Entropy **2013**, *15*, 3592-3601; doi:10.3390/e15093592

OPEN ACCESS

entropy

ISSN 1099-4300

www.mdpi.com/journal/entropy

Article

A Discrete Meta-Control Procedure for Approximating Solutions to Binary Programs

Pengbo Zhang , Wolf Kohn * and Zelda B. Zabinsky

Department of Industrial and Systems Engineering, University of Washington, Seattle, WA 98195, USA; E-Mails: pbzhang@u.washington.edu (P.Z.); zelda@u.washington.edu (Z.B.Z.)

* Author to whom correspondence should be addressed; E-Mail: wolfk@u.washington.edu; Tel.: +1-206-790-0624; Fax: +1-206-685-3702.

Received: 15 June 2013; in revised form: 30 August 2013 / Accepted: 30 August 2013 / Published: 4 September 2013

Abstract: Large-scale binary integer programs occur frequently in many real-world applications. For some binary integer problems, finding an optimal solution or even a feasible solution is computationally expensive. In this paper, we develop a discrete meta-control procedure to approximately solve large-scale binary integer programs efficiently. The key idea is to map the vector of n binary decision variables into a scalar function defined over a time interval $[0, n]$ and construct a linear quadratic tracking (LQT) problem that can be solved efficiently. We prove that an LQT formulation has an optimal binary solution, analogous to a classical bang-bang control in continuous time. Our LQT approach can provide advantages in reducing computation while generating a good approximate solution. Numerical examples are presented to demonstrate the usefulness of the proposed method.

Keywords: large-scale binary integer programs; linear quadratic tracking; optimal control

1. Introduction

Many decision problems in economics and engineering can be formulated as binary integer programming (BIP) problems. These BIP problems are often easy to state but difficult to solve due to the fact that many of them are NP-hard [1], and even finding a feasible solution is considered NP-complete [2,3]. Because of their importance in formulating many practical problems, BIP algorithms have been widely studied. These algorithms can be classified into exact and approximate algorithms as follows [4]:

(1) Exact algorithms: The exact algorithms are guaranteed either to find an optimal solution or prove that the problem is infeasible, but they are usually computationally expensive. Major methods for BIP problems include branch and bound [5], branch-and-cut [6], branch-and-price [7], dynamic programming methods [8], and semidefinite relaxations [9].

(2) Approximate algorithms: The approximate algorithms are used to achieve efficient running time with a sacrifice in the quality of the solution found. Examples of well-known metaheuristics, as an approximate approach, are simulated annealing [10], annealing adaptive search [11], cross entropy [12], genetic algorithms [13] and nested partitions [14]. Moreover, many hybrid approaches that combine both the exact and approximate algorithms have been studied to exploit the benefits of each [15]. For additional references regarding large-scale BIP algorithms, see [1,16–18].

Another effective heuristic technique that transforms discrete optimization problems into problems falling in the control theory and information theory or signal processing domains has also been studied recently. In [19,20], circuit related techniques are used to transform unconstrained discrete quadratic programming problems and provide high quality suboptimal solutions. Our focus is on problems with linear objective functions, instead of quadratic, and linear equality constraints, instead of unconstrained.

In our previous work [21], we introduced an approach to approximating a BIP solution using continuous optimal control theory, which showed promise for large-scale problems. The key innovation to our optimal control approach is to map the vector of n binary decision variables into a scalar function defined over a time interval $[0, n]$ and define a linear quadratic tracking (LQT) problem that can be solved efficiently. In this paper, we use the same mapping, but instead of solving the LQT problem in continuous time, we explore solving the LQT problem in discrete time, because the time index in our reformulation of the BIP represents the dimension of the problem, $\{0, 1, \ldots, n\}$, and a discrete time approach more accurately represents the partial summing reformulation than the continuous approach. In addition, in our previous work, the transformation into a continuous LQT problem was based on a reduced set of constraints, and a least squares approach was used to estimate the error due to the constraint reduction. The algorithm iteratively solved the LQT problem and the least squares problem until convergence conditions were satisfied. In this paper, instead of iteratively solving the LQT problem based on a reduced set of constraints, we solve the LQT problem only once for the full state space. This approach improves the flow of information for convergence.

