Decomposition Based Hybrid Metaheuristics

Günther R. Raidl[1]

Institute of Computer Graphics and Algorithms, Vienna Unviersity of Technology, Favoritenstr. 9/1861, 1040 Vienna, Austria

Abstract

Difficult combinatorial optimization problems coming from practice are nowadays often approached by hybrid metaheuristics that combine principles of classical metaheuristic techniques with advanced methods from fields like mathematical programming, dynamic programming, and constraint programming. If designed appropriately, such hybrids frequently outperform simpler "pure" approaches as they are able to exploit the underlying methods' individual advantages and benefit from synergy. This article starts with a general review of design patterns for hybrid approaches that have been successful on many occasions. More complex practical problems frequently have some special structure that might be exploited. In the field of mixed integer linear programming, three decomposition techniques are particularly well known for taking advantage of special structures: Lagrangian decomposition, Dantzig-Wolfe decomposition (column generation), and Benders' decomposition. It has been recognized that these concepts may also provide a very fruitful basis for effective hybrid metaheuristics. We review the basic principles of these decomposition techniques and discuss for each promising possibilities for combinations with metaheuristics. The approaches are illustrated with successful examples from literature.

Keywords: Combinatorial optimization, Metaheuristics, Mixed integer programming, Hybrid optimization approaches, Decomposition techniques

Email address: raidl@ads.tuwien.ac.at (Günther R. Raidl)

[1]The author's work is partly funded by the Vienna Science and Technology Fund (WWTF) under grant ICT10-027 and the Austrian Science Fund (FWF) under grant P24660.

1. Introduction

Difficult combinatorial optimization problems are in practice frequently approached by means of *metaheuristics*. This term has originally been introduced by Glover (1977) and essentially refers to a broad class of problem-independent strategies for approximate optimization and problem solving. In fact, the boundaries of this class are today somewhat fuzzy, but typically cited representatives include *neighborhood-search-based strategies* like simulated annealing, tabu search, and variable neighborhood search, *population-based methods* like evolutionary/genetic algorithms and scatter search, and *construction-oriented techniques* like the greedy randomized adaptive search procedure and ant colony optimization.

When one is confronted with a non-trivial combinatorial optimization problem from practice that should be solved "as well as possible", a typical approach is to start with a relatively simple constructive heuristic and try to improve obtained solutions by means of local search. To overcome the trap of local optimality, which inherently appears in almost all non-trivial problems, metaheuristics are frequently the next step. Assuming one is still not satisfied with the obtained solutions, one will then typically try to refine the so far developed approach by tuning parameters and certain design decisions, but from a conceptual point-of-view even more important by identifying special characteristics of the problem at hand and finding effective possibilities to exploit them. In this way, one frequently ends up with a more complex solver that is not a straight-forward, "pure" instantiation of a classical metaheuristic anymore, but rather a combination of cleverly chosen techniques that might even stem from diverse algorithmic research fields. This is what we refer to as *hybrid metaheuristic* (HM).

The obvious motivation behind such hybridizations is to obtain better performing systems that exploit and unite advantages of the individual components, i.e., to benefit from synergy. The large number of publications on HMs and dedicated scientific events such as the ongoing series of *Workshops on Hybrid Metaheuristics* (Blum et al., 2004) and *Workshops on Matheuristics* (Maniezzo

et al., 2006) document the popularity, success, and importance of this specific line of research. In fact, today it seems that choosing an adequate hybrid approach is determinant for achieving top performance in solving many real-world problems.

This does, of course, not imply that more complex approaches are always the better choice. Increased complexity also comes with disadvantages: The software is more difficult to maintain and tune and adaptions in problem specifications are frequently harder to adhere. Thus, a still very valid design goal also is to keep an optimization approach as simple as possible, and include extensions only if they indeed provide significant benefits. Such decisions are often difficult, and it shall be explicitly remarked that sometimes, especially when development time is limited or a large versatility is required due to expected changes in problem specifications, a simpler, pure approach might be the wiser choice.

The idea of hybridizing different optimization approaches is not new but dates back to the origins of metaheuristics themselves. For a long time, however, such hybrids were not so popular since several separated and even competing communities of researchers existed who considered "their" favorite class of optimizers "generally best" and followed specific philosophies too dogmatically. It is mostly due to the *no free lunch* theorems (Wolpert & Macready, 1997) that this situation fortunately changed and people recognized that there cannot exist a general optimization strategy that is globally best. To solve a problem at hand most effectively, it almost always requires a specialized algorithm that needs to be compiled of adequate parts.

Several publications exist that try to classify HMs or provide guidelines for their design. Talbi (2002) gives a general taxonomy, Cotta et al. (2005) concentrate on parallel hybrids, and El-Abd & Kamel (2005) consider cooperative search strategies. Combinations of metaheuristics with exact optimization techniques are particularly addressed by Puchinger & Raidl (2005). Talbi (2013b) further describes a unified taxonomy of HMs with mathematical programming, constraint programming, and machine learning. More recent general surveys on HMs pointing out prominent design principles are provided by Raidl et al.

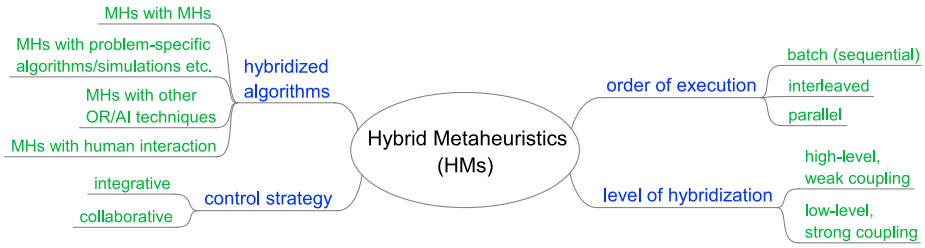

Figure 1: A basic taxonomy for hybrid metaheuristics.

(2010) and Blum et al. (2011).

Figure 1 illustrates the top levels of a basic taxonomy as suggested by Raidl (2006). First, we might distinguish *what* is hybridized: concepts of different metaheuristics; metaheuristics with problem-specific algorithms like simulations; metaheuristics with other techniques e.g. from operations research or artificial intelligence; or metaheuristics with human interaction. A second criterion for distinguishing HMs is the *order of execution* of the individual parts, which can be strictly sequential (i.e., a batch approach), interleaved, or parallel. A third criterion is what we call *control strategy*: In an integrative approach, we have some kind of master algorithm in which one or more subordinate components are embedded or called. In contrast, in collaborative hybrids individual components may run more independently and exchange information in some way. Last but not least a fourth criterion for characterizing HMs is the *level of hybridization*: In a high-level hybrid, individual components are coupled rather weakly and typically not so much communication takes place between them, while a low-level hybrid has strongly interwoven parts. For more details on such taxonomies we refer to the above publications; here we just introduced the nomenclature we will use later.

When considering combinations of metaheuristics with other optimization techniques, hybrids with mathematical programming approaches became particularly popular over the last decade. The reason behind this is that especially classical *mixed integer linear programming* (MIP) techniques have pros

and cons that are complementary to those of metaheuristics to a large degree. When a problem can be expressed by means of a linear objective function and linear constraints, chances are good that at least small instances can be solved to proven optimality by a state-of-the-art general purpose MIP solver like CPLEX[2], GUROBI[3], or the freely available SCIP[4]. While these modern solvers are sometimes astonishingly effective in solving smaller problems or problems with certain structures, they are also known to typically scale rather poorly. When it comes to large instances of hard problems running times or memory requirements often become intractable. Hybrid metaheuristics might exploit the advantages of MIP solvers, e.g., by applying them to smaller subproblems while relying on the power of metaheuristic search as outer framework. Combinations of metaheuristics and mathematical programming techniques are frequently also termed *matheuristics*; see Raidl & Puchinger (2008) for a survey and Maniezzo et al. (2009) for a comprehensive book on this topic. Further recommended books covering many more general aspects and applications of HMs are by Blum et al. (2008) and Talbi (2013a).

The following section reviews major classical *design patterns* of HMs that have proven to be successful already on many occasions. The main part of this article will concentrate on a class of hybrids which is not that commonly found but we nevertheless consider highly promising for many large real-world problems: Approaches that are based on *decomposition techniques* having the origins in solving large (mixed integer) linear programs: Lagrangian decomposition, Dantzig-Wolfe decomposition utilizing column generation, and Benders' decomposition. Sections 3 to 6 overview the principles of these techniques and discuss possibilities of exploiting them in a hybrid metaheuristic context. Section 7 concludes this article and points out promising future research directions.

[2] `http://www-01.ibm.com/software/integration/optimization/cplex-optimizer`
[3] `http://www.gurobi.com`
[4] `http://scip.zib.de`

2. Design Patterns for Hybrid Metaheuristics

The following design patterns describe typical strategies for obtaining hybrids by combining different simpler or more classical algorithms or concepts of such. This section summarizes the presentation of Raidl et al. (2010) and extends it with recent developments. Note that frequently, these design patterns also appear in combination, and sometimes there also is no clear borderline of what one calls a "real" hybrid and where a described pattern is actually part of some classical optimization approach.

