2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)
Munich, Germany, August 20-24, 2018

Enhanced branch and bound approach for receding horizon based planning

Michael Jäntsch, Naresh N. Nandola, Li Wang, Mathias Hakenberg and Ulrich Münz

Abstract— **In this work, we present an efficient planning algorithm for flexible manufacturing industries. In particular, we modified a traditional branch and bound approach to be used in a receding horizon manner by adopting the terminal cost concept from model predictive control domain. Thus, the proposed algorithm combines best practices from traditional planning and scheduling as well as from process control. The efficacy of the proposed algorithm is demonstrated on job shop problems of different sizes. Results are compared with traditional branch and bound based planning. The initial results are encouraging and demonstrate superior performance as well as scalability for large problems.**

I. Introduction

In today's world, scheduling and planning is a key for manufacturing industries to remain competitive and profitable. In manufacturing industries, planning and scheduling is seen as decision-making process, where scheduling only considers the allocation of resources while planning also considers matching specific task and resource skills [1]. Thus, scheduling and planning deal with efficient utilization of available resources to complete a given set of tasks in the most cost effective manner. Typically, a manufacturing process involves multiple number of tasks that have to be assigned to available machines based on their capabilities. These tasks are scheduled such that the desired objective, e.g., minimizing makespan, minimizing due date violation, etc., is met.

Numerous algorithms for optimal planning and scheduling are proposed by researchers from various areas. This has been an active topic of research since last couple of decades and still evolving. Kondil et al. [2] have proposed a generic State-Task-Network (STN) framework for planning and scheduling. The STN is a graphical representation with two types of nodes with different characteristics: (i) state node (ii) task node. The state node represents various process states such as final product, intermediate product, etc. while the task node represents various operations such as milling, drilling, etc. The STN framework is extended to Resource-Task-Network (RTN) by Pantelides [3] where the distinction between nodes is relaxed and assumed that each task transforms a set of resources to another set of resources. Thus, making uniform description for all available resources. It also captures the interaction of tasks with multiple resources. The RTN and STN results in a complex Mixed Integer Linear Program (MILP), which is computationally very expensive and requires sophisticated solvers for reliability. Another popular approach for the planning and scheduling in the manufacturing industries is the so called branch and bound method [4], [5]. The branch and bound approach works on a tree evolution principle where each branch is evaluated against specific criteria, e.g. minimize production time, and progresses in the direction which satisfies this criteria. Thus, at the end of complete tree-evolution, the optimal plan and schedule is generated. The branch and bound is relatively easy to implement but results in a combinatorial problem that may become intractable for large problem. In order to increase computational efficiency, Potocnik et al. [6] used reachability analysis from control theory to eliminate infeasible schedules (or combinations) and proposed optimal depth of the branch and bound tree.

There have also been efforts to develop computationally efficient framework for the planning and scheduling. Dimitriadis et al. [7] proposed RTN-based rolling horizon approach where the planning horizon is divided into a detailed time block (DTB) and an aggregate time block (ATB). The discrete time RTN [3] is considered for the DTB while an aggregate RTN [8] is applied for ATB and the resulting MILP is solved for the optimal solution. The algorithm starts with the small DTB and fixes some of the variables in DTB once the optimal solution is found. In the next step, DTB and ATB are increased by a number of time periods. This process is repeated until entire schedule is generated. Recently, Li et al. [9] used similar concept to solve a planning and scheduling problem that also include production capacity. Moreover, they applied heuristic process network decomposition techniques to achieve computational reduction. In yet another effort, reduction in computation time is achieved by formulating a rolling horizon based MILP problem to get the optimal solution for multiple urban energy hub system [10].