We have chosen a quadratic criterion for our approach because its formalism includes a measure of the residual entropy of the dynamics of the algorithm as it computes successive approximation to a solution. Because of the mapping used in our algorithm, the information measure is given by the inverse of the Riccati equation that we solve. That inverse of the solution of the Riccati equation is a Fisher information matrix of the algorithm as a dynamical system [22,23]. The information from the algorithm in the criterion determines the quality of the solution.

The computational complexity for solving the LQT problem is polynomial in the time horizon, the dimension of the state space and the number of control variables. In our LQT problem, the time horizon is n, the dimension of the state space is the number of constraints m, and the number of control variables is 1. Our meta-control approach solves the LQT problem to obtain an efficient approximate solution to the original BIP problem.

In Section 2, our approach is presented in detail, and numerical results are given in Section 3. In Section 4, we state the conclusions of this work.

2. Development of the Meta-Control Algorithm for BIP Problems

The original BIP problem is:

Problem 1.

$$\min_{\substack{u_j \\ j=0,\ldots,n-1}} \quad \sum_{j=0}^{n-1} \tilde{c}_j u_j \tag{1}$$

$$\text{s.t.} \quad \sum_{j=0}^{n-1} \tilde{a}_{ij} u_j = \tilde{b}_i \quad i = 1,\ldots,m \tag{2}$$

$$u_j \in \{0,1\} \qquad j = 0,\ldots,n-1 \tag{3}$$

where u_j for $j = 0,\ldots,n-1$ are binary decision variables. We assume $\tilde{c}_j, \tilde{a}_{ij}$, and $\tilde{b}_i$ are real known values for $i = 1,\ldots,m$ and $j = 0,\ldots,n-1$ and there exists at least one feasible point.

2.1. Partial Summing Formulation

We start by defining partial summing variables as in [21] from the original BIP problem as

$$f_{0,j+1} = f_{0,j} + \tilde{c}_j u_j \tag{4}$$

$$f_{i,j+1} = f_{i,j} + \tilde{a}_{ij} u_j \tag{5}$$

for $i = 1,\ldots,m$ and $j = 0,\ldots,n-1$, with initial conditions $f_{0,0} = f_{i,0} = 0$.

For ease of notation, we create a new $(m+1)\times 1$ vector $x_j = [f_{0,j}, f_{1,j},\ldots,f_{m,j}]^T$ and the i^{th} element of x_j is denoted $x_{j(i)}$ for $i = 1,\ldots,m+1$ and for $j = 0,\ldots,n$. We also define the $(m+1)\times 1$ vector $a_j = [\tilde{c}_j, \tilde{a}_{1j},\ldots,\tilde{a}_{mj}]^T$ for $j = 0,\ldots,n-1$, and the $(m+1)\times 1$ vector $b = \left[0, \tilde{b}_1,\ldots,\tilde{b}_m\right]^T$, where the i^{th} element of b is denoted $b_{(i)}$ for $i = 1,\ldots,m+1$. We define Problem 2 as follows, with initial conditions x_0 as a vector of zeros:

Problem 2.

$$\min_{\substack{u_j \\ j=0,\ldots,n-1}} \quad x_{n(1)}$$

$$\text{s.t.} \quad x_{j+1} = x_j + a_j u_j \qquad j = 0,\ldots,n-1 \tag{6}$$

$$x_0 = 0 \tag{7}$$

$$x_{n(i)} = b_{(i)} \qquad i = 2,\ldots,m+1 \tag{8}$$

$$u_j(u_j - 1) = 0 \qquad j = 0,\ldots,n-1 \tag{9}$$

Proposition 1. *Problem 2 exactly represents Problem 1.*

The proof is straight-forward; the constraints ensure feasibility and the objective function is equivalent to Problem 1.

2.2. Construct the LQT Problem

We construct an LQT problem, Problem 3, by first defining an error term, as a measure of unsatisfied constraints, an $(m+1)\times 1$ vector e_j for $j = 0,\ldots,n$, as

$$e_j = x_j - b \tag{10}$$

We develop the dynamics in terms of the measure e_j, by combining Equation (10) with Equation (6), yielding

$$e_{j+1} = e_j + a_j u_j \tag{11}$$

and note that $e_0 = -b$, given initial conditions $x_0 = 0$. The criterion is to minimize the measure of unsatisfied constraints using a terminal penalty for infeasibility and objective function value, which is given by

$$J(u) = \frac{1}{2}\sum_{j=0}^{n-1} e_j^T Q_j e_j + \frac{1}{2} e_n^T F e_n \tag{12}$$

We also relax constraint (9) with $0 \le u_j \le 1$.

