Journal of Heuristics, 10: 89–104, 2004

Recovering Beam Search: Enhancing the Beam Search Approach for Combinatorial Optimization Problems

F. DELLA CROCE,* M. GHIRARDI AND R. TADEI
D.A.I., Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129, Torino, Italy
email: federico.dellacroce@polito.it

Submitted in November 2002 and accepted by Edmund Burke in November 2003 after 1 revision

Abstract

A hybrid heuristic method for combinatorial optimization problems is proposed that combines different classical techniques such as tree search procedures, bounding schemes and local search. The proposed method enhances the classic beam search approach by applying to each partial solution corresponding to a node selected by the beam, a further test that checks whether the current partial solution is dominated by another partial solution at the same level of the search tree. If this is the case, the latter solution becomes the new current partial solution. This step allows to partially recover from previous wrong decisions of the beam search procedure and can be seen as a local search step on the partial solution. We present here the application to two well known combinatorial optimization problems: the two-machine total completion time flow shop scheduling problem and the uncapacitated p-median location problem. In both cases the method strongly improves the performances with respect to the basic beam search approach and is competitive with the state of the art heuristics.

Key Words: heuristics, recovering beam search, scheduling, location

1. Introduction

In this work, Recovering Beam Search (RBS), a hybrid heuristic method for combinatorial optimization (CO) problems, is proposed. This method in an enhancement of the beam search (BS) approach (Ow and Morton, 1988, 1989) which in turn is a well established heuristic approach originally invented in the AI community. BS consists of a truncated branch and bound with a breadth-first search strategy where only the most promising w nodes at each level of the search tree are selected as nodes to branch from: w is the so-called *beam width*. Obviously, the larger is the beam width the slower is the algorithm. The nodes evaluation process at each level is the main issue of any BS procedure: typically, a two-stage approach is applied. First, a crude evaluation (*filtering phase*) is applied to select a reduced number of nodes for the accurate evaluation. This crude evaluation is a one-shot evaluation (typically requires constant time) and is applied to reduce the computational burden of the procedure. Then, the selected nodes are accurately evaluated and the best w nodes, w being

*Author to whom all correspondence should be addressed.

the beam width, are retained for branching. Note that the crude evaluation is actually an optional component of the BS approach. The accurate evaluation is typically performed by means of bounding procedures for the given problem. The more time consuming are these procedures, the more time consuming will be the overall procedure.

An error in the nodes evaluation of any BS procedure that induces the pruning of a *good* node (namely a node leading to an optimal or nearly optimal solution) can never be recovered. This is the major drawback of the BS approach: whenever all the *best* nodes are pruned, the best feasible solution reached may be significantly far from the optimum. To avoid this, the only mean available for a BS procedure is to use a sufficiently large beam width slowing down sometimes dramatically the procedure's efficiency.

The proposed RBS method overcomes this issue by introducing a *recovering* step that searches for improved partial solutions with respect to those selected by the beam. In order to evaluate a limited number of nodes in the search tree, the recovering step searches only for partial solutions situated at the same level of the search tree with respect to those selected by the beam. This step, which allows to partially recover from wrong decisions, is applied in such a way to increase only slightly the CPU time required by the procedure.

The proposed RBS method is tested here on two classic CO problems, namely the two-machine total completion time ($F2||\sum C_j$) scheduling problem and the uncapacitated p-median location problem. For both problems the method strongly outperforms the corresponding BS procedure and is competitive with the state of the art neighborhood-search based algorithms. This suggests that the RBS method may become a valid alternative to the currently available metaheuristic approaches.

The paper proceeds as follows. In Section 2, the RBS method is outlined. In Sections 3 and 4, it is shown how the RBS approach can be applied to the $F2||\sum C_j$ problem and the uncapacitated p-median location problem respectively. Each of these sections includes computational tests to compare the proposed method with the current state of the art algorithms. Section 5 concludes the paper with final remarks.

2. The recovering beam search method

Classic BS procedures cannot recover from wrong decisions: if a branch leading to the optimal solution in the search tree is pruned in the nodes evaluation process, there is no way to reach afterwards that solution. The RBS method seeks to overcome this issue by means of a *recovering step* that searches for improved partial solutions dominating those selected by the beam.

Consider a CO problem where the objective function must be minimized. Assume that a branching scheme and correspondingly a search tree has been devised for that problem either by considering available branch and bound procedures or by having devised an ad hoc exact search tree algorithm. The node evaluation process is guided here both by lower and upper bound procedures. Each node is evaluated by means of a convex combination of lower (LB) and upper (UB) bounds. The simplest way to do this is to consider a linear combination, namely the weighted sum $V = (1 - \alpha)LB + \alpha UB$, where V is the evaluation function and $0 \leq \alpha \leq 1$ is a parameter generally defined by experimental testing. The

more accurate the evaluation function at each node is, the smaller the deviation of the final solution value from the optimal solution value will be. A correct tuning of parameter α allows to obtain high quality results also for problems where either the *LB* procedure or the *UB* procedure (but not both) are not too precise.

