IMPLEMENTATION OF A CONSTRAINED BASED SEARCH (CBS) APPROACH USING TIME WINDOWS

Maria Teresa Moreira Rodrigues, maite@desq.feq.unicamp.br
Universidade Estadual de Campinas, Faculdade de Engenharia Química
CP 6066 – 13083-970 – Campinas, SP, Brasil
Alessandro Ludgero Noal Borjas, borjas@desq.feq.unicamp.br
Universidade Estadual de Campinas, Faculdade de Engenharia Química
CP 6066 – 13083-970 – Campinas, SP, Brasil
Amarildo de Marchi Lopes, marchi@desq.feq.unicamp.br
Universidade Estadual de Campinas, Faculdade de Engenharia Química
CP 6066 – 13083-970 – Campinas, SP, Brasil

Abstract. *CBS approaches are cited as an alternative for short term scheduling when the constraints are so hard to be fulfilled that the main objective is not to find an optimal solution. Although this approach has been used in commercial packages, the implementation details are not clearly discussed in the reference papers. Once the scheduling problem is suitable to be resolved using a CBS approach, there are many important implementation steps that must be carefully discussed and analyzed in order to lead to an appropriate implementation. This paper is focused on the aspects concerning the choice of branching heuristics, backtracking heuristics, and constraint propagation in order to keep the problem feasible through the search procedure. The CBS approach can be based on two major strategies. The first one allocates each task to an instant time, and the second is based on an ordering decision. In the first case is easy to determine when the problem ends, but a backtracking procedure is not easy to define; by the other hand, in the second case the backtracking procedure is much easier to implement but is not easy to decide when the search procedure must end. In this paper it is proposed to control the search procedure based on the number of disjunctive arcs solved at each ordering decision. Two different heuristics have been tested to decide the pair of tasks to be ordered at each tree node: one based on slack time, and the second based on a capacity analysis.*

Keywords*: Scheduling, Constrained Based Search, Heuristics*

1. INTRODUCTION

A Constrained Based Search approach is a search tree procedure well suited for those problems where the constraints are hard. The main idea is: if the decisions of greater impact are taken earlier, infeasible scenarios are likely to appear sooner during the searching process. As the procedure develops, the bottlenecks become less critical and a feasible solution is more likely to be reached.

Constrained Based Approaches depend on the definition of a time window, that is, the definition of a time interval within a task must be performed. Due to the competitive nature of short term scheduling problems, two or more tasks could require a shared resource in overlapping time windows. In this case, an estimate resource loading measure could be used as a way to support the decision where the constraints are hard. Once a criterion has been defined, it is necessary to analyze if the constraints are really hard, and if they are, how to use this information to take an ordering decision. After a decision is taken, its impact must be propagated to keep the search space updated. If the decision is relevant to the problem, then a time windows shortening is observed, and the solution space is reduced. The branching process continues until an infeasible scenario is detected, or all the initial disjunctions are resolved.

The CBS approach can be based on two major strategies. The first one allocates each task to an instant of time, and the second is based on an ordering decision. In this paper it is proposed to control the search procedure based on the number of disjunctive arcs resolved at each ordering decision.

To implement a CBS approach, it is necessary:

- Define a procedure to determine initial tasks time windows
- Define a bottleneck identification strategy
- Define a branching strategy
- Define a constraint propagation mechanism
- Define a backtracking strategy
- Define an interruption criterion

1.1. Time Windows Definition

In this paper, a software developed earlier (Rodrigues et al. 2000a) was used to determine tasks time windows, by the means of a MRP-like procedure. To perform this calculation it is necessary to state: i) the State Task Network (Kondili, Pantelides and Sargent, 1993) representation of all products; ii) demands of all products; iii) raw material delivery plan. It is important to notice that this procedure is only useful for a fixed assignment task to equipment and fixed batch size. The procedure uses a backward exploding procedure to calculate the number of batches of all intermediates and final products to fulfill the demands. Combining this backward explosion procedure to a forward explosion procedure starting with the raw material delivery plan, a set of optimistic time windows for all tasks is calculated. This time windows are optimistic because it is not taken into account the competition among tasks requiring the same resource. Those time windows can be refined through a capacity analysis plus a constraint propagation mechanism.

1.2 Bottleneck identification strategy

The main idea behind a CBS approach is to identify bottlenecks using different strategies, and resolve progressively the dynamic bottlenecks at each node until there are no more bottlenecks left. The identification of a bottleneck can be made applying different tools; in this paper it is used loading measures to identify them.

A loading measure is a way to identify time intervals that most likely can lead to infeasible situations. The idea is first to identify the most critical time interval, then the piece of equipment where this high load occurs, and select the tasks which time windows are contained in this time interval.

In the literature there are at least two important loading measures. Keng, Yun and Rossi (1988) proposed the Cruciality measure, and Sadeh (1991) proposed the Aggregate Demand. One can choose any of them, but the most important aspect is the way they are implemented because it can be very time consuming. Application examples are shown in Rodrigues (2000a,b).