The parameters Q_j and F are positive semi-definite and user-specified. The $(m+1)\times(m+1)$ matrix Q_j is used to penalize the unsatisfied constraints. The $(m+1)\times(m+1)$ matrix F is used to penalize the terminating conditions and aid in minimizing the original objective function.

We now summarize our discrete LQT problem with the criterion in Equation (12) as follows:

Problem 3.

$$\min_{\substack{u_j \\ j=0,\ldots,n-1}} \quad J(u) = \frac{1}{2}\sum_{j=0}^{n-1} e_j^T Q_j e_j + \frac{1}{2} e_n^T F e_n \tag{13}$$

$$\text{s.t.} \quad e_{j+1} = e_j + a_j u_j \quad j = 0,\ldots,n-1 \tag{14}$$

$$0 \le u_j \le 1 \quad j = 0,\ldots,n-1 \tag{15}$$

$$e_0 = -b \tag{16}$$

It is known that solving Problem 3 directly is numerically unstable [24]. However, Theorem 1 suggests an algorithmic approach to solving Problem 3, by making a discrete analog to a bang-bang control with a switching function.

Theorem 1. *Analogous to a bang-bang control in continuous time, Problem 3 has an optimal binary solution with* $u_j \in \{0,1\}$ *for discrete times* $j = 0,1,\ldots,n-1$ *with non-singular arcs.*

Proof. We first construct the Hamiltonian function [24] as follows

$$H(e_j, \lambda_{j+1}, u_j) = \frac{1}{2} e_j^T Q_j e_j + \lambda_{j+1}^T (e_j + a_j u_j) \tag{17}$$

where λ_j is the $(m+1)\times 1$ costate vector, for $j = 0,\ldots,n-1$, and it satisfies

$$\lambda_j = \lambda_{j+1} + Q_j e_j \text{ and } \lambda_n = F e_n \tag{18}$$

Let e^*, λ^* and u^* be the optimal solution, by the necessary conditions for the optimality [24], we have:
$H(e_j^*, \lambda_{j+1}^*, u_j^*) \leq H(e_j^*, \lambda_{j+1}^*, u_j)$

$$\Rightarrow \frac{1}{2}e_j^{*T}Q_je_j^* + \lambda_{j+1}^{*T}\left(e_j^* + a_ju_j^*\right) \leq \frac{1}{2}e_j^{*T}Q_je_j^* + \lambda_{j+1}^{*T}\left(e_j^* + a_ju_j\right)$$
$$\Rightarrow \lambda_{j+1}^{*T}a_ju_j^* \leq \lambda_{j+1}^{*T}a_ju_j, \quad \forall u_j \in [0,1] \tag{19}$$

Thus, we have

$$u_j^* = \begin{cases} 1 & \text{if } \lambda_{j+1}^{*T}a_j < 0 \\ \in [0,1] & \text{if } \lambda_{j+1}^{*T}a_j = 0 \\ 0 & \text{if } \lambda_{j+1}^{*T}a_j > 0 \end{cases} \tag{20}$$

□

If $\lambda_{j+1}^{*T}a_j \neq 0$, binary values for u_j^* are determined by Equation (20). When $\lambda_{j+1}^{*T}a_j = 0$, the arc is singular, and we may reintroduce constraint (9), $u_j(1-u_j) = 0$, to force a binary solution.

To get an intuitive understanding of the singularity issue, suppose all $Q_j = 0$, and the element at row 1, column 1 of matrix F equals zero. Then Problem 3 reduces to minimize the infeasibility penalty term, $\frac{1}{2}\sum_{i=1}^{m}\left[\left(\sum_{k=0}^{n-1}\tilde{a}_{ik}u_k - \tilde{b}_i\right)^2 F_i\right]$. If this term equals zero, then $e_n = 0$, satisfying all of the original constraints (2), and $\lambda_n = 0$ from Equation (18), and because $Q_j = 0$, all $\lambda_j = 0$. Then $\lambda_{j+1}^{*T}a_j = 0$ for all j. However, if Q_j and the first element of F have positive values, then $\lambda_{j+1}^{*T}a_j$ may be positive or negative and Equation (20) is useful. An auxiliary problem to determine values for Q_j and F that resolve the singularity will be explored in future research.