2.1. Intelligent Initialization

The probably most natural way of boosting many optimization algorithms is by supplying an already good starting solution. Such initial solutions might be obtained from problem-specific constructive heuristics or essentially any other optimization approach that is considered fast enough. Note that also classical exact optimization methods like branch-and-bound typically depend heavily on a good initialization in order to be able to prune the search space effectively from the very beginning. For population based metaheuristics like evolutionary algorithms it is crucial to start with a set of initial solutions that is diverse enough in order to avoid premature convergence. Multiple initialization procedures may be utilized, or a deterministic procedure might be randomized in order to obtain different promising starting solutions. Note that the latter approach is also systematically applied in the *Greedy Randomized Adaptive Search Procedure* (GRASP) (Feo & Resende, 1995).

2.2. Embedded Improvement Methods

Another extremely commonly applied pattern is to embed some improvement method, e.g., a classical local search or more advanced metaheuristic like simulated annealing or tabu search, into a master algorithm. Such an approach thus follows an integrative control strategy. A typical representative that uses this pattern as main distinguishing feature is the *memetic algorithm* (Moscato, 1999; Hart et al., 2005), which essentially is an evolutionary algorithm including some

local improvement technique that is applied to all or part of the newly created solution candidates of each iteration. In this way intensification of the heuristic search is frequently successfully enhanced. As another example, *variable neighborhood search* approaches (Hansen et al., 2001) can be seen to systematically extend this pattern by applying a series of neighborhood searches in a nested way. Last but not least, also classical branch-and-bound often relies on primal local improvement procedures for finding sooner better heuristic solutions and corresponding global bounds in order to prune larger parts of the search space.

2.3. Multi-Stage Approaches

This kind of hybrids solves some larger problem by decomposing the whole optimization into multiple stages that are addressed by individual techniques in a typically sequential manner.

Especially more complex real-world problems frequently involve hierarchical decisions and corresponding sets of variables. A first stage of optimization may then be used to fix higher-level decisions while the remaining lower-level variables are determined in one or more successive phases where the higher-level variables are considered fixed. As an example consider some vehicle routing problem (Fisher & Jaikumar, 1981): The principal decisions which customers are visited by which vehicle might be considered the upper level which is decided first, e.g., according to a clustering. Further details such as the specific vehicle routes and the exact timing of the visits may be optimized in a second stage, where we have the advantage that the problem decouples into independent subproblems for all the vehicles. Obviously the stages in such approaches are in general not independent and suboptimal solutions are therefore usually obtained. Nevertheless, for very large and difficult problems such approaches may be meaningful or even be the only practically reasonable possibility. Note, however, that in the context of vehicle routing problems also the alternative possibility of deciding an order for visiting all customers in a first stage, i.e., building one large route, and splitting it into several tours in the second stage can be effective, especially as the splitting step might be done efficiently via dy-

namic programming; Prins et al. (2014) presents a review on such approaches.

As another example, consider the design of large telecommunication networks, where the decision process will again be hierarchical: Only after an optimization of the general layout and structure, technical details like link capacities and properties of routers will be derived (Martins & Ribeiro, 2006).

Multi-stage approaches may be meaningful even for problems that do not have a natural hierarchy of decision variables. *Multi-level refinement strategies* (Walshaw, 2008) apply a recursive coarsening to create a series of approximations to the original problem. An initial solution is then identified for the coarsest level and iteratively refined at each level – coarsest to finest – until a solution for the original problem is obtained. *Iterated multi-level algorithms* extend this concept by iteratively re-coarsening the problem based on obtained solutions; in this way poor decisions in the coarsening may be redeemed. Multi-level refinement strategies have been successfully applied on problems such as multilevel graph partitioning, graph coloring, and very large vehicle routing problems. In general, they are a promising technique for improving the scalability of some optimization algorithm. As a recent example, Valejo et al. (2014) applies multi-level refinement to partition social network graphs and suggests a coarsening method exploiting the neighborhood similarity.

Another kind of multi-stage hybrids are approaches involving (advanced) preprocessing techniques as well as kernelization methods. Especially in the latter, the original problem is reduced in polynomial time to a so-called problem kernel such that an optimal solution to it can, in polynomial time, be transformed back into an optimal solution to the original problem. Heuristic variants of such methods are variable fixing approaches, where in early stages variables are identified that are very likely to have certain values in good or optimal solutions and these variables are correspondingly fixed for the later stage(s). For examples of kernelization approaches to the to the vertex cover problem see Gilmour & Dras (2006) and to the multi-dimensional knapsack problem see Puchinger et al. (2010); Angelelli et al. (2010).

2.4. Decoder-Based Approaches

Especially in the field of evolutionary algorithms it is popular to represent candidate solutions in some indirect way, in this context often called *genotype*, and use some problem-dependent decoding algorithm to obtain the corresponding "real" solution (*phenotype*). The advantage of such an approach is mainly that the variation operators to construct new candidate solutions can be chosen more or less independently of the particular problem. Problem specificities like special constraints can often be nicely hidden from the optimization algorithm and possibly efficiently be handled by means of the decoder. Sometimes, genotypes represent only *incomplete* solutions, i.e., not all aspects are specified, and an "intelligent" decoding procedure augments the missing values by solving some remaining subproblem. Such approaches have been successfully used especially for (multi-dimensional) cutting and packing as well as scheduling and timetabling problems, where direct representations are often difficult to handle due to the many complex constraints. Indirect representations are in such cases frequently based on permutations, and decoders are derived from construction heuristics considering the objects in the orders specified by the permutations (Kellerer et al., 2004).

As another example consider solving a MIP involving integer variables as well as continuous variables. A metaheuristic might be used to tackle the integer variables only, while optimal values for the continuous variables are derived for each integer-candidate solution efficiently by means of a linear programming solver.

So-called *hyper-heuristics* (Burke et al., 2013) also make heavy use of ideas similar to decoder-based approaches: They operate on a search space of heuristics or heuristic components rather than directly on the space of solutions in order to find/compile a heuristic that best solves the target problem.

2.5. Exploring Large Neighborhoods

Sometimes local search is enhanced by using particularly large neighborhoods that are not investigated by naive enumeration but some more efficient

algorithm. If the neighborhood structure and the corresponding algorithm are chosen appropriately, large portions of the search space might be efficiently covered. Such techniques are generally known as *large neighborhood search* (Shaw, 1998) or *very large-scale neighborhood* search (Ahuja et al., 2002).

Many of today's combinations of metaheuristics with MIP approaches follow this scheme as it is relatively straight-forward to apply once a compact MIP model is available for the problem at hand: Part of the variables are fixed to the incumbent solution's values and the others are kept open and optimized via a MIP solver. Of course the selection of the variables to be kept open and optimized might be crucial. Typically, they are selected either randomly or according to some greedy heuristic trying to identify weak parts of the solution. Often it is also meaningful to select variables that are strongly related together. For examples see Büdenbender et al. (2000); Prandtstetter & Raidl (2008); Lopes et al. (2014).

Besides MIP solvers, also constraint programming is sometimes applied as, e.g., described by Shaw (1998) for vehicle routing problems and more recently by Di Gaspero & Urli (2014) for homecare scheduling problems. Furthermore, dynamic programming frequently is a promising candidate for identifying best solutions in large neighborhoods, providing the neighborhood structure is suitably chosen.

In the context of problems where objects need to be partitioned into disjoint sets, such as vehicle routing problems, machine scheduling problems, and other assignment problems, *cyclic and path exchange neighborhoods* can be highly effective (Ahuja et al., 2002). In these neighborhoods a series of objects is exchanged among an arbitrary number of partitions in a cyclic or path-like fashion; the best move is determined by constructing an improvement graph where an arc corresponds to the movement of a single object and applying a shortest path-like algorithm.

Obviously, it is not necessary to always find a best solution within a large neighborhood as it is the case in the above examples (except when prematurely terminating the MIP solver or dynamic programming). Sometimes also

simpler but faster heuristics such as greedy constructive methods are used to assign promising values to the open variables. Such approaches are frequently called *destroy-and-recreate* or *removal-and-insertion* techniques, as a solution is partially dissolved and the missing parts are redetermined. A good example is the *adaptive large neighborhood search* heuristic for pickup and delivery problems with time windows (Ropke & Pisinger, 2006), which involves several competing types of large neighborhoods and corresponding sub-heuristics whose application is controlled by their historic performance. In fact, this principle of adaptive large neighborhood search turned out to work particularly well on a larger variety of vehicle routing problems and became quite popular over the recent years. For example, Hemmelmayr et al. (2012) describe such an approach for two-echelon vehicle routing problems in city logistics and Azi et al. (2014) for a vehicle routing problem with multiple routes per vehicle.

Last but not least, note that decoder based approaches discussed in Section 2.4 might sometimes also be interpreted as large neighborhood search techniques, especially when incomplete solution representations in conjunction with more complex decoders are used.

2.6. Solution Merging

Here, the idea is to derive a new, possibly better solution from the properties (i.e., variable values) contained in two or more input solutions. The observation that high-quality solutions usually have many properties in common is exploited. In the simplest form this principle is applied in the classical *recombination operator* of evolutionary algorithms, where the attributes to be inherited are typically chosen in a computationally cheap, random fashion. A more advanced, but also computationally more expensive approach is *path relinking* (Glover et al., 2000), where a starting solution is more systematically transformed into a guiding solution by iteratively performing atomic changes. Thus, a path between the two solutions is traced in the search space, and a best solution on it is returned as result. For a recent example see Pessoa et al. (2013), where path relinking is applied within a GRASP for set k covering.