1.3 Branching Strategies

The choice of a branching strategy is perhaps the most important issue in a CBS approach. To the higher load peak of the Aggregate Demand in the time horizon is associated to a piece of equipment as well to a set tasks having overlapping time windows. If the time windows do not overlap, then there is no conflict to be managed.

Cheng and Smith (1993) suggest ordering decisions as a suitable strategy in CBS approaches. In this paper an ordering strategy has been implemented, and one advantage in this case is that the tree is binary because the only two decisions to be made are task A precedes task B, or task B precedes task A.

The first step is to identify the most critical pair of tasks at each tree node. There are different ways to select a pair of tasks. Smith and Cheng (1993) suggest the use of the Slack Time, which means that the pair is selected based on the difference between total available time (defined by the smaller earliest beginning time and the larger latest finishing time) and the sum of processing times of both tasks.

1.4 Constraint Propagation Mechanism

The constraint propagation mechanism is a consistency enforcing mechanism in order to keep updated the partial solution at each tree node. In this mechanism (ILOG 1997) two types of constraints are considered: i) mass balance constraints; ii) capacity constraints.

A scheduling solution is the definition, for each task, its starting time fulfilling the problem constraints (mass balance constraints, capacity constraints, storage constraints, etc). Looking at the scheduling problem in relation to its graph representation, a solution is reached when all the ordering decisions have been taken, which means that all disjunctive arcs at the root node have been oriented.

The mass pegging relations are fixed from the beginning of the search procedure, and there are a set of oriented arcs enforced by the time windows (that is, if time windows do not overlap or if the overlap is less than the sum of processing times, than there is a fixed arc between these tasks). In the case that overlapping time windows have Intervals of Total Reliance (Sadeh, 1991) a capacity analysis is performed and if an ordering between time windows is inferred, then a disjunctive arc is resolved. In this case, the ordering decision must be enforced, leading sometimes to a reduction in the search space through reduction in tasks time windows. The search space reduction is more likely to occur if the constraints are important. The complete set of conditions to implement a constraint propagation mechanism is shown in Erschler (1976) and Rodrigues (2000a,b).

1.5 Backtracking Strategy

The definition of a backtracking strategy is strongly connected to the chosen branching strategy. In the section 1.3 it was established ordering decisions as the branching strategy. The main advantage in this case is that it is easier to manage the search tree, and avoids the time discretization problem. In the case of ordering decisions there is a natural strategy for backtracking since in the scheduling solution an ordering must be kept for each pair of tasks competing for the same piece of equipment. So, if one branch, corresponding for example A precedes B lead to an infeasible node, the search procedure backtracks to the opposite ordering decision B precedes A. If this decision is also infeasible then the problem is to identify the most promising not investigated nodes at the search tree.

The implemented backtracking procedure is based on the following aspects: i) the analysis of the number of disjunctions in the choice of the most promising node for the backtracking; ii) the management of the necessary information for the backtracking. In this paper, the most promising node for backtracking is the one that have the lesser number of disjunctions, amongst all not investigated nodes at the search tree, because it is presumed that a possible final solution can be reached by the opening of a lesser number of nodes.

1.6 Interruption Criterion

The interruption of the searching procedure depends on the type of branching strategy. Of course if an allocation procedure is chosen, then the search terminates when all the tasks have been allocated. However, in the case of ordering decisions strategy, the procedure is terminated when all possible ordering decisions have been made. It is important to notice that in this case some time windows can remain not allocated.

2. CBS IMPLEMENTATION

The CBS approach proposed with the elements presented in section 1 was implemented in Visual Basic. The main reason is to allow a friendly interface to interact with the software, and most importantly, the software allows two different running modes: i) automatic mode, in which once the procedure starts it will only be interrupted when all disjunction have been resolved, and ii) manual mode, in which the software suggests the next possible decisions and the user can decide which ordering decision is going to be implemented, and the process can be terminated at any point of the searching process. In this last case, the system automatically updates the time windows and the data base in order to allow launching an APS (Advanced Planning and Scheduling) – like software. In the next section an example is presented.

