European Symposium on Computer Aided Process Engineering – 15
L. Puigjaner and A. Espuña (Editors)

A hybrid algorithm for solving two-stage stochastic integer problems by combining evolutionary algorithms and mathematical programming methods

Jochen Till[a*], Guido Sand[a], Sebastian Engell[a],
Michael Emmerich[b] and Lutz Schönemann[c]

[a]Process Control Lab (BCI-AST), Department of Biochemical and Chemical Engineering, Universität Dortmund, Emil-Figge-Str. 70, D-44221 Dortmund, Germany

[b]Leiden Institute of Advanced Computer Science (LIACS), University of Leiden, Niels Bohrweg 1, 2333 CA Leiden, The Netherlands

[c]Chair of Systems Analysis (LS 11), Department of Computer Science, Universität Dortmund, Joseph-von-Fraunhofer-Str. 20, D-44227 Dortmund, Germany

Abstract

We propose a new hybrid algorithm to solve linear two-stage integer programs (2SIPs) based on stage decomposition. The master algorithm performs a search on the first stage variables by an evolutionary algorithm (EA), the decoupled scenario problems are solved by mathematical programming. The approach is applied to a real-world scheduling problem with uncertainties. The performance of different EAs, namely a genetic algorithm and an evolution strategy for integer programming, is compared to that of the solution of a monolithic 2SIP using mathematical programming.

Keywords: stochastic integer programming, evolutionary algorithms, hybrid algorithms, batch scheduling, uncertainty

1. Introduction

Design and operational problems in the processing industries are usually characterized by uncertainties in parameters (Sahinidis, 2004). A promising approach to deal with these uncertainties is the use of stochastic integer programming (Birge and Louveaux, 1997). In a recent publication, Sand and Engell (2004) presented a linear two-stage stochastic integer program for the online-scheduling of an industrial multi-product batch plant. Sand and Engell (2004) applied the scenario decomposition algorithm of Carøe and Schultz (1999), which was shown to generate high quality solutions in reasonable computing times. However, most of the computing time is spend on generating bounds on the optimal solution rather than on the generation of good feasible solutions.

* Author/s to whom correspondence should be addressed: j.till@bci.uni-dortmund.de, tel.: +49-231-755-6080; fax: +49-231-755-5129

2. The hybrid algorithm

Two-stage stochastic programs represent optimization problems in which some of the decisions have to be made under uncertainty and the remainder can be made after the uncertainty has been realized. Birge and Louveaux (1997) state a linear two-stage stochastic integer program (2SIP), in which the uncertainty is represented by a finite number of scenarios ω with corresponding probabilities π_ω, as its deterministic equivalent mixed-integer linear program (MILP):

$$\max_{\boldsymbol{x},\, \boldsymbol{y}_\omega} \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} + \sum_{\omega=1}^{\Omega} \pi_\omega \boldsymbol{q}_\omega^{\mathrm{T}} \boldsymbol{y}_\omega \quad \text{s.t.} \quad \boldsymbol{T}_\omega \boldsymbol{x} + \boldsymbol{W}_\omega \boldsymbol{y}_\omega \le \boldsymbol{h}_\omega, \boldsymbol{x} \in X,\ \boldsymbol{y}_\omega \in Y,\ \omega = 1,\ldots,\Omega. \tag{1}$$

The variables are assigned to 1st and 2nd stage vectors $\boldsymbol{x}$ and $\boldsymbol{y}_\omega$, which belong to polyhedral sets X and Y with integer requirements. The $\boldsymbol{x}$-vector represents "here and now"-decisions which are applied regardless of the future evolution and thus have to be identical for all scenarios. In contrast, the $\boldsymbol{y}_\omega$-vectors denote scenario-dependent recourses under the assumption that the respective scenario realizes. The uncertainties may affect any parameter of the model, such that Ω different matrices and right hand sides $\boldsymbol{T}_\omega$, $\boldsymbol{W}_\omega$ and $\boldsymbol{h}_\omega$ may arise. The objective is to maximize the first stage profit plus the expected second stage profit computed using the weighting-vectors $\boldsymbol{c}$ and $\boldsymbol{q}_\omega$.
The 2SIP according to Eq. (1) exhibits a characteristic block-angular matrix structure (Fig. 1, left). With fixed first stage variables $\boldsymbol{x}$, the 2SIP decomposes into decoupled second stage scenario sub-problems, which can be solved separately (Fig. 1, right).

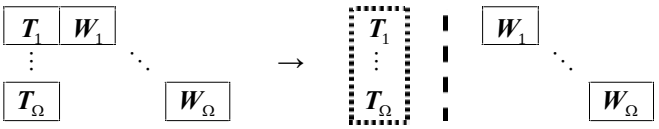

Figure 1. The matrix structure of the 2SIP (left) and stage decomposition with fixed $\boldsymbol{x}$ (right).

