17th European Symposium on Computer Aided Process Engineering – ESCAPE17
V. Plesu and P.S. Agachi (Editors)

Generic vs. Engineered Evolutionary Algorithms in Batch Scheduling with Recourse

Guido Sand,[a] Thomas Tometzki,[b] Jochen Till,[b] Maren Urselmann,[b]
Michael Emmerich,[c] and Sebastian Engell[b]

[a]*ABB Corporate Research, 68526 Ladenburg, Germany, guido.sand@de.abb.com*
[b]*Process Control Laboratory, Universität Dortmund, 44221 Dortmund, Germany,*
{t.tometzki | j.till | m.urselmann | s.engell}@bci.uni-dortmund.de
[c]*LIACS, University of Leiden, 2333 CA-Leiden, The Netherlands, emmerich@liacs.nl*

Abstract

This paper considers a case study of a batch chemical scheduling problem on a moving horizon with significant uncertainties in demand. The scheduling problem is represented as a two-stage stochastic integer program and solved by a stage-decomposition based hybrid algorithm with an evolutionary algorithm for the first-stage and mathematical programming for the second-stage. We describe an engineered evolutionary algorithm with systematic inclusion of process knowledge versus a generic evolutionary algorithm. The former exploits the hierarchical structure of *operation*, *batching* and *scheduling* decisions in the solution space representation and the mutation operator. Comparative numerical experiments show that the coverage of the feasible search space is significantly improved and the convergence to good solutions is faster.

Keywords: batch scheduling, evolutionary algorithms, knowledge integration

1. Introduction

The information and decision structure in scheduling on moving horizons with uncertainties can be reflected by a mixed-integer recourse model with a finite number of scenarios in the form of a two-stage stochastic integer program. The

here-and-now decisions (first-stage) which have to be made under uncertainty are compensated by *recourse* decisions (second-stage). In [1] the application of a hybrid stage decomposition based algorithm to a case study (see Section 2) was presented. Compared to an exact scenario decomposition based algorithm [2] the hybrid algorithm improves the initial solution faster for a while but then it stagnates at suboptimal solutions. It is supposed that the reason is that hybrid algorithm covers the highly constrained search space insufficiently. The aim of the present work is to remedy the shortcomings of the generic evolutionary algorithm for the case study by an *engineered* evolutionary algorithm.

2. Problem statement and generic evolutionary approach

2.1. Chemical Batch Scheduling Case Study

Fig. 1 shows the layout of a multi-product batch plant for the production of expandable polystyrene (EPS) [1]. Two types A and B of the polymer in five grain size fractions are produced from raw materials E. The preparation stage is not considered here. The polymerization stage is operated in batch mode and is controlled by ten recipes. Each recipe defines the product (A or B) and its grain size distribution. Each batch yields a main product and four coupled products. The capacity of the polymerization stage constrains the number of batches to 12 in each two-day period. The batches are transferred into two semi-continuously operated finishing lines which fractionate the grain sizes. The capacity of each finishing line is between 5 and 12 batches per period in case it is operated, and 0 otherwise. The operation mode can be changed every second period. The scheduling decisions which have to be made are *operation decisions* on the finishing lines in each period, *batching decisions* on the numbers of polymerizations of each EPS-type in each period, and *scheduling decision* on the receipies used in each polymerization. The decisions in periods 1 to 3 are considered as first-stage, those in periods 4 and 5 as second-stage decisions. The uncertainty in the demands is represented by 64 scenarios of equal probability. The profit to be maximized is calculated from sales revenues, production costs, storage costs, and penalties for lateness and for finishing line start-ups and shut-downs.

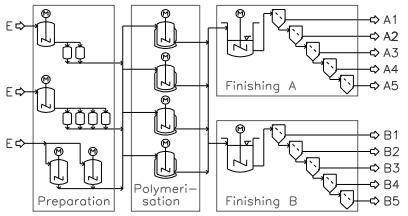

Fig. 1: The flow sheet of the multi-product batch

2.2. Stage decomposition based algorithmic approach

The main idea of stage decomposition is to remove the ties between the second-stage scenario subproblems by fixing the first-stage decisions. The scenario

subproblems are of significantly smaller size than the full two-stage problem. The master problem is a function of the vector of first-stage variables **x** only:

$$\min_{\mathbf{x}} \; f(\mathbf{x})=\mathbf{c}^T\mathbf{x}+\sum_{\omega=1}^{\Omega}\pi_\omega Q_\omega(\mathbf{x}) \quad \text{s.t.} \quad \mathbf{Ax} \le \mathbf{b}, \mathbf{x} \in X. \tag{1}$$

The evaluation of the second-stage value function $Q_\omega(\mathbf{x})$ for a given **x** requires the solution of Ω independent MILP subproblems over the second-stage variables y_ω:

$$Q_\omega(\mathbf{x}) = \min_{\mathbf{y}_\omega} \; \mathbf{q}_\omega^T\mathbf{y}_\omega \quad \text{s.t.} \quad \mathbf{W}_\omega\mathbf{y}_\omega \le \mathbf{h}_\omega - \mathbf{T}_\omega\mathbf{x}, \mathbf{y}_\omega \in Y \quad \forall\omega = 1,\ldots,\Omega. \tag{2}$$