Aforementioned algorithms consider planning and scheduling as an offline process and generate the optimal schedule for the entire planning horizon before process starts. For most of the cases this requires the solution of an MILP problem, which often becomes computationally intractable for large planning problems. On the other hand, uncertainties, such as order cancellation, machine failure, uncertain execution times, etc. are hallmark of any real world application, which makes rescheduling inevitable. This requires planning and scheduling in parallel with plan execution to accommodate any unforeseen disruptions and

M. Jäntsch, N. N. Nandola, M. Hakenberg and U. Münz are with the Autonomous Systems and Control Research Group in Siemens Corporate Technology, Princeton, NJ, USA. L. Wang is a graduate student at the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332 and did his internship with Siemens Corporate Technology, Princeton, NJ, USA.
jaentsch.michael@siemens.com

react accordingly. In other words, it demands active planning and scheduling at each time step or whenever a deviation between planned and executed schedule is detected. As a consequence, there is a need for computationally efficient algorithms that produce acceptable schedules (which are close to optimal if not the optimal one) quick enough to implement in online fashion. Moreover, it should be scalable to large plants and robust to react to changes. We address these issues in this work by borrowing the receding horizon and terminal cost concept from Model Predictive Control (MPC) literature [11] and propose an enhanced branch and bound approach with a shorter planning horizon and the introduction of estimated cost-to-go values. This estimated cost-to-go considers all unplanned activities and captures approximate cost beyond the finite planning horizon of h-step (which is significantly lower than the full planning horizon). Thus, a relatively small branch and bound problem is solved at each time step from which only the first few decisions are implemented on the plant. In this approach, the approximate cost-to-go values play an important role and facilitate: (i) tree exploration by considering costs beyond the limited horizon, (ii) decision on which states will be explored first within the planning horizon, and (iii) avoiding exploration of decision branches with inferior solutions.

The paper is organized as follows: Section II briefly gives and overview over concepts of the traditional branch and bound approach. The proposed receding horizon based branch and bound approach as well as the concept of estimated cost-to-go is discussed in Section III. Section IV demonstrates effectiveness of the proposed approach on job shop problems with varying size. Section V summaries the proposed work and future directions.

II. Overview of Branch and Bound Algorithm

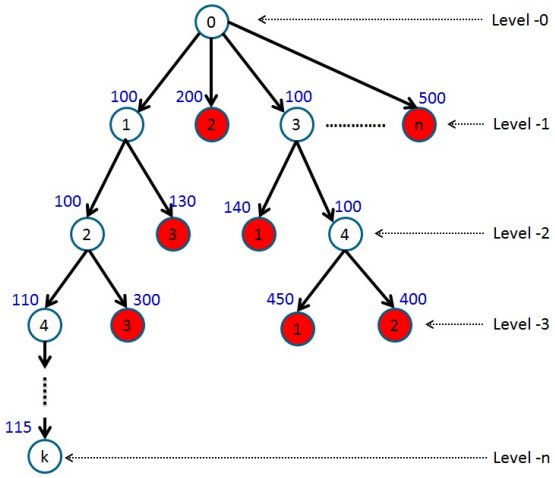

Fig. 1. Traditional branch and bound approach.

This section summarizes the traditional branch and bound approach using an illustrative example. Fig. 1 demonstrates a graphical representation of the classical branch and bound approach where P products consisting of n tasks have to be scheduled on a single machine such that overall due date violation is minimized. As shown in Fig. 1, the branch and bound enumerates nodes of a tree at each level. In this figure, the number inside the circle represents the label of the assigned task while number outside the circle represents the objective function values at the node. In the beginning at Level-0 no task is assigned. At Level-1, the objective function (e.g. worst case violation of due date) is calculated if particular task is assigned to the machine. As shown in Fig. 1, assignment of Task-1 and Task-3 results in lowest objective function value (100), while in all other cases the objective function value is higher. Thus, at Level-2 only two nodes, i.e., sequence starting with Task-1 and Task-3, are considered for further evaluation to assign remaining tasks and all other possibilities are discarded, i.e., those branches are fathomed. Similarly, at Level-2, sequence $(1, 2, \ldots)$ and sequence $(3, 4, \ldots)$ yield lowest objective function values. Hence, for the next step only these branches are considered for further evaluation. This process is repeated until all tasks are scheduled. Consequently, it results in a tree with n levels. At the end of the tree evaluation an optimal sequence of task is generated, which is $(1, 2, 4, \ldots, k)$ for this example. Here it should be noted that, if at any level the objective function value of the current branch goes beyond the value of earlier fathomed node then that particular node should be added to the set of active nodes and considered for further evaluation. For example, if the objective function value for sequence $(1, 2, 4, \ldots)$ goes beyond 130 at some level than the sequence $(1, 3, \ldots)$ would become active, which otherwise was fathomed at level-2. In summary, branch and bound algorithm consist of two stages: (i) a branching stage that enumerates all the possible scenarios at given level based on predefined objective (ii) bounding stage that systematically fathoms undesirable branches based on certain objective function. This process is repeated at each level and proceeds towards the most economical direction characterized by objective function till all task assignments are completed, i.e., when entire schedule is generated.