The main idea of the EA/MILP algorithm is to let a master algorithm perform the search on the first stage variables $\boldsymbol{x}$ by an evolutionary algorithm and to solve the decoupled sub-problems for a given $\boldsymbol{x}$ by mathematical programming. The objective value $f(\boldsymbol{x})$ is interpreted as the "fitness" of an individual of the EA. In contrast to stage decomposition based mathematical programming methods (e.g. the "L-shaped decomposition") EAs do not suffer from a lack of convexity of the value function, i.e. the optimal second stage objective as function of $\boldsymbol{x}$. However, the hybrid algorithm is not able to provide lower bounds in contrast to mathematical algorithms.
In this work we compare two different EAs: a genetic algorithm (GA), which typically operates on a bit-string representation with recombination as the main operator, and an evolution strategy (ES) for integer programming with geometric mutation, which adapts the mutation strength during the course of evolution (Beyer and Schwefel, 2002).
The GA used here (Goldberg, 1989) operates on a bit-string of the dimension n with a population size μ typically in the range of [30; 50] and generates an identical number of offspring $\lambda=\mu$. The recombination is defined as uniform crossover with a probability of p_c typically from the interval [0.6;1]. The mutation operator inverts single bits and its

probability is typically $p_m \cong 1/n$. The $(\mu+\lambda)$ selection chooses the best μ individuals from a set of μ parents and λ offspring. The strategy parameter vector is $\boldsymbol{s}_{GA}=(\mu, \lambda, p_c, p_m)$.
The ES employed here (Rudolph, 1994) uses a population size of μ=10 with a recommended number of offspring $\mu/\lambda \cong 1/7$. The dominant recombination takes the parameters from one of two uniformly selected parents with equal probability. The intermediate recombination uses the arithmetic mean of both parents' parameters. Usually, the object parameters (search space) are recombined by the dominant method and the strategy parameters by the intermediate method. The mutation operator with a constant mutation rate adds values from a geometric distribution scaled by the adaptive mean mutation step size σ to the integer parameters. The initial value of σ should be set to 10% of the range of the object parameters. If a parent individual exceeds the maximum age κ=5, it is not further considered in the (μ, κ, λ)-selection that chooses the best individuals from a set of μ parents and λ offspring. The vector of strategy parameters is $\boldsymbol{s}_{ES}=(\mu, \kappa, \lambda, \sigma)$.

3. The real world benchmark example

The aggregated scheduling model of a multi-product batch plant for the production of polymers with uncertainty in the demands serves as a real world example (Sand and Engell, 2004). As shown in Fig. 2, two types (A/B) of expandable polystyrene (EPS) in five grain size fractions each are produced from a number of raw materials (E).

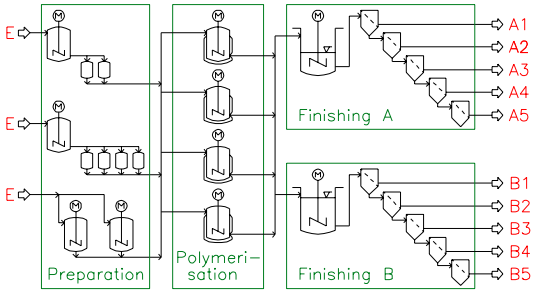

Figure 2. The flow sheet of the EPS-process.

The preparation stage and the polymerization stage operate in batch mode, the finishing lines operate continuously. For each EPS-type, five recipes exist which determine the grain size distribution such that each batch yields a main product and four coupled products. If a minimum feed to a finishing line cannot be provided, this line has to be shut down temporarily. The objective is to maximize the profit calculated from sales revenues, production costs, storage costs and penalties for lateness and for finishing line switches. The decisions are the discrete choice of recipes and their starting times.
The model used is based on a discrete representation of time with a scheduling horizon of up to n_i=10 intervals $i \in I = \{1,2,3,\ldots,n_i\}$. The uncertainty is represented by Ω demand scenarios of equal probability. The intervals $\{i \in I \mid i < 4\}$ are defined as the first stage intervals. The first stage decisions are the discrete choice of recipes in these intervals and are modelled by an vector of n=30 integer variables $\boldsymbol{x} \in X^n, X = \{0,1,\ldots,12\}$.

We define two test problems, which differ in the number of demand scenarios and in the number of second stage intervals (see Tab. 1). Since our work aims at real-time applications, the CPU-time is limited to 4 hours.

