Rigorous vs. Stochastic Algorithms for Two-stage Stochastic Integer Programming Applications

Jochen Till, Sebastian Engell, and Guido Sand

Process Control Lab (BCI-AST), Universität Dortmund, D-44221 Dortmund, Germany

S.Engell@bci.uni-dortmund.de

Abstract

We recently proposed a new hybrid algorithm to solve linear two-stage stochastic integer programs (2SIPs) based on stage wise (primal) decomposition [1]. The master algorithm performs a search on the first stage variables by an evolutionary algorithm (EA), the decoupled second stage scenario problems are solved by mathematical programming. In this paper, the performance of the EA approach is compared to that of the 2SIP-solver of Carøe and Schultz [2] which is based on scenario (dual) decomposition. A real-world batch scheduling problem with uncertainties serves as a test instance.

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

I. Introduction

Design and operational problems in the processing industries are often characterized by uncertainties in parameters [3]. A promising approach to deal with these uncertainties is the use of stochastic integer programming [4]. In recent publications, Sand and Engell [5], and Engell et. al. [6] presented a linear two-stage stochastic integer program for the online-scheduling of an industrial multi-product batch plant. They applied the rigorous scenario decomposition algorithm of Carøe and Schultz [2], which was shown to generate high quality solutions in reasonable computing times. However, the focus is on generating lower bounds on the optimal solution rather than on the generation of good feasible solutions. Alternatively, Till et. al. [1] proposed a stochastic algorithm to solve 2SIPs based on stage wise decomposition using evolutionary algorithms combined with mathematical programming methods. Basically, the rigorous algorithm performs an implicit enumeration of all solutions while the stochastic one is based on randomized metaheuristics. The aim of this paper is to discuss and to compare both approaches by solving the same instance of a real world scheduling problem.

II. The stochastic programming formulation

Two-stage stochastic programs represent optimization problems in which some of the decisions have to be made under uncertainty in the model parameters and the remaining decisions can be made after the uncertainty has been realized. When the uncertainty is represented by a finite number of scenarios $\omega = 1, \dots, \Omega$ with corresponding probabilities π_ω, the *deterministic equivalent program* of the linear optimization problem in the *extensive form* [4] is given as

$$\begin{aligned} z \;&= \min\nolimits_{\mathbf{x},\mathbf{y}_\omega} \;\; \{\mathbf{c}^T\mathbf{x} + \sum_{\omega=1}^{\Omega} \pi_\omega \mathbf{q}_\omega^T \mathbf{y}_\omega | \mathbf{A}\mathbf{x} \le \mathbf{b}, \mathbf{T}_\omega \mathbf{x} + \mathbf{W}_\omega \mathbf{y}_\omega \le \mathbf{h}_\omega, \\ &\qquad \mathbf{x} \in X, \mathbf{y}_\omega \in Y, \omega = 1, \dots, \Omega\}. \end{aligned} \tag{1}$$

The variables are assigned to the first and second stage decisions $\mathbf{x}$ and $\mathbf{y}_\omega$, which belong to the polyhedral sets X and Y. The first stage decisions $\mathbf{x}$ represent *here-and-now*-decisions which are applied regardless of the future evolution and thus have to be identical for all scenarios. In contrast, the sets of second stage decisions $\mathbf{y}_\omega$ denote scenario-dependent recourses which are associated to the corresponding realization of $\mathbf{q}_\omega$, $\mathbf{h}_\omega$, $\mathbf{T}_\omega$, and $\mathbf{W}_\omega$. The objective is to minimize the first stage costs plus the expected second stage costs. The program (1) can be written as the *intensive form* of the *deterministic equivalent program* [4]

$$z \;=\; \min_{\mathbf{x}} \{\mathbf{c}^T\mathbf{x} + \sum_{\omega=1}^{\Omega} \pi_\omega Q_\omega(\mathbf{x}) | \mathbf{A}\mathbf{x} \le \mathbf{b}, \mathbf{x} \in X\}, \tag{2}$$

where the *second stage value function* $Q_\omega(\mathbf{x})$ is defined by

$$Q_\omega(\mathbf{x}) \;=\; \min_{\mathbf{y}_\omega} \{\mathbf{q}_\omega^T \mathbf{y}_\omega | \mathbf{W}_\omega \mathbf{y}_\omega \le \mathbf{h}_\omega - \mathbf{T}_\omega \mathbf{x}, \mathbf{y}_\omega \in Y\}. \tag{3}$$

When all second stage variables are continuous, i.e. $Y \subseteq \mathbb{R}_+^{n_2}$, the value function is a piecewise linear convex function in $\mathbf{x}$. In a two-stage stochastic integer program (2SIP), a subset of the second stage variables are subject to integrality requirements, i.e. $Y \subseteq \mathbb{Z}_+^{n_2'} \times \mathbb{R}_+^{n_2''}$. Then the value function is in general non-convex and non-differentiable in $\mathbf{x}$ and has the same properties as the value function in integer programming [4, 7, 8].

