J Heuristics (2009) 15: 283–312
DOI 10.1007/s10732-007-9064-9

Benders decomposition, Lagrangean relaxation and metaheuristic design

Marco Boschetti · Vittorio Maniezzo

Received: 15 November 2006 / Revised: 30 March 2007 / Accepted: 3 April 2007 /
Published online: 23 October 2007

Abstract Large part of combinatorial optimization research has been devoted to the study of exact methods leading to a number of very diversified solution approaches. Some of those older frameworks can now be revisited in a metaheuristic perspective, as they are quite general frameworks for dealing with optimization problems. In this work, we propose to investigate the possibility of reinterpreting decompositions, with special emphasis on the related Benders and Lagrangean relaxation techniques. We show how these techniques, whose heuristic effectiveness is already testified by a wide literature, can be framed as a "master process that guides and modifies the operations of subordinate heuristics", i.e., as metaheuristics. Obvious advantages arise from these approaches, first of all the runtime evolution of both upper and lower bounds to the optimal solution cost, thus yielding both a high-quality heuristic solution and a runtime quality certificate of that same solution.

Keywords Combinatorial optimization · Lagrangean relaxation · Benders's decomposition · Metaheuristic

1 Introduction

It has been 20 years since metaheuristics were introduced as such (Glover 1986), mainly with the objective of overcoming the brittleness of problem-specific approaches and to enable the fast design of effective procedures. Several different metaheuristic paradigms have been presented in the literature, but the current trend seems

M. Boschetti (✉)
Department of Mathematics, University of Bologna, Bologna, Italy
e-mail: boshett@csr.unibo.it

V. Maniezzo
Department of Computer Science, University of Bologna, Bologna, Italy
e-mail: vittorio.maniezzo@unibo.it

to emphasize more and more the role of local search as an essential ingredient of any of them. In fact, most paradigms are either directly based on local search (simulated annealing, tabu search, variable neighborhood search, iterated local search, guided local search, ...) or definitely benefit from hybridization with it (genetic algorithms, scatter search, ...). The literature shows that such techniques are often able to produce the best known heuristic solutions on the instance sets used for benchmarking. However, the harder it is to find feasible neighbors, the more ineffective they become, the set partitioning problem being a paradigmatic example thereof.

All current metaheuristics are primal-only methods. Possibly, the use of dual information and more in general of mathematical programming based results, could help designing different approaches. This implies revisiting much mathematical programming theory, which was usually devised for exact approaches, in a metaheuristic framework. In this work, we propose to investigate the possibility of reinterpreting decompositions, with special emphasis on the related Benders and Lagrangean relaxation techniques, from a metaheuristic perspective.

Lagrangean and Benders heuristics have been around for decades and are mentioned in a number of papers. However, they were either seen as the poor relatives of accompanying exact methods—thus deserving at most some hints as a subsection of a paper—or they were proposed as very problem-specific heuristics, thereby reproducing the *ante*-metaheuristics setting. We propose to show here that, on the contrary, the structure of such approaches can be sufficiently robust to propose them as full-fledged metaheuristics, with the accompanying bonus of the entailed well-grounded structure.

The naming "metaheuristic" was introduced by F. Glover, who denoted tabu search "as a 'meta-heuristic' superimposed on another heuristic" (Glover 1986). This definition was made more normative by stating that a metaheuristic "refers to a master strategy that guides and modifies other heuristics to produce solutions beyond those that are normally generated in a quest for local optimality" (Glover and Laguna 1997). A discussion on this topic by S. Voss concludes that a metaheuristic is "an iterative master process that guides and modifies the operations of subordinate heuristics to efficiently produce high-quality solutions. (...) The family of meta-heuristics includes, but is not limited to, adaptive memory procedures, tabu search, ant systems, greedy randomized adaptive search, variable neighborhood search, evolutionary methods, genetic algorithms, scatter search, neural networks, simulated annealing, and their hybrids" (Voss 2001). The need for problem specialization is made even more explicit in P. Hansen's position, according to which "a metaheuristic is a schema for designing heuristics".

According to this, decompositions such as Benders' or Lagrangean (but the same would apply also to Dantzig-Wolfe) provide a rich framework for designing metaheuristics. There is an entrance barrier to acknowledge, in that these model-based approaches require some familiarity with basic decomposition techniques, but this is rewarded by the possibility to obtain:

- Both an upper and a lower bound, thus an a posteriori quality check of the solution obtained, which is a feature totally absent in primal-only methods but of both theoretical and practical significant importance;

- Optimality conditions, which could permit an early process closure, other than standard termination conditions;
- Reduction of search space, as dual information can be used to prune expensive decision variables;
- Multiple quality starting points for local search, provided by the different solutions constructed by the heuristics.

Examples of such model-based metaheuristics are proposed in Sect. 2 of this paper. Section 3 introduces the problems used for validating the algorithms put forth, a very standard one (Single Source Capacitated Facility Location) used as a direct example, and two more complex ones (Membership Overlay and Multi-Mode Project Scheduling) used to introduce some research issues. Section 4 shows the application and the results obtained with the Lagrangean approach, while Sect. 5 shows those obtained with the Benders approach.

2 Methodologies

This section briefly overviews the two decomposition techniques we will use as a basis for metaheuristics design and shows how metaheuristic frameworks can be directly derived. Notice that the proposed algorithms are not the only ones that could be derived from the used decompositions, but they represent reasonable frameworks we already use with success on different problems. Hopefully, this paper could also serve as a means to foster research on different or more general metaheuristic frameworks, including the main heuristic approaches deriving from decomposition techniques.

2.1 Benders' and Lagrangean decompositions

Both decompositions can be applied from continuous up to pure integer linear programming. Since the topic is quite standard in any basic mathematical programming textbook, we will only sketch here the basic formulae in the case of a MIP problem, where we will assume, for ease of presentation, that the feasibility region is non-null and bounded.
Consider the following problem P:

$$z_P = \min \ \mathbf{c_1 x} + \mathbf{c_2 y} \tag{1}$$

$$\text{s.t.} \ \mathbf{Ax} + \mathbf{By} \geq \mathbf{b} \tag{2}$$

$$\mathbf{Dy} \geq \mathbf{d} \tag{3}$$

$$\mathbf{x} \geq \mathbf{0} \tag{4}$$

$$\mathbf{y} \geq \mathbf{0} \text{ and integer} \tag{5}$$

Lagrangean relaxation suggests to penalize some constraints, for example constraints 2, by means of *Lagrangean penalties* $\boldsymbol{\lambda}$, $\boldsymbol{\lambda} \geq \mathbf{0}$, thereby obtaining the following formulation L:

$$z_{\mathrm{L}} = \min \ \mathbf{c_1 x} + \mathbf{c_2 y} + \boldsymbol{\lambda}(\mathbf{b} - \mathbf{Ax} - \mathbf{By}) \tag{6}$$
$$\text{s.t.} \ \ \mathbf{Dy} \geq \mathbf{d} \tag{7}$$
$$\mathbf{x} \geq \mathbf{0} \tag{8}$$
$$\mathbf{y} \geq \mathbf{0} \text{ and integer} \tag{9}$$

It is easily shown that $z_{\mathrm{L}} \leq z_{\mathrm{P}}$. Being interested in the best possible lower bound, the *Lagrangean dual* problem arises, asking to identify the lambdas which maximize the lower bound. Mathematically, the following formulation

$$z_{\mathrm{L}} = \max \ \boldsymbol{\lambda}\mathbf{b} + z_{\mathrm{LR}}(\boldsymbol{\lambda}) \tag{10}$$
$$\text{s.t.} \ \ \boldsymbol{\lambda} \geq \mathbf{0} \tag{11}$$

with

$$z_{\mathrm{LR}}(\boldsymbol{\lambda}) = \min \ (\mathbf{c_1} - \boldsymbol{\lambda}\mathbf{A})\mathbf{x} + (\mathbf{c_2} - \boldsymbol{\lambda}\mathbf{B})\mathbf{y} \tag{12}$$
$$\text{s.t.} \ \ \mathbf{Dy} \geq \mathbf{d} \tag{13}$$
$$\mathbf{x} \geq \mathbf{0} \tag{14}$$
$$\mathbf{y} \geq \mathbf{0} \text{ and integer} \tag{15}$$

shows the structure of the internal *subproblem* LR and of the external Lagrangean Dual. Notice that it is possible to add to problems L and LR constraints that are redundant in the original formulation, but can help the convergence.

Benders decomposition (Benders 1962) suggests to iterate through two phases, where the first one fixes the $\mathbf{y}$ variables to some $\bar{\mathbf{y}}$ value, then the second one solves the resulting subproblem. This translates into decomposing problem P as follows:

$$z_{\mathrm{B}} = \min \ \mathbf{c_2 y} + z_{\mathrm{SP}}(\mathbf{y}) \tag{16}$$
$$\text{s.t.} \ \ \mathbf{Dy} \geq \mathbf{d} \tag{17}$$
$$\mathbf{y} \geq \mathbf{0} \text{ and integer} \tag{18}$$

where

$$z_{\mathrm{SP}}(\mathbf{y}) = \min \ \mathbf{c_1 x} \tag{19}$$
$$\text{s.t.} \ \ \mathbf{Ax} \geq \mathbf{b} - \mathbf{By} \tag{20}$$
$$\mathbf{x} \geq \mathbf{0} \tag{21}$$

The internal subproblem SP can be equivalently formulated in dual form, by associating dual variables $\mathbf{w}$ to constraints 20. The result is the following problem DP:

$$z_{\mathrm{DP}} = \max \ \mathbf{w}(\mathbf{b} - \mathbf{By}) \tag{22}$$
$$\text{s.t.} \ \ \mathbf{wA} \leq \mathbf{c_1} \tag{23}$$
$$\mathbf{w} \geq \mathbf{0} \tag{24}$$

This is a standard LP problem that, under the assumption of boundedness, has the optimum at an extreme point. By denoting as $\{\mathbf{w}^t, t = 1, \ldots, T\}$ the set of the extreme points of the feasible region (which we have assumed finite and non-null), the subproblem becomes $\max_{t=1,\ldots,T} \mathbf{w}^t(\mathbf{b} - \mathbf{B}\mathbf{y})$. Therefore, upon denoting by $\mathbf{Y}$ the feasible region induced by constraints 3 and 5, i.e. $\mathbf{Y} = \{\mathbf{y} : \mathbf{D}\mathbf{y} \geq \mathbf{d}, \mathbf{y} \geq \mathbf{0}$ and integer$\}$, problem B can be reformulated as $\min_{\mathbf{y}\in\mathbf{Y}}\{\mathbf{c_2}\mathbf{y} + \max_{t=1,\ldots,T} \mathbf{w}^t(\mathbf{b} - \mathbf{B}\mathbf{y})\}$, or, equivalently, as $\max_{t=1,\ldots,T} \min_{\mathbf{y}\in\mathbf{Y}} \mathbf{c_2}\mathbf{y} + \mathbf{w}^t(\mathbf{b} - \mathbf{B}\mathbf{y})$.