Table 1. The test problems and their model dimension size.

test problem	*model type*	*1st ; 2nd stage intervals*	*demand scenarios Ω*	*continuous variables*	*integer variables*	*constraints*
EPS-I	monolithic 2SIP	3; 7	128	64,001	17,920	45,539
	single scenario MILP	3; 7	1	531	110	327
EPS-II	monolithic 2SIP	3; 4	16	5601	1,568	4,083
	single scenario MILP	3; 4	1	381	68	228

4. The experimental set-up of the hybrid algorithm

The algorithmic framework was implemented in C/C++ using 'TEA' (Emmerich and Hosenberg, 2000), a library for the design of standard and non-standard EAs. For the MILP sub-problems GAMS/CPLEX 8.1 (CPLEX, 2002) is used. The calculations were performed on a 2.4 GHz Linux machine with 1 GB of memory.

Since the first stage decisions $\boldsymbol{x}$ are a vector of integer variables, a n_{ES}=30 dimensional vector $\boldsymbol{x}_{ES} \in X_{ES}^{n_{ES}}, X_{ES} = \{0,1,\ldots,12\}$, was chosen as the ES representation. The GA representation is based on a 4-bit binary code for each integer and results in a n_{GA}=120 dimensional bit string $\boldsymbol{x}_{GA} \in X_{GA}^{n_{GA}}, X_{GA} = \{0,1\}$. The integer coding leads to $2.62 \cdot 10^{33}$ combinations, the binary coding to $1.33 \cdot 10^{36}$. Because of the resource constraints $\boldsymbol{g}(\boldsymbol{x}) \leq 0$ the search space is highly constrained and only $8.50 \cdot 10^{15}$ feasible solutions exist.

An initial population of μ feasible individuals with a high diversity helps the EA to converge to a good solution quickly. The search space is heavily constrained, so it is not possible to generate enough feasible solutions from random integer numbers in a reasonable time. Thus we applied constraint propagation with domain reduction techniques to the hierarchy of constraints within a decision tree representation. The creation of 50 feasible solutions takes only 0.5s and is independent of the size of the second stage.

The problem was formulated as a minimization problem. To handle the constraints, the original objective function $f(\boldsymbol{x})$ was replaced by $F(\boldsymbol{x})$ which penalizes the distance to the feasible region and always prefers feasible to infeasible solutions (Deb, 2001):

$$F(\boldsymbol{x}) = \begin{cases} f(\boldsymbol{x}) & \text{if } \boldsymbol{x} \text{ feasible} \\ f_{max} + \sum_{i=1}^{n} g_i(\boldsymbol{x}) & \text{otherwise} \end{cases} \tag{2}$$

The parameter f_{max} denotes the worst feasible solution (Deb, 2001). The evaluation of a feasible individual takes ~60s for *EPS-I* and ~2.5s for *EPS-II*, whereas the calculation of the penalty requires less than 10^{-2}s.

From 1,500 uniformly distributed feasible solutions generated by CP (see 4.2), the range of the objective function of *EPS-I* was [-78, -55] with the majority of solutions around -73, and of *EPS-II* it was [-48, -25] with the most solutions around -40. An absolute MILP integrality gap of 1.0 for *EPS-I* and 0.5 for *EPS-II* led to a sufficient accuracy of the sub-problem solutions.

In general, an EA does not provide a lower bound of $f(\boldsymbol{x})$. A guaranteed lower bound was computed by the solution of the 2SIP with integer relaxation (R-2SIP) and by the perfect information (PI-2SIP) assumption (Birge and Louveaux, 1997), see Tab 3.

5. Results and analysis

We started the numerical experiments for the GA and the ES with a feasible initial population and applied the recommended strategy parameters $\boldsymbol{s}_{GA}=(\mu=50,\ \lambda=50,\ p_c=1,\ p_m=1/120)$ and $\boldsymbol{s}_{ES}=(\mu=10,\ \kappa=5,\ \lambda=70,\ \sigma=1.2)$. To improve the results, a set of experiments was conducted with *EPS-II* by varying one strategy parameter at a time and keeping the others constant. For the GA the original parameters obtained the best performance, for the ES we found a better performance without recombination of the object parameters and with discrete recombination of the step size. Table 2 displays the best objective after a certain CPU user time for typical results out of several EA/MILP runs compared to the solution of a monolithic 2SIP.

Table 2. Best objective function values after a specified CPU user time.

	CPU-time	*0.5h*	*1h*	*2h*	*3h*	*4h*	R-2SIP	PI-2SIP
EPS-I	2SIP CPLEX *obj*	+59.75*	+59.75*	+59.75*	+59.75*	+59.75*	-92.48	-81.92
	GA/MILP *obj*	-74.68	-74.71	-74.71	-75.64	-75.64	--	--
	GA evaluations	25	39	141	252	336		
	ES/MILP *obj*	-76.24	-76.24	-76.52	-76.52	-76.63	--	--
	ES evaluations	33	88	181	254	314		
EPS-II	2SIP CPLEX *obj*	-46.75	-46.75	-48.14	-48.14	-48.14	-62.22	-52.11
	GA/MILP *obj*	-45.82	-47.28	-48.12	-48.12	-48.12	--	--
	GA evaluations	1,098	2,159	4,437	6,587	8,722		
	ES/MILP *obj*	-47.41	-47.56	-49.5	-50.41	-51.03	--	--
	ES evaluations	878	1,720	4,769	6,787	8,911		

* *Trivial solution, all integer variables are equal to zero.*

For the large model *EPS-I*, both EA/MILP approaches generated a feasible solution quickly, but did not improve beyond the best results of a random sampling of the feasible solutions (see Sec. 4.3). The reason is the large problem size; only 314 ES evaluations were performed within 4 hours, which is not more than four generations.