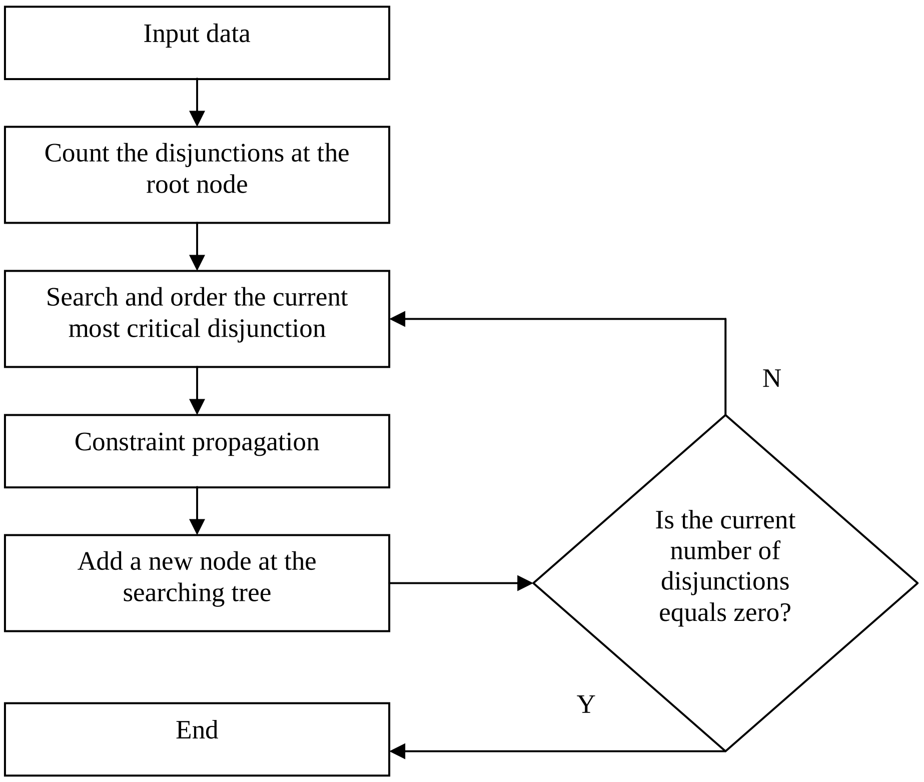

Input data
Count the disjunctions at the root node
Search and order the current most critical disjunction
Constraint propagation
Add a new node at the searching tree
Is the current number of disjunctions equals zero?
N
Y
End

Figure 1 – Implemented system general illustration

3. EXAMPLE

The STN (Kondili, Pantelides and Sargent, 1993) for the example is presented in Figure 2, and the data showing assignment task/equipment and the demands of final products are shown in Tables 1 and 2. The batches sizes (represented by X u.m), the processing times (represented by Y u.t.), as well the processing policy (represented by UIS - Unlimited Intermediate Storage and ZW – Zero Wait processing policy) are already shown in Figure 1.

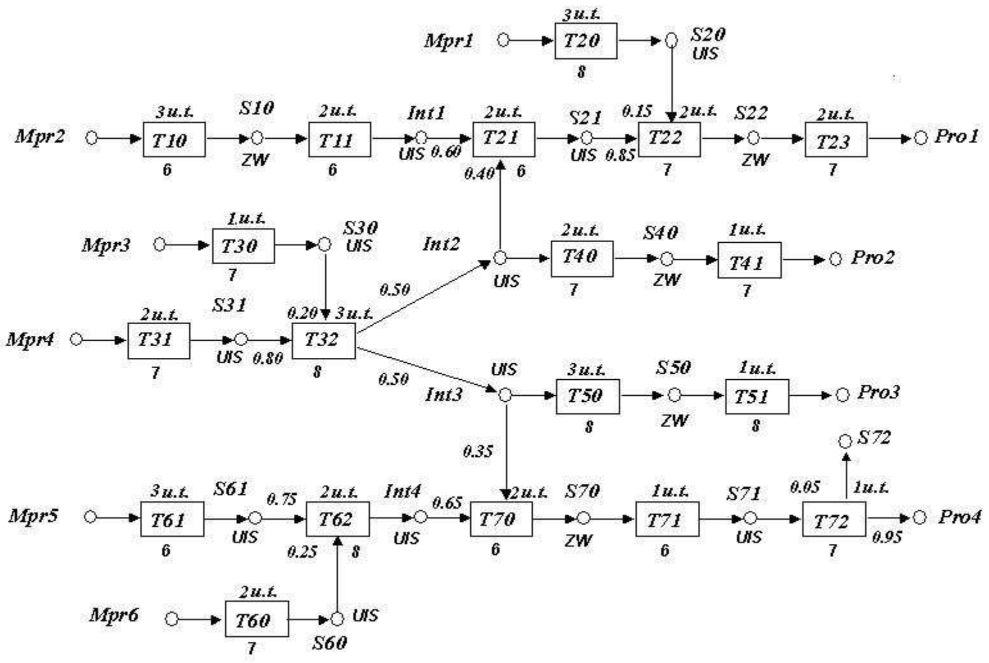


Figure 1 – STN for example (Papageorgiou L. G., Pantelides C. C. 1996)

Table 1 – Assignment task to equipment

Equipment	Task
P1	T10,T21
P2	T32
P3	T31,T72
P4	T23,T30,T60
P5	T20,T40,T50
P6	T61,T70
P7	T11,T22,T41
P8	T51,T62,T71

Table 2 – Final products demands and due dates

Products	Quantity	Due dates
Pro1	70	100
Pro2	50	100
Pro3	50	100
Pro4	50	100

The initial time windows as well the equipment loading based on the Aggregate Demand are shown in Figure 3 and 4.