The internal maximization problem can thus be reformulated as:

$$z_{\text{MP}} = \min \ z \tag{25}$$

$$\text{s.t.} \quad z \geq \mathbf{c_2}\mathbf{y} + \mathbf{w}^t(\mathbf{b} - \mathbf{B}\mathbf{y}), \qquad t = 1, \ldots, T \tag{26}$$

$$\mathbf{y} \in \mathbf{Y} \tag{27}$$

Usually one starts with a small number T' of *Benders' cuts* 26, i.e., $T' \ll T$. In this case problem MP is called the *master problem*, it is solved with the T' available cuts, then the *subproblem* DP is solved, in order to ascertain whether an additional cut should be added to the master or the solution is already optimal. Even for problems MP and SP, it is possible to add constraints that are redundant in the original formulation, but can help the convergence.

Notice that, upon refactoring formulation L, the Lagrangean relaxation of problem P becomes:

$$z_{\text{L}} = \min \ \mathbf{c_2}\mathbf{y} + z_{\text{SL}}(\mathbf{y}) \tag{28}$$

$$\text{s.t.} \quad \mathbf{D}\mathbf{y} \geq \mathbf{d} \tag{29}$$

$$\mathbf{y} \geq \mathbf{0} \text{ and integer} \tag{30}$$

where

$$z_{\text{SL}}(\mathbf{y}) = \max \ (\mathbf{c_1} - \boldsymbol{\lambda}\mathbf{A})\mathbf{x} + \boldsymbol{\lambda}(\mathbf{b} - \mathbf{B}\mathbf{y}) \tag{31}$$

$$\text{s.t.} \quad \mathbf{x}, \boldsymbol{\lambda} \geq \mathbf{0} \tag{32}$$

The analogy with problem B is apparent, all the more so when using $\mathbf{w}$ instead of $\boldsymbol{\lambda}$ to denote the penalties. Building upon this, van Roy (1986) developed his cross decomposition approach, an intriguing technique, unfortunately not directly applicable in our case.

2.2 A Lagrangean metaheuristic

The literature is rich with heuristics based on the decomposition structure outlined above. An excellent introduction to the whole topic of Lagrangean relaxation, and of related heuristics, can be found in Beasley (1993). A general structure is, however, common to most applications, which is the following.

```
LagrHeuristic()
1   identify an “easy” subproblem LR($\mathbf{x}$, $\boldsymbol{\lambda}$)
2   repeat
3         solve subproblem LR($\mathbf{x}$, $\boldsymbol{\lambda}$)
4         check for unsatisfied constraints
5         update penalties $\boldsymbol{\lambda}$
6         construct problem solution using $\mathbf{x}$ and $\boldsymbol{\lambda}$
7   until (end_condition)
```

The pseudocode is obviously underspecified for a direct application, being at the same abstraction level metaheuristics are usually presented, see Genetic Algorithms or Scatter Search. However, notice that this structure already shows the essential ingredients of a metaheuristics, i.e., it is “an iterative master process that guides and modifies the operations of a subordinate heuristic” at Step 6.

Steps 1 and 3 are problem-dependent, such as neighborhoods definition or crossover implementation in other contexts. Step 4 is trivial, while Step 5 can be implemented by means of any state of the art techniques, usually subgradient optimization or bundle methods. Notice moreover that some of these techniques have been proved to converge not only to the optimal $\boldsymbol{\lambda}$, but also to the optimal $\mathbf{x}$ of the linear relaxation (see Sherali and Choi 1996 and Barahona and Anbil 2000), thereby possibly providing a particularly “intelligent” starting point for Step 6.

2.3 A Benders’ metaheuristic

The identification of a common structure for Benders’ based heuristics is more difficult than for Lagrangean ones, since the proposals in the literature vary much, and usually Benders is utilized in a very problem-dependent fashion. We will thus propose here one possible structure, which already proved effective, but alternative ones are possible.

The structure can be applied both to MIP problems, as sketched in Sect. 2.1, and to pure IP problems. The subproblem SP($\mathbf{x}$) could be defined over integer or binary variables, in this case it is necessary its linear relaxation in order to obtain its dual DP($\mathbf{w}$).

```
BendHeuristic()
1   identify a master MP(z,$\mathbf{y}$) and an “easy” subproblem SP($\mathbf{x}$), set $t = 0$
2   repeat
3         solve (heuristically) master problem MP$^t$. Solution ($z^t$, $\mathbf{y}^t$)
4         if $\mathbf{x}$ are requested to be integer
5            then solve (heuristically) problem SP($\mathbf{x}$), solution ($z_{\mathrm{H}}$, $\mathbf{x}^t$)
6         solve problem DP($\mathbf{w}^t$), solution ($z_d$, $\mathbf{w}^t$), and add to MP the
          corresponding cut
7         if no more cuts can be added
8            then STOP else set $t = t + 1$
9   until (end_condition)
```

Taking into account the intrinsic difficulty of both MP and SP, we thus propose to consider solving them both heuristically.

The effect of solving heuristically MP at Step 3 is that it is not guaranteed to produce a lower bound to problem P. When a lower bound is needed, MP must be solved to optimality, or approximated from below. Notice, however, that the main purpose of MP is to produce alternative **y** sets, of possibly increasing qualities, and this can be effectively accomplished by heuristic solutions.

Step 5 provides an upper bound, i.e., a feasible solution, to the whole problem.

Step 6 finds a lower bound to the problem obtained by fixing the **y** variables.

The terminating condition at Step 7 depends on whether the master is solved heuristically or to optimality. In this last case, the condition would be "**if** $z^t \geq z_d$", which in fact implies the impossibility to generate new cuts. However, in a heuristic context such as that admitted by Steps 3 and 5, new cuts could be further generated, which could prove useful for continuing search.

3 Problems

The algorithms presented in Sect. 2 are meant as metaheuristics. They are relatively simple, yet effective robust structured approaches. To get state of the art results some sophisticated elements are needed, for these as for any other metaheuristic. In order to show robustness, we report about the application of each proposed approaches to two different problems. The first problem, which is used to show a simple straightforward application of the pseudocodes of Sect. 2, is the Single Source Capacitated Facility Location. Then, we show how very effective procedures can be obtained by including more advanced elements in the basic framework, reporting an application of LAGRHEURISTIC to the Membership Overlay problem and an application of BENDHEURISTIC to the Multi-Mode Project Scheduling problem.

In the following subsections we will introduce these problems, before describing the applications.

3.1 Single source capacitated facility location

The Single Source Capacitated Facility Location Problem (SCFLP), is a well-known problem that arises in many sectors, from clustering problems in data mining to networks design. The problem asks to locate a number of facilities (e.g., plants, warehouses or hubs), that must provide a service to a set of customers, minimizing a global cost. The cost includes fixed charges for opening the facilities and transportation costs for satisfying customer demands.

Let $J = \{1, \dots, n\}$ be the index sets of customers and $I = \{1, \dots, m\}$ the index set of possible facility locations. Each customer j has an associated demand, q_j, that must be served by a single facility; a facility located at site i has an overall capacity of Q_i.

The costs are composed of a cost c_{ij} for supplying the demand of a customer j from a facility established at location i and of a fixed cost, f_i, for opening a facility at location i.

Let x_{ij}, $i = 1, \dots, m$, $j = 1, \dots, n$, be binary variables such that $x_{ij} = 1$ if customer j is assigned to a facility located at i, 0 otherwise, and let y_i, $i = 1, \dots, m$, be binary variables such that $y_i = 1$ if a facility is located at site i, 0 otherwise.

A mathematical formulation of the SCFLP is as follows:

$$z_{\text{SCFLP}} = \min \sum_{i \in I, j \in J} c_{ij} x_{ij} + \sum_{i \in I} f_i y_i \tag{33}$$

$$\text{s.t.} \quad \sum_{i \in I} x_{ij} = 1, \quad j \in J \tag{34}$$

$$\sum_{j \in J} q_j x_{ij} \le Q_i y_i, \quad i \in I \tag{35}$$

$$x_{ij} \in \{0, 1\}, \quad i \in I, j \in J \tag{36}$$

$$y_i \in \{0, 1\}, \quad i \in I \tag{37}$$

The objective function 33 asks to minimize the sum of fixed and service costs. Assignment constraints 34 ensure that all customers are serviced by exactly one facility; knapsack constraints 35 are the facility capacity constraints and, finally, constraints 36 and 37 are the integrality constraints.

SCFLP is an NP-hard problem, and it has often been used for benchmarking new approaches given its simple structure. Some variants of it exist, the most studied one permitting a split assignment of customers to location, thus relaxing constraints 36 to $x_{ij} \ge 0$, $i \in I$, $j \in J$. Most approximation results, such as Chudak and Shmoys's 3-approximation algorithm (Chudak and Shmoys 1999), refer to this problem version. Closely related problems are also the Capacitated p-medians and the Generalized Assignment. Several exact approaches have been proposed for SCFLP, one of the best known being Neebe and Rao (1983), where a branch and bound scheme based on a partitioning formulation is proposed. However, exact methods cannot scale up to big problem size.

Large instances have been tackled by means of different kinds of heuristics, from VLSN (Ahuja et al. 2003) to reactive GRASP and tabu search (Delmaire et al. 1999). Extensive research has been devoted to Lagrangean heuristic for the SCFLP. Most authors start by relaxing assignment constraints, obtaining a Lagrangean subproblem which separates into n knapsack problems, one for each facility, whose combined solutions provide a lower bound to the problem (Barcelo and Casanova 1984; Pirkul 1987; Sridharan 1991; Holmberg et al. 1999; Holt et al. 1999). However, different relaxations have also been used. Klincewicz and Luss (1986) on the contrary relax the capacity constraints 35, thereby obtaining as Lagrangean subproblem an uncapacitated facility location problem, which is solved heuristically. Beasley (1993) and Agar and Salhi (1998) relax both the assignment and the capacity constraints, and obtain a very robust solution approach, which provides good quality solutions to a number of different location problems, including p-median, uncapacitated, capacitated and single source facility location problems.

3.2 Membership overlay

The Membership Overlay Problem (MOP) is a problem that arises in peer-to-peer (P2P) network design. P2P computing is a new field of distributed computing that supports several novel applications, such as file sharing, grid computing, distributed search, distributed hash tables, etc. This new field poses a large number of new optimization problems, most of which are dynamic in nature.

The MOP arises in all P2P applications that are large and fully decentralized: lacking a central service, participating nodes talk to each other directly, typically using a relatively small list of peer nodes they are aware of. The optimization problem arises through the fact that the peers have to intelligently select the other peers they communicate with, so that no participant gets overloaded but available bandwidth is reliably utilized by all network nodes. Current solutions are typically based on random or pseudo-random topologies (Ganesh et al. 2003). These topologies have a number of advantages from a theoretical point of view, but they ignore bandwidth and other constraints. We refer the interested reader to (Boschetti et al. 2006) for further details.

