Chronological Decomposition Heuristic for Scheduling: Divide and Conquer Method

Jeffrey Dean Kelly
Honeywell Hi-Spec Solutions, Toronto, Ontario, Canada, M2J 1S1

The chronological decomposition heuristic (CDH) is a simple time-based divide-and-conquer strategy intended to find rapidly, integer-feasible solutions to production scheduling optimization problems of a practical scale. It is not an exact algorithm in that it will not find the global optimum, although it does use either branch-and-bound or branch-and-cut. The CDH is specifically designed for production scheduling optimization problems found in the manufacture of petroleum distillates, petrochemicals, chemicals, and pharmaceuticals, which are formulated by discretizing the temporal dimension using a pre-specified time grid with fixed time-period spacing. However, the approach can be tailored easily to continuous-time formulations by properly slicing or cutting off any exogenous orders along the boundaries of the decomposition. Moreover, the CDH is intended to be used for those production scheduling problems which have inbound feedstock (supply), outbound product-stock (demand), and out-of-service equipment (maintenance) orders with specific release and due-dates occurring at any time within the scheduling horizon, that is, decisions must be made intermittently to receive raw materials and ship finished products.

The basic premise of the CDH is to chop the scheduling time horizon into aggregate time intervals or "time chunks," which are a multiple of the base time period. Each time chunk is solved using mixed-integer linear programming (MILP) techniques starting from the first time chunk and moving forward in time using the technique of chronological backtracking, if required (Marriott and Stuckey, 1998). (For more details, see the extensive literature on *constraint logic programming*.) The efficiency of the heuristic is that it decomposes the temporal dimension into smaller-size time chunks, which are solved in succession instead of solving one large problem over the entire scheduling horizon. The basic idea of such a decomposition strategy was partially presented in Bassett et al. (1996), whereby they provided a hierarchical interaction or collaboration between a planning layer and a temporally decomposed scheduling layer. For the CDH, we focus on the time-based decomposition of the scheduling layer without the need for a higher-level coordinating or planning layer.

For many industrial size problems, solving the MILP using commercial branch-and-bound or branch-and-cut optimization can be a somewhat futile exercise even for well-formulated problems of practical interest. Instead, many researchers such as Kudva et al. (1994), Wolsey (1998), Nott and Lee (1999), Blomer and Gunther (2000), and Kelly (2002) have devised elaborate primal heuristic techniques to enable the solution of problems of large scale and complexity; these techniques can also be augmented by other decomposition strategies such as Lagrangean and Bender's relaxation. Unfortunately, with these heuristics, global optimality or even global feasibility cannot be guaranteed; however, these methods and others not mentioned have proven useful for problems which are sometimes too large to be solved using conventional methods alone. Therefore, the CDH should be considered as a step in the direction of quickly aiding the scheduling user in finding integer-feasible solutions of reasonable quality.

Decomposition of Scheduling Time Horizon into Time Chunks with Cross Over

As mentioned, the CDH decomposes the scheduling time horizon, denoted as NTP, into NTC smaller time chunks of size NTPTC where, although not shown, the time chunks can be of nonuniform dimension depending on the problem class or instance. In this way it essentially mimics the way a human scheduler manually forms production schedules by ratcheting towards an overall solution one time chunk or time period at a time starting from the start of a schedule referred to as TPOFFSET + 1 in time periods; this is the first time period in the future. Typical of manual scheduling approaches using simulation (or spreadsheets), the scheduling heuristic procedure usually implemented by the user is to decide at each time period what decisions should be made in order to alleviate any quantity-type infeasibilities such as inventory underflows or overflows, and supply and demand order fulfillment excesses or deficits. In many cases, logic-type constraints and property-type stipulations concerning the operating details of the production and the quality of the products produced are often ignored or only mildly respected. To facilitate a better understanding of the decomposition, Figure 1 illustrates the time horizon for a six time chunk configuration.

The smaller dotted rectangles suffixed after the first five time chunks are the time periods denoted as cross over.

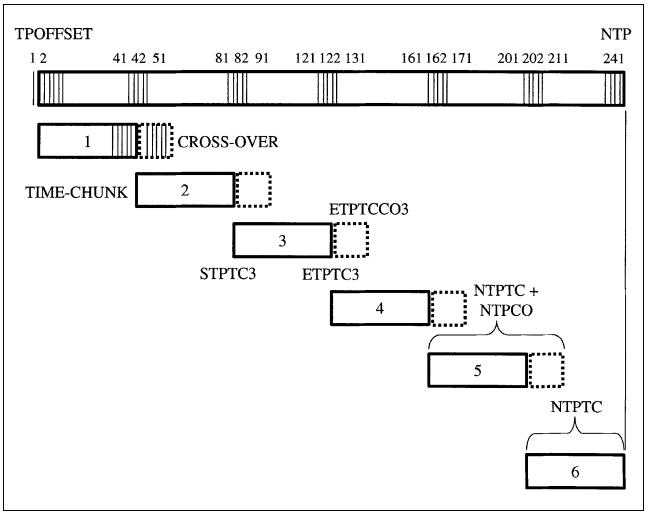

Figure 1. Scheduling time horizon (NTP) with six time chunks and cross-over.

