Quantified Combinatorial Optimization

Thorsten Ederer and Ulf Lorenz and Thomas Opfer

Abstract MIP and IP programming are state-of-the-art modeling techniques for computer-aided optimization. However, companies observe an increasing danger of disruptions that prevent them from acting as planned. One reason is input data being assumed as deterministic, but in reality, data is afflicted with uncertainties. Incorporating uncertainty in existing models, however, often pushes the complexity of problems that are in P or NP, to the complexity class PSPACE. Quantified integer linear programming (QIP) is a PSPACE-complete extension of the IP problem with variables being either existentially or universally quantified. With the help of QIPs, it is possible to model board-games like Gomoku as well as traditional combinatorial OR problems under uncertainty. In this paper, we present how to extend the model formulation of classical scheduling problems like the Job-Shop and Car-Sequencing problem by uncertain influences and give illustrating examples with solutions.

1 Introduction

In the past, there have been several efforts to add uncertainties into the model description of problems [1, 7]. We were involved into examinations of uncertainties in airline fleet assignment [6] and railway planning [2]. However, the ratio of implementing efforts to output was rather disappointing. Therefore, we came to the conclusion that a modeling language is needed that combines convenient MIP modeling with the ability to express uncertainties. In 2004, Subramani introduced the idea to enrich linear programs by universally quantified variables [8].

Thorsten Ederer, M. Sc. / Dipl.-Math. Thomas Opfer
TU Darmstadt, Discrete Optimization, Dolivostraße 15, 64293 Darmstadt
e-mail: ederer/opfer@mathematik.tu-darmstadt.de

PD Dr. Ulf Lorenz
TU Darmstadt, Fluid Systems Techology, Magdalenenstraße 4, 64289 Darmstadt
e-mail: ulf.lorenz@fst.tu-darmstadt.de

Definition 1 (Quantified Integer Program). Let $x = (x_1, \ldots, x_n)^T \in \mathbb{Q}^n$ be a vector with lower and upper bound vectors $l \in \mathbb{Z}^n$ and $u \in \mathbb{Z}^n$ such that $l_i \leq x_i \leq u_i$. $A \in \mathbb{Q}^{m \times n}$, $b \in \mathbb{Q}^m$ and x build a linear inequality system. Moreover, there is a vector of quantifiers $\mathcal{Q} = (\mathcal{Q}_1, \ldots, \mathcal{Q}_n)^T \in \{\forall, \exists\}^n$. Let $\mathcal{Q} \circ x \in [l,u] \cap \mathbb{Z}^n$ with the componentwise binding operator $\circ$ denote the *quantification vector* $(\mathcal{Q}_1 x_1 \in [l_1, u_1] \cap \mathbb{Z}, \ldots, \mathcal{Q}_n x_n \in [l_n, u_n] \cap \mathbb{Z})^T$ such that every quantifier $\mathcal{Q}_i$ binds the variable x_i. Let there also be a vector of objective coefficients $c \in \mathbb{Q}^n$. Each maximal consecutive subsequence of $\mathcal{Q}$ consisting of identical quantifiers is called a *quantifier block* – the corresponding i-th subsequence of x is called a *variable block* B_i.

$$\text{We call} \quad z = \min_{B_1}(c^1x^1 + \max_{B_2}(c^2x^2 + \min_{B_3}(c^3x^3 + \max_{B_4}(\ \ldots \min_{B_k} c^k x^k))))) \quad \begin{array}{c} \\ \mathcal{Q} \circ x \in [l,u] \cap \mathbb{Z}^n : Ax \leq b \end{array} \tag{QIP}$$

a *Quantified Integer Program with objective function* (for a minimizing existential player). Here we assume w.l.o.g. that $\mathcal{Q}_1 = \exists$ and $\mathcal{Q}_n = \exists$.

A QIP with objective can be interpreted as a two-person zero-sum game [3, 4] between a max-player who tries to make the instance infeasible or to maximize the objective function and a min-player who wants to make the instance feasible and to minimize the objective against all odds. Note that this is a short notation for a dynamic program where the players have recursively to find optimal vectors x_{B_i} with fixed $x_{B_1}, \ldots, x_{B_{i-1}}$ under consideration that the other player will set the next block of variables optimal concerning his own incentives.