The simplest approach to solve (1) is to consider it as a large scale monolithic mixed-integer linear program (MILP) and to apply a commercial standard solver, e.g. CPLEX [9]. However, the constraint matrix of (1) exhibits a characteristic block-angular structure (see Fig. 1) which is amenable to decomposition approaches. When $Y \subseteq \mathbb{R}_+^{n_2}$, this structure together with the convexity property of $Q_\omega(\mathbf{x})$ is exploited by efficient decomposition approaches like the rigorous L-shaped method [4]. When the second stage requires integer variables, the non-convexity of the value function prohibits the efficient use of these methods. For the general case of mixed-integer first and second stages, the dual decomposition algorithm of Carøe and Schultz [2] is considered to be the state-of-the-art for solving 2SIPs [3]. A recent survey of algorithms for 2SIPs can be found in [10].

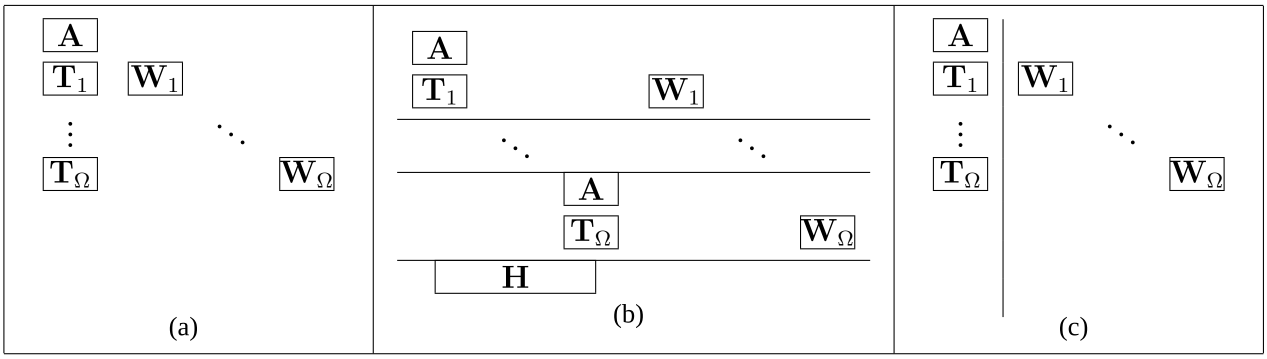

Figure 1: The structure of the constraints of the the two-stage stochastic program (a), the scenario decomposition (b), and the stage wise decomposition (c).

III. The EA/SIP approach for stage wise decomposition

The main idea of the EA/SIP algorithm is to consider the implicit form of the 2SIP (2) as the master problem and to let an evolutionary algorithm perform the search on the first stage decisions $\mathbf{x}$. With fixed first stage decisions the stages decompose (see Fig. 1) into Ω subproblems (3), which are solved separately for all $\omega \in \{1,\dots,\Omega\}$ using a standard MILP solver (e.g. CPLEX [9]). Each individual of the EA interprets $\mathbf{x}$ as its *object parameters* (phenotype) and the objective value $z(\mathbf{x})$ as its *fitness*. In contrast to stage decomposition based mathematical programming methods (e.g. the L-shaped decomposition) EAs do not require convexity of the value function. However, an EA is not able to provide lower bounds during the course of evolution in contrast to mathematical algorithms.

Till et. al. [1] proposed the use of an integer evolution strategy (ES) as a master algorithm. Evolution strategies are a class of evolutionary algorithms, which were originally developed for continuous search spaces and which adapt the mutation strength during the course of evolution [11]. The integer ES used here is designed for bounded search spaces X [12] and is based on the ES for unbounded integer programming of Rudolph [13].