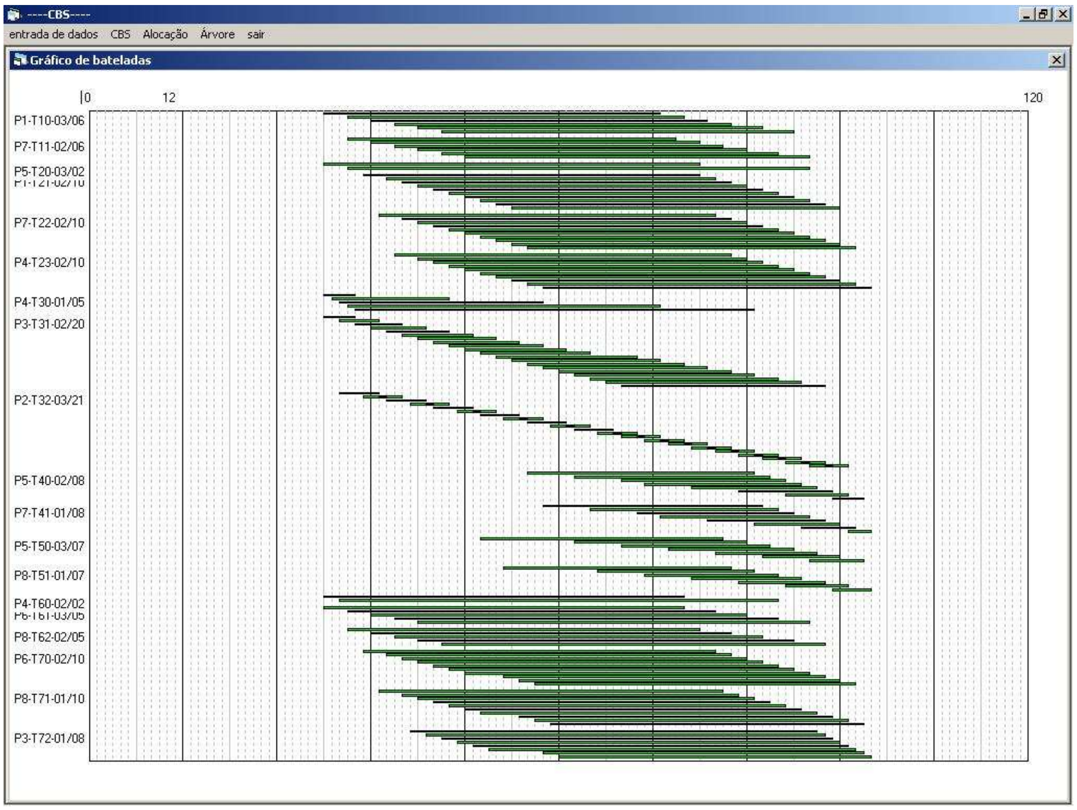


Figure 3 – Time windows at the root node

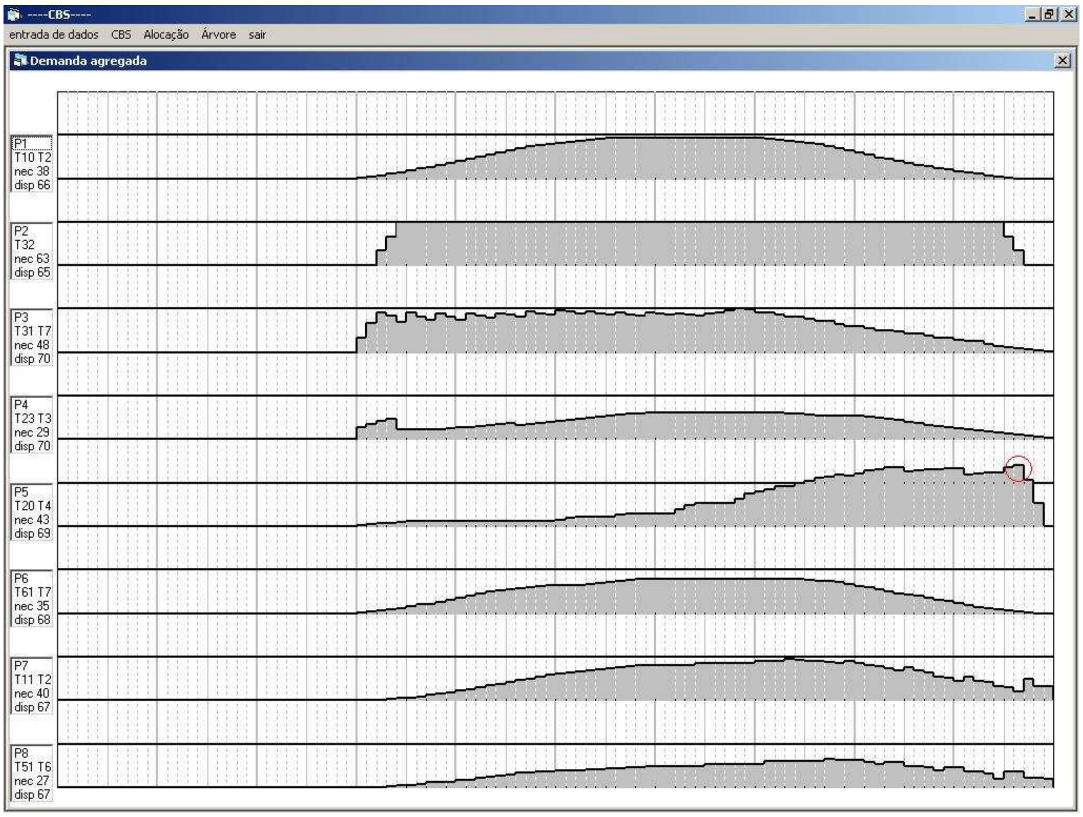


Figure 4 – Aggregate Demand graph of root node scenario

The disjunctive arcs that under the equipment loading criterion chosen (Aggregate Demand) contribute to the larger peak in the time horizon (circled spot at fig. 4). In this example, the pair (T50 7) – (that is, task 50; batch 7) – and (T40 7) was chosen, and the decision (T50 7) precedes (T40 7) was implemented.

A new time windows scenario is produced and is shown in Figure 5, after the propagation mechanism is launched.

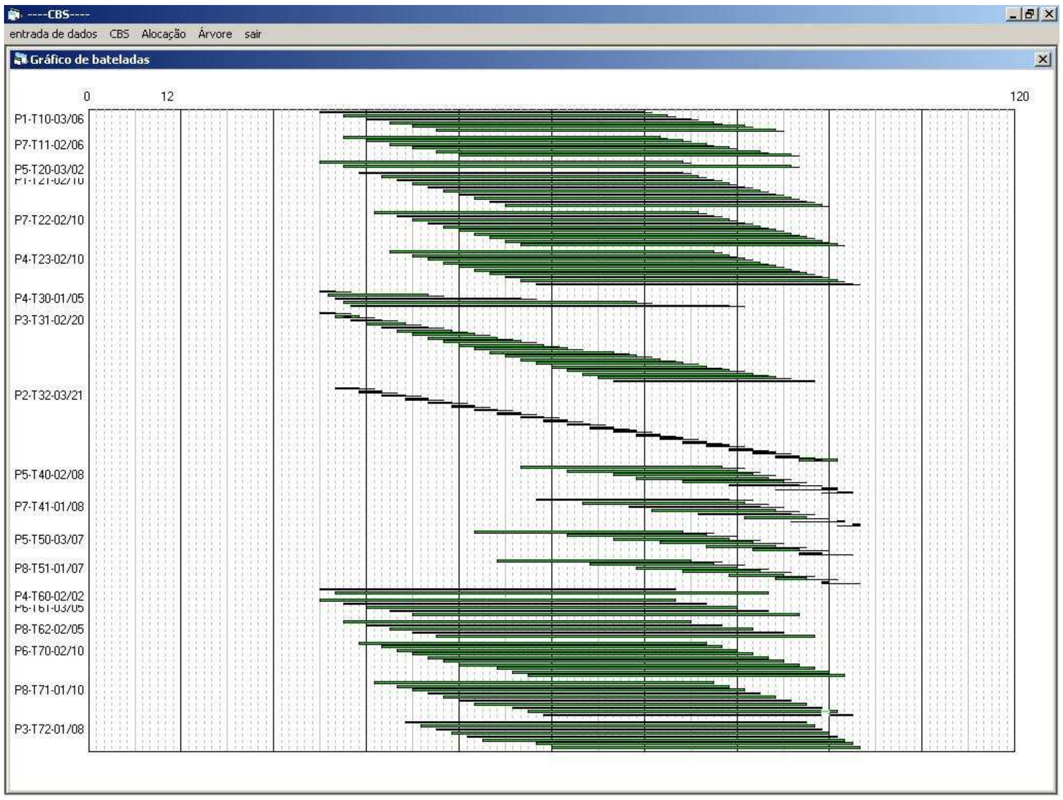


Figure 5 – Time windows configuration at node 1

A time windows reducing can be observed in Figure 5. The list of disjunctions is updated in order to proceed the selection of a new pair of tasks. The number of disjunctions falls from 602 at the root node to 578 at node 1.

In figure 6 is presented the Aggregate Demand graph of node 1 scenario, in which is possible to notice that, in terms of capacity analysis, the most critical bottleneck at the root node no longer exists at the node 1 due to the ordering imposed to tasks (T50 7) and (T40 7). The other shortenings in time windows are due to the constraint propagation mechanism that keeps the search space updated.