2 Job Shop Scheduling

Jop Shop Scheduling is a classical optimization problem in which jobs have to be assigned to several machines such that the total time until all jobs are finished (the *makespan*) is minimized. An assignment that a job has to be processed by a machine is called a task and is given a certain duration. If some tasks depend on one another, a full or partial order of tasks can be given.

Equation (2) defines the makespan. Equation (3) ensures the partial task order. The ordering indicator variables $y_{i,m,j}$ are defined by the following two equations.

For solution purposes it is relevant to use as few universally quantified variables as possible. To that end, we introduce a binary encoding $\tilde{r}$ of the retardation and add existentially quantified helper variables r as unary encoding of the retardation. Equations (6) represent a linear formulation of this translation.

Equations (7) to (10) are an adaption of the prior constraints for the second stage variables with uncertainly prolonged task durations. Equations (11) to (13) define the earliness caused by the existential players reaction, which is used as a penalty term for large rearrangements of the first stage planning.

$$\min \quad m^2 + k \cdot \bar{e} + \frac{1}{M} \cdot m^1 \qquad \text{s.t.} \qquad \exists\, s^1\, y^1\, m^1\, \forall\, \tilde{r}\ \exists\, r\, w,\, s^2\, y^2\, m^2,\, e: \tag{1}$$

$$s^1_{j,m} + d_{j,m} \leq m^1 \qquad \forall\, (j,m) \in T \tag{2}$$

$$s^1_{i,m} + d_{i,m} \leq s^1_{j,n} \qquad \forall\, (i,m,j,n) \in O \tag{3}$$

$$s^1_{i,m} + d_{i,m} \leq s^1_{j,m} + M \cdot (1 - y^1_{i,m,j}) \qquad \forall\, (i,m) \in T,\ (j,m) \in T \tag{4}$$

$$s^1_{j,m} + d_{j,m} \leq s^1_{i,m} + M \cdot y^1_{i,m,j} \qquad \forall\, (i,m) \in T,\ (j,m) \in T \tag{5}$$

$$\sum_{(u,j,m) \in U} u \cdot r_{j,m} = \sum_{b \in B} 2^b \cdot \tilde{r}_b - |T| \cdot w \quad \wedge \quad \sum_{\substack{u \in U \\ (j,m) = T_u}} r_{j,m} \leq 1 \tag{6}$$

$$s^2_{j,m} + d_{j,m} + \delta_{j,m} \cdot r_{j,m} \leq m^2 \qquad \forall\, (j,m) \in T \tag{7}$$

$$s^2_{i,m} + d_{i,m} + \delta_{i,m} \cdot r_{i,m} \leq s^2_{j,n} \qquad \forall\, (i,m,j,n) \in O \tag{8}$$

$$s^2_{i,m} + d_{i,m} + \delta_{i,m} \cdot r_{i,m} \leq s^2_{j,m} + M \cdot (1 - y^2_{i,m,j}) \qquad \forall\, (i,m) \in T,\ (j,m) \in T \tag{9}$$

$$s^2_{j,m} + d_{j,m} + \delta_{i,m} \cdot r_{i,m} \leq s^2_{i,m} + M \cdot y^2_{i,m,j} \qquad \forall\, (i,m) \in T,\ (j,m) \in T \tag{10}$$

$$e_{i,m} \geq s^1_{i,m} - s^2_{i,m} \qquad \forall\, (i,m) \in T \tag{11}$$

$$e_{i,m} \geq 0 \tag{12}$$

$$\bar{e} = \frac{1}{|T|} \cdot \sum_{(i,m) \in T} e_{i,m} \tag{13}$$

An example[1] is given in tables 2 and 3. Table 4 depicts an optimal solution. The first-stage scheduling has the property that the planner can find a rescheduling to each possible redardation such that the worst-case makespan is minimized.

3 Car Sequencing

In flexible manufacturing systems, varying models of same basic product are produced. They usually require different processing times, so sequences which alternately produce different models are preferable. We consider so called $r_k : s_k$ sequencing rules [5] that restrict too frequent production of work intensive models at certain stations, that is, option k may only be produced at most r_k times per each s_k successively sequenced models.