To create an LQT problem that is practical to solve, we introduce a penalty term $u_j(u_j-1)R_j$ in the criterion, where R_j is a Lagrangian multiplier associated with constraint (9):

Problem 4.

$$\min_{\substack{u_j \\ j=0,\ldots,n-1}} \quad \frac{1}{2}\sum_{j=0}^{n-1}\left(e_j^TQ_je_j + u_j(u_j-1)R_j\right) + \frac{1}{2}e_n^TFe_n \tag{21}$$
$$\text{s.t.} \quad e_{j+1} = e_j + a_ju_j \qquad j = 0,\ldots,n-1 \tag{22}$$
$$e_0 = -b \tag{23}$$

The optimal control for Problem 4 $\hat{u}_j$ can be solved by the standard dynamic programming method [25] (see appendix for details). The computation associated with solving Problem 4 is $O(nm^3)$. We then obtain an approximate binary solution to the original BIP problem as follows:

$$u_j^* = \begin{cases} 0 \text{ for } \hat{u}_j < 0.5 \\ 1 \text{ for } \hat{u}_j \geq 0.5 \end{cases} \tag{24}$$

for $j = 0, 1, \ldots, n-1$.

Motivated by the successive overrelaxation method [26], we introduce a weighting factor ω to improve the stability of our proposed method. Rather than applying quantization at the final step as shown in Equation (24), we did quantization at each step and propagate the binary value $\bar{u}_j$ during the dynamic programming procedure (see appendix for details). At the final step, we then replace $\hat{u}_j$ in Equation (24) with $\omega\hat{u}_j + (1-\omega)\bar{u}_j$ to get the approximate binary solution.

3. Numerical Results

We explore the limits of the algorithm with some test problems obtained from MIPLIB [27]. MIPLIB is a standard and widely used benchmark for comparing the performance of various mixed integer programming algorithms, and most of the problems in the MIPLIB arise from real-world applications. We have presented 6 tests in our numerical result section, where $air01$, $air03$, $air04$, $air05$ and $nw04$ are airline crew scheduling type problems. The dimensions and the optimal solutions for the test problems and the numerical results are shown in Table 1. The CPU time is given for a single run with branch-and-cut with CPLEX, branch-and-bound in MATLAB, and our method in MATLAB. In Table 1, the feasibility measure is the summation of the absolute differences of feasibility over all constraints, and the optimality measure is defined as $\frac{\hat{f}-f^*}{f_W-f^*}$ [28], where f^* denotes the true objective function value, $\hat{f}$ denotes the function value found by our proposed method and f_W denotes the worst (largest) function value. All tests are done on an Intel(R) Core(TM) i3 CPU @2.4 GHz machine under 64bit Windows7 with 4 GB RAM.

Table 1. Test Problems from MIPLIB.

Problem	n	m	**Time(sec) with branch-and-cut in CPLEX**	**Time(sec) with branch-and-bound in MATLAB**	**Time(sec) with our method in MATLAB**	**Feasibility measure**	**Optimality measure (%)**
enigma	21	100	0.23	4.02	0.03	18	0
air01	771	23	0.28	2.86	0.22	13	2.55%
air03	124	10,757	1.05	17.64	34.00	138	-11.68%
air04	8,904	823	34.35	too large to run	3231.5	811	1.43%
air05	426	7,195	26.66	too large to run	698.6	322	-0.55%
nw04	87,482	36	9.83	too large to run	37.9	19	1.36%

In the numerical tests, we experimented with different values for parameters Q_j, R_j and F on the small problems $enigma$ and $air01$. The diagonal elements of Q_j were set to 0, 1 and 10, and we found that smaller values were better, so we report results with $Q_j = 0$ in Table 1. We also tested values for parameter R_j set to 1, 10, 100 and 1000, and there was not much difference in performance, so we set $R_j = 10$. As for parameter F, we found that bigger values were better, so we set the diagonal elements of F to $100,000$. The parameters Q_j penalize the intermediate error values whereas the parameter F penalizes the terminal error at n. Since the terminal error better reflects the original BIP optimality and infeasibility measures, intuitively, it makes sense to set $Q_j = 0$ and F large.