Even more systematic are *optimal merging* procedures, where the subspace of all solutions that can be constructed out of the properties appearing in a set of given input solutions is considered and a best solution returned. Depending on the underlying problem, identifying such an optimal offspring might be a hard optimization problem on its own, but due to the typically limited number of different properties appearing in the parents, it can often be solved in practical time. Applegate et al. (1998) were one of the first describing such an optimal merging: For the traveling salesman problem, they derive a set of different high-quality tours by means of the chained Lin-Kernighan iterated local search algorithm. The sets of edges of all these solutions are merged and the problem is finally solved to optimality by means of a MIP approach on this strongly restricted graph. While merging appears in this example as second stage in a sequential two-stage approach and is only performed once, there are also cases of intertwined hybrids where optimal merging is used as a more systematic variation operator replacing classical recombination. Such operators are then also called *optimal recombinations*.For example, Blum & Blesa (2008) apply solution merging within a large neighborhood search for the k-cardinality tree problem, and Eremeev (2008) studies optimal recombination operators for the travelling salesman problem. Also the commercial MIP solver CPLEX contains optimal merging as heuristic procedure for obtaining improved incumbent solutions (Lodi, 2013).

2.7. Strategic Guidance of Metaheuristics by Other Approaches

Many successful HMs exploit information on promising areas of the search space obtained by other techniques by intensifying or restricting the heuristic search to these regions.

Problem relaxations are most frequently used for such purposes: Some of the problem's constraints are dropped in order to obtain a relaxation that can be solved efficiently. The obtained solution may then be a good guiding point. If a (compact) MIP formulation exists for the problem at hand, its linear programming (LP) relaxation often is an obvious choice. An obtained fractional

solution can frequently be rounded or in some other way repaired to obtain a feasible integral solution in its proximity, variables that have already integral values in the LP solution might be fixed, or LP values may be used to bias variation operators, e.g., by choosing close variable values with higher probabilities. Occasionally, dual variable information of LP solutions may provide even more helpful guidance. Chu & Beasley (1998), for example, make use of it in a genetic algorithm for the multi-dimensional knapsack problem by calculating so-called *pseudo-utility ratios* for the primal variables and using them in similar ways as described above for the primal LP solution values. Also Puchinger et al. (2010) point out that at least for this problem pseudo-utility ratios are significantly better indicators for the likeliness of the corresponding items to appear in an optimal integer solutions than primal LP variable values.

Apart from the LP relaxation also other relaxations are sometimes exploited in conjunction with metaheuristics. Besides problem-specific approaches, Lagrangian relaxation has been particularly successful (Haouari & Siala, 2006; Jeet & Kutanoglu, 2007; Leitner & Raidl, 2008; Pessoa et al., 2013). In comparison to the LP relaxation, Lagrangian relaxation has the advantage of frequently yielding tighter optimality bounds. This, however, comes at the cost of a usually higher computational effort. We will consider Lagrangian relaxation in more detail in the context of decomposition approaches in Section 4.

Also other information is sometimes exploited for guiding metaheuristics, e.g., when constructing candidate solutions, lower and/or upper bounds determined for partial solutions (Dowsland et al., 2006) or reduced variable domains determined by constraint propagation techniques (Meyer & Ernst, 2004). More generally, in collaborative approaches solution migration can also be considered a technique where one optimization task provides guidance for another.

2.8. Strategic Guidance of Branch-and-Bound by Metaheuristics

Branch-and-bound is a fundamental tree-search approach for solving difficult optimization problems systematically to proven optimality. It relies on the calculation of lower and upper bounds for partial solutions in order to prune the

search tree as far as possible. As already mentioned, good initial solutions and local improvement techniques frequently play an important role to obtain primal bounds. Besides these aspects, principles of local search based metaheuristics are sometimes mimicked by special control strategies for selecting the tree nodes (i.e., open subproblems) to be processed next or special branching strategies focusing the tree search to the neighborhoods of promising solutions.

Danna et al. (2005) proposed *guided dives*, where the tree search temporally switches to a depth-first strategy and always considers next a subproblem where the branching variable has the value of the incumbent solution. Guided dives are repeatedly applied in regular intervals, results indicate a strongly improved heuristic performance.

Fischetti & Lodi (2003), on the contrary, suggested *local branching*. Given an incumbent solution again, branching is performed by adding on the one hand a so-called local branching constraint that restricts the search space to the incumbent's k-OPT neighborhood and on the other hand its inverse representing the remaining search space. The MIP solver is then forced to completely solve the k-OPT neighborhood before considering the remaining open nodes of the branch-and-bound tree. If an improved solution has been found, a new subproblem corresponding to the k-OPT neighborhood of the new incumbent is split off, otherwise a larger k may be tried. When no further improvements are achieved, the remaining problem is processed in a standard way. While local branching is often beneficial, it was also shown that the addition of the inverse local branching constraints frequently is counterproductive as many of these dense constraints degrade performance.

Danna et al. (2005) further proposed *relaxation induced neighborhood search* (RINS), where occasionally a sub-MIP is spawned from a search-tree node corresponding to another special neighborhood of an incumbent solution: Variables having the same values in the incumbent and the current solution to the LP relaxation are fixed and an objective value cutoff corresponding to the current LP value is set. The subproblem on the remaining variables is then solved with limited time. In the authors' experimental comparison of guided dives, local

branching, and RINS on a collection of MIP models originating from diverse sources including job-show scheduling, network design, crew scheduling, lot-sizing, and railway line planning problems and MIPLIB-3.0[5], RINS performed best; it has been included in CPLEX as a standard strategy. More recently, Gomes et al. (2013) suggested an extension of RINS that explicitly explores pre-processing techniques. This method systematically searches for a suitable number of fixations to produce subproblems of controlled size, which are explored in a variable neighborhood descent fashion.

Also somehow related are metaheuristics that utilize a *complete solution archive* to avoid reconsiderations of candidate solutions. Raidl & Hu (2010) proposed such an extension for a genetic algorithm based on a trie data structure, which actually closely resembles an explicitly stored branch-and-bound tree. When an already evaluated solution would be reconsidered, it is efficiently transformed into a usually similar but guaranteed new solution by appropriately traversing the trie. In this way, the metaheuristic is in principle turned into an exact tree-search based approach. Such a solution archive may be particularly effective in cases where the solution evaluation is expensive or decoder-based approaches with incomplete representations are used.

3. Decomposition Techniques

Problem decomposition techniques are a class of approaches particularly aimed at solving very large or complex problems, frequently also involving different types of variables. The basic idea is to solve such a large problem by solving a series of smaller problems and appropriately combining the results.

In fact, the previous sections already introduced several such techniques: Sequential multi-stage methods described in Section 2.3 follow this principle in a straight-forward way. Large-neighborhood search methods, cf. Section 2.5, obviously also fall into this category and in general have the advantage to avoid the

[5] `http://www.or.deis.unibo.it/research pages/ORinstances/MIPs.html`

problem of fixing possibly suboptimal decisions too early. The same holds for advanced solution merging strategies, cf. Section 2.6, which in fact might also be considered special large neighborhood search approaches where the neighborhood is induced by more than one parent solutions.

Certain metaheuristics have been suggested that explicitly make decomposition to their primary principle: *Variable neighborhood decomposition search* (Hansen et al., 2001) extends classical variable neighborhood search by fixing parts of incumbent solutions and applying some improvement method to the corresponding subproblems. *Partial Optimization Metaheuristic Under Special Intensification Conditions* (POPMUSIC) from Taillard & Voß (2001) is another general approach to solve very large problems by iteratively solving smaller parts with an effective problem-specific method. A more specific, recent example where such an approach has been applied to efficiently compute Steiner trees on large graphs is given by Leitner et al. (2014). *Decomposition guided variable neighborhood search* (Fontaine et al., 2011) utilizes the graph of clusters provided by a tree decomposition of the constraints graph to guide the exploration of large neighborhoods within a variable neighborhood search. More recently, S. Loudni (2013) improved this method by new strategies for better controlling intensification and diversification, and Ouali et al. (2014) applied this method in a cooperative parallel way for solving weighted constraint satisfaction problems. Last but not least, the already mentioned *multi-level refinement strategies* (Walshaw, 2008), cf. Section 2.3, also fall into this category, although they follow a different bottom-up approach.

All these approaches explicitly depend on the embedding of other, effective optimization procedures and can thus be said to be hybrids already by definition.

In the following sections we will consider more specific decomposition approaches that build upon classical techniques coming from (mixed integer) linear programming. *Lagrangian decomposition*, *Dantzig Wolfe decomposition* with its related *column generation*, and *Benders' decomposition* are prominent approaches for solving large LPs or MIPs having certain structures. We will review their basic principles and discuss successful and promising combinations

with metaheuristic approaches. For an overview that also discusses these three MIP decomposition approaches and tries to reinterpret them as more general metaheuristic frameworks, see Boschetti et al. (2009). For general in-depth introductions to MIP techniques and related topics the text books by Nemhauser & Wolsey (1988); Wolsey (1998) are recommended.