We add uncertainty to this problem by incorporating a malfunction in the production process. It may happen that a car cannot be processed in the prescheduled order and has to be reinserted a few timesteps later after the malfunction has been corrected. The uncertainty is given by a tuple (t,t'), $t < t'$ with the new timestep t' at which the model originally scheduled at t will be processed. The cars inbetween will each be processed one timestep earlier. The resulting schedule is modeled by stage two ($s = 2$) variables – note that they may be chosen differently for each possible malfunction. The planer may react to this uncertainty by rescheduling yet another

[1] The model formulation and example data are adapted from the work of Jeffrey Kantor, Christelle Gueret, Christian Prins and Marc Sevaux, cf. http://estm60203.blogspot.com/ .

Table 1: Jobshop model notation.

J	set of jobs
M	set of machines
T	set of tasks, $T \subseteq J \times M$
O	taskorder, $O \subseteq T \times T$
$s_{j,m}$	start time (integer) of task (j,m)
$d_{j,m}$	duration of task (j,m)
$\delta_{j,m}$	additional duration of task (j,m) in case of delay
$e_{j,m}$	earliness of task (j,m), i.e., $e = \max\{d^1 - d^2, 0\}$
$\bar{e}$	mean earliness
m	makespan
$r_{u,j,m}$	indicator of unary encoding of retarded task
$\tilde{r}_b$	indicator of binary encoding retarded task
w	wrapping indicator for binary to unary translation

Table 2: Jobshop tasks.

job	machine	duration	extra
Paper1	Blue	45	5
Paper1	Yellow	10	0
Paper2	Blue	20	5
Paper2	Green	10	10
Paper2	Yellow	34	0
Paper3	Blue	12	0
Paper3	Green	17	0
Paper3	Yellow	28	20

Table 3: Jobshop order.

prior task		later task	
Paper1	Blue	Paper1	Yellow
Paper2	Green	Paper2	Blue
Paper2	Blue	Paper2	Yellow
Paper3	Yellow	Paper3	Blue
Paper3	Blue	Paper3	Green

Table 4: Solution of the Jobshop Example.

sc.	start times								m.s.
	1B	1Y	2B	2G	2Y	3Y	3B	3G	
	first stage solution								
	0	45	0	45	65	0	70	82	99
1	0	50	0	50	70	0	70	82	104
2	0	45	0	45	65	0	65	82	99
3	0	45	0	45	70	0	70	82	104
4	0	45	0	45	65	0	65	82	99
5	0	45	0	45	65	0	65	82	99
6	0	45	0	45	65	0	65	82	99
7	0	45	0	45	65	0	70	82	99
8	0	48	0	50	70	0	75	87	104

Table 5: Car Instance.

opt	r	s
1	1	2
2	2	3

class	cars	opt 1	opt 2
0	2	0	0
1	3	0	1
2	1	1	0
3	4	1	1

model, i.e., he chooses a tuple (u,u'), $t' < u < u'$ such that the car which was originally scheduled at u will be processed at u'. This final reschedule is given by stage three variables.

The first three equations and the first stage variables ($s = 1$) give a formulation of the original problem without uncertainty. Equation (15) ensures that for each class c the produced amount equals the given demand D_c. Equation (16) specifies that exactly one unit is produced in each time step. The $r_k : s_k$ sequencing rules are not strictly enforced – instead violations are counted by the indicator variables y^s_{k,t_0} in Equation (17).

Similar to the job shop model, we introduce a binary encoding $\tilde{m}$ and helper variables m for the uncertain machine malfunctions.

Given unary encodings of the malfunction m_u and the answer a_u, we can encode the change of schedule with constraints similar to equation (19) (which model which

parts of the schedule do *not* change) and further constraints which ensure that the cars are processed in the correct new order. These equations are rather long but not very insightful, so we skip them here.

$$\min \quad \sum_{k\in O}\sum_{t_0\in T^k} y^1_{k,t_0} \qquad \text{s.t.} \qquad \exists\, x^1\, y^1\, \forall\, \tilde{m}\ \exists\, m\, w\,,\, x^2\, y^2\,,\, a\,,\, x^3\, y^3 : \tag{14}$$

$$\sum_{t\in T} x^s_{t,c} = D_c \qquad \forall\, c\in C\,,\, s\in S \tag{15}$$

$$\sum_{c\in C} x^s_{t,c} = 1 \qquad \forall\, t\in T\,,\, s\in S \tag{16}$$

$$\sum_{t=t_0}^{t_0+s_k}\sum_{c\in C} A_{k,c}\cdot x^s_{t,c} \le r_k + M\cdot y^s_{k,t_0} \qquad \forall\, k\in O\,,\, t_0\in T^k\,,\, s\in S \tag{17}$$