Cross-over or time-period overlap is used to diminish the influence of the end-effects of truncating the MILP subproblem scheduling horizon (that is, so the end is less abrupt). It can be found in Wilkinson et al. (1994) and Bassett et al. (1996), but it was also used as a concept in the work of Baker and Goldmann (1970) to provide some look ahead beyond the current time period when determining feasible or suitable directions to follow. It is an option used in the CDH where it is possible for each time chunk to have a variable amount of cross-over. The last time chunk does not have any cross-over in order to match the scheduling horizon ending time period. It should also be stated that the notion of cross-over can also be found embedded in the fundamental idea of hierarchical production planning of Bitran and Hax (1977) who propose the use of a rolling window or horizon in the attempt to manage the scope, complexity, and, most of all, the uncertainty, of the problem.

The other variables shown in the figure are STPTC, ETPTC, and ETPTCCO. These are defined as the start of the time chunk in time periods, the end of the time chunk, and the end of the time chunk including the cross-over time periods, respectively. The time periods spanning NTPTC + NTPCO are included into the MILP subproblem solved for the first NTC—1 time chunks. The last time chunk is of size NTPTC plus any extra time periods that remain after dividing (NTP-TPOFFSET) by NTC. The initial or starting conditions such as inventory positions for the MILP for the next time chunk are given by the solution at ETPTC for the previous time chunk only; the cross-over time periods results are discarded.

Depth-First with Backtracking Search

When solving any MILP problem, it is expected to find more than one integer-feasible solution during the search for any given time chunk or scheduling horizon. We make use of these solutions in CDH to increase the possibility of finding better solutions, and to mitigate the possibility of encountering dead-ends or infeasibilities. The simplest form of a dead-end recovery strategy is called "chronological" or "last-in-first-out" backtracking, which consists of going back to the most recently instantiated time chunk node with at least one alternative left to branch on. Figure 2 shows our simple search heuristic for a five time chunk CDH slicing.

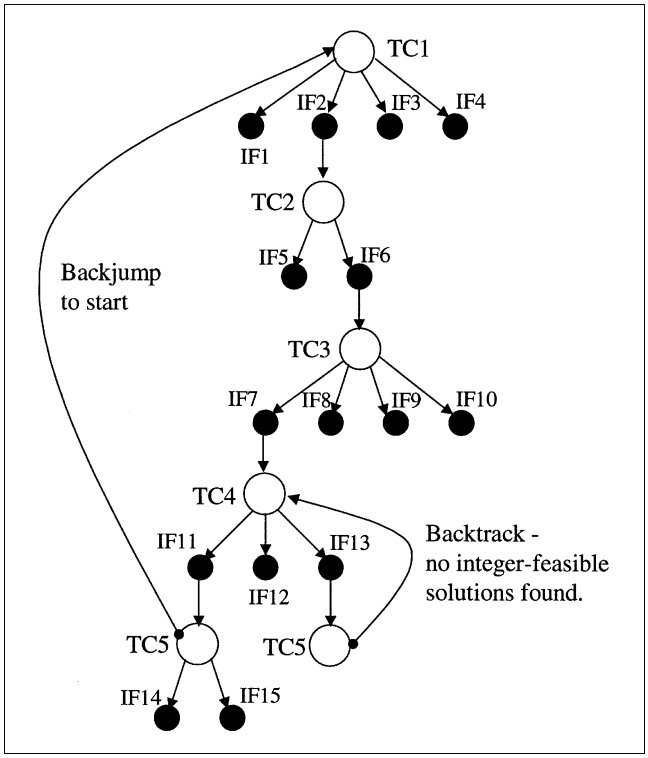

Figure 2. Depth-first with backtracking search tree with five time chunks.

