Solving the Short Run Economic Dispatch Problem Using Concurrent Constraint Programming

Juan Francisco Díaz F.[1], Iván Javier Romero[1], and Carlos Lozano[2]

[1] Universidad del Valle, Cali, Colombia
Escuela de Ingeniería de Sistemas y Computación
{jdiaz,ivromero}@univalle.edu.co
[2] Escuela de Ingeniería Eléctrica y Electrónica
clozano@univalle.edu.co

Abstract. This paper shows a description of an application for solving the Short-Run Economic Dispatch Problem. This problem consists of searching the active power hourly schedule generated in electrical networks in order to meet the demand at minimum cost. The solution cost is associatted to the inmediate costs of thermal units and the future costs of hydropower stations. The application was implemented using Mozart with real-domain constraints and a hybrid model among real (XRI) and finite domains (FD). The implemented tool showed promising results since the found solution costs were lower than those found in the literature for the same kind of problems. On the other hand, in order to test the tool against real problems, a system with data from real networks was implemented and the solution found was good enough in terms of time efficiency and accuracy. Also, this paper shows the usability of Mozart language to model real combinatory problems.

1 Introduction

The Hydrothermal Economic Dispatch Problem has as one of its purposes to generate the active power supply schedule for generating units in an electrical system in order to meet the demand at minimum cost. This can be seen as an optimization constraint satisfaction problem (*OCSP*) with a combinatory space of solutions.
An OCSP is defined by a set of variables, their domains, a series of constraints among them and an objective function to be optimized. The general objective is to assign values to all variables in such a way that they meet all the constraints among them and the objective function evaluated for these values be optimized. The Concurrent Constraint Programming (CCP) approach is widely used for solving this type of problems.
Solving an OCSP using CCP languages, consists of two steps: modeling the problem (logical specification) and specifying the search strategy for finding its solutions. Modeling involves basically writing in the con-

current constraint programming language the variables, their domains and the constraints among them.
In the CCP paradigm a procedure, called a propagator, is associated to each constraint. This procedure tries to reduce the domain of the variables associated to it (i.e. it discards values of the domains of the variables that can not be part of any solution.)
The collection of propagators acting on the variables may cause a domain to become empty. This means that the problem specification is contradictory and so, there is no solution. A propagator is considered solved when any combination of values in their domains is a solution. Propagators may instead reach a fixed point state in which nothing new about the variables can be deduced (i.e. the domains cannot be further reduced). In that case, a search stage is necessary.
Specifying the search strategy consists in defining the criteria for the search when the propagation has taken computation to a fixed point state in which undetermined variables (i.e. those having domains with more than one value) remain. The main idea is to add to the current fixed point state one or more constraints that allow the propagation process to advance a little bit more (i.e. allowing it to reduce some domain of some variable). Since these constraints are not part of the problem constraints, it is also necessary to explore what happens if the opposite constraints are added to the fixed point state. In that way all possibilities are taken into account and no solution is missed. The process of adding these new constraints is called distribution.
The search strategy specifies the constraints that must be added and the order in which the searching process is carried out; whatever they are, adding them creates two new search states. In each of them, the propagation process is applied again until a new fixed point state is reached and the procedure is repeated.
If all variables are determined, a solution (not necessarily optimal) has been found. The latter is taken, a constraint saying that the next solution must be better than the last solution found is added, and a searching process for a new solution is started. This procedure is repeated until the optimal (or near optimal) solution is found or the systems resources, time, memory be exhausted. Otherwise, if, after exploring all possibilities, all states are contradictory states, then the problem does not have a solution.
The efficiency of the search is directly linked to the number of explored states. The more the domains of the variables in the propagation stage are efficiently reduced, the less states are generated and, therefore the problem is solved more quickly. The number of generated states also depends on the distribution strategy.
Searching for the best solution in the CCP model uses a branch and bound technique. A valuation (in this case, the cost of generation) of a previously found solution provides an upper bound. A constraint asserting that the next solution must have a better valuation than the upper bound is then added. This constraint has also an associated propagator that further reduces the search tree.