The integer ES employed here uses a population size of μ parents with a recommended number of offspring $\lambda = 7\mu$. For the recombination, two of the μ parents are randomly selected. Similar to the GA crossover-operator, the dominant recombination operator generates an offspring by randomly taking each of the offsprings parameters from one of the selected parents with the same probability. The intermediate recombination uses the arithmetic mean of both parents' parameters. Usually, the object parameters (search space, phenotype) are recombined by the dominant method and the strategy parameters (e.g. mutation step size) 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. If a parent individual exceeds the maximum age κ, it is not further considered in the selection that chooses the best μ individuals from a set of μ parents and λ offspring.

There is no general efficient method for constraint handling in EAs [14]. To handle the feasibility constraints on $\mathbf{x} \in X$ given in (2), we propose the use of a modified objective function. The original objective $f(\mathbf{x})$ in (2) is replaced by $F(\mathbf{x})$ which always prefers feasible to infeasible solutions [15]:

$$F(\mathbf{x}) \quad = \quad \left\{ \begin{array}{ll} f(\mathbf{x}) & \text{if } \mathbf{x} \text{ is feasible} \\ f_{max} + p(\mathbf{x}) & \text{otherwise} \end{array} \right\} \tag{4}$$

The parameter f_{max} denotes the worst feasible solution and is found by maximizing the integer relaxation of (2). The positive penalty term $p(\mathbf{x})$ is a measure of the constraint violation and is typically calculated by $p(\mathbf{x}) = \sum_{j=1}^{J} r_j \max[0, (g_j(\mathbf{x}))]$, a weighted sum of the right hand sides of the violated constraints using a positive scaling factor r_j for each constraint. In our approach the penalty term is explicitly calculated using simple linear equations and logic conditions.

The EA/SIP algorithm is finite if a termination criterion is used. In general, an EA does not provide a lower bound. Since the EA/SIP algorithm is based on a probabilistic technique, there is a positive probability that the optimum will not be visited in a given time [11]. Thus for the evaluation of the best objective found by the EA the median of several runs is considered.

IV. The scenario (dual) decomposition approach

The first stage decisions $\mathbf{x}$ cannot anticipate which scenario realizes and must be feasible for each scenario. The condition that the first stage decision $\mathbf{x}$ is taken before realizing the random outcome can be stated explicitly by the *nonanticipativity constraints* on copies of the first stage variables $\mathbf{x}_\omega$ for each scenario ω. An equivalent formulation of (1) is given by

$$\begin{aligned} z \ &= \min\nolimits_{\mathbf{x}_\omega, \mathbf{y}_\omega} \ \{\sum_{\omega=1}^{\Omega} \pi_\omega(\mathbf{c}^T\mathbf{x}_\omega + \mathbf{q}_\omega^T\mathbf{y}_\omega) | \mathbf{x}_1 = \ldots = \mathbf{x}_\Omega, \mathbf{A}\mathbf{x}_\omega \le \mathbf{b}, \\ &\qquad \mathbf{T}_\omega\mathbf{x}_\omega + \mathbf{W}_\omega\mathbf{y}_\omega \le \mathbf{h}_\omega, \mathbf{x}_\omega \in X, \mathbf{y}_\omega \in Y, \omega = 1, \ldots, \Omega\}. \end{aligned} \tag{5}$$

This formulation suggests a decomposition based on dropping the nonanticipativity constraints since the stochastic program then becomes separable in the Ω subproblems corresponding to the individual scenarios ω (see Fig. 1).

The main idea of the dual decomposition algorithm of Carøe and Schultz [2] is the Lagrangian relaxation of the nonanticipativity constraints and a branch-and-bound algorithm to reestablish nonanticipativity. With the nonanticipativity constraints formulated as $\sum_{\omega=1}^{\Omega} \mathbf{H}_\omega\mathbf{x}_\omega = 0$, $\mathbf{H} = (\mathbf{H}_1, \ldots, \mathbf{H}_\Omega)$ and the vector of Lagrangian multipliers $\boldsymbol{\lambda}$, the Lagrangian dual problem of (5) is written as

$$z_{LD} = \max_{\boldsymbol{\lambda}} D(\boldsymbol{\lambda}), \tag{6}$$

$$\begin{aligned} D(\boldsymbol{\lambda}) &= \min_{\mathbf{x}_\omega, \mathbf{y}_\omega} \{\sum_{\omega=1}^{\Omega} \pi_\omega(\mathbf{c}^T\mathbf{x}_\omega + \mathbf{q}_\omega^T\mathbf{y}_\omega) + \boldsymbol{\lambda}\mathbf{H}_\omega\mathbf{x}_\omega | \mathbf{A}\mathbf{x}_\omega \le \mathbf{b}, \\ &\qquad \mathbf{T}_\omega\mathbf{x}_\omega + \mathbf{W}_\omega\mathbf{y}_\omega \le \mathbf{h}_\omega, \mathbf{x}_\omega \in X, \mathbf{y}_\omega \in Y, \omega = 1, \ldots, \Omega\} \end{aligned} \tag{7}$$