The linear constraints of the master problem (1) are scenario independent, while the parameters of the linear second-stage constraints in (2) may vary from scenario to scenario. The vector of the first-stage variables **x** appears as a vector of fixed parameters in the constraints of the second-stage scenario problems. First-stage feasible solutions do not necessarily have feasible solutions in the second-stage due to the implicit constraints in (2). The objective is to minimize the sum of the costs of the first-stage decisions and to the expected costs of the second-stage decisions, weighted by the vectors **c** and $\mathbf{q}_\omega$. The finite sets X and Y may contain integrality requirements. The main algorithmic idea is to address the master problem given by (1) by an evolutionary algorithm. To evaluate f(**x**), the Ω subproblems given by (2) are solved independently by a MILP solver.

2.3. Generic evolutionary algorithm

A realization of this algorithmic approach was presented in [1] using the mixed-integer (μ,κ,λ)-evolution strategy from [3]. Each individual of the population represents a search point $\mathbf{x}_k=(x_1,\ldots,x_n)$ by its object parameters, in addition to mutation strength parameters $\mathbf{s}_k=(s_1,\ldots,s_n)$ which affect the mutation operator. In the evaluation of **x**, for unsatisfied constraints $\mathbf{Ax}\le\mathbf{b}$ the fitness function f(**x**) is replaced by the penalty function $p(\mathbf{x})+f_{max}$ which is defined as the sum of constraint violations according to $p(\mathbf{x})=\sum_j(\mathbf{A}_j\mathbf{x}-\mathbf{b}_j)$ and an upper bound f_{max} of f(**x**) for feasible solutions **x**. After the evaluation, λ offsprings are generated by λ-fold application of the mutation operator. It perturbes each variable x_i by a random number drawn from the symmetric difference of two discrete geometric distributions. The distribution depends on the dimension n and the parameter s_i which is modified log-normally [3]. To maintain the bounds for x_i, values outside the bounds are mapped onto the next bound. A truncation selection chooses the μ best ($1\le\mu\le\lambda$) individuals out of the union of μ parents and λ offsprings which do not exceed the maximum age of κ for the next interation loop.

2.4. Analysis of the search space of the generic algorithm

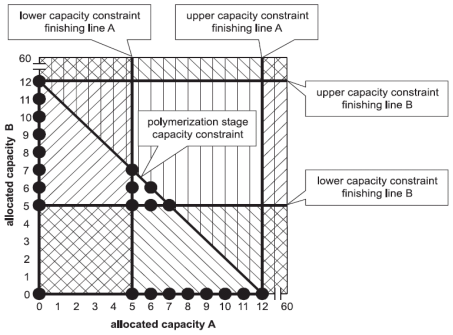

Fig. 2: Capacity constraints and feasible solutions in the space of allocated capacities.

The natural representation of the search space for the case study is given by a 30-dimensional integer vector (one variable for the number of each recipe in the 3 first-stage periods). Each variable is bounded between 0 and 12 leading to a 30-dimensional box-constrained search space with a cardinality of $13^{30} \approx 2.62 \cdot 10^{33}$. The capacity of the polymerization stage constrains each sum of 10 variables belonging to one period to 12. Analogously, the capacity of each finishing line constrains the sum of 5 variables to the disjunctive sets $\{0\} \cup \{5,\ldots,12\}$. The sum of the allocated capacities of both finishing lines in one period is equal to the allocated capacity of the polymerization stage. Fig. 2 shows the subsets of points which satisfy all capacity constraints for one period. Each aggregated point represents a 10-dimensional subspace. The cardinalities of points for the subsets are 1 for the set in the origin (no finishing line operated, symbolized as 00), 6,062 for each set on the axes (one finishing line operated, symbolized as 01 or 10), and 196,056 for the set in the center (both finishing lines operated, symbolized as 11). The geometry of capacity constraints is identical for all periods. The operations state constraints apply to pairs of operation states of the same finishing line in two successive periods leading to 25 (out of 64) feasible operation state combinations. The cardinality of the set of points which satisfy all constraints in all three periods is approx. $8.50 \cdot 10^{15}$. A ratio of feasible to infeasible solutions of only $1{:}3 \cdot 10^{19}$ highlights that the optimization problem is highly constrained.