The set of known peers at each node defines the *overlay network*, where there is a directed link between nodes i and j if i has the IP address of j in its list of peers. That is, node j is connected to, or a neighbor of, i if i allocates non-zero bandwidth for communicating with j. This network defines the membership in the P2P group. The structure of this network has a major impact on the performance of the communication.

A mathematical model of the MOP can be formulated on a graph $G = (V, E)$ of n vertices, where the nodes correspond to peers and the edges correspond to possible connections: if there is an edge (i, j) then i can possibly send a message to j using the underlying routing infrastructure, and j to i. When two nodes i and j establish a connection, each one must allocate part of its bandwidth. Each node can dynamically enter and exit the network, and when it is connected it can make use of a limited bandwidth. In the model, each node has two associated weights, p_i and w_i, $i = 0, \dots, n$, corresponding to its *uptime* (measured as the percentage of time that the peer is available and responding to traffic (Saroiu et al. 2002), normalized to 1) and to the available bandwidth of its connection to the Internet, respectively.

The MOP asks to find a subgraph $G' = (V, E')$ of G. The edges in the graph G' define the fact that two nodes actually decide to allocate some bandwidth to communicate with each other. If b_i and b_j are the bandwidths which could be allocated by i and j, then the bandwidth of the connection can be at most $b_{ij} = \min\{b_i, b_j\}$. The two values b_i and b_j could be equal to w_i and w_j or could be less than that, due to other connections already maintained by the peers. Moreover, there is a lower bound l_{ij} on the bandwidth of acceptable connections and it is anyway possible to put a limit u_{ij} on the maximal value that b_{ij} can take.

The algorithm for solving this problem has to be local: no global knowledge of the network is provided, each node i can exchange information only with the nodes in $\delta'(i)$, that is, with its neighbors in G'. A mathematical formulation of the static version of the problem (SMOP), where nodes cannot enter or exit the network, is as follows.

Two sets of decision variables are used: $\{x_{ij}\}$ and $\{y_{ij}\}$, $(i, j) \in E$. The decision variables x_{ij} specify the bandwidth allocated to the connection between peers i and j. Therefore they are continuous variables $0 \leq x_{ij} \leq u_{ij}$, which will be further constrained when they are not 0 to be at least l_{ij}. Decision variables y_{ij} are binary variables which are 1 if arc (i, j) is used for a connection, 0 otherwise.

The formulation is the following:

$$z_{\mathrm{SMOP}} = \max \sum_{(i,j)\in E} p_{ij} x_{ij} \tag{38}$$

$$\text{s.t.} \quad \sum_{j\in\delta(i)} x_{ij} \leq b_i, \quad i \in V \tag{39}$$

$$\sum_{i\in S_h} \sum_{j\in V\setminus S_h} y_{ij} \geq 1, \quad \forall S_h \subset 2^V \tag{40}$$

$$l_{ij} y_{ij} \leq x_{ij} \leq u_{ij} y_{ij}, \quad (i, j) \in E \tag{41}$$

$$y_{ij} \in \{0, 1\}, \quad (i, j) \in E \tag{42}$$

where $p_{ij} = p_i \times p_j$, for each edge $(i, j) \in E$, and $\delta(i)$ represents the neighborhood of i in G (e.g., $\delta(i) = V \setminus \{i\}$ if graph G is complete). Constraints 40 enforce connectivity, but they have to be dropped in dynamic settings, such as MOP's, since the probability of survival of any path is small. Connectivity emerges as a property of networks having average node outdegree sufficiently greater than 1. Problem SMOP is NP-hard even without constraints 40. An effective LP-relaxation of problem P can be obtained by replacing constraints 42 with constraints in the form $0 \leq y_{ij} \leq 1$, for each $(i, j) \in E$.

The literature on MOP is currently limited to Maniezzo et al. (2005) and Boschetti et al. (2006), apart from Maniezzo et al. (2004) were a preliminary version of the problem was reported.

3.3 Multi-mode project scheduling

The Multi-Mode Project Scheduling Problem (MPSP) asks to determine the starting times of the activities of a project with the objective of minimizing the total project duration (*makespan*). The activities cannot be interrupted once put in progress, and are subject to constraints that impose a partial precedence relation among them and a limit to their resource usage. Moreover, the activities can be executed in different *modes*, involving different durations and resource requirements. This problem is NP-hard being a generalization of the Resource-Constrained Project Scheduling Problem (RCPSP), which is also NP-hard.

The MPSP problem can be formulated as follows (Maniezzo and Mingozzi 1999). A set $X = \{1, \dots, n\}$ of activities (jobs) and two sets RR and NR of resources are given. The set RR is the set of *renewable resources*, resources whose quantities are renewed from period to period, while NR is the set of *nonrenewable resources*, i.e., the set of resources which have a limited overall availability over all project duration.

In each period of the planning horizon $t, t = 1, \dots, T_{\max}$, K_r units of renewable resource r are available, $r \in \mathrm{RR}$, while there is a global availability of W_s units of nonrenewable resource $s, s \in \mathrm{NR}$. We assume that time period t corresponds to the time interval $[t, t+1)$.

Every activity i is associated with a set M_i of possible modes it can be executed in. When activity i is executed in mode m, it has a processing time (duration) d_{im} and it requires a constant amount k_{imr} of renewable resource r, during each time interval of its execution, and an amount w_{ims} of nonrenewable resource s. All quantities, d_{im}, k_{imr}, w_{ims}, K_r and W_s, are assumed to be non-negative integers, no pre-emption is allowed and setup times are included in the processing times.

With each activity j is associated a set $\delta(j)^{-1} \subseteq X \backslash \{j\}$ of immediate predecessors. The precedence constraints can be represented by an acyclic digraph $G = (X, H)$ where $H = \{(i, j) : i \in \delta(j)^{-1}, j \in X\}$. We assume, without loss of generality, that activities are topologically ordered. Activities 1 and n are used to represent the beginning and the end of the whole project: activity 1 must be completed before starting any other activity and activity n can start after the completion of all other activities. Let $X' = X\backslash\{1, n\}$ and $n' = |X'|$. Both activities 1 and n have only one execution mode, no duration and no resource consumption.

The objective is to assign a mode to each activity and to find a schedule that minimizes the project completion time (project makespan), satisfying precedence and resource constraints.

An obvious upper bound $T_{\max}$ to the project makespan is given by the sum of the maximum possible duration of each activity. A time window $[es_i, ls_i]$ of earliest and latest start times for each activity $i \in X$ can be computed by performing a forward and backward recursion on the graph G, by setting $es_1 = 0$ and $ls_n = T_{\max}$ (see Elmaghraby 1977).

A (0–1) integer programming formulation of the MPSP can be obtained as follows.

Let $B_i = \{(i, m) : m \in M_i\}$ and $B = \bigcup_{i \in X'} B_i$. For a given subset $F \subseteq B$, we denote with $X(F) = \{i : (i, m) \in F\}$ the different activities of F. A subset F is called a *feasible subset* if the activities in F satisfy precedence and resource constraints and if $|X(F)| = |F|$. Any feasible MPSP solution of cost t^* can be represented by a sequence $S = (F_{\ell_1}, F_{\ell_2}, \ldots, F_{\ell_{t^*}})$ where each F_{ℓ_t} of S represents the feasible subset of activities in progress at time t. Let $F = \{1, 2, \ldots, n_f\}$ be the index set of all feasible subsets of B, and let $F_{im} \in F$ be the index set of all feasible subsets containing the pair $(i, m) \in B$.

Let $\zeta_{\ell t}$ be a (0–1) variable equal to 1 if and only if all activities of the feasible subset F_ℓ are being executed at time period t, and let x_{it} be a (0–1) variable equal to 1 if and only if activity i terminates its execution at time period t and y_{im} be a (0–1) binary variable equal to 1 if and only if job i is executed in mode m.

The mathematical formulation of MPSP becomes:

$$z_{\mathrm{MPSP}} = \min \sum_{t=es_n}^{ls_n} t x_{nt} \tag{43}$$

$$\text{s.t.} \quad \sum_{\ell \in F_{im}} \sum_{t=es_i}^{ls_i} \zeta_{\ell t} = d_{im} y_{im}, \quad m \in M_i, i \in X' \tag{44}$$

$$\sum_{\ell \in F} \zeta_{\ell t} \leq 1, \quad t = 1, \ldots, T_{\max} \tag{45}$$

$$\sum_{m \in M_i} y_{im} = 1, \quad i \in X \tag{46}$$

$$x_{it} \geq \sum_{\ell \in F_i} \zeta_{\ell t} - \sum_{\ell \in F_i} \zeta_{\ell t+1}, \quad t = es_i, \dots, ls_i, i \in X' \tag{47}$$

$$\sum_{t=es_j}^{ls_j} t x_{jt} - \sum_{t=es_i}^{ls_i} t x_{it} \geq \sum_{m \in M_j} d_{jm} y_{jm}, \quad (i, j) \in H \tag{48}$$

$$\sum_{i \in X} \sum_{m \in M_i} w_{ims} y_{im} \leq W_s, \quad s \in \text{NR} \tag{49}$$

$$x_{it} \in \{0, 1\}, \quad i \in X, t = 1, \dots, T_{\max} \tag{50}$$

$$\zeta_{\ell t} \in \{0, 1\}, \quad \ell \in F, t = 1, \dots, T_{\max} \tag{51}$$

$$y_{im} \in \{0, 1\}, \quad m \in M_i, i \in X \tag{52}$$

Constraints 44 force the solution to have feasible subsets contained in F_{im} in progress for exactly d_{im} time periods, if activity i is executed in mode m. Constraints 45 ensure that at each time period t at most one feasible subset is in progress. Constraints 46 force each activity to be executed in only one mode. Constraints 47 ensure that the execution of activity i terminates at time period t if the activity i is contained in the feasible subset in execution at time period t but is not contained in the feasible subset in execution at time period $t + 1$. Finally, constraints 48 and 49 are the precedence and the nonrenewable resource constraints, respectively.

Note that, since a feasible subset F_ℓ is executed for a number of time periods equal to $\sum_t \zeta_{\ell t}$, the objective function 43 can be rewritten as $z_{\text{MPSP}} = \min \sum_{\ell \in F} \sum_{t=1}^{T_{\max}} \zeta_{\ell t}$.

Several exact and heuristic techniques have been proposed for the MPSP. Talbot (1982) was the first to propose an exact enumeration scheme. These early methods were able to solve instances up to 15 activities. Sprecher and Drexl (1998) proposed new dominance criteria, which enabled their branch and bound algorithm to solve problems up to 20 activities.