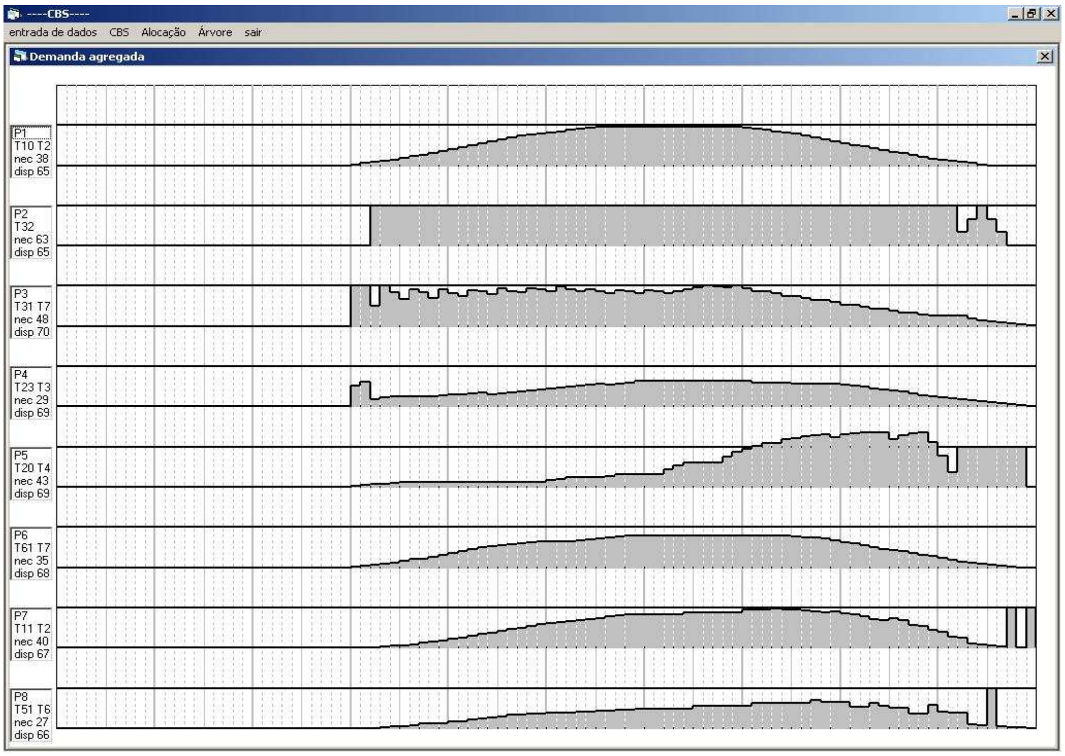


Figure 6 – Aggregate Demand graph of node 1 scenario

The final solution to this problem is shown in Figure 7. It corresponds to node 105. The search tree is shown in Figure 8.