Using the CCP paradigm thus in general leads to smaller search trees. Furthermore, the tree rarely has to be traversed exhaustively, even when all possible solutions must be computed.
In this work, the Concurrent Constraint Programming (CCP) technique is used to search for approximate solutions to the *Economic Dispatch Problem* by using Mozart, a concurrent constraint programming language.
Mozart language was used because the specification of the search strategy can be done by the programmers in a very high level reflecting easily the intuition of the expert in the subject or problem to be solved. This makes Mozart suitable for programming the intelligent, reasonable strategies defined by the experts.
The developed application was used to solve problems found in both the literature and real life, with excellent perspective related to time efficiency. However, the application shows a high use of RAM memory. The results found allows to be optimistic about real problems being solved by this technique.
This paper is organized as follows: Section 2 makes an introduction to the economic dispatch problem; in Section 3, the proposed model in natural lenguage and its implementation in Mozart are described. In Section 4 the implemented distribution strategies are described. In Section 5 the solved problems results given by the developed application are shown, and finally, in Section 6 the conclusions and future work are ennumerated.

2 The Economic Dispatch Problem

The purpose of the power system economic operation is to use the energy resources (thermal, hydro, solar, wind, among others) available for energy generation in an optimal manner such that the electricity demand is met at minimum cost and under certain degree of reliability, quality and security [1]. Among the generating power units that we might find in a power system are: thermal power plants that work with fossil fuel, run-of-river power plants, hydro power plants and chain plants.

Solution Cost. The solution cost is due mainly to the sum of all dams future costs at the end of the study and, the inmediate costs of the thermal power plants for each of the study analisys stages. When the stored water rises, less water is used for energy production in the correspondig stage; as a consequence, more thermal generation is required, and the inmediate costs are higher. In turn, the future cost functions are associated to the thermal generation costs from the following stage until the final stage of planning[2, 3]. When the systems thermal power plants operate at the same incremental costs, the power production cost is the lower [5].

3 The Model and its implementation in Mozart

This section shows a general model description in natural language as well as the transformations of some of the mathematical equations into Mozart.

3.1 Modelling in natural language

Objective Function. The objective function is composed by a sum of all inmediate costs for each of the period study stages; these inmediate costs are composed by the thermal units production costs in each of the operating areas, plus curtailment plant costs, plus the water releases costs for each of the areas with hydropower plants. Also, the dam's future costs from each area is added to the objective function.
Demand Constraint. The power demand for each area should be met. The amount of power that cannot be met in an area is represented by a curtailment plant. The produced power by an area to meet its demand, plus the transmission line losses is composed by the amount of power generated by its power plants, plus the imported power from other areas, less the exported power to other areas. The curtailment plants are ficticious plants used to guarantee the solution feasibility when it is not possible to meet the demand.
Units Operating Constraint. The power produced by both hydroplants with dams and run-of-river power plants is limited by the generator turbine efficiency. The thermal power plants can only operate between upper and lower boundaries.
Dams Dynamics. The storage volume of water in a dam at the end of each study period stage should be equivalent to the stored volumen at beggining, plus the water inflows due to tributary rivers and the discharges from upstream reservoirs, less the flow rate through the turbine, less the spillage discharge rate, less the leakages and evaporations.
Constraints over Run-of-River Power Plants. The power produced by this kind of power plant is due mainly to the water inflows due to both the tributary rivers and the discharges from upstream reservoirs. There must be a correpondence between the amount of inflows and the amount of produced power, with the discharges and spillages from the upstream power plants.
Importing and Exporting Constraints. The main constraint imposed over the interarea transactions is related to the limitation on the transmission lines. What an importing is for one area, it is exporting for other and so, there is a limit on exporting due to the transmission line limitations. Also, the amount of power imported by an area a_1 should be equal to the exported power from area a_2.