Values for the weighting factor ω ranged between 0.5 to 0.9 in our exploratory tests, and the best results were typically for ω between 0.5 and 0.6.

CPLEX ran very quickly and always found an optimal solution; branch-and-bound in MATLAB was slower and only found a feasible solution for $enigma$, $air01$ and $air03$; our method in MATLAB ran slower than CPLEX, but generally faster than branch-and-bound in MATLAB. Even though our numerical results are "worse" than CPLEX, our methodology has a potential for extension with polynomial computational complexity.

4. Summary and Conclusion

The meta-control algorithm for approximately solving large-scale BIPs shows much promise because the computational complexity is linear in n (the number of variables) and polynomial in m (the number of constraints), specifically on the order of $O(nm^3)$. An LQT approach is suggested by the result in Theorem 1, which proves the existence of an optimal binary solution to the LQT problem. We provide numerical results with experimentally chosen parameter values that demonstrate the effectiveness of our approach.

In our future research, we will develop an auxiliary iterative method that can provide an explicit algorithm for detecting valid parameter values automatically and investigate other ways to integrate the quantization into the meta-control algorithm to improve the performance of this algorithm. We will also develop a stochastic decomposition method to reduce the computation time.

Acknowledgements

This research is sponsored, in part, by the National Science Foundation through Grant CMMI-0908317.

Conflicts of Interest

The authors declare no conflict of interest.

Appendix

We solve for $\hat{u}_j$ in Problem 4 using a dynamic programming approach. We write the cost-to-go equation as:

$$V\left(e_j, j\right) = \min_{u_j} \left\{ \ \tfrac{1}{2} e_j^T Q_j e_j + \tfrac{1}{2} u_j (u_j - 1) R_j + V(e_{j+1}, j+1) \ \right\} \tag{25}$$

with $V(e_n, n) = \frac{1}{2} e_n^T F e_n$, and equate it to the Riccati form

$$V\left(e_j, j\right) = \frac{1}{2} e_j^T \Sigma_j e_j + e_j^T \Psi_j + \Omega_j \tag{26}$$

where Σ_j represents a symmetric positive-definite $(m+1) \times (m+1)$ matrix, Ψ_j is a positive $(m+1) \times 1$ vector, and Ω_j is a positive scalar.

Combining the Equations (25), (26) and the dynamics in Equation (22), we have

$$\begin{aligned} V\left(e_j, j\right) = \min_{u_j} \Bigg\{ & \frac{1}{2} e_j^T Q_j e_j + \frac{1}{2} u_j (u_j - 1) R_j + \frac{1}{2} \Big(e_j + a_j u_j\Big)^T \Sigma_{j+1} \Big(e_j + a_j u_j\Big) \\ & + \Big(e_j + a_j u_j\Big)^T \Psi_{j+1} + \Omega_{j+1} \Bigg\} \end{aligned} \tag{27}$$

In order to minimize this expression we isolate the terms with u_j in them

$$\frac{1}{2} u_j (u_j - 1) R_j + \frac{1}{2} u_j^2 a_j^T \Sigma_{j+1} a_j + u_j a_j^T \Sigma_{j+1} e_j + u_j a_j^T \Psi_{j+1}$$

and take the derivative with respect to u_j and set the value to 0,

$$(u_j - \frac{1}{2})R_j + a_j^T \Sigma_{j+1} a_j u_j + a_j^T \Sigma_{j+1} e_j + a_j^T \Psi_{j+1} = 0$$

This yields the solution u_j for the optimal control

$$\hat{u}_j = \frac{\frac{1}{2}R_j - a_j^T \Sigma_{j+1} e_j - a_j^T \Psi_{j+1}}{R_j + a_j^T \Sigma_{j+1} a_j} \tag{28}$$