With reference to heuristic approaches, Talbot (1982) and Sprecher and Drexl (1998) proposed the truncated versions of their tree searches. Different heuristic strategies have also been investigated, such as biased random sampling (Drexl and Grunewald 1993), local search (Kolisch and Drexl 1994) and constraint-based approaches (Ulusoy and Ozdamar 1994).

3.3.1 Reformulations and lower bounds

The reported results are not directly derived from formulation MPSP, but on a more manageable one (see Maniezzo and Mingozzi 1999). First of all, a lower bound, called BND3, can be obtained from MPSP by removing precedence and non-preemption constraints 45, 47 and 48, defining $h_\ell = \sum_{t=1}^{T_{\max}} \zeta_{\ell t}$ to be the total execution time of feasible subset ℓ, and, finally, by relaxing equations 44 by replacing "=" with "$\geq$". Let $M \subseteq F$ be the set of *maximal feasible subsets*, i.e., subsets which are not contained in any subset of F. We denote with $M_{im} \subseteq M$ the index set of all maximal feasible subsets containing job i executed in mode m. The relaxed problem RP2 is the following:

$$z_{\mathrm{RP2}} = \min \sum_{\ell \in M} h_\ell \tag{53}$$

$$\text{s.t.} \quad \sum_{\ell \in M_{im}} h_\ell \geq d_{im} y_{im}, \quad m \in M_i, i \in X' \tag{54}$$

$$\sum_{m \in M_i} y_{im} = 1, \quad i \in X \tag{55}$$

$$\sum_{i \in X} \sum_{m \in M_i} w_{ims} y_{im} \leq W_s, \quad s \in \mathrm{NR} \tag{56}$$

$$h_\ell \geq 0, \quad \ell \in M \tag{57}$$

$$y_{im} \in \{0, 1\}, \quad m \in M_i, i \in X \tag{58}$$

The cost of the optimal solution of the LP-relaxation of RP2 gives lower bound BND3.

4 Solving problems with Lagrangean metaheuristics

Having introduced the basic techniques and the problem they have been applied to, we move on describing how the basic pseudocode for the Lagrangean metaheuristic can be specialized for each problem. Specifically, we first present a very straightforward application of LagrHeuristic to the SCFLP. This is not enough to produce edge-level results, but it shows that already by means of a simple code it is possible to get good results. A more sophisticated algorithm will be later described for the MOP.

4.1 Single capacitated facility location

The steps of LagrHeuristic have been specified as follows, for the case of the Single Capacitated Facility Location problem.

Step 1: identify an "easy" subproblem LR. The relaxation of assignment constraints 34 in problem SCFLP yields the following problem.

$$z_{\mathrm{LR}}(\boldsymbol{\lambda}) = \min \sum_{i \in I, j \in J} (c_{ij} - \lambda_j) x_{ij} + \sum_{i \in I} f_i y_i + \sum_{j \in J} \lambda_j \tag{59}$$

$$\text{s.t.} \quad \sum_{j \in J} q_j x_{ij} \leq Q_i y_i, \quad i \in I \tag{35}$$

$$x_{ij} \in \{0, 1\}, \quad i \in I, j \in J \tag{36}$$

$$y_i \in \{0, 1\}, \quad i \in I \tag{37}$$

where λ_j, $j \in J$, is an unrestricted penalty.

Step 3: solve subproblem LR. Problem LR decomposes naturally into $|I|$ knapsack problems, with objective function $\sum_{i\in I}(\sum_{j\in J}(c_{ij}-\lambda_j)x_{ij}+f_i y_i)$. Thus, for each $i\in I$ for which $\sum_{j\in J}(c_{ij}-\lambda_j)x_{ij}<-f_i$, the corresponding y_i is set to 1, it is set to 0 otherwise.

Step 4: check for unsatisfied constraints. The solution of LR can have customers assigned to multiple or to no location. This can be ascertained by direct inspection.

Step 5: update penalties $\boldsymbol{\lambda}$. We used a standard subgradient algorithm for updating penalties, see Pirkul (1987) for details.

Step 6: construct problem solution using $\mathbf{x}$ *and* $\boldsymbol{\lambda}$. Let $\bar{I}$ be the set of locations chosen in the solution obtained at Step 3. The SCFLP is transformed into a Generalized Assignment Problem (GAP) as follows:

$$z_{\mathrm{GAP}} = \min \sum_{i\in\bar{I},j\in J} c_{ij}x_{ij} \tag{60}$$

$$\text{s.t.} \quad \sum_{i\in\bar{I}} x_{ij} = 1, \quad j\in J \tag{61}$$

$$\sum_{j\in J} q_j x_{ij} \le Q_i, \quad i\in\bar{I} \tag{62}$$

$$x_{ij}\in\{0,1\}, \quad i\in\bar{I}, j\in J \tag{63}$$

This is still an NP-hard problem, but efficient codes exist to solve it, which we did once per Lagrangean iteration (see the following computational results section for further details).

Notice that for some iterations Step 3 may provide a set of locations $\bar{I}$ for which GAP is unfeasible. In this case no feasible SCFLP solution is generated and LagrHeuristic goes on.

4.1.1 SCFLP, computational results

Computational testing of the above described algorithm was carried out after implementing it in C# and Fortran, where the latter was used by linking algorithms MT1R for solving knapsack problems and MTHG for getting a heuristic solution of GAP problems (Martello and Toth 1990).[1] The code was run on a 1.7 GHz laptop with 1 Gb of RAM and .NET framework 2.0.

The benchmark instances are those used by Holmberg et al. (1999), they consist of 71 instances whose size ranges from 50 to 200 customers and from 10 to 30 candidate facility locations. The instances are divided into 4 subsets. Set 1 has customers and locations with coordinates randomly generated in (10, 200), problems p1 to p12 have 50 customers and 10 possible locations, problems p13 to p24 have 50 customers and 20 possible locations. Set 2 has locations generated in (10, 300). The assignment costs are based on a vehicle routing problem cost distribution (see Holmberg et al. 1999 for details). Set 3 is based on vehicle routing test problems used by Solomon

[1]Codes are freely available at the page http://www.or.deis.unibo.it/knapsack.html.

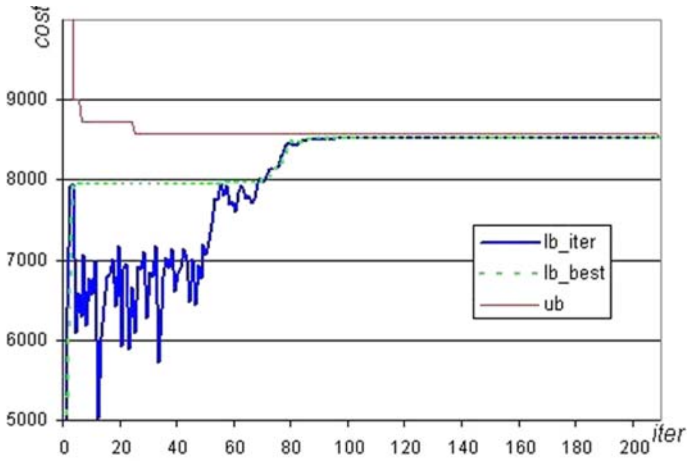

Fig. 1 SCFLP, upper and lower bound evolution

Table 1 SCFLP, computational results, whole testset

Probl.	%ΔLP	%ΔHL	%ΔLL	t_{lagr}	t_{tot}	%Δdfs	t_{dfs}
1–24 avg	20.14	0.01	0.13	0.09	0.66	0.00	0.54
1–24 max	31.01	0.06	0.66	0.31	1.73	0.00	1.85
25–40 avg	7.41	1.47	1.22	0.11	2.42	0.13	12.67
25–40 max	12.82	14.30	8.47	0.30	4.62	0.79	34.08
41–55 avg	24.19	0.38	0.31	0.14	0.71	0.03	1.62
41–55 max	51.81	2.02	1.86	0.46	1.39	0.18	5.47
56–71 avg	27.08	1.13	0.65	1.02	4.18	0.02	15.97
56–71 max	38.81	13.91	3.48	4.99	8.02	0.14	46.60

(1987), while set 4 is generated as set 1 but the number of potential locations is 30 and the number of customers is 200.

When running LagrHeuristic we let both an upper bound z_{ub} and a lower bound z_{lb} evolve. The algorithm terminated either when an optimal solution was found, i.e. when $z_{lb} = z_{ub}$, or when 1500 subgradient iterations were made. The initial *alpha* subgradient step control parameter (Polyak 1969) was set to 1.5 and multiplied by 0.9 when 5 consecutive non-improving iterations were detected.

Figure 1 shows the evolution of upper and lower bounds on one run (problem p52), which was solved to optimality. We mention here again how the inclusion of dual information into the metaheuristic permits to determine the quality of the best solution found, and possibly of its optimality.

Complete computational results are reported in Tables 1 and 2. The results of LagrHeuristic are compared against those obtained by the best so far metaheuristic for SCFLP, which to the best of our knowledge is the VLSN by Ahuja et al. (2003).

Table 2 SCFLP, computational results, set 3

Probl.	opt	LP	HL	LL	t_{lagr}	t_{tot}	%ΔLP	%ΔHL	%ΔLL	dfs	%Δdfs	t_{dfs}
p25	11630	10732	11649	11570	0.04	1.79	7.72	0.16	0.52	11630	0.00	9.63
p26	10771	10022	10773	10722	0.05	1.76	6.95	0.02	0.45	10771	0.00	5.08
p27	12322	11348	12373	12199	0.06	1.80	7.90	0.41	1.00	12322	0.00	12.75
p28	13722	12654	13916	13592	0.27	3.49	7.78	1.41	0.95	13727	0.04	13.60
p29	12371	11721	12544	12317	0.30	2 42	5.25	1.40	0.44	12379	0.07	30.09
p30	11331	10860	11408	11068	0.18	2.15	4.16	0.68	2.32	11392	0.54	23.54
p31	13331	12840	13608	13111	0.01	4.62	3.68	2.08	1.65	13436	0.79	30.01
p32	15331	14820	15808	15078	0.16	2.29	3.33	3.11	1.65	15436	0.69	34.08
p33	11629	10775	11629	11542	0.06	1.87	7.34	0.00	0.75	11629	0.00	11.70
p34	10632	10117	10632	10631	0.04	1.96	4.84	0.00	0.01	10632	0.00	6.30
p35	12232	11602	13981	11196	0.01	2.90	5.15	14.30	8.47	12232	0.00	10.31
p36	13832	13087	13832	13651	0.27	4.10	5.39	0.00	1.31	13832	0.00	10.50
p37	11258	9828	11258	11258	0.10	1.56	12.70	0.00	0.00	11258	0.00	1.52
p38	10551	9374	10551	10551	0.06	1.55	11.16	0.00	0.00	10551	0.00	1.54
p39	11824	10364	11824	11824	0.02	1.51	12.35	0.00	0.00	11824	0.00	0.20
p40	13024	11354	13024	13024	0.09	2.94	12.82	0.00	0.00	13024	0.00	1.79

Notice however that the CPU times for these last results have been obtained on a PC with a Athlon/1200 Mhz processor and 512 Mb RAM, under the RedHat Linux 7.1.