4. Lagrangian Decomposition and Metaheuristics

Consider we are given a problem that can be modeled in the form

$$z_{\text{MIP}} = \min\{c^T x \mid Ax \geq b,\ Dx \geq d,\ x \in \mathbb{D}^n\}, \tag{1}$$

where x is a vector of n non-negative decision variables of domain $\mathbb{D}^n$, $c^T x$ is the linear function to be minimized, and there are two sets of constraints $Ax \geq b$ and $Dx \geq d$. Assume that the constraints $Ax \geq b$ are easy in the sense that we could solve the problem efficiently when dropping the "complicating" constraints $Dx \geq d$. Such an approach would be a feasible relaxation but typically yield only a weak lower bound. *Lagrangian relaxation* (Fisher, 1981) replaces these constraints by corresponding penalty terms in the objective function:

$$z_{\text{LR}}(\lambda) = \min\{c^T x + \lambda^T (d - Dx) \mid Ax \geq b,\ x \in \mathbb{D}^n\}. \tag{2}$$

Vector λ is the vector of real-valued Lagrangian multipliers, and for any $\lambda \geq 0$, $z_{\text{LR}}(\lambda) \leq z_{\text{MIP}}$ holds; i.e., we have a valid relaxation of the original MIP (1). We are now interested in finding a specific instantiation of λ yielding the best—i.e., largest—possible lower bound, which leads to the *Lagrangian dual problem*

$$z^*_{\text{LR}} = \max_{\lambda \geq 0}\{z_{\text{LR}}(\lambda)\}. \tag{3}$$

This Lagrangian dual is a piecewise linear, convex function which can usually be well solved by iterative procedures like subgradient methods; see, e.g., Barahona & Anbil (2000) for an advanced approach called *volume algorithm*.

Given a solution λ to the Lagrangian dual problem (3) and a corresponding optimal solution x^* to the Lagrangian relaxation (2) that is also feasible for

the original problem (1), i.e., $Dx^* \geq d$, the following complementary slackness condition holds: x^* is an optimal solution to the original problem (1) iff

$$\lambda^T(d - Dx^*) = 0. \tag{4}$$

Provided the Lagrangian dual problem is solved to optimality, it can be shown that the Lagrangian relaxation always yields a bound that is at least as good as the one of the corresponding LP relaxation; in practice it often is far better.

In case of *Lagrangian decomposition* (LD), the Lagrangian relaxation (2) decouples into a series of k subproblems that can be independently solved:

$$z_{\mathrm{LD}}(\lambda) = \sum_{i=1,\ldots,k} \min\{c^{iT}x^i + \lambda^T(d^i - D^i x^i) \mid A^i x^i \geq b^i,\ x^i \in \mathbb{D}^{n^i}\}. \tag{5}$$

These subproblems can be of the same type or different.

While Lagrangian relaxation in principle only yields a lower bound to the original minimization problem, it is usually not hard to extend the approach to a *Lagrangian heuristic* also yielding a feasible approximate solution and corresponding upper bound, e.g., by problem-specific repairing.

We illustrate LD on the *knapsack-constrained maximum spanning tree problem* (KCMST) following Pirkwieser et al. (2007). Consider a graph $G = (V, E)$ with node set V and edge set E; each edge has associated a weight $w_e \geq 0$ and a price p_e. The aim is to find a subgraph $T = (V, E'),\ E' \subseteq E$ corresponding to a spanning tree whose total weight does not exceed a limit $W \geq 0$ and whose total price is a maximum. Obviously, this problem is a combination of the well known *minimum cost spanning tree problem* (MST) when taking negative prices as costs and the *0–1 knapsack problem* (KP).

We express KCMST in the following way in order to apply LD:

$$\max \quad \sum_{e \in E} p_e x_e \tag{6}$$

$$\text{s.t.} \quad x \mathrel{\hat{=}} \text{a spanning tree on } G \tag{7}$$

$$\sum_{e \in E} w_e y_e \leq W \tag{8}$$

$$x_e = y_e, \qquad \forall e \in E \tag{9}$$

$$x_e,\, y_e \in \{0,1\} \qquad \forall e \in E \tag{10}$$

Binary variables $x_e = y_e$ indicate whether or not the corresponding edges e belong to the solution, i.e., $e \in E'$. In this model, duplicating x with y obviously is redundant, but it facilitates LD as the spanning tree condition (7) is expressed only on variables x and the knapsack constraint (8) only on variables y. Equalities (9) provide the linkage.

By relaxing now these linking constraints (9) in a Lagrangian manner, we obtain the decomposition into the classical MST and KP:

$$\begin{aligned} z_{\text{LD}}(\lambda) = &\max\{(p-\lambda)^T x \mid x \mathrel{\hat{=}} \text{a spanning tree on } G,\, x \in \{0,1\}^E\} + \\ &\max\{\lambda^T y \mid w^T y \leq W,\; y \in \{0,1\}^E\}. \end{aligned} \tag{11}$$

The MST can be efficiently solved by well known algorithms like Prim's MST algorithm, and the KP, which is only weakly NP-hard, by dynamic programming approaches like the COMBO algorithm (Martello et al., 1999). These subproblems are iteratively solved within the subgradient procedure in order to find best possible Lagrangian multipliers λ and a corresponding upper bound for the optimal KCMST solution value.

Feasible heuristic solutions are obtained as follows:

- Occasionally, the intermediate spanning trees obtained by solving the MST subproblems may comply with the knapsack constraint (8), and thus we may directly obtain approximate KCMST solutions. This is not guaranteed, however.

- Infeasible intermediate spanning trees can be repaired, e.g. by a greedy approach that iteratively removes an edge with highest weight and reconnects the separated components by an edge with lower weight.

- As these possibly repaired intermediate solutions are typically only of moderate quality, it becomes natural to additionally apply some improvement heuristic. A straight-forward extension is a local search utilizing an edge-exchange neighborhood, which considers all feasible single-edge replacements. Obviously, also more advanced metaheuristics might be applied here to obtain even better solutions.

Besides this classical Lagrangian heuristic approach, metaheuristics may further benefit by more rigorously exploiting information obtained by the LD, in particular Lagrangian dual variable values. Haouari & Siala (2006) describe an effective strategy in the context of the prize collecting Steiner tree problem, which Pirkwieser et al. (2007) apply in a similar spirit to our example of the KCMST problem as follows.

Consider a memetic algorithm for the KCMST, which represents candidate spanning trees directly by storing their edges. Initial solutions are random spanning trees, recombination is achieved by applying a random spanning tree algorithm on the merged edge sets of two parent trees, and mutation performs a random edge exchange. Any candidate tree not complying with the knapsack constraint is greedily repaired as described above. Edge-exchange local search is applied to each new offspring solution. This memetic algorithm is improved by making use of the above LD's results as follows:

- A few best solutions directly obtained by the LD are used to seed the initial population.

- The edge set E is reduced to only contain edges that appeared in at least one of the LD's intermediate spanning trees (feasible or not). Thus, the memetic algorithm can also be said to act as solution merging strategy over all intermediate LD solutions, cf. Section 2.6.

- Most importantly, original edge profits p_e are replaced by *reduced profits* $p'_e = p_e - \lambda_e$, and recombination and mutation are biased in their random decisions to include edges with higher reduced profits more likely.

- Should a solution be found whose objective value corresponds to the LD upper bound, the search is terminated as the solution is proven optimal.

Large-scale experiments on graphs with up to 8 000 nodes and over 23 000 edges have shown excellent results: More than 67% of all solutions could be solved to proven optimality, and average remaining relative gaps between the LD's upper bounds and the found solutions' objective values were less than 10^{-6}. The pure memetic algorithm without the guidance by the LD could not compete at all, and the feasible solutions obtained from the pure LD were only moderately good. These surprising results of course also indicate that the KCMST problem, although NP-hard, can be solved relatively well in practice. It turned out that especially the LD's reduced profits are an excellent indicator for how beneficial edges really are. In contrast, original edge prices p_e, edge weights w_e, and even relative prices p_e/w_e may be frequently highly misleading when used for guiding heuristic search. Note that there are also some parallels to the works on the multi-constrained knapsack problem by Chu & Beasley (1998) and Puchinger et al. (2010) who used pseudo-utility ratios determined from dual LP values for guiding heuristic search.

Leitner & Raidl (2008) describe another successful hybrid of LD and a variable neighborhood search in the context of a fiber optic network design problem, and Leitner & Raidl (to appear) utilize large neighborhood search in combination with LD for solving the capacitated connected facility location problem. Boschetti & Maniezzo (2009) illustrate Lagrangian heuristics on the single source capacitated facility location problem and the membership overlay problem arising in the context of P2P networks. Thiruvady et al. (2014) utilize LD in combination with ant colony optimization to solve a resource constrained job scheduling problem on multiple machines. By relaxing the linking constraintes each machine's scheduling problem can be solved independently, and the solu-

tion to the LD is used to effectively guide the ACO.

Last but not least, we remark that (meta-)heuristics may also become attractive for solving harder subproblems in Lagrangian relaxation approaches, although one has to keep in mind that only optimal solutions to the subproblems will guarantee the validity of a bound obtained for an original problem.

5. Column Generation Approaches and Metaheuristics

Dantzig-Wolfe decomposition (Dantzig & Wolfe, 1960) has originally been introduced for solving very large LPs having a special block-diagonal structure and relies on *delayed column generation* (CG). In mathematical programming, CG is a well-known technique to approach in particular MIP models involving an exponential number of variables. Such MIPs frequently arise in set covering or set partitioning formulations for, e.g., vehicle routing, network design, cutting and packing, and scheduling problems. Often such models can be shown to be substantially stronger than some compact formulation w.r.t. the LP relaxation. The challenge, however, is their very large number of variables which usually prohibits a direct application of a MIP solver. CG provides a possible practical solution approach at least for solving the LP relaxation of such MIP models. For in-depth information on CG we refer to Desaulniers et al. (2005) and Lübbecke & Desrosiers (2005).