In order to simplify notation, we let

$$S_j = \frac{-a_j^T \Sigma_{j+1}}{R_j + a_j^T \Sigma_{j+1} a_j} \tag{29}$$

$$\delta_j = \frac{\frac{1}{2}R_j - a_j^T \Psi_{j+1}}{R_j + a_j^T \Sigma_{j+1} a_j} \tag{30}$$

and we can now write

$$\hat{u}_j = S_j e_j + \delta_j \tag{31}$$

We equate the Riccati form Equation (26) with the value function in Equation (27) evaluated at $\hat{u}_j$ from Equation (31), yielding

$$\begin{aligned}\frac{1}{2}e_j^T \Sigma_j e_j + e_j^T \Psi_j + \Omega_j = {} & \frac{1}{2}e_j^T Q_j e_j + \frac{1}{2}(S_j e_j + \delta_j)(S_j e_j + \delta_j - 1)R_j \\ & + \frac{1}{2}\Big(e_j + a_j(S_j e_j + \delta_j)\Big)^T \Sigma_{j+1} \Big(e_j + a_j(S_j e_j + \delta_j)\Big) \\ & + \Big(e_j + a_j(S_j e_j + \delta_j)\Big)^T \Psi_{j+1} + \Omega_{j+1}\end{aligned}$$

We now solve for Σ_j and Ψ_j by separating the quadratic terms from the linear terms in e_j. Isolating the quadratic terms in e_j, we have

$$\frac{1}{2}e_j^T \Sigma_j e_j = \frac{1}{2}e_j^T Q_j e_j + \frac{1}{2}e_j^T S_j^T R_j S_j e_j + \frac{1}{2}e_j^T \big(I + a_j S_j\big)^T \Sigma_{j+1} \big(I + a_j S_j\big) e_j$$

which yields the Riccati equation corresponding to Σ_j

$$\Sigma_j = Q_j + S_j^T R_j S_j + \big(I + a_j S_j\big)^T \Sigma_{j+1} \big(I + a_j S_j\big) \tag{32}$$

Isolating the linear terms in e_j, we have

$$e_j^T \Psi_j = e_j^T S_j^T (\delta_j - \frac{1}{2})R_j + e_j^T \big(I + a_j S_j\big)^T \Sigma_{j+1} a_j \delta_{j+1} + e_j^T \big(I + a_j S_j\big)^T \Psi_{j+1}$$

and factoring out e_j^T, the tracking equation for Ψ_j is

$$\Psi_j = S_j^T (\delta_j - \frac{1}{2})R_j + \big(I + a_j S_j\big)^T \Sigma_{j+1} a_j \delta_j + \big(I + a_j S_j\big)^T \Psi_{j+1} \tag{33}$$

Therefore, Σ_j and Ψ_j can be found backwards in time by Equations (32) and (33) from initial conditions $\Sigma_n = F, \Psi_n = 0$.

Given Σ_j and Ψ_j, we can calculate $\hat{u}_j$ from Equations (28), (22) and (23). To calculate $\bar{u}_j$ for our implementation with quantization, we use the same Σ_j and Ψ_j, but introduce rounding to the nearest integer in Equations (28), (22) and (23) to obtain:

$$\bar{u}_j = \text{int}\left[\frac{\frac{1}{2}R_j - a_j^T\Sigma_{j+1}\hat{e}_j - a_j^T\Psi_{j+1}}{R_j + a_j^T\Sigma_{j+1}a_j}\right] \tag{34}$$

and

$$\bar{e}_{j+1} = \text{int}[\bar{e}_j + a_j\bar{u}_j] \tag{35}$$

with $\bar{e}_0 = -\text{int}[b]$.