For the smaller problem *EPS-II*, reasonable solutions were found after a short period of time. The monolithic 2SIP and the GA/MILP algorithm converge slowly whereas the ES/MILP improves faster towards the global optimum. The guaranteed gap is less than 2%. For the GA it was found that the object parameters converge also and the population loses diversity. This is not the case in the ES, probably because of the maximum life span of the individuals and because the recombination of the object parameters is omitted.

Fig. 4 displays the fitness $F(\boldsymbol{x})$ of the feasible offspring (left) and of the infeasible offspring (right) plotted versus the number of EA fitness evaluations for a typical ES/MILP run. Note, that due to the penalty function the fitness of infeasible solutions is always greater than $f_{max}=0$. In the average, more than one third of the offspring is feasible.

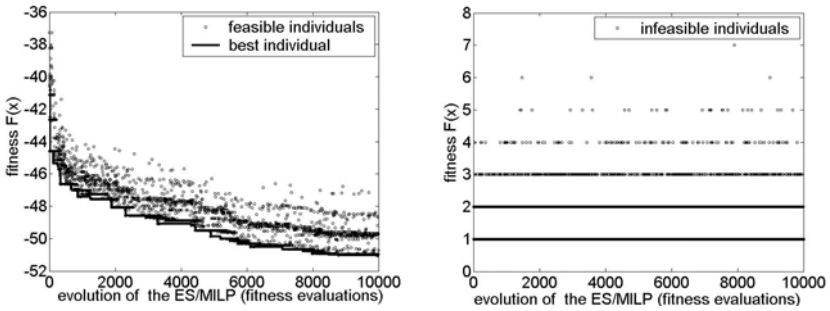

Figure 4.Evolution of the fitness during an ES/MILP run (EPS-II).

6. Conclusions and recommendations

A hybrid EA/MILP algorithm for 2SIPs was presented and applied to a real-world scheduling example. As a next step, the problem specific effects of the EA operators as well as the influence of the shape of the fitness function will be analyzed. In general, the efficiency of the EA/MILP depends (1) on an efficient EA and (2) the fast solution of the sub-problems. In future work (1) is tackled by a problem specific EA based on a decision tree that represents only feasible solutions. The second part will be tackled by adaptive accuracy and approximate solutions. The EA/MILP approach is easy and general to use and can be extended to multi-layer problems. The EA master problem can handle any logic or non-linear constraints. For real-time applications the decomposition enables the parallel solving of the individuals or sub-problems.

References

Beyer, H.-G.; Schwefel, H.-P.: Evolution strategies - A comprehensive introduction, Natural Computing, 1, 3-52, 2002.

Birge, J. F.; Louveaux, F.: Introduction to stochastic programming. Springer, New York, 1997.

Carøe, C.C.; Schultz, R.: Dual decomposition in stochastic integer programming. Operations Research Letters 24 (1999), 37-45.

CPLEX: Using the CPLEX callable library. ILOG Inc., Mountain View, CA, 2002.

Deb, K.: Multi-Objective Optimization using Evolutionary Algorithms. Wiley, Chichester, 2001.

Emmerich, M.; Hosenberg, R.: TEA - A Toolbox for the Design of Parallel Evolutionary Algorithms in C++, Tech. Report, CI-106/01, DFG-SFB 531 , University of Dortmund, 2000.

Goldberg, David E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading, Mass., 1998.

Rudolph, G.: An Evolutionary Algorithm for Integer Programming. In: Davidor, Y.; Schwefel, H.-P.; Männer, R. (Eds.) : Parallel problem Solving from Nature - PPSN III, Lecture Notes in Computer Science., Springer, Berlin, 1994, 193-197.

Sahinidis, N. V.: Optimization under uncertainty: state-of-the-art and opportunities. Computers and Chemical Engineering 28 (2004), 971-983.

Sand, G.; Engell, S.: Modelling and solving real-time scheduling problems by stochastic integer programming. Computers and Chemical Engineering 28 (2004), 1087-1103.

Acknowledgement

The financial support by the German Research Foundation (DFG), Collaborative Research Center "Design and Management of Complex Technical Processes and Systems by Means of Computational Intelligence Methods" (SFB 531), project C10, is gratefully acknowledged.