To illustrate the principles of CG, let us consider an abstract network design problem, in which we are given a graph $G = (V, E)$. The node set V consists of a root node 0 (e.g., a central server), clients $C \subset V$, and possibly further nodes $S = V \setminus \{0\} \setminus C$. The edge set E represents potential links that may be installed for connecting the respective nodes at given costs $c_e > 0$. The objective is to find a minimum cost subgraph $G' = (V', E'),\ V' \subseteq V,\ E' \subseteq E$ corresponding to a network that provides for each customer $c \in C$ a connection to the root 0. In the simplest form this problem corresponds to the classical rooted Steiner tree problem in which each client needs to be connected to the root by a simple path. More generally, various complicating conditions may be

considered for the individual connections such as hop- or length-constraints or redundancy requirements like having some customers more reliably connected via two independent paths. Thus, our abstract network design problem covers many classes of more specific problems. We can approach it by the following *connection formulation*:

$$\min \quad \sum_{e \in E} c_e x_e \tag{12}$$

$$\text{s.t.} \quad \sum_{p \in P_k} f_p^k \geq 1 \qquad \forall k \in C \tag{13}$$

$$x_e - \sum_{p \in P_k | e \in p} f_p^k \geq 0 \qquad \forall k \in C,\ e \in E \tag{14}$$

$$x_e \in \{0,1\} \qquad \forall e \in E \tag{15}$$

$$f_p^k \in \{0,1\} \qquad \forall k \in C,\ p \in P_k \tag{16}$$

Most remarkably, this formulation considers for each customer $k \in C$ explicitly the set of all possible feasible connections, referred to by P_k. Corresponding binary variables f_p^k for all $p \in P_k$ and $k \in C$ indicate which connections are realized. Binary variables x_e indicate the edges that are part of the solution. The objective function (12) minimizes the costs of these selected edges. Inequalities (13) ensure that (at least) one connection from P_k is realized for each customer (cover constraints), and inequalities (14) provide the linkage between the f_p^k and x_e variables, ensuring that all edges that are part of a chosen connection are indeed selected.

Obviously, this MIP has far too many f_p^k variables – corresponding to columns in the matrix notation – to be directly solved even for small instances, as there are in general exponentially many possible connections. In short, CG solves the LP relaxation of above model, called *master problem* (MP), by starting with a *restricted master problem* (RMP) that is obtained from MP by considering just a small subset of all connections and corresponding variables f_p^k, e.g., those contained in an initial heuristic solution. This RMP can be efficiently solved and is then iteratively extended by adding further variables for connections that likely lead to better solutions. Each time one or more variables have

been added, the RMP is resolved until no further improvements are possible.

The task of finding suitable variables to be added is the so-called *pricing subproblem* and directly derives from the mechanisms of the simplex method for solving LPs: When having a solution to the current RMP, we consider the corresponding dual variable values μ_k^* and $\pi_{k,e}^*$ associated with the cover inequalities (13) and linking constraints (14), respectively. They define *reduced costs*

$$\overline{c}_{k,p} = -\mu_k^* + \sum_{e \in p} \pi_{k,e}^* \qquad \forall k \in C,\ p \in P_k \tag{17}$$

for all possible connections, and we have to identify a variable/connection with negative reduced costs, i.e., $\overline{c}_{k,p} < 0$, as only such variables may yield an improvement in the RMP. If we can prove that no such variable exists anymore, the current RMP solution is also optimal for the MP and CG terminates.

In our abstract network design problem, the pricing problem is thus to find for some $k \in C$ a feasible connection $p \subseteq E$ having costs $\sum_{e \in p} \pi_{k,e}^* < \mu_k^*$ or to prove that none exists. Depending on the problem's specific requirements for connections, this task may be as simple as finding a shortest path or more complex when, e.g., hop- or length-constraints or redundancy requirements need to be satisfied. The conceptually nice aspect, however, is that these specificities only have to be dealt with in this pricing subproblem.

As pointed out, CG only solves the LP relaxation of a MIP model. To obtain guaranteed optimal solutions for the MIP, the approach needs to be extended to a branch-and-price, which is an LP based branch-and-bound in which CG is performed at every tree-node. However, as such MIP models are typically relatively strong, good heuristic solutions can usually be directly obtained from the LP solution by simple rounding or repairing.

With respect to metaheuristic hybrids several possibilities exist for boosting the performance.

- If the pricing subproblem is hard, meta-heuristics may be well suited for identifying variables with negative reduced costs. It should just be kept in mind that in the end an exact approach is necessary to prove that

no further variables with negative reduced cost exist in order to have the master problem solved exactly. Therefore, frequently a chain of algorithms is used for the pricing problem, starting with a fast greedy heuristic over some metaheuristic approaches up to a usually slowest exact approach; for example Puchinger & Raidl (2007) describe such a chained approach for a two-dimensional cutting problem.

- Metaheuristics may also become useful in conjunction with solving the integer master problem. On the one hand, the set of variables initially provided to the RMP obviously has a crucial impact on the performance, and thus deriving them from one or more good starting solutions originally determined by (meta-)heuristics often is beneficial. On the other hand, metaheuristics can also be used to derive a better final solution based on the results of CG than those obtained by simple rounding or repairing, cf. Section 2.7. In this context, several aspects may be exploited for guiding the heuristic search:
  - obviously, the LP solution;
  - the overall set of variables finally contained in the RMP; when considering our network design problem, one may, e.g., restrict the heuristic search to only those edges that appear in some connection corresponding to an included variable; note here the strong parallels to the LD-based hybrids described in Section 4;
  - reduced costs of variables may provide a more promising guidance than their LP values, again cf. Section 4;
  - dual variable values may provide more fine-grained guidance, e.g., in the case of our network design problem they may indicate which edges are more likely to yield improvements or should better be spared.
- Instead of applying CG and a metaheuristic sequentially and providing guidance in only one way, a collaborative approach may be considered, with both algorithms running intertwined or in parallel and mutually ex-

changing information; i.e., while the metaheuristic continuously exploits current LP solutions, dual variable information etc., it also sends newly found good solutions to the CG and corresponding variables are inserted in the RMP.

For successful examples of above mentioned concepts, see Filho & Lorena (2000), who describe a constructive genetic algorithm in conjunction with CG for graph coloring, Pirkwieser & Raidl (2010), who consider a variable neighborhood search and an evolutionary algorithm collaborating with CG to solve the periodic vehicle routing problem with time windows, and Massen et al. (2012), who apply ant colony optimization for heuristic CG to solve a black-box vehicle routing problem. Massen et al. (2013) show how the latter perhomone-based heuristic CG can further be improved by automatic algorithm configuration methods.

Alvelos et al. (2013) describe a general hybrid strategy called SearchCol, where CG and a metaheuristic are iteratively performed and information is exchanged between both. The metaheuristic works in a problem-independent way trying to find a best integral solution by searching over combinations of variables identified in CG, while the CG is perturbed in each iteration based on the metaheuristic's result by fixing subproblem variables with special constraints.

6. Benders' Decomposition and Metaheuristics

Benders' decomposition (BD) has been originally suggested for solving large MIPs involving "complicating" variables (Benders, 1962). It can be regarded dual to CG, as instead of iteratively adding variables to a RMP, inequalities, i.e., rows, are added.

We consider a MIP of the form

$$z_{\text{MIP}} = \min\{c^T x + c'^T y \mid Ax + By \geq b,\ Dx \geq d,\ x \in \mathbb{D}^n,\ y \geq 0\} \qquad (18)$$

with two kinds of decision variable vectors x and y. The x variables are "complicating" in the sense that when they are temporarily fixed the remaining problem

becomes considerable more tractable, e.g., because the remaining problem decouples into multiple independent problems with only continuous variables.

Benders' decomposition reformulates the MIP by considering only the x variables with their specific inequalities $Dx \geq d$ and accounting for the impact of the y variables by adding a function $z_{\mathrm{SP}}(x)$ to the objective

$$z_{\mathrm{B}} = \min\{c^T x + z_{\mathrm{SP}}(x) \mid Dx \geq d,\ x \in \mathbb{D}^n\}, \tag{19}$$

where $z_{\mathrm{SP}}(x)$ is the result of the subproblem

$$z_{\mathrm{SP}}(x) = \min\{c'^T y \mid By \geq b - Ax,\ y \geq 0\}. \tag{20}$$

In this subproblem, x is thus assumed to be a given constant vector. As the remaining variables y are continuous, this subproblem is an LP, for which we can also consider its dual form

$$z_{\mathrm{DP}}(x) = \max\{w^T(b - Ax) \mid w^T B \leq c',\ w \geq 0\}, \tag{21}$$

where w is the vector of dual variables associated with the inequalities $By \geq b - Ax$. Assuming the feasible region of this dual problem is bounded and not empty, let W be the set of corresponding extreme points. Then, we may also write $z_{\mathrm{DP}}(x) = \max_{w \in W} w^T(b - Ax)$ and rewrite our master problem (18) as

$$z_{\mathrm{MIP}} = \min\{z \mid z \geq c^T x + w^T(b - Ax)\ \forall w \in W,\ Dx \geq d,\ x \in \mathbb{D}^n\}. \tag{22}$$

Inequalities $z \geq c^T x + w^T(b - Ax)$ for all extreme points $w \in W$ are called *Benders' cuts*.