The first time chunk or the TC1 node finds four integer-feasible solutions recorded as the IF1, IF2, IF3, and IF4 nodes. It then chose the IF2 node as the one to branch on for the time chunk TC2. The integer-feasible solution selection rule can be implemented in two ways. We can select the best objective function value, or we could have simply randomized the choice. The search then found two solutions for TC2, and IF6 was chosen. TC3 found four solutions, and IF7 was selected. TC4 found three solutions and IF13 was initially selected; however, it did not find any integer-feasible solutions due to either an integer-infeasibility or a time-out on the implicit enumerative search. The search then backtracked up to the next time chunk and chose IF11 as the next best alternative. If there were no alternatives left for node TC4, then the search would have backtracked to TC3 and so on. The CDH search would terminate without finding any global or overall integer-feasible solutions if no alternatives are left for the TC1 node. Fortunately, TC5 with parent IF11 found two solutions and the number of global solutions found is recorded as two with the solution path genealogy defined by the sequence of TC to IF node connections. Given that there are three more solutions available for the first time chunk, the CDH "backjumps" to the TC1 node and selects the next best integer-feasible solution remaining and continues the search

for other overall integer-feasible solutions perhaps of better or worse quality than the ones previously found. The term given for this type of simple search mechanism is "depth-first with backtracking," because it greedily proceeds down a branch of the search tree until an overall solution is found for the entire time horizon or when infeasibilities or time-outs cause it to fathom or terminate the branch. The number of overall integer-feasible solutions that the CDH can find may be greater than the number of solutions found for the first time chunk due to the fact that the last, or NTC, time chunk may find more than one solution as well. However, this number may be decremented if backtracking is required for the reasons noted earlier. An interesting offshoot of the CDH search is its ability to be used in a parallelization environment. It is very easy to configure different CDH parameters on several different computers linked together over a local-area-network or even the internet. Suggested parameter setting variations would be to change the number of time chunks NTC, the number of cross-over time periods, and to use a randomized branching selection rule instead of, for example, the best first rule. The end goal is to generate as many global or overall integer-feasible solutions in as short an amount of time as possible from which the scheduling user would choose the one which aligns best with the current production scenario. Moreover, for the production scheduling problems encountered in the hydrocarbon processing industries such as in oil refineries and petrochemical complexes, the quality aspects or the "intensive" variables of the production such as sulfur and octane levels must then be taken into consideration after the quantity and logic aspects have been accounted for. This implies that even the best quantity-logic solutions found by the CDH may not yield the best quantity-quality production solution.

Numerical Example

With the purpose of elaborating more clearly the efficiency of the chronological decomposition heuristic, a relatively small crude-oil *blendshop* scheduling optimization problem is presented. This production scheduling example models the situation where intermittent supply orders of crude oils are being sent to an oil refinery whereby continuous demands of crude oil mixtures are required to charge an atmospheric and vacuum distillation pipestill in a particular operating or production mode as depicted in Figure 3. The modeling of the system is described as a *fixed-charge network flow* problem with binary or 0-1 logic variables used to represent flow or no flow between the equipment with aspects of the *capacitated lot-sizing* problem also embedded in the formulation concerning the tank material balance modeling. Several detailed logic

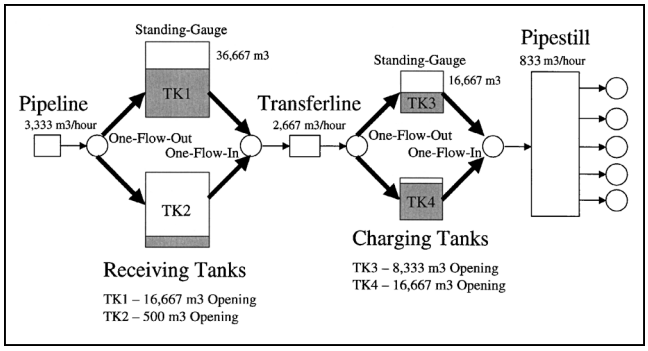

Figure 3. Small crude oil blendshop network.
Bold arrows are logic move decision variables.

or operating rule constraints are posed such as standing-gauge (mutually exclusive flow in and out of a tank for any given time period); one-flow-in and one-flow-out at a time for a piece of equipment are included in order to properly represent the details of the blendshop operation, as well as other constraints and variables imposed to model the startup and shutdown of any move or movement occurring between any two pieces of equipment (used to model temporal and spatial switchovers/changeovers). The Archimedean-type objective function for this optimization problem is composed of three terms: profit, performance, and penalties. The profit term is used to maximize the difference between demand revenues and supply costs including any inventory holding costs. The performance term is used to minimize the number of movements or active logic variables and to minimize transitions. The penalty term is used to minimize any artificial variable associated with a hard quantity or logic constraint that must be softened in order to present the scheduling user with some form of a solution when infeasibilities exist. More details of this formulation can be also found in Lee et al. (1996), Shah (1996), and Jia et al. (2002).