3.2 Implementation of the model in the Mozart language

Figure 1 shows the implementation of the dam storage volume Equation (2) at the end of a specific period of time, using Mozart language. The

description of both model decision variables and variables with known values for Equation (2) are presented in Table 1. It is possible to notice the expresivity of the Mozart language for easily translating this kind of formula.

Table 1. Elements of the Equation (2)

$UCH_{t,r,m}$	Undammed water done by the hydro plant m in area r during stage t (decision variable)
$SCH_{t,r,m}$	Releases done by hydro plant m in area r during stage t (decision variable)
$PCH_{r,i}$	Set of hydro plants upstream directly connected with hydroelectric plant i in area r (known value)
$V_{t,i,r}$	Volume of stored water in the hydroelectric plant dam i from area r to the final period of time t (decision variable)
$A_{t,r,i}$	Water streams per hour entering to the dam in hydro plant i in area r in stage t (known value)
$S_{t,i,r}$	Volume of released water per hour by hydro plant with dam i in area r during stage t (decision variable)
$Ph_{t,r,i}$	Delivered Power per hour by Hydro Power Plant with Dam i from area r in stage t (decision variable)
$\rho_{r,i}$	Hydro Power Plant i production coefficient in area r
$f_{t,i,r}$	Leakings in dam per hour by hydro plant i in area r (known value)
$e_{t,i,r}$	Steaming in dam per hour by hydro plant i in area r (known value)
dur	Duration of the stage in hours

$$IU = \sum_{m \in PCH_{r,i}} [UCH_{t,r,m} + SCH_{t,r,m}] \tag{1}$$

$$V_{t,r,i} = V_{(t-1),r,i} + \left(A_{t,r,i} + IU - \frac{Ph_{t,r,i}}{\rho_{r,i}} - S_{t,r,i} - f_{r,i} - e_{r,i} \right) \cdot dur \ \forall t \forall r \forall i \tag{2}$$

4 Distribution Strategies Implementation

All the distribution strategies are based mainly on two functions, *Order* and *Value*. The function *Order* decides which is the next variable to be distributed, comparing among pairs of variables from a list of those that have not been determined yet. The function *Value* decides what value to assign to the chosen variable in *Order*. Distribution strategies for both thermal units and hydro units were developed.

Distribution strategies for thermal units Two distribution strategies were developed for thermal units. The first deals with the first-order

```
for T in 2..NPeriods do
   CArea={MakeTuple c NAreas} in
   for R in 1..NAreas do
      ConstHydro={MakeTuple c RArea.R.nHydroPlant} in
      for I in 1..RArea.R.nHydroPlant do
         Spill HydroPower VolDam IU
      in
         %%Domain declaration for Spill HydroPower
         %%VolDam ...
         IU = {InflowUpstream T R I}
         {XRI.Consistency
          eq(VolDam
             plus(
                CPeriod.(T-1).area.R.rHydro.I.volDam
                times(
                   sub(plus(RArea.R.rHydro.hydrol.I.T
                            IU)
                       plus(
                          divide(HydroPower
                                 RArea.R.rHydro.cf.I)
                          Spill RArea.R.rHydro.filt.I
                          RArea.R.rHydro.evap.I)
                     ) Dur ) ) ) }
         ConstHydro.I =
         constHydro(volDam:VolDam
                    spill:Spill hidroPower:HydroPower)
      end CArea.R = c(rHydro:ConstHydro)
   end CPeriod.T = c(area:CArea) end
```

Fig. 1. Representation of the dam dynamics equation in Mozart

cost functions. This strategy selects first the thermal units in order of priority (lower coefficients in the cost function). The value of the variable to distribute will be the nearest to the upper boundary. The second strategy was designed to deal with second-order cost functions. This strategy chooses first by stage order, then by area, then by incremetal costs between the limits and at last, it chooses the incremental costs below lower boundary. If the chosen variable by the function *Order* generates incremental costs considering the power constraint, then it distributes by the nearest value to the generated power at the same incremental costs. If the chosen variable has its ideal power to operate at the same incremental costs under the lower boundary, then it will distribute by the lower limit of the variable. If the chosen variable has its ideal power to operate at the same incremental costs over the upper limit, then it will distribute by the upper limit of the variable.
If the selected variable to distribute can generate incremental costs among the limits then, it chooses the nearest value. Both distribution

strategies for thermal units were implemented for the variables within FD and XRI domains.