The columns show: probl identifier of a problem or of a problem set, LP result of LP-relaxation, HL best result of LagrHeuristic, LL best found lower bound, %ΔLP percentage distance from optimality of the LP-relaxation value, %ΔHL percentage distance from optimality of LagrHeuristic result, %ΔLL percentage distance from optimality of the lower bound, t_{lagr} CPU time in seconds to find the best LagrHeuristic result, t_{tot} CPU time in seconds used by LagrHeuristic before terminating. Concerning the results obtained by the "dfs" variant of the VLSN heuristic proposed by Ahuja et al. (2003), the columns show: %Δdfs percentage distance from optimality of dfs, t_{dfs} CPU time in seconds taken by dfs.

As repeatedly pointed out, the results we report here are not for showing that we have the best heuristic in the literature, but for showing that even a straightforward implementation of algorithm LagrHeuristic can get very close to the state of the art. This is already apparent on Table 1, and it is made evident in Table 2, where it appears that the big difference in set 2 (and analogously in set 4) is mainly due to one single instance. It would be rather easy to close that gap by means of some trick on the subgradient algorithm, such as an alpha restart or an adaptive anneal (not to mention a local search on the upper bound), but again, this would obfuscate our point.

One last remark is in order: out of the 71 instances, 3 could be solved to optimality by the subgradient alone, which evolved weights that lead to the satisfaction also of the relaxed constraints, while 16 other ones were solved to proved optimality since the lower and the upper bound converged to the same cost. In all these cases the computation terminated before the maximum available CPU time, an option which is not available for primal-only heuristics.

4.2 Membership overlay

In actual practice of P2P networks it can be observed that nodes continuously enter and exit the network, some nodes spending more time in the network while others join only for a short time. The problem formulation is not affected by dynamicity, in the sense that at any moment in time the problem formulation is as described in formulae 38–42. The only effect is that the graph G and all related elements are time-varying.

Whenever the average uptime of a node in the network is higher than the optimization time, it becomes feasible to re-optimize the network, taking into account the new network conditions. This could be done periodically or whenever significant network topology changes are detected. All algorithms have been thus formulated in order to work in a local setting, where processes are run asynchronously on each network node and no global data structure is maintained.

Problem-specific step details are as follows.

Step 1: identify an "easy" subproblem LR. This relaxation is obtained by associating non negative penalty λ_i to each constraint 39 and by removing constraints 40. The resulting relaxed problem is as follows:

$$z_{\text{LR}}(\boldsymbol{\lambda}) = \max \sum_{(i,j)\in E} p'_{ij}x_{ij} + \sum_{i\in V} b_i\lambda_i \tag{64}$$

$$\text{s.t.} \quad l_{ij}y_{ij} \le x_{ij} \le u_{ij}y_{ij}, \quad (i,j)\in E \tag{65}$$

$$y_{ij} \in \{0,1\}, \quad (i,j)\in E \tag{66}$$

where $p'_{ij} = p_{ij} - \lambda_i - \lambda_j$. It is straightforward to see that in an optimal $z_{\text{LR}}(\boldsymbol{\lambda})$ solution, for each edge $(i,j)\in E$, $x_{ij} = u_{ij}$, if $p'_{ij} \ge 0$, and $x_{ij} = 0$, if $p'_{ij} < 0$. Therefore, variables y_{ij} are unnecessary and the mathematical formulation 64–66 is equivalent to problem:

$$z_{\text{LR}}(\boldsymbol{\lambda}) = \max \sum_{(i,j)\in E} p'_{ij}x_{ij} + \sum_{i\in V} b_i\lambda_i \tag{67}$$

$$\text{s.t.} \quad 0 \le x_{ij} \le u_{ij}, \quad (i,j)\in E \tag{68}$$

Notice that penalties are associated to nodes and that each node $i\in V$ can compute its contribution to the global cost if it knows its penalty λ_i and the penalties λ_j of its direct neighbors $j\in\delta(i)$. The problem thus decomposes into n subproblems LR$_i$, $i\in V$, defined as follows:

$$z_{\text{LR}_i} = \max \frac{1}{2}\sum_{j\in\delta(i)} p'_{ij}x_{ij} + b_i\lambda_i \tag{69}$$

$$\text{s.t.} \quad 0 \le x_{ij} \le u_{ij}, \; j\in\delta(i) \tag{70}$$

Step 3: solve subproblem LR. The optimal LR solution $(\mathbf{x},\mathbf{y})$ of value $z_{\text{LR}}(\boldsymbol{\lambda}) = \sum_{i\in V} z_{\text{LR}_i}$ is computed by each node i according the following observations:

- If $p'_{ij} \ge 0$ it uses as much bandwidth as possible, i.e. $y_{ij} = 1$ and $x_{ij} = u_{ij}$;

- If $p'_{ij} < 0$ it does not use the connection, i.e. $y_{ij} = 0$ and $x_{ij} = 0$.

The above steps lead to an agreement on the values to assign to x_{ij} and y_{ij}, even when they are independently executed by two neighboring nodes i and j.

Step 4: check for unsatisfied constraints. The solution of the subproblems could be infeasible for the MOP because some nodes could globally allocate more bandwidth than they have available or less than the lower limit to a single connection. This is straightforward to check by direct inspection.

Step 5: update penalties $\boldsymbol{\lambda}$. In order to find the values of $\boldsymbol{\lambda}$ that minimize the upper bound $z_{\mathrm{LR}}(\boldsymbol{\lambda})$ we solved the Lagrangean Dual $\min\{z_{\mathrm{LR}}(\boldsymbol{\lambda}) : \boldsymbol{\lambda} \geq \mathbf{0}\}$ by means of a subgradient algorithm. Special attention had to be put on the fact that the usual step-defining formula has the norm of all infeasibilities at the denominator. We modified that obtaining a local step-defining equation, which preserves the property of guaranteeing asymptotic optimality (see Boschetti et al. 2006 for details). The optimal solution of the Lagrangean Dual is equivalent to the optimal solution of the LP relaxation of problem MOP when connectivity constraints are removed.

Step 6: construct problem solution using **x** *and* $\boldsymbol{\lambda}$. The computation of a feasible global solution is based on two interaction moments: each node i iteratively polls its neighbors j, $j \in \delta(i)$, for their current λ_j values. Having these, a local subgradient algorithm can update all its local variables and compute its contribution to the upper bound and to the corresponding solution. The bound solution can be infeasible, thus a negotiation phase is needed so that each node i can agree with each neighbor j the bandwidth, if any, to allocate to connection (i, j) to get a globally feasible solution.

Let $z^*_{\mathrm{LR}_i}$ be the solution obtained by the subgradient optimization of problem LR_i using penalties λ^*_j, $j \in \delta(i) \cup \{i\}$. The solution could be infeasible because of the relaxed constraints. A heuristic solution is obtained by considering the optimized costs $p^*_{ij} = (p_{ij} - \lambda^*_i - \lambda^*_j)$, ranking all arcs $(i, j) \in E$ by non increasing p^*_{ij} values and allocating all possible bandwidth to each successively considered connection. More in detail, the algorithm is the following.

```
HeuMOP($\mathbf{p}^*$)
1  Order all arcs $(i, j) \in E$ by decreasing $p^*_{ij}$
2  Initialize $s_i = b_i$ for each $i \in V$
3  for each arc $(i, j)$ in $E$ in nonincreasing $p^*_{ij}$ order
4      do slack $= \min\{s_i, s_j\}$
5          if slack $\geq l_{ij}$
6              then $x_{ij} =$ slack
7                   $y_{ij} = 1$
8                   $s_i = s_i -$ slack; $s_j = s_j -$ slack
```

This approach is derived from the exact method for solving continuous knapsack problems (Martello and Toth 1990). In our case it is not guaranteed to be optimal, but it consistently produces good quality solutions in time $O(m \log m)$, where m is the number of arcs, the highest cost operation being the ordering of the arcs.

HeuMOP is reported for a centralized environment. In P2P networks, the negotiation process to allocate bandwidth is a local process where each node has only knowledge of the neighborhood data. Therefore, in a distributed and dynamic environment

HeuMOP must be split in separate threads, one for each node, where network changes are handled programmatically as events and local data are exchanged between them (see Boschetti et al. 2006 for a detailed description).

4.2.1 MOP, computational results

Testing of the above procedures is still ongoing. We implemented and tested it on a desktop, then moved on to distributed environments. On a desktop, the algorithms were coded partly in C# and partly in C++ and run on a 1000 MHz Pentium III machine, under Windows XP. Distributed and dynamic tests are being performed both in a network simulation environment (Peersim 2006) and on the actual Internet, by means of Planetlab (2006). We have been conducting a number of experiments on different scenarios. For simulations, we generated two sets of instance graphs (Maniezzo et al. 2005); the first one (set A) has parameters which match those measured on real P2P networks as reported in Saroiu et al. (2002), the second set (set B) has the nodes randomly generated on a x-y plane and p_{ij} inversely proportional to the Euclidean distance of nodes i and j, $\forall i, j \in V$, in order to produce meaningful visualizations, besides modeling location-aware networks.

The computational testing was carried out separately for the static and the dynamic case. In the static case we wanted to determine the solution quality disruption, w.r.t. upper bound z_{P}, due to the successive heuristics, while in the dynamic case the ease of adaptation to a mutated environment was of interest. Table 3 summarizes the results obtained (see also Maniezzo et al. 2005). The columns show:

- Id: an instance identifier;
- n: number of nodes;
- z^*_{LR}: cost of best solution found for problem LR;
- $\%z^*_{\mathrm{LH}}$: percentage gap for solution of algorithm LagrHeuristic;
- t^*_{LH}: time to find the solution of algorithm LagrHeuristic.

Table 3 Results on different MOP instances

Problem			LagrHeuristic					
			10 iter		20 iter		30 iter	
Id	n	z^*_{LR}	$\%z^*_{\mathrm{HL}}$	t^*_{HL}	$\%z^*_{\mathrm{HL}}$	t^*_{HL}	$\%z^*_{\mathrm{HL}}$	t^*_{HL}
A20	20	842	0.96	0.00	0.59	0.04	0.27	0.05
A50	50	2137	0.88	0.09	0.52	0.14	0.36	0.17
A100	100	8162	0.83	0.21	0.67	0.33	0.24	0.43
A500	500	20192	0.97	4.08	0.66	7.28	0.23	10.29
A1000	1000	53765	0.91	19.42	0.58	34.15	0.30	49.81
B20	20	53295	4.50	0.16	4.08	0.24	3.74	0.30
B50	50	131891	4.38	0.16	4.02	0.27	3.69	0.36
B100	100	281836	4.50	0.19	4.06	0.33	3.93	0.43
B500	500	1413301	4.63	3.74	4.19	7.52	3.99	9.91
B1000	1000	2940334	4.39	19.29	4.01	37.10	3.88	56.72

The last two elements are reported for a number of subgradient optimization iterations which is 10, 20 and 30, respectively.