Thus, a branch and bound is a combinatorial problem with intense computational requirement, which increases exponentially with the number of tasks. It becomes more complex in practice, where a typical job shop problem consists of m machines and n tasks to be assigned among all machines. Moreover, computational efficiency of this approach relies on branching strategies and fathoming criteria to a large extent. There is a number of branching strategies and heuristics available for efficient branching [1], [12]–[14]. However, a well formulated branch and bound algorithm generates only active schedules [1], which contains optimal schedule. As per [4], the following simple criteria is used to pick machines, i.e., branching, for a particular task at each stage:

$$C^* = \min_{\substack{1 \le i \le n, \\ 1 \le j \le m}} (r_{ij} + s_{ij}) \tag{1}$$

where m and n stands for number of machines and tasks, respectively, r_{ij} and s_{ij} denotes possible start time and processing time of task i on machine j, C^* is optimal

objective function value for optimization problem (1) and j^* is selected machine (which is obtained by solving problem (1)). Thus, (1) gives a combination of the machine and task whose next processing step can be completed at the earliest possible moment. In next step, a partial schedule is generated by selecting one task on machine j^* from set of all task that can be performed on it such that it satisfies the following criteria:

$$r_{ij^*} < C^*, \quad \forall i \tag{2}$$

These steps are repeated until all the tasks are assigned to particular machine. As discussed earlier, certain nodes are fathomed without further evaluation based on predefined criteria. Many heuristics such as the disjunctive programming method, shifting bottleneck method, etc. are developed for fathoming. For more details on these heuristics, readers are refereed to [1], [14].

III. Cost-to-go estimation augmented branch and bound for receding horizon planning

As discussed in the previous section, existing branch and bound approaches calculate the entire schedule in one shot and suffer from the curse of dimensionality such that it may lead to computationally intractable problems for large manufacturing units. Therefore, in this work, a receding horizon based branch and bound approach for flexible manufacturing and planning is proposed. In particular, we use a fixed horizon h for planning instead of looking until end of the plan (i.e. up to depth n). And at each stage, only the first few tasks are implemented and again a plan is calculated h steps ahead after updating the current state. This process is repeated until all tasks are assigned and entire schedule is generated. Thus, the overall computational effort is significantly reduced due to a shorter planning horizon of the branch and bound algorithm. However, without any modification in the existing branch and bound approach, there is a risk of generating schedules that may look optimal in the beginning for a few steps but later would lead to an extremely suboptimal schedule, as only a short planning horizon is considered. To avoid such a scenario, inspired by the terminal cost concept in model predictive control literature, we introduce "cost-to-go" function in the objective function used for fathoming criteria while using a similar branching rule as explained in the previous section. The cost-to-go function is an estimation of the remaining cost that would incur if a particular task is assigned to a selected node at the current step. As a consequence, each node in a particular layer is associated with two costs: (i) actual cost due to already assigned tasks in the current branch (ii) estimated future cost (i.e., cost-to-go) to complete the entire schedule if particular branch is selected for the specific task. Total cost is considered as summation of these two cost components. Thus, due to inclusion of an additional "cost-to-go" penalty in the objective function, the solution always progresses towards the optimal even when considering a smaller planning horizon.

Accuracy and success of the proposed approach rely on the accuracy of the cost-to-go value. There may be many ways to define cost-to-go estimation. In this work, we consider cost-to-go from product perspective, as well as from machine perspective and the maximum among these is considered as the estimated cost-to-go value. The cost-to-go from the product perspective can be calculated as maximum completion time among all products. Thus, cost-to-go calculation from product perspective (CTG_P) is easy to calculate. On the other hand, cost-to-go from the machine perspective (CTG_M) can be thought as minimum of the maximum makespan of all machines, which leads to solving multiple complex equations simultaneously. Size and complexity of these equations depend on the number of available machines and products. Finally, effective cost-to-go is calculated as follows:

$$CTG_e = \max(CTG_P,\ CTG_M) \tag{3}$$

The estimated cost-to-go obtained from Eq.(3) is considered as a termination criteria for the h-step ahead horizon branch and bound, which is solved at each stage in receding horizon manner. The size of the planning horizon h is a tuning parameter of the algorithm. The next section demonstrates the effectiveness of our proposed algorithm.