Distribution strategies for hydroelectric units Three distribution strategies were developed for hydroelectric units in the XRI domain. The first chooses the variable in order of priority: the power from an hydro plant, the spillage from the dam, and the level of the dam. The second strategy, considers not only the order of the first, but gives priority to the variables that represent the power from hydro plants with dam with the greater amount of outflows power plants that benefit from its discharges and spillages; then, it gives the priority to the plant with the greater dam. The third strategy gives priority to the plant with the greater dam. These strategies were also implemented in the finite domain but only considering the distribution of the variables representing the power from hydro power plants.

In Figures 2 and 3, the implementation in Mozart language of the first strategy in the finite domain is shown. Again, from these figures it can be seen the expresivity of the language for coding these strategies. In addition, as it can be seen in Figure 4, higher order programming gives the possibility for easily changing from one strategy to another.

The function *Value* is the same for the six strategies. For the variables representing the hydropower plants with dam, the value of the power with which the distribution will be done, will be according to a percentage of power at which the plants are required to operate. For variables representing the run-of-river power plants, the distribution will be done always over its upper limit. For variables representing the discharges and the levels of the dam, the distribution will be carried out using their lower limits and upper limits, respectively, in order to keep their discharges to the minimum possible value and the level of the dam at the highest possible limit, respectively.

5 Tests

The tests were carried out in a PC with Linux Operating System, Debian 3.1, with two Xeon processors at 2.8 GHz each and 2 GB RAM memory. Some of the problems were taken from the literature, others were considered using the data found in [4]. The future cost curves of the solved problems from [4] were taken by multiplying the future cost curves from the Laja lake found in the same factor.

The conventions BSCL,BSC,STBS and MUBS used in Table 2 mean cost of the best solution found in literature, cost of the best solution found by the application, time employed in searching the best solution, and used memory in the best solution time, respectively. The best solution found by the application uses a notation *xri(<limInfCosto limSupCosto>)*, which is a variable in the *XRI* domain.

As it can be seen in Figure 5 (Problem 8 in Table 2), the solution (generation costs) of the implemented tool chooses hydropower generation over thermal generation so the cost is minimized. It is also seen that the

```
Order =
proc{$ V1 V2 C}
   if V1.idPeriod < V2.idPeriod then
      C = true
   else
      if V1.idPeriod > V2.idPeriod then
         C = false
      else
         if V1.MaxLevDam > 0.0 then
            if V2.MaxLevDam > 0.0 then
               if V1.nConRiverDown >
                  V2.nConRiverDown then
                  C = true
               else
                  if V1.nConRiverDown <
                     V2.nConRiverDown then
                     C = false
                  else
                     C = V1.MaxLevDam >=
                     V2.MaxLevDam
                  end
               end else C = true end
         else C = V2.MaxLevDam <= 0.0
         end end end end
```

Fig. 2. Representation of function order in Mozart.

```
Value =
proc{$ V R}
   VAux VAux2
in
   {FD.reflect.max V.hydroPowerPlantFinit VAux}
   VAux2 =
   {Float.toInt {Float.´/´
                 {Number.´*´ Nivel V.maxPower} 100.0}}
   if VAux < VAux2 then
      R = VAux
   else
      {FD.reflect.nextLarger
       V.hydroPowerPlantFinit VAux2 R}
   end end
```

Fig. 3. Representation of function value in Mozart.

```
proc {Strategy Data}
   . . .
   R=generic(order:     Order
             filter:    Filter
             select:    Select
             value:     Value
             procedure: Proc)
in
   %%This call the distributor
   {Distribuidor.distribute R Data} end
```

Fig. 4. Calling the distributor with a especific strategy.

hydropower generation follows closely the demand behaviour, allowing to flatten the thermal generation and keeping it at its lowest levels.