Computationally, Benders' decomposition starts by solving (22) with no or only a small set of initial Benders' cuts. This *reduced master problem* (RMP) yields initial values for x for which the subproblem $z_{\mathrm{SP}}(x)$ and its dual are solved. The dual solution corresponds to an extreme point $w \in W$ for which a respective Benders' cut can be derived and added to the RMP, usually cutting off its current solution x. The phases of (re-)solving the RMP and the subproblem are iterated until no new cut violated by x is found. The finally obtained solution x is then a minimum for the original MIP (18). More generally, the

primal subproblem (20) may sometimes be infeasible, yielding an unbounded dual subproblem (21). In these cases, extreme rays are used to derive *feasibility cuts* (in contrast to *optimality cuts*), driving the process towards feasibility by forbidding the current RMP solution x.

Note that in contrast to LD and CG, BD directly yields an optimal solution to an original MIP and not only to a relaxation of it; it is therefore not necessary to further embed it in a branch-and-bound to obtain an exact approach.

As an example consider a special variant of the abstract network design problem (12)–(16) from Section 5, called the *local access network design problem* (Randazzo & Luna, 2001). Each customer needs to be connected to the dedicated root node by a simple path and in addition to the fixed costs c_e for installing a link e also capacity-dependent costs c'_e arise. This problem can be modeled by the following *multi-commodity flow formulation*:

$$\min \quad \sum_{e \in E} c_e x_e + \sum_{(i,j) \in E_{\mathrm{D}}} c'_{(i,j)} \sum_{k \in C} f^k_{i,j} \tag{23}$$

$$\text{s.t.} \quad \sum_{(0,j) \in E_{\mathrm{D}}} f^k_{0,j} = b_k \qquad \forall k \in C \tag{24}$$

$$\sum_{(i,k) \in E_{\mathrm{D}}} f^k_{i,k} = b_k \qquad \forall k \in C \tag{25}$$

$$\sum_{(i,j) \in E_{\mathrm{D}}} f^k_{i,j} - \sum_{(j,i) \in E_{\mathrm{D}}} f^k_{j,i} = 0 \qquad \forall i \in V \setminus \{0,k\},\ k \in C \tag{26}$$

$$f^k_{i,j} \leq b_k x_{(i,j)} \qquad \forall k \in C,\ (i,j) \in E_{\mathrm{D}} \tag{27}$$

$$x_e \in \{0,1\} \qquad \forall e \in E \tag{28}$$

$$f^k_{i,j} \geq 0 \qquad \forall k \in C,\ (i,j) \in E_{\mathrm{D}} \tag{29}$$

The model sends for each customer $k \in C$ an individual commodity of size $b_k > 0$ (corresponding to the required capacity) from the root 0 to node k. For this purpose, arc set E_{D} is defined to contain two reversely directed arcs for each edge in E, and for each such arc and each commodity $k \in C$ a flow variable $f^k_{i,j}$ is used. Equations (24)–(26) are the classical flow conservation constraints. Inequalities (27) link the design variables x_e with the flow variables so that a flow may only occur over edges that are part of the solution.

We perform BD by considering variables x as the "complicating" ones and obtain as reformulated master problem

$$z_{\text{MP}} = \min\{z \mid z \geq c^T x + w^T(b - Ax)\ \forall w \in W,\ x \in \{0,1\}^{|E|}\}, \qquad (30)$$

where $Ax + Bf \geq b$ corresponds to the flow conservation and linking constraints (24)–(27). Thus, we have to select a minimum cost subset of edges only constrained by the Benders' cuts $z \geq c^T x + w^T(b - Ax)$. The respective Benders' subproblem becomes

$$z_{\text{SP}}(x) = \min\{c'^T f \mid Bf \geq b - Ax,\ f \geq 0\}. \qquad (31)$$

Due to the fixed x, the flow variables for different commodities $k \in C$ are here not linked anymore, and this subproblem decouples into $|C|$ independent problems of finding a minimum cost path from the root to each customer $k \in C$. Some of these problems may be infeasible as the root and respective customers might not be connected in the given solution x. Feasibility cuts based on extreme rays of the dual are then identified for moving towards feasibility. For example, such a Benders' cut might enforce to have at least a certain number α of edges from a specific subset $E' \subseteq E$ selected, i.e., $\sum_{e \in E'} x_e \geq \alpha$.

Randazzo & Luna (2001) present a comparison of a Lagrangean relaxation-based branch-and-bound, a branch-and-cut, and the above sketched BD extended with certain strengthening constraints. In their experiments, BD was the only algorithm able to solve to optimality all instances within a day, leading to the conclusion that it is the most robust method. More generally, different classes of Benders' cuts can be derived for this and related problems, and they have specific, partly complementary impacts on the overall performance (Magnanti et al., 1986; Costa, 2005). It is thus important to carefully choose a suitable set of Benders' cuts that are actually added in each iteration.

In classical BD as introduced above, the subproblem must be an LP involving only continuous variables. However, BD has also been generalized to certain kinds of non-linear problems (Geoffrion, 1972). Hooker & Ottosson (2003) proposed *logic-based BD*, an approach that is applicable to an even wider class of

subproblems including in particular also discrete ones. This is achieved by generalizing the LP dual to an *inference dual*. Constraint programming techniques turn out to be especially useful for handling discrete subproblems.

Similarly as with LD and CG, there are several possibilities for metaheuristics to come into play with BD:

- Although smaller than the original problem, the RMP is frequently still difficult to solve, especially when already many Benders' cuts have been added. Metaheuristics may provide good approximate RMP solutions in much shorter time, and these solutions may be sufficient in order to achieve substantial speedups of the overall approach. If finally, when the metaheuristic cannot identify an improved master solution, the RMP is solved to optimality and no further Benders' cuts can be identified, the whole approach is still complete. Poojari & Beasley (2009) describe such an approach for solving general MIPs in which a genetic algorithm together with a feasibility pump heuristic are applied to the RMP. The authors argue that a population based metaheuristic like a genetic algorithm is particularly useful as it provides multiple solutions in each iteration giving rise to more Benders' cuts. Similarly in spirit, Lai et al. (2010); Lai & Sohn (2012) propose a genetic algorithm/BD hybrid for solving the capacitated plant location problem; results indicate a tremendous saving of computation time in comparison to classical BD. Lai et al. (2012) further discuss such an approach for a vehicle routing problem. Rei et al. (2008) suggest to use local branching, cf. Section 2.8, for solving a MIP master problem in order to sooner find improved upper as well as lower bounds.

- Initial solutions obtained by some (meta-)heuristic may be used to derive an initial set of Benders' cuts in order to start with a more meaningful first RMP. For example, Easwaran & Üster (2009) apply a tabu search to warm-start a BD approach for a supply chain network design problem.

- Subproblems need not necessarily always to be solved to optimality in order to obtain useful Benders' cuts, even when completeness of the whole

approach shall be retained (Zakeri et al., 1999). Especially when considering difficult subproblems in logic-based BD, constraint programming and metaheuristics have a great potential for speeding up the overall approach by providing helpful cuts much faster. For example Hooker (2007) describes a constraint programming based BD that substantially outperforms pure MIP as well as constraint programming approaches on a large class of planning and scheduling problems. Cordeau et al. (2001) solve an aircraft routing and crew scheduling problem by applying BD and use CG-based heuristics for the RMP as well as the integer subproblem.

Raidl et al. (2014) proposed an exact logic-based BD approach for a bi-level capacitated vehicle routing problem and sped it up considerably by first solving all instances of the master problem as well as all subproblems by means of a fast variable neighborhood search heuristic. Invalid Benders' cuts possibly cutting off feasible solutions might be created. In a second phase, all these heuristically generated Benders' cuts are verified by re-solving the corresponding subproblems exactly by means of MIP, yielding possibly corrected cuts that replace the invalid ones. When finally also the master problem is solved exactly and no further Benders' cuts can be derived, a proven optimal solution is obtained.

7. Conclusions

Hybrid metaheuristics have shown to be successful advanced approaches for solving a wide range of practically relevant problems. On many occasions they are leading methods when dealing with large, complex combinatorial problems that cannot be solved to proven optimality in reasonable time. When designed appropriately HMs can significantly benefit from the advantages of the underlying basic strategies and exploit synergy.

The design of successful hybrids, however, usually is not trivial, and one should be aware that a more complex system is not necessarily always better. In fact, the principle of *"keeping things as simple as possible but not simpler"*

still holds also in this context. To develop an "as good as possible" heuristic optimization software for a new complex problem typically requires a considerable amount of experience, research of existing work on related problems, testing, comparing, and fine-tuning. This review intended to give an overview on the most successful design patterns of HMs and particularly focused on decomposition-based hybrids with origins in mathematical programming.

Decomposition approaches, in general, are particularly useful to address large problems possibly consisting of several dependent, difficult subproblems. Lagrangian decomposition, Dantzig-Wolfe decomposition with its column generation , and Benders' decomposition are well-known and frequently applied methods. The great potential they have when extended and combined with metaheuristics, however, has only been recognized more widely over the last decade. Besides, e.g., just providing initial approximate solutions for primal bounds, we have seen that far more possibilities exist for fruitful combinations. Metaheuristics are able to substantially enhance the applicability of these classical decomposition techniques – and vice versa.

More work is necessary to get an even better understanding of the conditions under which the individual design patterns are best suited. From the author's point of view, exploiting dual information in metaheuristics provided by Lagrangian relaxations and column generation approaches as well as metaheuristic approaches for solving separation problems in column generation and Benders' subproblems in logic-based Benders' decompositions may be particularly fruitful for future research.

References

Ahuja, R., Ergun, Ö., Orlin, J., & Punnen, A. (2002). A survey of very large-scale neighborhood search techniques. *Discrete Applied Mathematics*, *123*, 75–102.