The results show how the proposed problem relaxation is highly effective on both, structurally very different, instance sets. The gap between the upper bound z^*_{LR} and the lower bound z^*_{HL} is always reasonably small. No alternative approaches so far have been proposed for this problem, to further validate the results.

The validation of the effectiveness of the distributed algorithm in a dynamic setting has been so far completed only on the PeerSim simulation environment (2006). This is a java-based environment which has already been used for evaluating other P2P protocols; it takes care of the multithreading on the nodes and of managing all message-passing, leaving to the user the task of defining the code to be run at each node. We linked our code and tested the system under different operational conditions of problems of type A, the only type permitted by the simulator. We used two settings: one which allows the peers that are much more powerful than others ("*superpeers*" in the literature) to have high u_{ij} on outgoing connections (set A1), while the second settings permits limited u_{ij} on all connections, independently of the power of the peer (set A2).

Unfortunately the simulator did not prove able to efficiently scale up to more than a few thousands nodes, however the results obtained over 1000 nodes are already significant. Figure 2 shows the evolution of the global bandwidth allocated by the whole network for a problem A1 (top) and A2 (bottom). Initially there was no allocated bandwidth and both cases, after a short startup, there is a fast convergence to a stable level. This was already expected from the results of Table 3, as testified by the solution quality obtained after different numbers of internal iterations. There are wide oscillations left around the stable level due to the great dynamicity of the settings, where a high percentage of nodes continuously enter and exit the network (the average uptime being in both cases equal to time needed for performing 30 subgradient iterations). Oscillations are wider on the left graphic as dynamicity on superpeers has a greater impact on global throughput. Figure 3 shows a visual interface for two small instances of type A1 (top) and A2 (bottom). The radius of a node i is proportional to its bandwidth b_i, the thickness of an edge (i, j) is proportional to the bandwidth x_{ij} allocated for that connection. Notice on the left how a backbone among super peers emerges, and small peers are connected almost only to one or two superpeers. On the contrary, on the right, when connections are anyway constrained, there is a much higher connectivity and no backbone. This is coherent to actual P2P networks structures.

5 Solving problems with Benders metaheuristics

As it was the case for Lagrangean decomposition, we now present the application of BENDHEURISTIC to two problems: the SCFLP as a simple application and the MPSP as a more research-oriented work.

5.1 Single capacitated facility location

Step 1: identify a master MP and an "easy" subproblem SP. A possible Benders decomposition of SCFLP involves keeping in the master the decision of which facilities

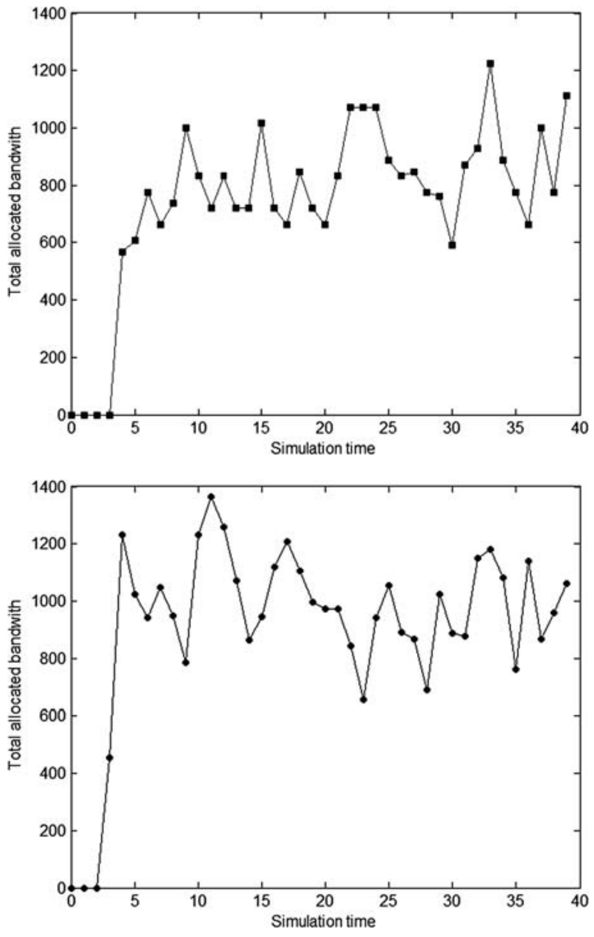

Fig. 2 Global throughput, A1 instance (top) and A2 instance (bottom)

to open, and assigning clients to open facilities as a subproblem. The subproblem is therefore a GAP again.

More in detail, the master problem is:

$$z_{\mathrm{MP}} = \min \sum_{i\in I} f_i y_i + z_{\mathrm{SP}}(\mathbf{y}) \tag{33}$$

$$\text{s.t.} \quad y_i \in \{0,1\}, \quad i \in I \tag{37}$$

and the subproblem becomes:

$$z_{\mathrm{SP}}(\mathbf{y}) = \min \sum_{i\in I, j\in J} c_{ij} x_{ij} \tag{71}$$

$$\text{s.t.} \quad \sum_{i\in I} x_{ij} = 1, \quad j \in J \tag{34}$$

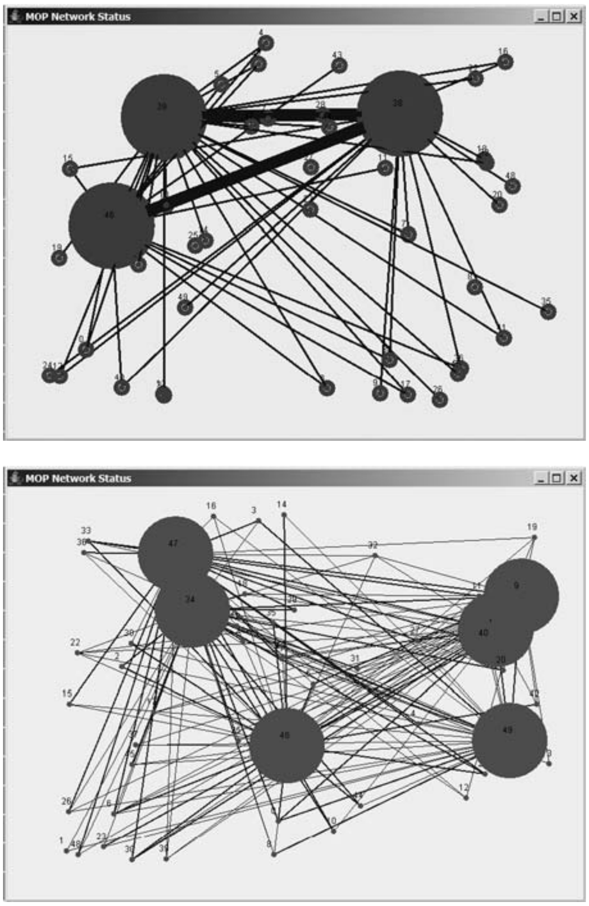

Fig. 3 Connectivity, A1 instance (top) and A2 instance (bottom)

$$\sum_{j \in J} q_j x_{ij} \leq Q_i y_i, \quad i \in I \tag{35}$$

$$x_{ij} \in \{0, 1\}, \quad i \in I, j \in J \tag{36}$$

Step 3: solve master problem MP. As the master problem, even though NP-hard after the addition of Bender's cuts, was relatively easy to solve, we solved it to optimality at each iteration.

Step 5: solve subproblem SP. The x_{ij} are required to be integer, but the subproblem is the same GAP we met in Sect. 4.1. The same considerations apply.

Step 6: solve problem DP to add cuts to MP. To get the subproblem's dual we relaxed constraints 36 into $x_{ij} \geq 0$, $i \in I$, $j \in J$. After associating dual variables w'_j, $j \in J$, to constraints 34 and w''_i, $i \in I$, to constraints 35, problem DP becomes:

$$z_{\mathrm{DP}}(\mathbf{y}) = \max \sum_{j\in J} w'_j + \sum_{i\in I} Q_i y_i w''_i \tag{72}$$

$$\text{s.t.} \quad w'_j + q_j w''_i \le c_{ij}, \quad i \in I, j \in J \tag{73}$$

$$w''_i \le 0, \quad i \in I \tag{74}$$

therefore yielding the following master formulation, which includes the added cut.

$$z_{\mathrm{MP}} = \min \ z$$

$$\text{s.t.} \quad z \ge \sum_{i\in I}(f_i + Q_i w''_i) y_i + \sum_{j\in J} w'_j \tag{75}$$

$$y_i \in \{0,1\}, \quad i \in I \tag{37}$$

5.1.1 SCFLP, computational results

We implemented the BendHeuristic for SCFLP and run it under the same setting as the LagrHeuristic reported in Sect. 4.1.1. The results show a lower competitiveness of the basic BendHeuristic, when compared to its Lagrangean counterpart. However, the general working is as expected, as we witness a convergence of upper and lower bounds. Figure 4 shows a trace in the case of instance p50 of test set 3, one of the instances where BendHeuristic is particularly effective.

Global results are shown in Table 4. The columns show: probl identifier of a problem or of a problem set, LP result of LP-relaxation, HB best result of BendHeuristic, BL best found lower bound, %ΔLP percentage distance from optimality of the LP-relaxation value, %ΔHB percentage distance from optimality of BendHeuristic result, %ΔBL percentage distance from optimality of the lower bound, t_{Bend} CPU time in seconds to find the best BendHeuristic result, t_{tot} CPU time in seconds used

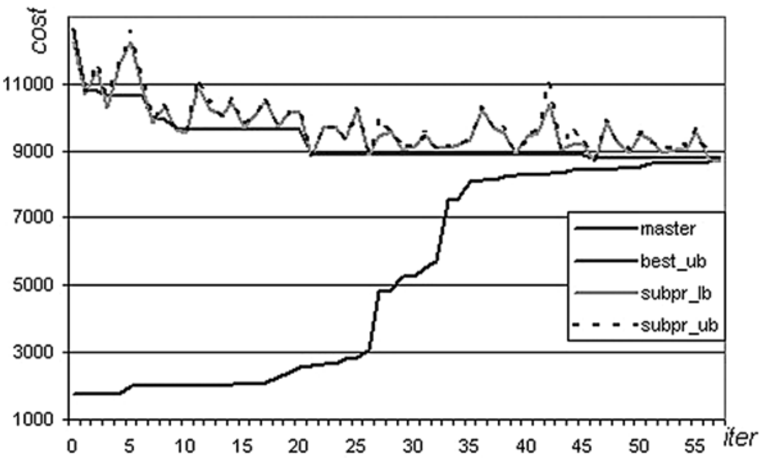

Fig. 4 BendHeuristic, upper and lower bound evolution on instance p50

Table 4 Computational results, whole testset

Probl.	%ΔLP	%ΔHB	%ΔLB	t_{Bend}	t_{tot}	%Δdfs	t_{dfs}
1–24 avg	20.14	0.67	4.74	408.92	1802.44	0.00	0.54
1–24 max	31.01	3.73	11.68	2224.76	5048.00	0.00	1.85
25–40 avg	7.41	0.62	0.48	112.50	202.00	0.13	12.67
25–40 max	12.82	3.21	5.18	715.44	1418.68	0.79	34.08
41–55 avg	24.19	1.54	14.20	189.73	518.23	0.03	1.62
41–55 max	51.81	5.41	70.60	1532.17	2538.99	0.18	5.47
56–71 avg	27.08	9.74	58.37	229.62	331.73	0.02	15.97
56–71 max	38.81	25.71	80.52	1805.79	1985.05	0.14	46.60