While (6) is a nonlinear concave maximization usually solved by a subgradient method [2], the subproblem (7) is separable in the scenarios ω which represent a MILP problem that can be solved e.g. by CPLEX [9].

The Lagrangian dual provides a tight lower bound which is not smaller than the lower bound z_{LP} obtained by the LP relaxation of (1), i.e. $z_{LP} \le z_{LD}$ [2]. Let the optimal solution of (1) be z^{opt}, then the duality gap is defined as $z^{opt} - z_{LD}$ and the integrality gap as $z^{opt} - z_{LP}$.

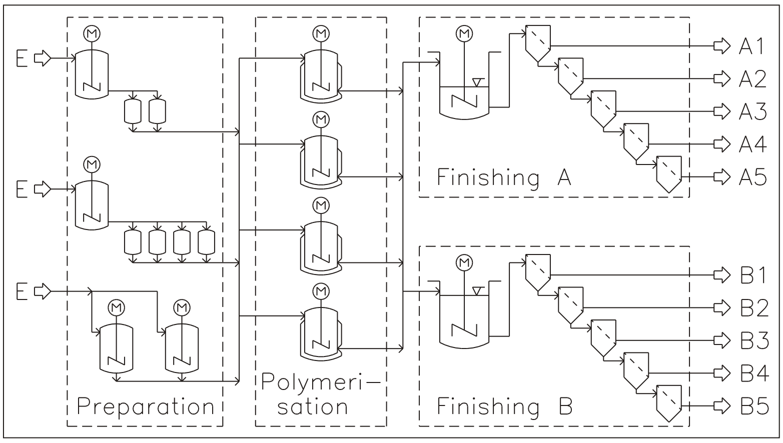

Figure 2: The flow sheet of the EPS-process.

The implementation of the finite dual decomposition algorithm [2] called DDSIP [16] uses a standard branch and bound procedure [8] on the first stage variables:

1. Initialization: Set $z^* = \infty$ and let $\mathcal{P}$ consist of problem (5).

2. Termination: If $\mathcal{P} = \emptyset$ then $\mathbf{x}^*$ with $z^*(\mathbf{x}^*)$ is optimal.

3. Node selection: Select and delete $P \in \mathcal{P}$ and solve its Lagrangian dual. If P is infeasible, $z_{LD} = \infty$. If $z_{LD} \geq z^*$ then goto 2.

4. Bounding: If the first stage solutions of the dual subproblems (7) are different, compute a solution suggestion heuristically $\hat{\mathbf{x}}(P) = Heu(\mathbf{x}_1, \ldots, \mathbf{x}_\Omega)$. Compute $z(\hat{\mathbf{x}})$ from (2) and (3), and set $z^* := min\{z^*, z(\hat{\mathbf{x}})\}$, delete all $P' \in \mathcal{P}$ with $z_{LD}(P') \geq z^*$.

5. Branching: Select a component x_k of $\mathbf{x}$ and add two new problems to $\mathcal{P}$ that differ from P by the additional constraints $x_k \leq \lfloor \hat{x}_k \rfloor$ and $x_k \geq \lfloor \hat{x}_k \rfloor + 1$ (if x_k is integer), or $x_k \leq x_k - \epsilon$ and $x_k \geq x_k + \epsilon$, where $\epsilon > 0$. Go to 3.

V. 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 [5]. 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). The availability of raw materials and the product storage capacity are assumed to be unlimited. The preparation stage is not limiting the production process and is thus neglected in the sequel.