$$\sum_{u\in U} u\cdot m_u = \sum_{b\in B} 2^b\cdot \tilde{m}_b - |F|\cdot w \quad \wedge \quad \sum_{u\in U} m_u \le 1 \tag{18}$$

$$|x^2_{t,c} - x^1_{t,c}| \le \sum_{\substack{u\in U\\ (t_i,t_j)=F_u\\ t_i<t<t_j}} m_u \quad \forall\, c\in C\,,\, t\in T \tag{19}$$

$$\textit{further stage-connecting constraints}\ldots \tag{20}$$

An example is given by table 5 and table 7 shows an optimal solution. The first-stage solution has the property, that the production planner can respond (column 3) to each possible malfunction (column 2) such that the second-stage production sequence has a worst-case minimal number of violated sequencing constraints.

4 Conclusion

We presented Quantified Integer Programming as an intuitive modelling language to generate recoverable robust solutions for classical scheduling problems that were extended by various uncertain influences.

References

[1] A. Ben-Tal, L. El Ghaoui, and A. Nemirovski. *Robust Optimization*. Princeton University Press, 2009.

[2] A. Berger, R. Hoffmann, U. Lorenz, and S. Stiller. Online railway delay management: Hardness, simulation and computation. *Simulation*, 87(7):616–629, 2011.

[3] T. Ederer, U. Lorenz, A. Martin, and J. Wolf. Quantified linear programs: A computational study. In *Part I*, ESA'11, pages 203–214. Springer, 2011.

[4] T. Ederer, U. Lorenz, T. Opfer, and J. Wolf. Modelling games with the help of quantified integer linear programs. In *ACG 13*. Springer, 2011.

[5] Malte Fliedner and Nils Boysen. Solving the car sequencing problem via branch & bound. *European Journal of Operational Research*, 191(3):1023–1042, 2008.

[6] S. Grothklags, U. Lorenz, and B. Monien. From state-of-the-art static fleet assignment to flexible stochastic planning of the future. In *Algorithmics of Large and Complex Networks*, pages 140–165, 2009.

[7] C. Liebchen, M.E. Lübbecke, R.H. Möhring, and S.Stiller. The concept of recoverable robustness, linear programming recovery, and railway applications. *Robust and online large-scale optimization*, pages 1–27, 2009.

[8] K. Subramani. Analyzing selected quantified integer programs. *Springer, LNAI 3097*, pages 342–356, 2004.

Table 6: Notation of the car sequencing model.

O	set of options
C	set of classes, $C \subseteq \mathcal{P}(O)$
T	set of timesteps ($\|T\|$ equals number of models)
T^k	set of intervals (by first timestep) for option k , $T^k = \{1, \ldots, \|T\| - s_k + 1\}$
$r_k : s_k$	at most r_k out of s_k successively sequenced models may require option k
D_c	demand of models of class c
$A_{k,c}$	indicator, if models of class c require option k
$x_{t,c}$	indicator, if a model of class c is produced at timestep t
y_{k,t_0}	indicator, if the sequencing rule $r_k : s_k$ beginning at timestep t_0 is satisfied
S	set of stages (duplicates of the original variables after each move)
σ	stage index: 1 pre-scheduling, 2 state after delay, 3 re-scheduling
F	ordered set of possible delays, $F = \{(t,t') \in T \times T,\ t < t'\}$
U	unary encoding vector of F
d_u	indicator of unary encoding for the delay from timestep t to t', $(t,t') \in U$
B	binary encoding of possible delays
$\tilde{d}_b$	indicator of binary encoding for the delay, $b \in B$
w	wrapping indicator for binary to unary translation

Table 7: Solution of the Car Sequencing Example.