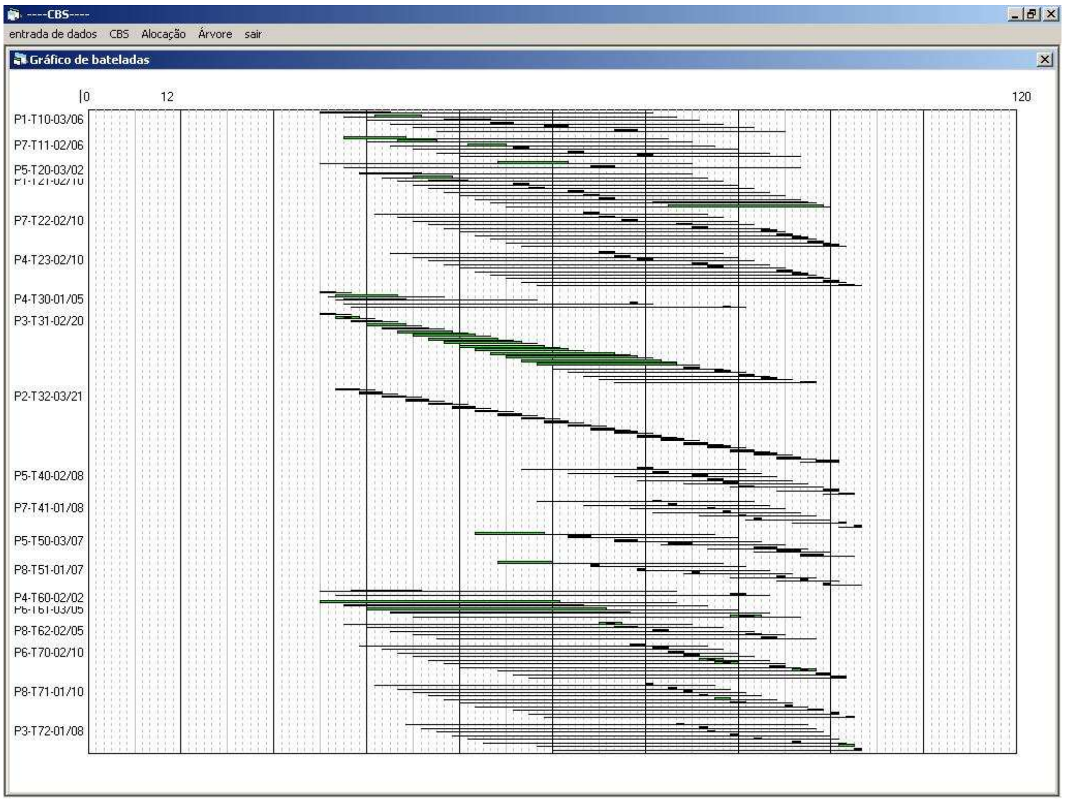
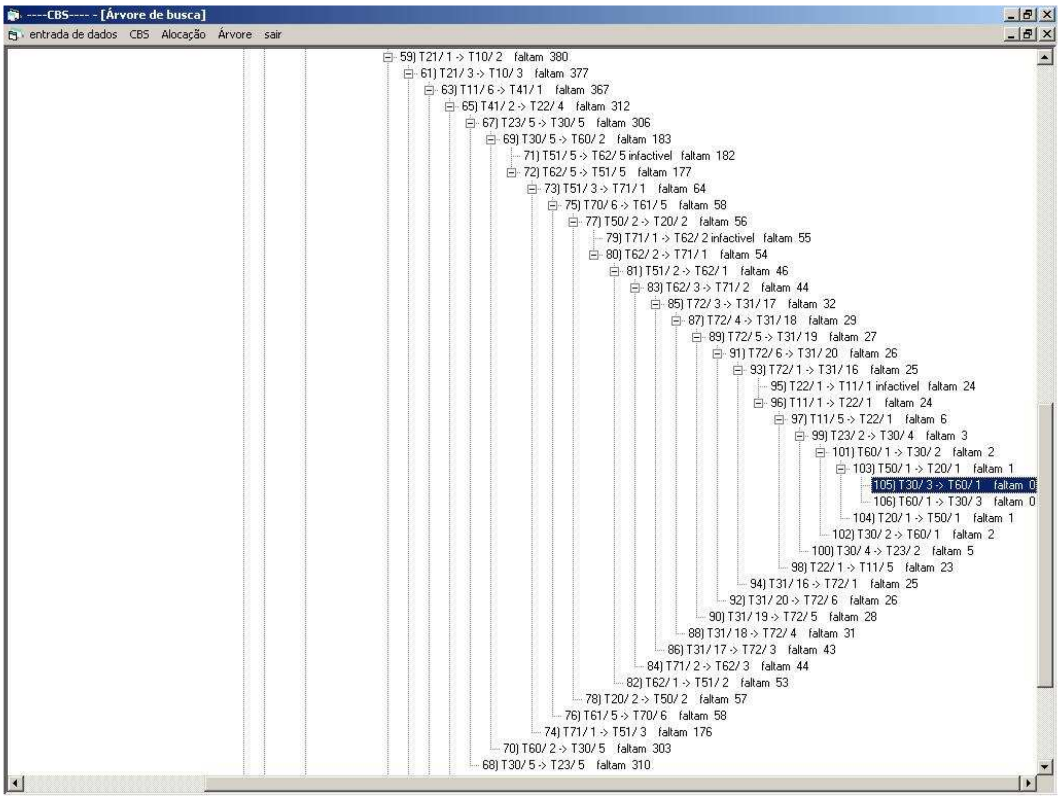


Figure 7 – Final Solution (node 105)