Alvelos, F., de Sousa, A., & Santos, D. (2013). Combining column generation and metaheuristics. In Talbi (2013a).

Angelelli, E., Mansini, R., & Grazia Speranza, M. (2010). Kernel search: A general heuristic for the multi-dimensional knapsack problem. *Computers and Operations Research*, *37*, 2017–2026.

Applegate, D. L., Bixby, R. E., Chvátal, V., & Cook, W. J. (1998). On the solution of the traveling salesman problem. *Documenta Mathematica*, *Extra Volume ICM III*, 645–656.

Azi, N., Gendreau, M., & Potvin, J.-Y. (2014). An adaptive large neighborhood search for a vehicle routing problem with multiple routes. *Computers & Operations Research*, *41*, 167–173.

Barahona, F., & Anbil, R. (2000). The volume algorithm: Producing primal solutions with a subgradient method. *Mathematical Programming, Series A*, *87*, 385–399.

Benders, J. F. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, *4*, 238–252.

Blesa, M. J., Blum, C., Festa, P., Roli, A., & Sampels, M. (Eds.) (2013). *Proceedings of HM 2013 – Eigth International Workshop on Hybrid Metaheuristics* volume 7919 of *LNCS*. Springer.

Blesa, M. J., Blum, C., & Voss, S. (Eds.) (2014). *Proceedings of HM 2014 – Ninth International Workshop on Hybrid Metaheuristics* volume 8457 of *LNCS*. Springer.

Blum, C., & Blesa, M. J. (2008). Solving the KCT problem: Large-scale neighborhood search and solution merging. In E. Alba et al. (Eds.), *Optimization Techniques for Solving Complex Problems* (pp. 407–421). John Wiley & Sons.

Blum, C., Blesa Aguilera, M. J., Roli, A., & Sampels, M. (Eds.) (2008). *Hybrid Metaheuristics – An Emerging Approach to Optimization* volume 114 of *Studies in Computational Intelligence*. Springer.

Blum, C., Puchinger, J., Raidl, G. R., & Roli, A. (2011). Hybrid metaheuristics in combinatorial optimization: A survey. *Applied Soft Computing*, *11*, 4135–4151.

Blum, C., Roli, A., & Sampels, M. (Eds.) (2004). *Proceedings of HM 2004 – First International Workshop on Hybrid Metaheuristics*.

Boschetti, M., & Maniezzo, V. (2009). Benders decomposition, Lagrangian relaxation and metaheuristic design. *Journal of Heuristics*, *15*, 283–312.

Boschetti, M., Maniezzo, V., & Roffilli, M. (2009). Decomposition techniques as metaheuristic frameworks. In Maniezzo et al. (2009).

Büdenbender, K., Grünert, T., & Sebastian, H.-J. (2000). A hybrid tabu search/branch-and-bound algorithm for the direct flight network design problem. *Transportation Science*, *34*, 364–380.

Burke, E. K., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., zcan, E., & Qu, R. (2013). Hyper-heuristics: a survey of the state of the art. *Journal of the Operational Research Society*, *64*, 1695–1724.

Chu, P. C., & Beasley, J. E. (1998). A genetic algorithm for the multidimensional knapsack problem. *Journal of Heuristics*, *4*, 63–86.

Cordeau, J.-F., Stojković, G., Soumis, F., & Desrosiers, J. (2001). Benders decomposition for simultaneous aircraft routing and crew scheduling. *Transportation Science*, *35*, 375–388.

Costa, A. M. (2005). A survey on Benders decomposition applied to fixed-charge network design problem. *Computers & Operations Research*, *32*, 1429–1450.

Cotta, C., Talbi, E. G., & Alba, E. (2005). Parallel hybrid metaheuristics. In E. Alba (Ed.), *Parallel Metaheuristics: A New Class of Algorithms* (pp. 347–370). Wiley.

Danna, E., Rothberg, E., & Le Pape, C. (2005). Exploring relaxation induced neighborhoods to improve MIP solutions. *Mathematical Programming, Series A*, *102*, 71–90.

Dantzig, G. B., & Wolfe, P. (1960). Decomposition principle for linear programs. *Operations Research*, *8*, 101–111.

Desaulniers, G., Desrosiers, J., & Solomon, M. M. (2005). *Column Generation*. Springer.

Di Gaspero, L., & Urli, T. (2014). A CP/LNS approach for multi-day homecare scheduling problems. In Blesa et al. (2014).

Dowsland, K. A., Herbert, E. A., Kendall, G., & Burke, E. (2006). Using tree search bounds to enhance a genetic algorithm approach to two rectangle packing problems. *European Journal of Operational Research*, *168*, 390–402.

Easwaran, G., & Üster, H. (2009). Tabu search and Benders decomposition approaches for a capacitated closed-loop supply chain network design problem. *Transportation Science*, *43*, 301–320.

El-Abd, M., & Kamel, M. (2005). A taxonomy of cooperative search algorithms. In M. J. Blesa Aguilera, C. Blum, A. Roli, & M. Sampels (Eds.), *Proceedings of HM 2005 – Second International Workshop on Hybrid Metaheuristics* (pp. 32–41). Springer volume 3636 of *LNCS*.

Eremeev, A. V. (2008). On complexity of optimal recombination for binary representations of solutions. *Evolutionary Computation*, *16*, 127–147.

Feo, T. A., & Resende, M. G. C. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization*, *6*, 109–133.

Filho, G. R., & Lorena, L. A. N. (2000). Constructive genetic algorithm and column generation: an application to graph coloring. In L. P. Chuen (Ed.), *Proceedings of APORS 2000, the Fifth Conference of the Association of Asian-Pacific Operations Research Societies within IFORS*.

Fischetti, M., & Lodi, A. (2003). Local branching. *Mathematical Programming, Series B*, *98*, 23–47.

Fisher, M. L. (1981). The Lagrangian relaxation method for solving integer programming problems. *Management Science*, *27*, 1–18.

Fisher, M. L., & Jaikumar, R. (1981). A generalized assignment heuristic for vehicle routing. *Networks*, *11*, 109–124.

Fontaine, M., Loudni, S., & Boizumault, P. (2011). Guiding VNS with tree decomposition. In *Proceedings of the IEEE International Conference on Tools wirth Artificial Intelligence* (pp. 505–512). IEEE.

Geoffrion, A. M. (1972). Generalized Benders decomposition. *Journal of Optimization Theory and Applications*, *10*, 237–260.

Gilmour, S., & Dras, M. (2006). Kernelization as heuristic structure for the vertex cover problem. In M. Dorigo et al. (Eds.), *Ant Colony Optimization and Swarm Intelligence* (pp. 452–459). Springer volume 4150 of *LNCS*.

Glover, F. (1977). Future paths for integer programming and links to artificial intelligence. *Decision Sciences*, *8*, 156–166.

Glover, F., Laguna, M., & Martí, R. (2000). Fundamentals of scatter search and path relinking. *Control and Cybernetics*, *39*, 653–684.

Gomes, T. M., Santos, H. G., & Souza, J. F. (2013). A pre-processing aware RINS based MIP heuristic. In Blesa et al. (2013).

Hansen, P., Mladenovic, N., & Perez-Britos, D. (2001). Variable neighborhood decomposition search. *Journal of Heuristics*, *7*, 335–350.

Haouari, M., & Siala, J. C. (2006). A hybrid Lagrangian genetic algorithm for the prize collecting Steiner tree problem. *Computers & Operations Research*, *33*, 1274–1288.

Hart, W. E., Krasnogor, N., & Smith, J. E. (2005). *Recent Advances in Memetic Algorithms* volume 166 of *Studies in Fuzziness and Soft Computing*. Springer.

Hemmelmayr, V. C., Cordeau, J.-F., & Crainic, T. G. (2012). An adaptive large neighborhood search heuristic for two-echelon vehicle routing problems arising in city logistics. *Computers & Operations Research*, *39*, 3215–3228.

Hooker, J. N. (2007). Planning and scheduling by logic-based Benders decomposition. *Operations Research*, *55*, 588–602.

Hooker, J. N., & Ottosson, G. (2003). Logic-based Benders decomposition. *Mathematical Programming*, *96*, 33–60.

Jeet, V., & Kutanoglu, E. (2007). Lagrangian relaxation guided problem space search heuristic for generalized assignment problems. *European Journal of Operational Research*, *182*, 1039–1056.

Kellerer, H., Pferschy, U., & Pisinger, D. (2004). *Knapsack Problems*. Springer.

Lai, M.-C., & Sohn, H.-S. (2012). Using a genetic algorithm to solve the Benders master problem for capacitated plant location. In S. Gao (Ed.), *Bio-Inspired Computational Algorithms and Their Applications* (pp. 405–420). InTech.

Lai, M.-C., Sohn, H.-S., Tseng, T.-L., & Bricker, D. L. (2012). A hybrid Benders/genetic algorithm for vehicle routing and scheduling problem. *International Journal of Industrial Engineering*, *19*, 33–46.

Lai, M.-C., Sohn, H.-s., Tseng, T.-L., & Chiang, C. (2010). A hybrid algorithm for capacitated plant location problem. *Expert Systems with Applications*, *37*, 8599–8605.

Leitner, M., Ljubić, I., Luipersbeck, M., & Resch, M. (2014). A partition-based heuristic for the Steiner tree problem in large graphs. In Blesa et al. (2014).

Leitner, M., & Raidl, G. R. (2008). Lagrangian decomposition, metaheuristics, and hybrid approaches for the design of the last mile in fiber optic networks.