The polymerization stage operates in batch mode. The production of each batch is controlled by a recipe. 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. The duration of a polymerization is the same for all recipes. After the polymerization of a batch is finished, this batch is directly transferred to the corresponding mixer of the finishing stage A or B. The mixers are semi-continuous storage tanks, the finishing lines operate continuously. If a mixing tank runs empty, the corresponding finishing line has to be shut down temporarily. After a shutdown, the line has to stopped for a certain period of time. The objective is to maximize the profit calculated from sales revenues, production costs, storage costs and penalties for lateness and for finishing line start-ups and shut-downs. The main production decisions are the discrete choice of recipes and their starting times.

The aggregated scheduling model [5] is a multi-period model considering a scheduling horizon of $i \in I = \{1, \ldots, 7\}$ periods; each period corresponds to two days. The production decisions are modelled by integer variables denoting the numbers of batches according to recipe r_p to be started in period i: $N_{i,r_p} \in \{0, \ldots, N^{max} = 12\}$. The first stage decisions $\mathbf{x}$ consist of all N_{i,r_p} with $i \in \{1, \ldots, 3\}$. The remaining production decisions together with all continuous variables including sales, storages etc. form the set of mixed-integer second-stage variables. The uncertainty is represented by $\Omega = 16$ demand scenarios of equal probability π_ω. The resulting 2SIP is characterized by: integer first stage, mixed-integer second stage, uncertainty only in the right-hand side $\mathbf{h}_\omega$.

The monolithic MILP corresponding to (1) contains 5601 real variables, 1586 integer variables, and 4083 constraints. Each subproblem corresponding to a second stage value function $Q_\omega(\mathbf{x})$ in (3) contains 321 real variables, 68 integer variables, and 228 constraints. From a practical point of view, the CPU-time is limited to 4 hours.

VI. Implementation and experimental setup

The algorithmic framework of EA/SIP was implemented in C/C++ using *TEA* [12], a library for the design of standard and non-standard EAs. For the MILP subproblems GAMS/CPLEX 8.1 [9] is used. The calculations were performed on a 2.4 GHz Linux machine.

The strategy parameters of the ES were set to $\mathbf{s}_{ES} = (\mu = 10, \lambda = 70, \kappa = 5, \sigma_{init} = 1.2)$ and the termination criterion of the value function subproblems (3) is set to a relative integrality gap of 1%. The EA/SIP was initialized (I) by randomly generated feasible first stage solutions and (II) by the best μ solutions of the wait-and-see problem (WS) [4], i.e. (5) with dropped nonanticipativity constraints.

For the dual decomposition algorithm [2] we used the C implementation DDSIP [16]. The Lagrangian dual problem is solved by a conic bundle algorithm. The CPLEX 9.03 callable library [9] is used to solve the mixed-integer subproblems in the branch-and-bound tree. The calculations were performed on a 3.06 GHz Linux machine.

In the root node of DDSIP, the Lagrangian dual (6) is solved to obtain an initial lower bound z_{LD} and values of the Lagrange multipliers $\boldsymbol{\lambda}$. For all subsequent nodes, the lower bounds are calculated from (7). The termination criterion of CPLEX is set to a relative optimality gap of 1%. The initialization was done by the solution of the expected value (EV) problem. The EV problem consists of a single scenario where the random parameters are replaced by their expected value [4].

VII. Numerical results

In Fig. 3, the objective values of the best feasible solutions found in the course of the computations are plotted over the CPU-time. The line denoted by 'CPLEX' shows the solution of the example problem in form of a monolithic MILP (1) using CPLEX [9]. DDSIP denotes the result of the dual decomposition algorithm with the best heuristic found in several experiments, namely *Take the average of all subsolutions, round the integer components to the next integer*.

DDSIP requires about $0.7h$ of CPU-time to compute a lower bound by solving the Lagrangian dual in the root node. Then the branch-and-bound procedure starts.

For the given problem instance, both algorithms generate solutions with good values of the objective

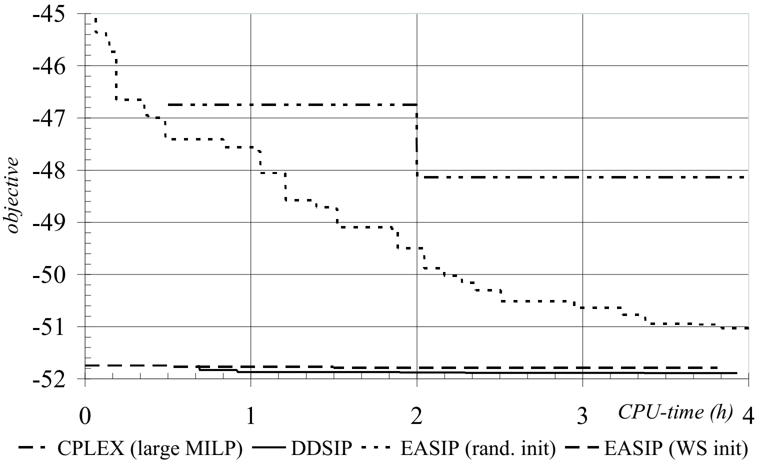

Figure 3: Best solutions (objective) versus CPU-time of CPLEX, DDSIP, and EA/SIP.