IV. case study

This section documents an application of the proposed receding horizon planning (RHP) approach. In order to demonstrate effectiveness of RHP, we consider job shop scheduling problems of different sizes between 6 and 20, where size x means x machines and x products with each product consisting of x tasks. The results are compared with the traditional branch and bound approach in terms of objective function values (see Fig. 2) and computational time (See Fig. 3). The branch and bound approach converged to the optimal solutions for problem sizes up to 9 within computation time of 15min while for problem size larger than 9, it did not converge in reasonable time hence solver is terminated randomly between 9 to 11 min with suboptimal solution (i.e., when objective function value is comparable to the corresponding RHP solution). In the figures, blue circle indicates branch and bound with the optimal solution, red diamond indicates branch and bound with the suboptimal solution due to early termination and black + indicates corresponding solution using the proposed RHP approach.

Figure 2 documents objective function values for different size of the problem using branch and bound (blue circle and red diamond) and RHP (black +) approach. From the figure it can be seen that the quality of results using our proposed RHP is comparable with the results a using traditional branch and bound approach for the problem of all sizes considered in this study. Although in some cases the traditional branch and bound produces marginally better results, the RHP approach outperforms branch and bound significantly in terms of the computation time. Figure 3 documents comparison of computational time using branch and bound (blue circle

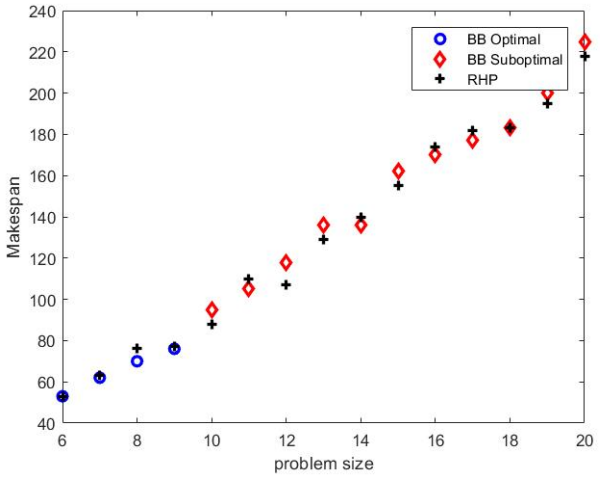

Fig. 2. Objective function comparison between traditional Branch and Bound (blue circle and red diamond) and RHP (black +). Branch and Bound (BB) is terminated at suboptimal stage for the problem size 10 and more (red diamond).

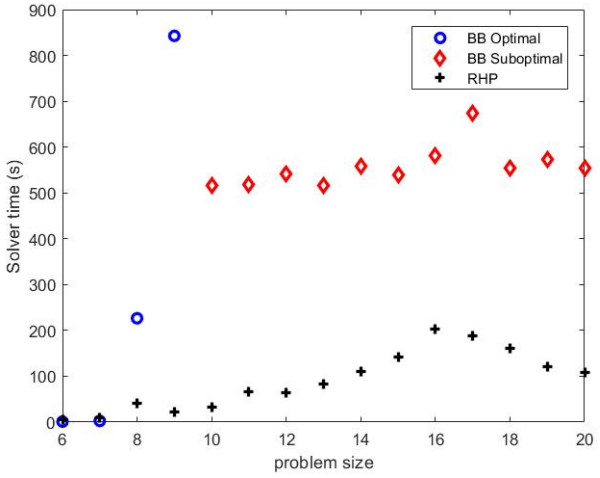

Fig. 3. Computation time comparison between traditional Branch and Bound (blue circle and red diamond) and RHP (black +). Branch and Bound (BB) is terminated at suboptimal stage for the problem size 10 and more (red diamond).

and red diamond) and RHP (black +). It can be seen that computation time for RHP is significantly less than the corresponding solution using branch and bound. Further, by looking at problem sizes up to 9 (where branch and bound converged to the optimal and denoted by blue circle in the figures), it can be seen that the computational time increases exponentially with problem size in case of the branch and bound approach while for RHP it increases marginally. Thus, the proposed RHP approach is scalable for large problem sizes, which is very critical for its practical use.