In M. J. Blesa Aguilera, C. Blum, C. Cotta, A. J. Fernández, J. E. Gallardo, A. Roli, & M. Sampels (Eds.), *Proceedings of HM 2008 – Fifth International Workshop on Hybrid Metaheuristics* (pp. 158–174). Springer volume 5296 of *LNCS*.

Leitner, M., & Raidl, G. R. (to appear). Combining Lagrangian decomposition with very large scale neighborhoood search for capacitated connected facility location. In *Post-Conference Book of the Eight Metaheuristics International Conference – MIC 2009*.

Lodi, A. (2013). The heuristic (dark) side of MIP solvers. In Talbi (2013a).

Lopes, R., Morais, V. W. C., Noronha, T. F., & Souza, V. A. A. (2014). Heuristics and matheuristics for a real-life machine reassignment problem. *International Transactions in Operational Research*, *40*, 179–200.

Lübbecke, M. E., & Desrosiers, J. (2005). Selected topics in column generation. *Operations Research*, *53*, 1007–1023.

Magnanti, T. L., Mireault, P., & Wong, R. T. (1986). Tailoring Benders decomposition for uncapacitated network design. In G. Gallo, & C. Sandi (Eds.), *Netflow at Pisa* number 26 in Mathematical Programming Studies (pp. 112–154). Springer.

Maniezzo, V., Hansen, P., & Voss, S. (Eds.) (2006). *Proceedings of Matheuristics 2006: First International Workshop on Mathematical Contributions to Metaheuristics*. Bertinoro, Italy.

Maniezzo, V., Stützle, T., & Voss, S. (Eds.) (2009). *Matheuristics – Hybridizing Metaheuristics and Mathematical Programming* volume 10 of *Annals of Information Systems*. Springer.

Martello, S., Pisinger, D., & Toth, P. (1999). Dynamic programming and strong bounds for the 0–1 knapsack problem. *Management Science*, *45*, 414–424.

Martins, S. L., & Ribeiro, C. C. (2006). Metaheuristics and applications to optimization problems in telecommunications. In M. G. C. Resende, & P. M. Pardalos (Eds.), *Handbook of Optimization in Telecommunications* (pp. 103–128). Springer.

Massen, F., Deville, Y., & Hentenryck, P. V. (2012). Pheromone-based heuristic column generation for vehicle routing problems with black box feasibility. In N. Beldiceanu et al. (Eds.), *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems – CPAIOR 2012* (pp. 260–274). Springer volume 7298 of *LNCS*.

Massen, F., López-Ibáñez, M., Stützle, T., & Deville, Y. (2013). Experimental analysis of pheromone-based heuristic column generation using irace. In Blesa et al. (2013).

Meyer, B., & Ernst, A. (2004). Integrating ACO and constraint propagation. In M. Dorigo et al. (Eds.), *Proceedings of ANTS 2004 – Fourth International Workshop on Ant Colony Optimization and Swarm Intelligence* (pp. 166–177). Springer volume 3172 of *LNCS*.

Moscato, P. (1999). Memetic algorithms: A short introduction. In D. Corne et al. (Eds.), *New Ideas in Optimization* (pp. 219–234). McGraw Hill.

Nemhauser, G. L., & Wolsey, L. A. (1988). *Integer and Combinatorial Optimization*. Wiley & Sons.

Ouali, A., Loudni, S., Loukil, L., Boizumault, P., & Lebbah, Y. (2014). Cooperative parallel decomposition guided VNS for solving weighted CSP. In Blesa et al. (2014).

Pessoa, L. S., Resende, M. G. C., & Ribeiro, C. C. (2013). A hybrid Lagrangean heuristic with GRASP and path-relinking for set k-covering. *Computers and Operations Research*, *40*, 3132–3146.

Pirkwieser, S., & Raidl, G. R. (2010). Matheuristics for the periodic vehicle routing problem with time windows. In *Proceedings of Matheuristics 2010:*

Third International Workshop on Model-Based Metaheuristics (pp. 83–95). Vienna, Austria.

Pirkwieser, S., Raidl, G. R., & Puchinger, J. (2007). Combining Lagrangian decomposition with an evolutionary algorithm for the knapsack constrained maximum spanning tree problem. In C. Cotta, & J. I. van Hemert (Eds.), *Evolutionary Computation in Combinatorial Optimization – EvoCOP 2007* (pp. 176–187). Springer volume 4446 of *LNCS*.

Poojari, C. A., & Beasley, J. E. (2009). Improving Benders decomposition using a genetic algorithm. *European Journal of Operational Research*, *199*, 89–97.

Prandtstetter, M., & Raidl, G. R. (2008). An integer linear programming approach and a hybrid variable neighborhood search for the car sequencing problem. *European Journal of Operational Research*, *191*, 1004–1022.

Prins, C., Lacomme, P., & Prodhon, C. (2014). Order-first split-second methods for vehicle routing problems: A review. *Transportation Research Part C*, *40*, 179–200.

Puchinger, J., & Raidl, G. R. (2005). Combining metaheuristics and exact algorithms in combinatorial optimization: A survey and classification. In *Proceedings of the First International Work-Conference on the Interplay Between Natural and Artificial Computation, Part II* (pp. 41–53). Springer volume 3562 of *LNCS*.

Puchinger, J., & Raidl, G. R. (2007). Models and algorithms for three-stage two-dimensional bin packing. *European Journal of Operational Research*, *183*, 1304–1327.

Puchinger, J., Raidl, G. R., & Pferschy, U. (2010). The multidimensional knapsack problem: Structure and algorithms. *INFORMS Journal on Computing*, *22*, 250–265.

Raidl, G. R. (2006). A unified view on hybrid metaheuristics. In F. Almeida, M. J. Blesa Aguilera, C. Blum, J. M. Moreno Vega, M. P. Pérez, A. Roli, &

M. Sampels (Eds.), *Proceedings of HM 2006 – Third International Workshop on Hybrid Metaheuristics* (pp. 1–12). Springer volume 4030 of *LNCS*.

Raidl, G. R., Baumhauer, T., & Hu, B. (2014). Speeding up logic-based Benders' decomposition by a metaheuristic for a bi-level capacitated vehicle routing problem. In Blesa et al. (2014).

Raidl, G. R., & Hu, B. (2010). Enhancing genetic algorithms by a trie-based complete solution archive. In P. Cowling, & P. Merz (Eds.), *Evolutionary Computation in Combinatorial Optimisation – EvoCOP 2010* (pp. 239–251). Springer volume 6022 of *LNCS*.

Raidl, G. R., & Puchinger, J. (2008). Combining (integer) linear programming techniques and metaheuristics for combinatorial optimization. In Blum et al. (2008).

Raidl, G. R., Puchinger, J., & Blum, C. (2010). Metaheuristic hybrids. In M. Gendreau, & J. Y. Potvin (Eds.), *Handbook of Metaheuristics* (pp. 469–496). Springer volume 146 of *International Series in Operations Research & Management Science*. (2nd ed.).

Randazzo, C. D., & Luna, H. P. L. (2001). A comparison of optimal methods for local access uncapacitated network design. *Annals of Operations Research*, *106*, 263–286.

Rei, W., Cordeau, J.-F., Gendreau, M., & Soriano, P. (2008). Accelerating Benders decomposition by local branching. *INFORMS Journal on Computing*, *21*, 333–345.

Ropke, S., & Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, *40*, 455–472.

S. Loudni, P. B., M. Fontaine (2013). Intensification/diversification in decomposition guided VNS. In Blesa et al. (2013).

Shaw, P. (1998). Using constraint programming and local search methods to solve vehicle routing problems. In M. Maher, & J.-F. Puget (Eds.), *Principle and Practice of Constraint Programming – CP98* (pp. 417–431). Springer volume 1520 of *LNCS*.

Taillard, É. D., & Voß, S. (2001). POPMUSIC: Partial optimization metaheuristic under special intensification conditions. In C. C. Ribeiro, & P. Hansen (Eds.), *Essays and Surveys in Metaheuristics* (pp. 613–629). Kluwer Academic Publishers.

Talbi, E.-G. (2002). A taxonomy of hybrid metaheuristics. *Journal of Heuristics*, *8*, 541–565.

Talbi, E.-G. (Ed.) (2013a). *Hybrid Metaheuristics* volume 434 of *Studies in Computational Intelligence*. Springer.

Talbi, E.-G. (2013b). A unified taxonomy of hybrid metaheuristics with mathematical programming, constraint programming and machine learning. In Talbi (2013a).

Thiruvady, D., Singh, G., & Ernst, A. T. (2014). Hybrids of integer programming and ACO for resource constrained job scheduling. In Blesa et al. (2014).

Valejo, A., Valverde-Rebaza, J., Drury, B., & Lopes, A. d. A. (2014). Multilevel refinement based on neighborhood similarity. In *Proceedings of the 18th International Database Engineering & Applications Symposium* IDEAS '14 (pp. 67–76). ACM.

Walshaw, C. (2008). Multilevel refinement for combinatorial optimisation: Boosting metaheuristic performance. In Blum et al. (2008).

Wolpert, D., & Macready, W. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, *1*, 67–82.

Wolsey, L. A. (1998). *Integer Programming*. Wiley-Interscience.

Zakeri, G., Philpott, A. B., & Ryan, D. M. (1999). Inexact cuts in Benders decomposition. *SIAM Journal on Optimization*, *10*, 643–657.