Table 1: Best solution found by EA/SIP and DDSIP after $4h$ of CPU-time.

Algor.	best first stage solution $\mathbf{x}$ period $i = 1$	period $i = 2$	period $i = 3$	z^*
EA/SIP	0 0 4 0 2 6 0 0 0 0	0 2 0 0 5 2 0 1 0 2	0 1 0 0 6 2 0 3 0 0	-51.79
DDSIP	0 0 3 0 2 7 0 0 0 0	0 1 1 0 5 2 0 1 0 2	0 2 0 0 5 1 0 4 0 0	-51.89

with respect to the requirements of the application problem. The EA/SIP algorithm with WS initialization has already found a good solution before DDSIP has finished solving the Lagrangian dual. The solution vector of the first stage variables are not identical what leads to a small difference in the objective of $\Delta z = 0.1$ (see Tab. 1).

A detailed view on the course of computation is given in Fig. 4 where the evolution of the lower bound during the solution process of the Lagrangian dual in the DDSIP root node is also displayed. After $0.7h$, the branch and bound algorithm in DDSIP starts and computes upper bounds.

The performance of a branch-and-bound algorithm depends on efficient pruning of the search space which requires tight lower bounds (see. e.g. [8]). The initial lower bound of DDSIP is obtained from the Lagrangian dual which is computationally expensive. On the other hand the dual bound is much tighter than the lower bound obtained by integer relaxations e.g. at the root node after 20 seconds, $z_{LP} = -62.22$ and $z_{LD} = -52.60$.

The observations have shown, that the EA/SIP Algorithm is capable of solving the stochastic program much faster than CPLEX and in a similar range of the CPU-time and solution quality as DDSIP. From the random initial solution it can be seen that EA/SIP works also well without knowledge of the optimal solution of particular scenarios or lower bounds. For both decomposition approaches, the efficiency depends on a careful tradeoff between the termination criterion of the subproblems and the algorithmic parameters of the master algorithm.

VIII. Summary and conclusions

In this paper, two conceptually different decomposition algorithms for 2SIPS were presented and compared for a real world example. The EA/SIP approach of Till et. al. [1] is based on stage wise (primal)

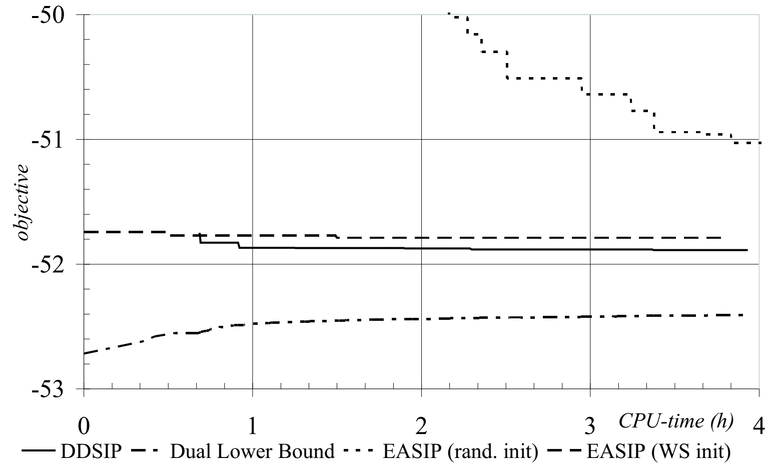

Figure 4: A detailed view on the course of the computation and the dual lower bound ($z_{LP} = -62.22$).

decomposition combined with randomized search techniques. The algorithm of Carøe and Schultz [2] is based on scenario (dual) decomposition combined with a rigorous search technique. It was shown that both decomposition algorithms perform considerably better than an algorithm that does not exploit the special structure of 2SIPs. Both decomposition based algorithms find similar solutions.