Table 2. Found Solutions

Problem	BSCL	BSC	STBS(ms)	MUBS(MB)
1 [6, p. 32-33]	8194,36	xri(8165.01 8165.08)	10	2,5
2 [6, p. 32-33]	7008,32730	xri(6998.98 6999.06)	60000	30
3 [5, p. 504]	86657,0736	xri(86621.3 86621.3)	8000	6
4 [6, p. 368]	32246.44	xri(32242.5 32242.5)	85000	40
5 [6, p. 368]	31984.82	xri(32005.7 32005.8)	25000	12
6 [4]	–	xri($3.91541e^8$ $3.91541e^8$)	116000	250
7 [4]	–	xri($2.79907e^8$ $2.79907e^8$)	100000	1600
8 [4]	–	xri($4.03688e^8$ $4.03688e^8$)	35000	900
9 [1, p. 93]	-21662.4	xri(-21662.4 -21662.4)	198000	90
10 [1, p. 98]	-12660	xri($-11394.9 - 11393.9$)	3000	2
11 [4]	–	xri($2.90519e^8$ $2.90519e^8$)	200000	800

As it can be seen in Table 2, in many problems the generation costs found with the application were lower than in the literature and in a few, the solution found was a little higher than in literature. Furthermore, it is good to notice that it was possible to solve problems with real data, i.e., the problems with data taken from [4].

6 Conclussions and Future Work

In the development of this work, it was possible to show that the Concurrent Constraint Programming is applicable to the Short-term Hydrothermal Economic Dispatch Problem. The Economic Dispatch Problem is a combinatory, non linear problem, that offers real challenges for searching the solution.
Also, it is evident the possibility to develop intelligent strategies in a very high level and modular form using Mozart language.

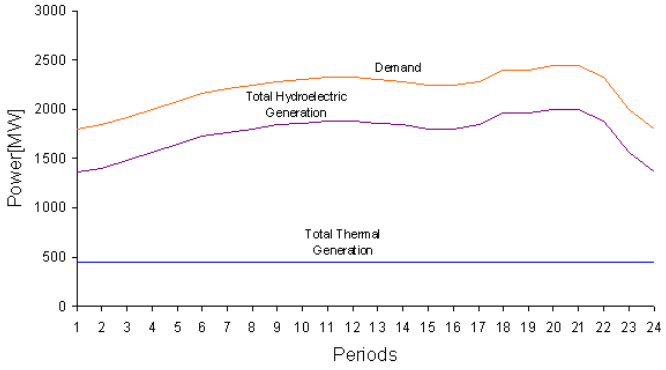

Fig. 5. Hydro Power and thermal generation for the problem 8 of Table 2

In the implemented model, the obtained results (generation costs) were better than those reported in literature. In several problems, the solution found had lower costs and in others, the results were very closed to those in the literature. At the same time, the model was applied for solving problems with real data, representing a real electrical energy supply system.
A hybrid model was implemented for distribution in both real and finite domains. In the future, it would be interesting to solve the *Economic Dispatch* along with the *Unit Commitment Problem*.

References

1. Pablo Hernán Corredor. *Operación Económica de Sistemas de Potencia*. Universidad Potinficia Bolivariana, 1992.
2. Mario Pereira, Nora Campodonico, and Rafael Kelman. Long-term Hydro Scheduling based on Stochastics Models. *EPSOM*, 1998.
3. Mario Pereira, Nora Campodonico, and Rafael Kelman. Application of Stochastic Dual DP and Extensions to Hydrothermal Scheduling. Research Report 012-99, PSRI, 1999.
4. Esteban Manuel Gil Sagás. Programación de la generación de corto plazo en sistemas hidrotérmicos usando algoritmos géneticos, tesis para optar al grado de magister en ingeniería eléctrica, 2001.
5. William Stevenson. *Análisis de Sistemas Eléctricos de Potencia*. Mc Graw Hill, 1996.
6. Allen Wood. *Power generation, operation and control*. Mc Graw Hill, 1996.