V. Summary and Future Work

In this paper we proposed a computationally efficient receding horizon based branch and bound approach for flexible manufacturing planning. The effectiveness of the proposed algorithm is demonstrated on the jobshop problem and results are compared to a traditional branch and bound approach. From the results it can be concluded that the proposed approach provides marginally suboptimal solutions as compared to the branch and bound approach. However, it is scalable in terms of computational time, which remains almost the same when increasing the problem size while traditional branch and bound shows exponential increase in the computation time. The success of the proposed approach is greatly relies on the design of cost-to-go function. A bad design of cost-to-go function would lead to a solution which looks optimal in the beginning but later, after implementing a few steps, it may even lead to infeasible solutions. This is called a dead-lock situation. In such a scenario entire manufacturing unit would go for a toss and may lead to huge economical loss. Our future work involves optimal design of the cost-to-go function to avoid such a situation for various other scheduling application such as batch scheduling in process industries.

References

[1] M. Pinedo, *Scheduling: Theory, Algorithms, and Systems*. New York: Springer, 2012.

[2] E. Kondili, C. Pantelides, and R. Sargent, "A general algorithm for short-term scheduling of batch operationsi. milp formulation," *Computers & Chemical Engineering*, vol. 17, no. 2, pp. 211 – 227, 1993.

[3] C. Pantelides, "Undefined frameworks for optimal process planning and scheduling," in *Proc. Second International Conference of Foundations of Computer-Aided Process Operations*, 1994, pp. 253–274.

[4] B. Giffler and G. L. Thompson, "Algorithms for solving production-scheduling problems," *Operations research*, vol. 8.4, pp. 487–503, 1960.

[5] P. Y. Gan and K. S. Lee, "Scheduling of flexible-sequenced process plans in a mould manufacturing shop," *International Journal of Advanced Manufacturing Technology*, vol. 20, no. 3, pp. 214–222, 2002.

[6] B. Potočnik, A. Bemporad, F. D. Torrisi, G. Mušič, and B. Zupančič, "Hybrid modelling and optimal control of a Multiproduct Batch Plant," *Control Engineering Practice*, vol. 12, no. 9, pp. 1127–1137, 2004.

[7] A. Dimitriadis, N. Shah, and C. Pantelides, "Rtn-based rolling horizon algorithms for medium term scheduling of multipurpose plants," *Computers & Chemical Engineering*, vol. 21, pp. S1061 – S1066, 1997, supplement to Computers and Chemical Engineering.

[8] S. J. Wilkinson, *Aggregate formulations for large-scale process scheduling problems*. PhD thesis, Imperial College London, 1996.

[9] Z. Li and M. G. Ierapetritou, "Rolling horizon based planning and scheduling integration with production capacity consideration," *Chemical Engineering Science*, vol. 65, no. 22, pp. 5887 – 5900, 2010.

[10] J. F. Marquant, R. Evins, and J. Carmeliet, "Reducing computation time with a rolling horizon approach applied to a milp formulation of multiple urban energy hub system," *Procedia Computer Science*, vol. 51, pp. 2137 – 2146, 2015, international Conference On Computational Science, ICCS 2015.

[11] X. David-Henriet, L. Hardouin, J. Raisch, and B. Cottenceau, "Model predictive control for discrete event systems with partial synchronization," *Automatica*, vol. 70, no. June, 2016.

[12] B. S. Peter Brucker, Bernd Jurisch, "A branch and bound algorithm for the job-shop scheduling problem," *Discrete Applied Mathematics*, vol. 49, pp. 107–127, 1994.

[13] A. Hariri and C. N. Potts, "A branch and bound algorithm for the job-shop scheduling," *J.K.A. U.: Sci*, vol. 3, pp. 201–209, 1991.

[14] C. A. Floudas, *Nonlinear and Mixed-Integer Optimization: Fundamentals and Applications citation information*. New York: Oxford, 1995.