Table 5 Computational results, set 4

Probl.	opt	LP	HB	BL	t_{Bend}	t_{tot}	%ΔLP	%ΔHB	%ΔBL	dfs	%Δdfs	t_{dfs}
p56	21103	16477	22052	9455	31.02	267.89	21.92	4.50	55.20	21120	0.08	15.77
p57	26039	21034	27287	14430	19.57	117.69	19.22	4.79	44.58	26075	0.14	26.32
p58	37239	31667	38037	25209	5.41	42.46	14.96	2.14	32.30	37240	0.00	46.60
p59	27282	23356	29981	19678	15.60	77.45	14.39	9.89	27.87	27282	0.00	29.97
p60	20534	13946	23322	5500	6.17	14.16	32.08	13.58	73.22	20534	0.00	3.27
p61	24454	16984	27935	8800	5.87	14.40	30.55	14.23	64.01	24454	0.00	6.27
p62	32643	24072	34649	16500	6.43	16.64	26.26	6.15	49.45	32648	0.02	28.68
p63	25105	18634	26110	10313	1.33	115.50	25.78	4.00	58.92	25108	0.01	15.22
p64	20530	12680	25809	4000	10.50	13.34	38.24	25.71	80.52	20530	0.00	0.18
p65	24445	14958	29132	6400	5.46	13.73	38.81	19.17	73.82	24445	0.00	0.18
p66	31415	20275	34331	12000	10.39	12.84	35.46	9.28	61.80	31429	0.04	8.14
p67	24848	18826	25522	10513	1.30	125.45	24.24	2.71	57.69	24849	0.01	10.34
p68	20538	14181	23507	5376	1305.88	1438.16	30.95	14.46	73.82	20538	0.00	3.32
p69	24532	17304	26413	8000	1805.79	1985.05	29.46	7.67	67.39	24532	0.00	6.15
p70	32321	24465	33373	15000	198.78	734.43	24.31	3.25	53.59	32323	0.01	27.04
p71	25540	18719	29198	10265	244.39	318.49	26.71	14.32	59.81	25540	0.00	28.10

by BendHeuristic before terminating. Furthermore, the columns show the results obtained by the dfs variant of the VLSN heuristic Ahuja et al. (2003): %Δdfs percentage distance from optimality of dfs, t_{dfs} CPU time in seconds taken by dfs.

BendHeuristic found difficulties in solving set 3 and set 4, especially this last one, which have a cost structure that makes the master hard to solve when cuts begin to be added. Notice that in our "basic" implementation of BendHeuristic we let the master be solved to optimality, while in these cases—as suggested in Step 5—it would be worthwhile to solve the master only heuristically. However, we feel that this is a suggestion for a research line, rather than a contribution to estimate the effectiveness of the basic schema.

Table 5 shows detailed results on set 4, the one that was solved worse by BendHeuristic. Clearly this basic schema is not competitive on these instances, more sophisticated considerations are required. However, we believe that, since research on Benders based heuristics counts much less contributions than for instance Lagrange based ones, there is a wide space available for gaining insight on how to improve this basic functioning.

5.2 Multi-mode project scheduling

Step 1: identify a master MP and an "easy" subproblem SP. Problem RP2 is decomposed into master problem MP and subproblem SP. At iteration t, a feasible solution of MP corresponds to an assignment of modes to the activities; therefore, it defines a RCPSP instance whose solution cost provides an upper bound to the MRCPSP.

More in detail, the master is:

$$z_{\mathrm{MP}} = \min\ z \tag{76}$$

$$\text{s.t.}\quad z \geq \sum_{i\in X'}\sum_{m\in M_i} u^t_{im} d_{im} y_{im}, \qquad \mathbf{u}^t \in U \tag{77}$$

$$\sum_{m\in M_i} y_{im} = 1, \qquad i \in X' \tag{78}$$

$$\sum_{i\in X}\sum_{m\in M_i} w_{ims} y_{im} \leq W_s, \qquad s \in \mathrm{NR} \tag{79}$$

$$y_{im} \in \{0,1\}, \qquad m \in M_i, i \in X \tag{80}$$

where U is the set of the extreme points of the dual of the following subproblem:

$$z_{\mathrm{SP}}(\mathbf{y}) = \min \sum_{\ell\in M} h_\ell \tag{81}$$

$$\text{s.t.}\quad \sum_{\ell\in M_{im}} h_\ell \geq d_{im} y_{im}, \qquad m \in M_i, i \in X' \tag{82}$$

$$h_\ell \geq 0, \qquad \ell \in M \tag{83}$$

Step 3: solve master problem MP. Finding an optimal solution of MP may be too time-consuming, due to the number of Benders' cuts. We use therefore the following heuristic method. Let z^{t-1} be the solution found for problem MP at iteration $t-1$. At iteration t we start by setting $y^t = y^{t-1}$, where y^t is a feasible solution of MP. The solution y^t is iteratively improved by assigning to an activity i a feasible mode that minimizes the cost function when every other activity $j \in X\backslash\{i\}$ is assigned to the mode $\mu^t_j = \sum_{m\in M_j} m y^t_{im}$. This iterative procedure is repeated until no further improvement is achieved.

Step 5: solve subproblem SP. As the subproblem defined by a mode assignment is NP-hard, we solved it using a heuristic method, specifically we used the same heuristic proposed in Mingozzi et al. (1998). Moreover, in order to improve the quality of the final MRCPSP upper bound, we stored the K best RCPSP instances and solved them to optimality, at the end of HBEND, using the branch and bound method described by Mingozzi et al. (1998) for the RCPSP.

Step 6: Solve problem DP to add cuts to MP. A feasible solution of problem SP, at iteration t, provides a valid lower bound to the RCPSP defined by the solution of MP. Therefore, it can be obtained by any of the bounding procedures described in Mingozzi et al. (1998). In our computational results we used bound LB3.

The dual of SP is:

$$z_{\mathrm{DP}}(\mathbf{y}) = \max \sum_{i \in X'} \sum_{m \in M_i} u_{im} d_{im} y_{im} \tag{84}$$

$$\text{s.t.} \quad \sum_{(i,m) \in F_\ell} u_{im} \le 1, \quad \ell \in M \tag{85}$$

$$u_{im} \ge 0, \quad m \in M_i, i \in X' \tag{86}$$

where the cut to add to MP ant the tth iteration is in Eq. 84.

The full pseudocode of the heuristic we used, named HBEND, is as follows.

```
HBEND()
 1  identify a master MP(z,$\mathbf{y}$) and an “easy” subproblem SP($\mathbf{x}$),
 2  initialize $\mathbf{u}^0$, set $t = 0$, $W(0) = \emptyset$ and $z_{\mathrm{UB}} = \infty$
 3  repeat
 4          Solve (heuristically) master problem MP. Solution ($z^t$, $\mathbf{y}^t$)
 5          assign to each job $i$ the mode $\mu_i = \sum_{m \in M_i} m y^t_{im}$,
 6          solve heuristically the resulting RCPSP instance, solution ($z_{\mathrm{H}}$, $\mathbf{x}^t$)
 7          Set $z_{\mathrm{UB}} = \min(z_{\mathrm{UB}}, z_{\mathrm{H}})$.
 8          Solve in a heuristic way the subproblem SP,  solution ($z_d$, $\mathbf{u}^t$)
 9          Set $u^{t+1}_{i\mu_i} = 0, m \in M_i \backslash \mu_i, i \in X'$
10          if $z^t \ge z_d$
11             then STOP else set $t = t + 1$
12  until ($t$ = NITER)
```

5.2.1 MPSP, computational results

Algorithm HBEND has been implemented in C, results reported here have been obtained by running it on the same 1.7 GHz laptop used in the previous computational results subsections. We actually re-run the same code described in Maniezzo and Mingozzi (1999), except for allowing more iterations. As benchmark problems we used the instances generated by the project generator ProGen developed by Kolisch et al. (1995). We used the benchmarks with 10 and 20 activities, as these are still the only sets for which a significant number of results have been published in the literature. In these projects, each nondummy activity can make use of two renewable resources ($|\mathrm{RR}| = 2$), two nonrenewable resources ($|\mathrm{NR}| = 2$) and three possible modes ($|M_i| = 3$, $\forall i \in X$). Figure 5 shows a trace of a run of BendHeuristic on a 20 activities instance, sampled every 25 iterations, where upper and lower bound effectively converge.

Results are compared against those produced, on the same set of problems, by the best performing metaheuristics so far published, namely the particle swarm optimization by Zhang et al. (2006, PSO), the genetic algorithm by Hartmann (2001, GA-H),

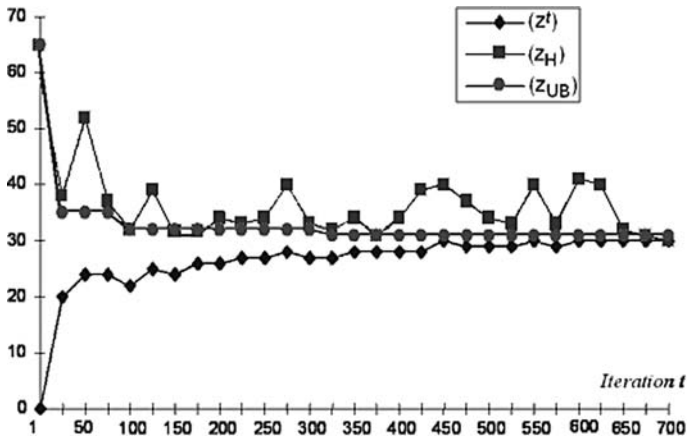

Fig. 5 Upper and lower bound evolution for a J20 problem instance

Table 6 Comparative results, 10 activities problem instances

10 activities	%Δavg	%Δmax	CPUavg	CPUmax
HE 100 cuts	6.07	58.10	0.03	0.05
HE 500 cuts	1.34	27.27	0.15	0.29
HE 1000 cuts	1.34	27.27	0.48	0.90
HE 2000 cuts	0.84	27.27	0.82	4.38
HB-z^t	3.68	19.04		
HSD	0.00	0.00	0.14[a]	2.31[a]
SA-J	0.20	22.20	7.00[b]	14.50[b]
PSO	0.11	5.10	n.a.	n.a.
SA-BL	0.21	7.80	n.a.	n.a.
GA-H	0.06	6.30	n.a.	n.a.

the simulated annealing by Jozefowska et al. (2001, SA-J) and the simulated annealing by Bouleimen and Lecocq (1998, SA-BL). Moreover we report the results of the branch and bound by Sprecher and Drexl (1998, HSD).