Specifically we have for this example a 3,333 m^3/h capacity pipeline delivering four different types of crude oils at non-overlapping times which can be deposited into only one of the two crude-oil receiving tanks, as detailed in Table 1. The designation one-flow-out, as mentioned, shown beside the circle after the pipeline is a logic constraint enforcing that simultaneous flow out of the pipeline to the two tanks is not allowed. The two receiving or storage tanks have upper capacities shown of 36,667 m^3, and both have standing-gauge logic constraints indicating that they can have only flow in or flow out, but not both, within the same time period. There is an intermediate 2,667 m^3/h transfer line of negligible holdup

Table 1. Receipt or Supply Orders of Crude Oil Over Horizon

Crude Oil	Start Time Period	End Time Period	Flow Rate	Duration	Flow	Valid Destination
1	82	91	3,333 m^3/h	10 h	33,333 m^3	TK2
1	163	172	3,333	10	33,333	TK2
2	7	16	3,333	10	33,333	TK2
3	43	52	3,333	10	33,333	TK1
3	221	230	3,333	10	33,333	TK1
4	132	141	3,333	10	33,333	TK1

Table 2. Problem Statistics Including Presolve Results as the Last Row

No. Rows	No. Columns	No. Nonzeros	No. Var.	No. SOS1	Root LP OBJ Value
18,850	26,481	68,644	1,500	3,000	−280.7617
7,080	10,736	22,432	—	—	—

which can only receive crude oil from one receiving tank and can only discharge crude oil to one charging tank. There are two charging tanks of upper capacity 16,667 m^3, which also run in standing-gauge and must charge 833 m^3/h continuously to the oil refinery pipestill or atmospheric distillation production unit from only one tank at a time; the logistic requirement is not imposed that the charging tank needs to be empty before switching over to the other charging tank. The eight bolded arrows shown are the movement logic decisions variables which have associated with them semi-continuous flow variables, and there is transition or switch-over minimization for the binary move variables modeled, as can be found on pp. 232–236 of Wolsey (1998), for example. The scheduling time horizon NTP is 241 1-h time periods (a 10-day horizon) with a TPOFFSET equaling a one time-period to account for the opening inventories and movement logic. The basic multiperiod flow modeling of the example uses appropriate reformulation, disaggregating, and tightening techniques also found in Wolsey (1998), as well as temporal directives or priorities applied to the binary variables and special-ordered-sets one (SOS1).

Table 2 provides the necessary problem statistics for the overall case without time chunking (that is, corresponds to a NTC = 1) where the root LP OBJ value column states the maximized objective function value with the binary variables being relaxed between zero and one. The last row of the table shows the number of rows, columns, and non-zeros left after the technique known as *presolve* has been performed (Dash Optimization Inc., 2001). The problem size of the individual time chunk plus cross-over is proportional to (NTPTC + NTPCO)/NTP. All experiments were programmed using *Xpress-Mosel* from Dash Optimization Inc. and were performed on a Pentium III 750 MHz laptop computer.

The CDH results are shown in Table 3. The first row of the table records the results of the branch-and-cut without time chunking and cross-over. It took 55 s to find the first integer-feasible solution, which was proved to be globally optimal in 5 additional s. The model seems to be well-formulated, as indicated by the small integrality gap between the root LP objective function value of −280.7617 and the global optimum of −280.8000. Five cases were performed with an arbitrary cross-over amount of 12 time periods or one-half of a day of look-ahead where no backtracking was required for any of the cases during the CDH search. As is apparent, the heuristic finds the optimal solution for NTCs of 2 and 10 where NTCs of 4 and 6 are near optimal. The NTC of 8 is less fortunate given that penalties or artificial variables were activated for the problem solution found and is declared as being integer-feasible primal infeasible. The efficiency of the CDH is clearly demonstrated given that five integer-feasible solutions of comparable quality were found in 50 s compared to the one found in 55 s for NTC = 1. In addition a second example, not shown, was also tested with a less logic constraint complexity, but with a more spatial scope (that is, more crude oils and tanks) was solved where we obtained similar computational performance and solution results.