Scenario	Mal.	Ans.	Production Sequence
first stage solution:			1, 3, 0, 3, 0, 3, 1, 2, 1, 3
1	–	(4, 5)	1, 3, 0, 3, 3, 0, 1, 2, 1, 3
2	(0, 1)	(2, 7)	3, 1, 3, 0, 3, 1, 2, 0, 1, 3
3	(0, 2)	–	3, 0, 1, 3, 0, 3, 1, 2, 1, 3
4	(0, 3)	–	3, 0, 3, 1, 0, 3, 1, 2, 1, 3
5	(0, 4)	–	3, 0, 3, 0, 1, 3, 1, 2, 1, 3
6	(0, 5)	(6, 8)	3, 0, 3, 0, 3, 1, 2, 1, 1, 3
7	(0, 6)	(7, 8)	3, 0, 3, 0, 3, 1, 1, 1, 2, 3
8	(0, 7)	(8, 9)	3, 0, 3, 0, 3, 1, 2, 1, 3, 1
9	(0, 8)	–	3, 0, 3, 0, 3, 1, 2, 1, 1, 3
10	(0, 9)	–	3, 0, 3, 0, 3, 1, 2, 1, 3, 1
11	(1, 2)	–	1, 0, 3, 3, 0, 3, 1, 2, 1, 3
12	(1, 3)	–	1, 0, 3, 3, 0, 3, 1, 2, 1, 3
13	(1, 4)	–	1, 0, 3, 0, 3, 3, 1, 2, 1, 3
14	(1, 5)	–	1, 0, 3, 0, 3, 3, 1, 2, 1, 3
15	(1, 6)	–	1, 0, 3, 0, 3, 1, 3, 2, 1, 3
16	(1, 7)	–	1, 0, 3, 0, 3, 1, 2, 3, 1, 3
17	(1, 8)	–	1, 0, 3, 0, 3, 1, 2, 1, 3, 3
18	(1, 9)	–	1, 0, 3, 0, 3, 1, 2, 1, 3, 3
19	(2, 3)	(4, 9)	1, 3, 3, 0, 3, 1, 2, 1, 3, 0
20	(2, 4)	–	1, 3, 3, 0, 0, 3, 1, 2, 1, 3
21	(2, 5)	–	1, 3, 3, 0, 3, 0, 1, 2, 1, 3
22	(2, 6)	–	1, 3, 3, 0, 3, 1, 0, 2, 1, 3
23	(2, 7)	–	1, 3, 3, 0, 3, 1, 2, 0, 1, 3
24	(2, 8)	–	1, 3, 3, 0, 3, 1, 2, 1, 0, 3
25	(2, 9)	–	1, 3, 3, 0, 3, 1, 2, 1, 3, 0
26	(3, 4)	–	1, 3, 0, 0, 3, 3, 1, 2, 1, 3
27	(3, 5)	–	1, 3, 0, 0, 3, 3, 1, 2, 1, 3
28	(3, 6)	–	1, 3, 0, 0, 3, 1, 3, 2, 1, 3
29	(3, 7)	–	1, 3, 0, 0, 3, 1, 2, 3, 1, 3
30	(3, 8)	–	1, 3, 0, 0, 3, 1, 2, 1, 3, 3
31	(3, 9)	–	1, 3, 0, 0, 3, 1, 2, 1, 3, 3
32	(4, 5)	–	1, 3, 0, 3, 3, 0, 1, 2, 1, 3
33	(4, 6)	–	1, 3, 0, 3, 3, 1, 0, 2, 1, 3
34	(4, 7)	(8, 9)	1, 3, 0, 3, 3, 1, 2, 0, 3, 1
35	(4, 8)	–	1, 3, 0, 3, 3, 1, 2, 1, 0, 3
36	(4, 9)	–	1, 3, 0, 3, 3, 1, 2, 1, 3, 0
37	(5, 6)	(7, 9)	1, 3, 0, 3, 0, 1, 3, 1, 3, 2
38	(5, 7)	–	1, 3, 0, 3, 0, 1, 2, 3, 1, 3
39	(5, 8)	–	1, 3, 0, 3, 0, 1, 2, 1, 3, 3
40	(5, 9)	–	1, 3, 0, 3, 0, 1, 2, 1, 3, 3
41	(6, 7)	(8, 9)	1, 3, 0, 3, 0, 3, 2, 1, 3, 1
42	(6, 8)	–	1, 3, 0, 3, 0, 3, 2, 1, 1, 3
43	(6, 9)	–	1, 3, 0, 3, 0, 3, 2, 1, 3, 1
44	(7, 8)	–	1, 3, 0, 3, 0, 3, 1, 1, 2, 3
45	(7, 9)	–	1, 3, 0, 3, 0, 3, 1, 1, 3, 2
46	(8, 9)	–	1, 3, 0, 3, 0, 3, 1, 2, 3, 1