----CBS---- - [Árvore de busca]
entrada de dados CBS Alocação Árvore sair

```
59) T21/ 1 -> T10/ 2   faltam  380
  61) T21/ 3 -> T10/ 3   faltam  377
    63) T11/ 6 -> T41/ 1   faltam  367
      65) T41/ 2 -> T22/ 4   faltam  312
        67) T23/ 5 -> T30/ 5   faltam  306
          69) T30/ 5 -> T60/ 2   faltam  183
            71) T51/ 5 -> T62/ 5 infactivel  faltam  182
            72) T62/ 5 -> T51/ 5   faltam  177
              73) T51/ 3 -> T71/ 1   faltam  64
                75) T70/ 6 -> T61/ 5   faltam  58
                  77) T50/ 2 -> T20/ 2   faltam  56
                    79) T71/ 1 -> T62/ 2 infactivel  faltam  55
                    80) T62/ 2 -> T71/ 1   faltam  54
                      81) T51/ 2 -> T62/ 1   faltam  46
                        83) T62/ 3 -> T71/ 2   faltam  44
                          85) T72/ 3 -> T31/ 17   faltam  32
                            87) T72/ 4 -> T31/ 18   faltam  29
                              89) T72/ 5 -> T31/ 19   faltam  27
                                91) T72/ 6 -> T31/ 20   faltam  26
                                  93) T72/ 1 -> T31/ 16   faltam  25
                                    95) T22/ 1 -> T11/ 1 infactivel  faltam  24
                                    96) T11/ 1 -> T22/ 1   faltam  24
                                      97) T11/ 5 -> T22/ 1   faltam  6
                                        99) T23/ 2 -> T30/ 4   faltam  3
                                          101) T60/ 1 -> T30/ 2   faltam  2
                                            103) T50/ 1 -> T20/ 1   faltam  1
                                              105) T30/ 3 -> T60/ 1   faltam  0
                                              106) T60/ 1 -> T30/ 3   faltam  0
                                            104) T20/ 1 -> T50/ 1   faltam  1
                                          102) T30/ 2 -> T60/ 1   faltam  2
                                        100) T30/ 4 -> T23/ 2   faltam  5
                                      98) T22/ 1 -> T11/ 5   faltam  23
                                    94) T31/ 16 -> T72/ 1   faltam  25
                                  92) T31/ 20 -> T72/ 6   faltam  26
                                90) T31/ 19 -> T72/ 5   faltam  28
                              88) T31/ 18 -> T72/ 4   faltam  31
                            86) T31/ 17 -> T72/ 3   faltam  43
                          84) T71/ 2 -> T62/ 3   faltam  44
                        82) T62/ 1 -> T51/ 2   faltam  53
                  78) T20/ 2 -> T50/ 2   faltam  57
                76) T61/ 5 -> T70/ 6   faltam  58
              74) T71/ 1 -> T51/ 3   faltam  176
          70) T60/ 2 -> T30/ 5   faltam  303
        68) T30/ 5 -> T23/ 5   faltam  310
```

Figure 8 – Search tree

In the final scenario it can be seen that a set of batches are not yet allocated, but all the disjunctions have been analyzed in terms of ordering decisions. In general the capacity analysis done in each node is not complete since it implies that all the possible pairs must be investigated in terms of their impact on equipment capacity. Moreover, an optimal solution only makes sense if there is an optimality criterion to be pursued. Thus, the only concern one can have at this point is if the scheduling solution is feasible or not.

One aspect of the implemented backtracking mechanism can be seen at figure 8. There are some infeasible scenarios at the search tree (node 95, for example), in which the searching procedure backtracks to the opposite ordering decision that has been intended before the infeasible scenario was reached.

4. CONCLUSION

Constrained Based Search can be an important tool to solve completely or partially scheduling problems with a high competition among tasks. However one limiting boundary to its application is that there are only few affordable options of languages and software to support its implementation. In this paper it was discussed important aspects of the framework to develop a CBS approach, allowing user intervention during the search process. In our point of view this is the most important aspect to be kept in a CBS approach since it is very suitable to deal with problems where the constraints are very difficult to define or even implement.

REFERENCIES

CHENG C.C., SMITH S.F., 1993. “Slack-based heuristics for constraint satisfaction scheduling”. Proc. 11th National Conf. on Artificial Intelligence, Wash DC, 139-144.

ERSCHLER J., 1976. “Analyse sous contraintes et aide à la décision pour certains problèmes d’ordonnancement”. PHD Thesis, Université Paul Sabatier, Toulouse, France.

ILOG., 1997. “Scheduler 4.0 User’s Manual”, Mountain View, EUA.

KENG N.P., YUN D.Y.Y., ROSSI M., 1988. “Interaction sensitive planning system for job shop scheduling”. Expert systems and intelligent manufacturing, 57-69. Elsevier. Amsterdam, Netherlands.

KONDILI E., PANTELIDES C.C., SARGENT R.W.H., 1993. A general algorithm for short term scheduling of batch operations – I. MILP formulation. Computers and Chemical Engineering, Vol. 17 (2), 211-227.

PAPAGEORGIOU L.G., PANTELIDES C.C., 1996. “Optimal campaign planning/ scheduling of multipurpose batch/semicontinuous plants. 2. A mathematical decomposition approach”. Industrial Engineering and Chemical Research, Vol. 35, 510-529.

RODRIGUES M.T.M., LATRE L.G., RODRIGUES L.C.A., 2000a. “Short-term planning and scheduling in multipurpose batch chemical plants: A multi-level approach”. Computers and Chemical Engineering, Vol. 24, 2247-2258.

RODRIGUES M.T.M., LATRE L.G., RODRIGUES L.C.A., 2000b. “Production planning using time windows for short term multipurpose batch plants scheduling problems”. Industrial Engineering and Chemical Research, Vol. 39, 3823-3834.

SADEH N., 1991. “Look-ahead techniques for micro-opportunistic job shop scheduling”, PhD Thesis, School of Computer Science, Carnegie Mellon University.

RESPONSABILITY NOTICE

The authors are the only responsible for the printed material included in this paper.