Conclusion

This article presents a simple and effective way to divide the temporal dimension of certain production scheduling optimization problems into smaller, more manageable time chunks which can be solved and brought together to form overall or globally integer-feasible solutions using a depth-first search. These time chunks are augmented with user specified cross-over time periods in order to help dampen the effects of the scheduling end conditions enabling more look-ahead or far-sightedness to be added to the search. The CDH is also able to handle other more complicated logic-type constraints such as mixing delays (a time lag between flows in and flows out of a tank) and production run-lengths with straightforward modifications. These require extra time periods to be included at the start of each time chunk and also require the minimum number of cross-over time periods to be at least as large as the maximum mixing delay and run-length. Although guidelines or rules-of-thumb for determining the number of time chunks have not been explicitly provided, it does seem reasonable to choose the number based on how quickly one of the sub-problems can be solved including cross-over. For very large problems with many time periods, and a considerably large number of time chunks may be necessary in order to solve in reasonable time just one sub-problem where it is expected that the CDH computation time

Table 3. Results of the CDH for 1, 2, 4, 6, 8 and 10 Time Chunks Using Branch-and-Cut

NTC	NTPT	NTPCO	Backtracked?	No. LP Nodes	No. Seconds	No. I-F Sols	Best MILP OBJ
1	240	0	N/A	555	55	1	−280.8000
2	120	12	No	872	28	2	−280.8000
4	60	12	No	345	8	1	−281.0000
6	40	12	No	388	7	1	−281.0000
8	30	12	No	511	10	1	−480.0000
10	24	12	No	357	7	1	−280.8000

extrapolates linearly by the number of time chunks configured. Similarly, for the number of cross-over time periods, its choice must trade-off computational speed vs. obtaining heuristically better solutions as the cross-over length increases enabling more look-ahead. It should also be mentioned that for plants which operate as jobshop or flowshop-type processes with many production stages where a finished product is made typically at the end of the scheduling horizon, the CDH becomes less appropriate if the due-dates for product shipping are greater than the scheduling horizon. For these plants, the time chunks boundaries must be carefully timed to include an adequate number of finished product due-dates and the scheduling horizon must be of sufficient duration to include a reasonable production context. Finally, the CDH can also be used to provide a reasonable or *approximate* lower-bound (for maximization problems) on the objective function, when, for example, a post branch-and-bound implicit-enumeration search is performed to see if the CDH solution can be improved further.

Literature Cited

Baker, T. E., and S. F. Goldmann, "DYAL—A New Computer Tool for Operations Scheduling," *Esso Mathematics & System Inc.*, Report No. EMS. 31M70 (Aug., 1970).

Bassett, M. H., J. F. Pekny, and G. V. Reklaitis, "Decomposition Techniques for the Solution of Large-Scale Scheduling Problems," *AIChE J.*, **42**, 3373 (1996).

Bitran, G. R., and A. C. Hax, "On the Design of Hierarchical Production Planning," *Decision Sci.*, **8**, 28 (1977).

Blomer, F., and H.-O. Gunther, "LP-Based Heuristics for Scheduling Chemical Batch Processes," *Int. J. of Production Res.*, **38**, 1029 (2000).

Dash Optimization Inc., *Xpress-Optimizer Reference Manual*, Release **13** (2001).

Jia, Z., M. Ierapetritou, and J. D. Kelly, "Refinery Short-Term Scheduling Using Continuous-Time Formulation—Crude-Oil Operations," *Ind. Eng. Chemistry Res.*, in press (Feb., 2002).

Lee, H., J. M. Pinto, I. E. Grossmann, and S. Park, "Mixed-Integer Linear Programming Model for Refinery Short-Term Scheduling of the Crude Oil Unloading with Inventory Management," *Ind. Eng. Chemistry Res.*, **35**, 1630 (1996).

Kelly, J. D., "Smooth-and-Dive Accelerator: A Pre-Milp Primal Heuristic Applied to Production Scheduling Problems," *Comput. Chem. Eng*, in press (2002).

Kudva, G., A. Elkamel, J. F. Pekny, and G. V. Reklaitis, "Heuristic Algorithm for Scheduling Batch and Semi-Continuous Plants with Production Deadlines, Intermediate Storage Limitations and Equipment Change-Over Costs," *Comput. Chem. Eng.*, **18**(9), 859 (1994).

Marriott, K., and P. J. Stuckey, *Programming with Constraints: An Introduction*, MIT Press, Cambridge, MA (1998).

Nott, H. P., and P. L. Lee, "An Optimal Control Approach for Scheduling Mixed Batch/Continuous Process Plants with Variable Cycle Time," *Comput. Chem. Eng.*, **23**, 907 (1999).

Shah, N., "Mathematical Programming Techniques for Crude Oil Scheduling," *Comput. Chem. Eng.*, **20**, Suppl. B, S1227 (1996).

Wilkinson, S. J., N. Shah, and C. C. Pantelides, "Scheduling of Multisite Flexible Production Systems," AIChE Meeting, San Francisco (1994).

Wolsey, L. A., *Integer Programming*, Wiley, New York (1998).

Manuscript received Mar. 15, 2002, and revision received July 10, 2002.