References

1. Wolsey, L.A. *Integer Programming*; Wiley: New York, NY, USA, 1998.
2. Danna, E.; Fenelon, M.; Gu, Z.; Wunderling, R. Generating Multiple Solutions for Mixed Integer Programming Problems. In *Integer Programming and Combinatorial Optimization*; Fischetti, M., Williamson, D.P., Eds.; Springer: Berlin, Germany, 2007; pp. 280–294.
3. Jarre, F. Relating Max-Cut Problems and Binary Linear Feasibility Problems. Available online: http://www.optimization-online.org (accessed on 15 June 2013).
4. Bertsimas, D.; Tsitsiklis, J.N. *Introduction to Linear Optimization*; Athena Scientific: Nashua, NH, USA, 1997.
5. Mitten, L.G. Branch-and-bound methods: General formulation and properties. *Oper. Res.* **1970**, *18*, 24–34.
6. Caprara, A.; Fischetti, M. Branch-and-Cut Algorithms. In *Annotated Bibliographies in Combinatorial Optimization*; Wiley: Chichester, UK, 1997; pp. 45–64.
7. Barnhart, C.; Johnson, E.L.; Nemhauser, G.L.; Savelsbergh, M.W.P.; Vance, P.H. Branch-and-price: Column generation for solving huge integer programs. *Oper. Res.* **1998**, *46*, 316–329.
8. Lew, A.; Holger, M. *Dynamic Programming: A Computational Tool*; Springer: New York, NY, USA, 2007; Volume 38.
9. Jünger, M.; Liebling, T.; Naddef, D.; Nemhauser, G.; Pulleyblank, W.; Reinelt, G.; Rinaldi, G.; Wolsey, L. *50 Years of Integer Programming 1958–2008: From the Early Years to the State-of-the-Art*; Springer: Berlin, Germany, 2009.
10. Kirkpatrick, S.; Gelatt, C.D., Jr.; Vecchi, M.P. Optimization by simulated annealing. *Science* **1983**, *220*, 671–680.
11. Zabinsky, Z.B. *Stochastic Adaptive Search for Global Optimization*; Kluwer Academic Publishers: Boston, MA, USA, 2003.
12. Rubinstein, R.Y.; Kroese, D.P. *The Cross Entropy Method: A Unified Combinatorial Approach to Combinatorial Optimization, Monte-Carlo Simulation and Machine Learning*; Springer: Berlin, Germany, 2004.
13. Haupt, R.L.; Sue, E.H. *Practical Genetic Algorithms*; Wiley: New York, NY, USA, 2004.
14. Shi, L.; Ólafsson, S. Nested partitions method for global optimization. *Oper. Res.* **2000**, *48*, 390–407.

15. Hoffman, K.L. Combinatorial optimization: Current successes and directions for the future. *J. Comput. Appl. Math.* **2000**, *124*, 341–360.
16. Grötschel, M.; Krumke, S.O.; Rambau, J. *Online Optimization of Large Scale Systems: State of the Art*; Springer: Berlin, Germany, 2001.
17. Martin, R.K. *Large Scale Linear and Integer Optimization*; Kluwer: Hingham, MA, USA, 1998.
18. Schrijver, A. *Combinatorial Optimization: Polyhedra and Efficiency*; Springer: Berlin, Germany, 2003.
19. Callegari, S.; Bizzarri, F.; Rovatti, R.; Setti, G. On the Approximate solution of a class of large discrete quadratic programming problems by $\Delta\Sigma$ modulation: The case of circulant quadratic forms. *IEEE Trans. Signal Process.* **2010**, *58*, 6126–6139.
20. Callegari, S.; Bizzarri, F. A Heuristic Solution to the Optimisation of Flutter Control in Compression Systems (and to Some More Binary Quadratic Programming Problems) via $\Delta\Sigma$ Modulation Circuits. In Proceedings of the 2010 IEEE International Symposium Circuits and Systems (ISCAS), Paris, France, 30 May–2 June 2010; pp. 1815–1818.
21. Von Haartman, K.; Kohn, W.; Zabinsky, Z.B. A meta-control algorithm for generating approximate solutions to binary programming problems. *Nonlinear Anal. Hybrid Syst* **2008**, *2*, 1232–1244.
22. Frieden, B.R. *Science from Fisher Information: A Unification*; Cambridge University Press: Cambridge, UK, 2004.
23. Zhen, S.; Chen, Y.; Sastry, C.; Tas, N.C. *Optimal Observation for Cyber-Physical Systems: A Fisher-Information-Matrix-Based Approach*; Springer: Berlin, Germany, 2009.
24. Lewis, F.L.; Syrmos, V.L. *Optimal Control*; Wiley: New York, NY, USA, 1995.
25. Bertsekas, D.P. *Dynamic Programming and Optimal Control*, 3rd ed.; Athena Scientific: Nashua, NH, USA, 2005; Volume I.
26. Varga, R.S. *Matrix Iterative Analysis*; Springer: Berlin, Germany, 2000.
27. MIPLIB—Mixed Integer Problem Library. Available online: http://miplib.zib.de/ (accessed on 15 June 2013).
28. Ali, M.M.; Khompatraporn, C.; Zabinsky, Z.B. A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems. *J. Glob. Optim.* **2005**, *31*, 635–672.