All algorithms operated on different machines and used different terminating conditions: HBEND was executed for a predefined number of Benders cuts, HSD was run for a given CPU time interval, SA-J for a given number of iterations, a disjunction of different factors for the others. We report CPU times, in seconds, when we have the data. caution should be taken, because our results were obtained on the cited machine, HSD was run on a PC 486 dx 66 MHz (a in Tables 6 and 7) and SA-J on a PC with 133 MHz Pentium processor and 32 MB RAM (b in Tables 6 and 7).

Both Tables 6 and 7 show that HBEND requires a number of cuts before achieving good results. However, a threshold on their number exists, after which adding more cuts does not produce any improvement. Permitting to add even more cuts would not improve the results on the tables. On the 10 activities projects, HSD performs

Table 7 Comparative results, 20 activities problem instances

20 activities	%Δavg	%Δmax	CPUavg	CPU max
HE 100 cuts	8.08	80.11	0.25	0.33
HE 500 cuts	4.24	50.20	1.25	1.67
HE 1000 cuts	2.13	16.66	3.12	5.49
HE 2000 cuts	1.89	13.22	4.99	32.30
HB-z^t	4.52	26.66		
HSD	2.38	51.61	60.00[a]	60.00[a]
SA-J	0.15	6.90	30.60[b]	75.00[b]
PSO	1.79	13.70	n.a.	n.a.
SA-BL	2.10	13.20	n.a.	n.a.
GA-H	1.21	14.20	n.a.	n.a.

better than HBEND, when the number of activities per project increases, and HBEND outperforms what becomes a truncated version of HSD. The absolute performance of HBEND is dominated by more recent metaheuristics, but not by much. Notice how HBEND results have been obtained in relatively short CPU times, contrary to those reported in Sect. 5.1, thanks to the heuristic master solution, and how, on the average, both upper and lower bounds converged to interesting values.

6 Conclusions

This papers advocates the possibility of deriving metaheuristic frameworks from mathematical programming (MP) techniques, originally designed for optimal solving. So far, metaheuristics derived their inspiration from natural phenomena or from boosting of local search; very little if anything comes from MP, and dual based information has very rarely been included. However, there are techniques which have been successfully used for decades even for heuristic solving, and which have been largely neglected by the metaheuristic community. We concentrated on two of them, Lagrangean and Benders decomposition, and showed how their basic working can be proposed as a metaheuristic schema, enjoying the usual properties of simplicity, robustness and effectiveness, but with the added bonus of a runtime quality check. Computational results were aimed at showing that often even a basic implementation of the MP-based metaheuristic schemata can lead to state-of-the-art performances, and that there are cases were more elaborate considerations can lead to top performing codes.

The same intuition that lays behind our two examples can be applied to other MP techniques, such as Dantzig-Wolfe or surrogate relaxation. We hope that this paper will help in the debate of the usability of MP components in metaheuristics.

References

Agar, M., Salhi, S.: Lagrangean heuristics applied to a variety of large capacitated plant location problems. J. Oper. Res. Soc. **49**, 1072–1084 (1998)

Ahuja, R.K., Orlin, J.B., Pallottino, S., Scaparra, M.P., Scutellà, M.G.: A multi-exchange heuristic for the single source capacitated facility location problem. Manag. Sci. **50** (2003)

Barahona, F., Anbil, R.: The volume algorithm: producing primal solutions with a subgradient method. Math. Program. **87**, 385–399 (2000)

Barcelo, J., Casanova, J.: A heuristic Lagrangean algorithm for the capacitated plant location problem. Eur. J. Oper. Res. **15**, 212–226 (1984)

Beasley, J.E.: Lagrangean relaxation. In: Reeves, C.R. (ed.) Modern Heuristic Techniques for Combinatorial Problems, pp. 243–303. Blackwell Scientific, Oxford (1993)

Benders, J.F.: Partitioning procedures for solving mixed-variables programming problems. Numer. Math. **4**, 280–322 (1962)

Boschetti, M.A., Jelasity, M., Maniezzo, V.: A local approach to membership overlay design. Working paper, Department of Computer Science, University of Bologna (2006)

Bouleimen, K., Lecocq, H.: A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem. In: Barbarosoglu, G., Karabati, S., Ozdamar, L., Ulusoy, G. (eds.) Proceedings of the Sixth International Workshop on Project Management and Scheduling, pp. 1922. Bogazici University (1998)

Chudak, F.A., Shmoys, D.B.: Improved approximation algorithms for a capacitated facility location problem. In: Proc. 10th Annu. ACM-SIAM Sympos. Discrete Algorithms, pp. S875–S876 (1999)

Delmaire, H., Diaz, J.A., Fernandez, E., Ortega, M.: Reactive grasp and tabu search based heuristics for the single source capacitated plant location problem. INFOR **37**, 194–225 (1999)

Drexl, A., Grunewald, J.: Nonpreemptive multi-mode resource-constrained project scheduling. IIE Trans. **25**(5), 74–81 (1993)

Elmaghraby, S.E.: Activity Networks: Project Planning and Control by network Models. Wiley, New York (1977)

Ganesh, A.J., Kermarrec, A.-M., Massoulié, L.: Peer-to-peer membership management for gossip-based protocols. IEEE Trans. Comput. **52**(2) (February 2003)

Glover, F.: Future paths for integer programming and links to artificial intelligence. Comput. Oper. Res. **13**, 533–549 (1986)

Glover, F., Laguna, M.: Tabu Search. Kluwer Academic, Boston (1997)

Hartmann, S.: Project scheduling with multiple modes: a genetic algorithm. Ann. Oper. Res. **102**, 111–135 (2001)

Holmberg, K., Ronnqvist, M., Yuan, D.: An exact algorithm for the capacitated facility location problems with single sourcing. Eur. J. Oper. Res. **113**, 544–559 (1999)

Holt, J., Ronnqvist, M., Tragantalerngsak, S.: A repeated matching heuristic for the single source capacitated facility location problem. Eur. J. Oper. Res. **116**, 51–68 (1999)

Jozefowska, J., Mika, M., Royzycki, R., Waligora, G., Weglarz, J.: Simulated annealing for multi-mode resource-constrained project scheduling. Ann. Oper. Res. **102**, 137–155 (2001)

Klincewicz, J., Luss, H.: A Lagrangean relaxation heuristic for capacitated facility location with single-source constraints. J. Oper. Res. Soc. **37**, 495–500 (1986)

Kolisch, R., Drexl, A.: Local search for nonpreemptive multi-mode resource constrained project scheduling. Technical Report 360, Institut fr Betriebswirtschaftslehre, Christian-Albrechts-Universitt zu Kiel, Kiel (1994)

Kolisch, R., Sprecher, A., Drexl, A.: Characterization and generation of a general class of resource constrained project scheduling problems. Manag. Sci. **41**, 1693–1703 (1995)

Maniezzo, V., Mingozzi, A.: A heuristic procedure for the multi-mode project scheduling problem based on benders decomposition. In: Weglarz, J. (ed.) Project Scheduling: Recent Models, Algorithms and Applications, pp. 179–196. Kluwer Academic, Dordrecht (1999)

Maniezzo, V., Boschetti, M.A., Jelasity, M.: An ant approach to membership overlay design. In: Ant Colony, Optimization and Swarm Intelligence: Proc. ANTS 2004. Lecture Notes in Computer Science, vol. 3172, p. 3748. Springer, Berlin (2004)

Maniezzo, V., Boschetti, M.A., Jelasity, M.: A fully distributed Lagrangean metaheuristic for a P2P overlay network design problem. In: Proceedings of the 6th Metaheuristics International Conference (MIC 2005), Vienna, Austria (2005)

Martello, S., Toth, P.: Knapsack Problems: Algorithms and Computer Implementations. Wiley, New York (1990)

Mingozzi, A., Maniezzo, V., Ricciardelli, S., Bianco, L.: An exact algorithm for the resource constrained project scheduling problem based on a new mathematical formulation. Manag. Sci. **44**, 715–729 (1998)

Neebe, A., Rao, M.: An algorithm for the fixed-charge assigning users to sources problem. J. Oper. Res. Soc. **34**, 1107–1113 (1983)

Peersim: A peer-to-peer simulator, http://peersim.sourceforge.net/ (2006)

Pirkul, H.: Efficient algorithm for the capacitated concentrator location problem. Comput. Oper. Res. **14**, 197–208 (1987)

Planetlab: An open platform for developing, deploying, and accessing planetary-scale services, http://www.planet-lab.org/ (2006)

Polyak, B.T.: Minimization of unsmooth functionals. USSR Comput. Math. Phys. **9**, 14–29 (1969)

Saroiu, S., Krishna Gummadi, P., Gribble, S.D.: A measurement study of peer-to-peer file sharing systems. In: Proceedings of Multimedia Computing and Networking 2002 (MMCN'02), San Jose, CA (2002)

Sherali, H.D., Choi, G.: Recovery of primal solutions when using subgradient optimization methods to solve Lagrangian duals of liner programs. Oper. Res. Lett. **19**, 105–113 (1996)

Solomon, M.: Algorithms for the vehicle routing and scheduling problem with time window constraints. Oper. Res. **35**, 254–365 (1987)

Sprecher, A., Drexl, A.: Solving multi-mode resource-constrained project scheduling by a simple, general and powerful sequencing algorithm. Eur. J. Oper. Res. **107**, 431–450 (1998)

Sridharan, R.: A Lagrangian heuristic for the capacitated plant location problem with single source constraints. Eur. J. Oper. Res. **66**, 305–312 (1991)

Talbot, B.: Resource-constrained project scheduling with time-resource tradeoffs: the nonpreemptive case. Manag. Sci. **28**, 1197–1210 (1982)

Ulusoy, G., Ozdamar, L.: A constraint-based perspective in resource constrained project scheduling. Int. J. Prod. Res. **32**, 693–705 (1994)

van Roy, T.J.: A cross decomposition algorithm for capacitated facility location. Oper. Res. **34**(1), 145–163 (1986)

Voss, S.: Meta-heuristics: The state of the art. In: Nareyek, A. (ed.) Local Search for Planning and Scheduling. LNAI, vol. 2148, pp. 1–23. Springer, Berlin (2001)

Zhang, H., Tam, C.M., Li, H.: Multimode project scheduling based on particle swarm optimization. Comput. Aided Civ. Infrastruct. Eng. **21**, 93–103 (2006)