3. Engineered approach

3.1. Drawbacks of the generic evolutionary algorithm

In previous work [1] it was found that the generic evolutionary algorithm typically converges towards a solution with operation state vector (11 11 11) even though a better solution is known that has a different operation state vector, e.g. (01 11 11) (the tupels represent periods 1, 2, 3). The aggregated representation of the search space as developed in section 2.4 can help to identify the reasons for the poor results. Fig. 3 shows a typical evolution of the allocated capacities in the first period for both finishing lines during 2 CPU-hours. The different shades of grey represent the quartiles and the median of the population. The integer variables were initialized randomly according to a uniform distribution

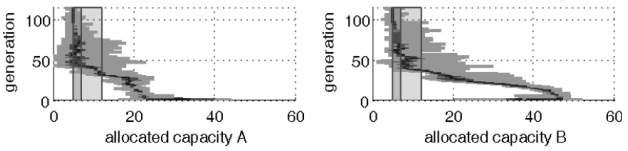

Fig. 3: Evolution of the allocated capacities (generic algorithm).

with an expected value of 6. Consequently, the initially allocated polymerizations (sums of 5 variables) are nearly normally distributed with an expected value of 30. It can be observed that the penalty function forces the object parameters towards solutions with the operation state vector (11 11 11). The corresponding subset of polymerization vectors is the largest one and nearest to the initial population. After reaching this subset the population stays in this subset due to the following reasons:

1. Other subsets of feasible solutions are at least one order of magnitude smaller than the subset corresponding to the operation vector (11 11 11).
2. Another feasible subset must be hit directly due the truncation selection operator and the "feasible over infeasible"-penalty in the fitness function.
3. The distance between the feasible subsets in the space of allocated capacities is large compared to the variance of the mutation distribution.
4. The mapping function after the undirected mutation introduces a bias into the offspring distribution away from the boundaries.

3.2. Engineered evolutionary algorithm

The analysis in Section 3.1 confirmed the hypothesis that the space of feasible solutions is not well covered by the generic evolutionary algorithm. The aim of algorithm engineering here is to improve the coverage of the feasible search space. In contrast to the natural representation used by the generic algorithm, a specific representation of the individuals is used here.

The representation reflects the hierarchy of decisions as mentioned in Section 2.1 namely *operation*, *batching* and *scheduling*. A decision tree is constructed by propagating the operation and capacity constraints from the root to the leaves while the decisions are disaggregated such that the full tree exactly represents the total feasible set of polymerization vectors. On each hierarchical layer, the feasible decisions are represented by layer-specific decision sets. Altogether, each solution is represented by twelve object parameters, in addition to one strategy parameter representing the mutation strength. The object parameters are initialized such that all paths in the decision tree have the same probability.

According to this initialization scheme, the largest subset with the operation state vector (11 11 11) is still privileged, but the probability for other subsets is significant. Corresponding to the hierarchical representation of the feasible set of solutions, a hierarchical mutation operator was designed for the variation of individuals. Its design is based on *minimal moves* which are applied sequentially to each hierarchical layer from the root to the leaves. In each mutation, minimal moves are executed as long as the sum of their weights does not exceed the mutation strength, where the weights decrease from the root to the leaves.

Each weight is proportional to the estimated impact on the change of the objective function of the corresponding minimal move. A minimal move changes the solution on a layer to a randomly chosen neighbour of the decision set. Possible minimal moves for the operation state 00 in the first period are 01 and 10 with the same probability, whereas state 11 is reached by at least two minimal moves. The mutation strength is adapted similar to the generic algorithm.

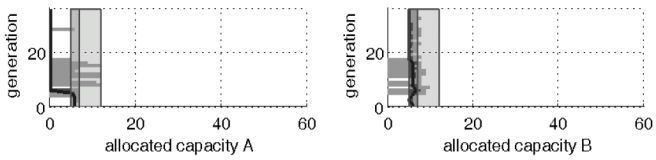

Fig. 4: Evolution of the allocated capacities (engineered algorithm).

The engineered algorithm covers the set of feasible solutions significantly better than the generic algorithm. The main reason is that according to the hierarchical mutation scheme the allocated capacities are controlled by the operation states and not by the polymerizations. The probability of a mutation of an operation state does not depend on the cardinality of the corresponding feasible subset of polymerizations. Fig. 4 shows the evolution of the allocated capacities corresponding to Fig. 3. After a few generations in operation state (11 11 11), the first solution with operation state vector (01 11 11) is found very fast. The fitness of the best solution found is shown in Fig. 5 on a logarithmic scale over CPU-time. The fitness converges significantly faster to a significantly better level for the engineered algorithm than in the generic case.

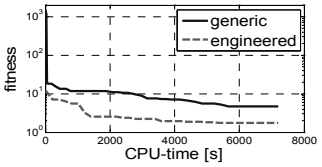

Fig. 5: Evolution of the fitness.

4. Conclusions and further work

The present work demonstrated that the inclusion of problem specific knowledge can significantly improve the efficiency of an evolutionary scheduling algorithm. The analysis showed that the highly constrained search space in not well covered by the generic algorithm. An improvement is expected from combining its metaheuristics with the ability of specific mutation operators for constrained aggregated parameters to cover all feasible subsets of the search space.

References

1. J. Till, G. Sand, M. Urselmann and S. Engell, Computers and Chemical Engineering, (2006) in press
2. C. Carøe and R. Schultz, Operations Research Letters, 24 (1999) 37
3. I.C. Parmee (ed.), Evolutionary Design and Manufacture, Springer, NY, 2000, pp. 55-67
4. Y. Davidor, H.P. Schwefel and R. Männer (eds.), Parallel Problem Solving from Nature, Springer, Berlin, 1994, pp. 193-197