic Programming as a new modelling tool that combines these complementary strengths of the AI and OR approaches. It is argued that the integration of inference techniques with mathematical computation of CLP provides relative advantages over the Mixed Integer Programming approach when constraint satisfaction problems include domain-specific heuristic rules as well as quantitative constraints. These advantages can be summarized as follows:

- CLP can represent complex domain-specific heuristic rules more efficiently than MIP because a large number of integer variables and constraints are replaced by a small number of predicate rules and symbolic variables, and
- CLP can return partial solutions that are not available in MIP but very useful to decision makers, and
- CLP is superior to MIP in making revisions to a plan whose underlying assumptions often change.

A case example of constraint satisfaction problems is implemented by MIP and CLP for comparison of the two approaches. The result exhibit that modellers can exploit the advantages of CLP with computational efficiency comparable to MIP.

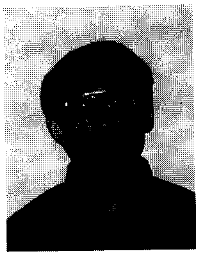
Most mathematical models have operated on a stand-alone basis under the implicit assumption that the user community has sufficient mathematical expertise. In many organization decision making situations, however, this assumption is not guaranteed. Often, opportunities for OR applications are not exploited. The managers with real-life problems usually have little, if any, formal computing or mathematical expertise and demand simple representation. In contrast, OR scientists are mainly interested in elegant solutions and pay more attention to algorithmic rationality than to non-mathematical aspects such as representation. The Structured Modelling [10], Production Modelling [15], and Formal/Unified Modelling [11] can be viewed as efforts to overcome this practicality gap and to help users build mathematical models.

The need for simple modelling environments together with our experience with CLP provides a future study issue that is worth pursuing: Modelling Language for Constraint Satisfaction based on the CLP scheme. The language will allow decision makers to represent constraint satisfaction problems efficiently and facilitate revision of the model. In addition users may be able to interactively communicate with the system through the partial solutions. Since many decision situations in organizations involve constraints and require numeric analysis as well as inference, such a modelling language should have many useful applications.

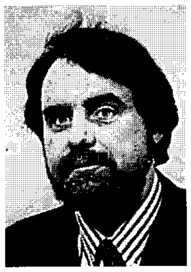
References

[1] R.H. Bonczek, C.W. Holsapple and A.B. Whinston, A Generalized Decision Support System Using Predicate Calculus and Network Data Base Management, Operations Research 29, No. 2 (1981), 263–281.
[2] A. Brook, A. Kendrick and A. Meeraus, GAMS: A User's Guide (The Scientific Press, Redwood City, CA, 1988).
[3] G. Chandrasekaran and R. Remesh, Microcomputer Based Multiple Criteria Decision Support System for Strategic Planning, Information and Management 12 (1987), 163–172.
[4] J. Cohen, Constraint Logic Programming Languages, Communications of the ACM 33, No. 7 (July 1990), 52–68.
[5] A. Colmerauer, An Introduction to Prolog III, Communications of the ACM 33, No. 7 (July 1990), 69–90.
[6] G. Dantzig, Linear Programming and Extentions (Princeton University Press, 1963).
[7] V. Dhar and N. Ranganathan, Integer Programming vs. Expert Systems: An Experimental Comparison, Communications of the ACM 33, No. 3 (March 1990), 323–336.
[8] M. Dincbas, P. Van Hentenryck, H. Simonis, A. Aggoun, T. Graf and F. Berthier, The constraint logic programming language CHIP, Proceedings of the International Conference on Fifth Generation Computer Systems (Tokyo, 1988), 693–702.
[9] D. Dolk and B. Konsynski, Knowledge Representations for Model Management Systems, IEEE Transactions on Software Engineering 10, No. 6 (November 1984), 619–628.

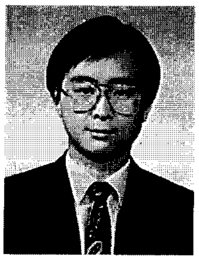
[10] A.M. Geoffrion, An Introduction to Structured Modelling, Management Science 33, No. 5 (May 1987), 547–588.
[11] S.N. Hong, A Formal, Unified Modelling Framework: Knowledge Representation and Automated Reasoning, Ph.D. thesis (1991), The University of Texas at Austin.
[12] J.N. Hooker, A Quantitative Approach to Logical Inference, Decision Support Systems 4 (1988), 45–69.
[13] J. Jaffar and J-L Lassez, Constraint Logic Programming, Proceedings of the Fourteenth ACM Symposium of the Principles of Programming Languages (Munich, 1987), 111–119.
[14] J. Jaffar and S. Michaylov, Methodology and Implementation of a CLP System, Proceedings of the Fourth International Conference on Logic Programming (Melbourne, 1987), 196–218.
[15] R. Krishnan, Knowledge Based Aids for Model Construction, Ph.D. thesis (1987), The University of Texas at Austin.
[16] C. Lassez, Constraint Logic Programming, BYTE (August 1987), 171–176.
[17] J.K. Lee and E.G. Hurst, Jr, Multiple-Criteria Decision Making Including Qualitative Factors: The Post-Model Analysis Approach, Decision Sciences 19, No. 2 (Spring 1988), 334–352.
[18] J.K. Lee and H.G. Lee, Interaction of Strategic Planning and Short-term Planning: An Intelligent DSS by the Post-Model Analysis Approach, Decision Support Systems 3 (1987), 141–154.
[19] F.S. Roberts, Measurement Theory with Applications to Decisionmaking, Utility, and the Social Sciences, Encyclopedia of Mathematics and the Applications 7.G (Addison-Wesley Publishing Company, Reading, Massachusetts, 1979).
[20] R.R. Trippi, Decision Support and Expert Systems for Real Estate Investment Decisions: A Review, INTERFACES 20, No. 5 (Sep. - Oct. 1990), 50–60.



Ho Geun Lee is Assistant Professor of Information and Systems Management Department at Hong Kong University of Science and Technology. Prior to that he was Visiting Scholar at the Erasmus University Research Institute for Decision and Information Systems (EURIDIS) of Erasmus University. He received his Ph.D. in Management Information Systems from the University of Texas at Austin in 1993 and his M.S. in Management Science from Korea Advanced Institute of Science and Technology in 1986. His research interests include (1) integration of AI and OR approaches for management science applications, (2) electronic commerce and electronic market structures, and (3) telecommunications and multi-media applications such as distance collaboration and distance education.



Ronald M. Lee is currently Director of the Erasmus University Research Institute for Decision and Information Systems (EURIDIS) of Erasmus University. Formerly, he was Associate Professor of Management Science and Information Systems Department at the University of Texas at Austin. He has a Ph.D. in Decision Sciences (Wharton, 1980), and has previously served as a research scholar at the International Institute for Applied Systems Analysis in Vienna, Austria, and as Visiting Professor of Management at the Universidade Nova de Lisboa, in Lisbon, Portugal. His research focuses on the use of formal logic representations for management science applications. Current projects involve the use of logic modelling to represent and manage formal business communications systems focusing on bureaucratic systems (formalized communications within institutions) and electronic contracting systems (formalized communications between enterprise).



Gang Yu is Associate Professor at the Graduate School of Business, University of Texas at Austin. His research involves industrial applications of large scale combinatorial and network optimization models. During the past six years, he has focused on designing and building decision support systems for industry and government agencies, including United Airlines, IBM, Electronic Data Systems, Continental Airlines, Tracor, and U.S. Navy. Gang Yu has published numerous papers on *Management Science, Operations Research, Interfaces, ORSA Journal on Computing, Transportation, Science*, and other journals.