The decomposition approaches are complementary. The dual decomposition obtains tight lower bounds while the ES can be seen as self-adjusting heuristic. A combination of both, e.g. a dynamic reduction of the ES search space based on tight bounds may lead to an improved decomposition algorithm and is subject to current research. Further current research addresses the investigation of the of relationship between the initialization, the stochastic properties of the 2SIP model, and the performance of the algorithms.

Acknowledgment

We gratefully acknowledge the fruitful cooperation with Eva Clostermann and Rüdiger Schultz of the Department of Mathematics, Universität Duisburg-Essen, Germany, and with Michael Emmerich of the Leiden Institute of Advanced Computer Science, University of Leiden, The Netherlands, and with Lutz Schönemann of the Chair of Systems Analysis (LS 11), Department of Computer Science, Universität Dortmund, Germany.

The financial support by the German Research Foundation (DFG) for the Collaborative Research Center (Sonderforschungsbereich) *Design and Management of Complex Technical Processes and Systems by Means of Computational Intelligence Methods* (SFB 531) at the Universität Dortmund is gratefully acknowledged.

References

[1] Till, J., Sand, G., Engell, S., Emmerich, M., Schönemann, L.: A hybrid algorithm for solving two-stage stochastic integer problems by combining evolutionary algorithms and mathematical programming methods. Volume 20A of Computer-Aided Chemcal Engineering., Amsterdam, Elsevier (2005) 187–192

[2] Carøe, C., Schultz, R.: Dual decomposition in stochastic integer programming. Operations Research Letters **24** (1999) 37–45

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

[4] Birge, J., Louveaux, F.: Introduction to Stochastic Programmig. Springer (1997)

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

[6] Engell, S., Märkert, A., Sand, G., Schultz, R.: Aggregated scheduling of a multiproduct batch plant by two-stage stochastic integer programming. Optimization and Engineering **5** (2004) 335–359

[7] Carøe, C.C., Tind, J.: L-shaped decomposition of two-stage stochastic programs with integer recourse. Mathematical Programming **83** (1998) 451–464

[8] Nemhauser, G., Wolsey, A.: Integer and Combinatorial Optimization. Wiley, New York (1999)

[9] CPLEX: Using the CPLEX Callable Library. ILOG Inc., Mountain View, CA. (2002)

[10] Louveaux, F.V., Schultz, R.: Stochastic Integer Programming. In:. Volume 10 of Handbooks in Operations Research and Management Science. Amsterdam (2003)

[11] Beyer, H., Schwefel, H.: Evolution strategies. Natural Computing (2002) 3–52

[12] Emmerich, M., Schütz, M., Groß, B., Grötzner, M.: Mixed-integer evolution strategy for chemical plant optimization with simulators. In Parmee, I., ed.: Evolutionary Design and Manufacture (ACDM), New York, Springer (2000) 55–67

[13] 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 edn. Springer, Berlin (1994) 193–197

[14] Coello Coello, C.A.: Theoretical and numerical constraint handling techniques used with evolutionary algorithms: A survey of the state of the art. Computer Methods in Applied Mechanics and Engineering **191** (2002) 1245–1287

[15] Deb, K.: Multi-Objective Optimization using Evolutionary Algorithms. Wiley-Interscience Series in Systems and Optimization. WILEY, Chichester (2001)

[16] Märkert, A.: User's guide to DDSIP - A C package for the Dual Decomposition of Stochastic Programs with Mixed-Integer Recourse. Institut für Mathematik, Universität Duisburg, Duisburg. (2004) (http://www.uni-duisburg.de/FB11/disma/ddsip-eng.shtml, March 23, 2005).

Jochen Till is a research associate in the Department of Biochemical and Chemical Engineering, Universität Dortmund, Germany. He received his Dipl.-Ing. from RWTH Aachen in 2001. His research interests are chemical batch scheduling, evolutionary algorithms, and process optimization.

Sebastian Engell is a Full Professor in the Department of Biochemical and Chemical Engineering of Universität Dortmund and heads the Process Control Laboratory. He also is Vice-Rector for Research and International Relations of Universität Dortmund. His research interests are chemical process operations and control.

Guido Sand is a senior researcher at the Department of Biochemical and Chemical Engineering, Universität Dortmund, Germany. He received his Dr.-Ing. degree from Universität Dortmund in 2003. His current research is on the application of optimization techniques to the design and logistics of chemical processes.