While the *LB* is computed exclusively for evaluation purposes, the *UB* in each node represents also a feasible solution. One way to exploit this feature is to apply Lagrangean relaxation to derive the *LB* and generate the *UB*, namely a feasible solution, directly from the Lagrangean solution as shown in Beasley (1993). This is what occurs for the two applications presented in this paper. In both cases, with this approach, for each node of the search tree, the Lagrangean bound is the best bound obtained by exploring several values of the Lagrangean multipliers which determines several different Lagrangean solutions and correspondingly several distinct feasible solutions that constitute, in a sense, a *node neighborhood*: the best among these solutions represents the node upper bound.

Note that Lagrangean relaxation is not a unique option for the computation of the LB and not always a feasible solution can be immediately derived from the LB solution. We do not tackle this issue in this paper, but we refer to Della Croce and T'kindt (2002), where non Lagrangean bounds were successfully applied to solve a single machine scheduling problem with an RBS method. In that case the LB corresponded to the optimal solution (computable in polynomial time) of the preemptive version of the considered problem and the UB was derived by heuristically modifying the LB solution.

In the BS approach of Ow and Morton (1988), prior to the accurate nodes evaluation, a crude evaluation representing the filtering phase is applied. In the RBS method, the filtering phase works as follows. Problem dependent dominance conditions, denoted as *valid dominance conditions*, when available, are applied together with so-called *pseudo dominance conditions*, holding in a heuristic context only. Whenever a valid dominance condition or a pseudo dominance condition applies for a given node, that node is pruned. Examples of valid and pseudo dominance conditions are given in the following section for the $F2||\sum C_j$ problem.

In the RBS method (like in the classic BS approach), the beam width is constant and is kept generally fairly low ($\leq$10) in order to minimize the overall procedure CPU time.

The main feature of the proposed method is the so-called *Recovering Phase* that is applied at each search tree level. Let $S = \{\sigma_k, k = 1, \ldots, l \leq w\}$ be the vector of current partial solutions at a given level. These solutions are considered one at a time. The recovering phase checks, typically by means of interchange operators applied to the current partial solution x, whether solution x is dominated by another partial solution y sharing the same search tree level. If so, x is discarded. Further, if y does not belong to set S, then it becomes a new current partial solution. If y already belongs to S, then there is room for another partial solution to be examined by means of the recovering step and then retained so as to maintain, when possible, exactly w nodes.

Indeed, this step often allows to *recover* from previous wrong decisions of the procedure. Note that, in the recovering step, a partial solution may be only substituted by another partial solution sharing the same search tree level: this guarantees that the total number of explored nodes is polynomial provided that the search tree depth is polynomial. Note also that the dominance of a partial solution vs another partial solution may be also considered in

a heuristic fashion by means of pseudo dominance conditions: this issue will be discussed in detail in the section devoted to the uncapacitated p-median problem.

Consider a minimization problem with search tree depth equal to u. The main steps of the RBS method are as follows.

RBS method (beam width $= w$, search tree depth $= u$)

1. Initialization:

   l = search tree level $= 0$;
   σ_1 = best current partial solution = root node (typically no variable has been fixed, namely $\sigma_1 = \{\}$);
   S = vector of current partial solutions $= \{\sigma_1\}$;
   x = incumbent best solution value $= +\infty$.

2. FOR ($k = 1, k \leq \min\{|S|, w\}, k{+}{+}$)

   - Branch σ_k generating the corresponding children.
   - Filtering phase: prune all child nodes that are dominated by means of *valid* or *pseudo* dominance conditions

3. Empty set S: $S = \{\}$.
4. FOR each remaining child node:

   - Compute LB and UB. IF $UB < x$, THEN $x = UB$.
   - Compute the evaluation function $V = (1 - \alpha)LB + \alpha UB$ with $0 \leq \alpha \leq 1$.

5. Sort the set T of remaining children nodes in non-decreasing order of their evaluation function: let σ_k be the k-*th* best node.
6. Set $k = 1$.
   WHILE ($|S| < w$) AND ($k \leq |T|$), DO

   - Recovering step: search for a partial solution $\bar{\sigma}_k$ that dominates σ_k (and shares with σ_k the same search tree level) by means of interchange operators. IF $\bar{\sigma}_k$ is found, THEN set $\sigma_k = \bar{\sigma}_k$. IF $\sigma_k \notin S$, THEN $S = S \bigcup \{\sigma_k\}$, ELSE prune σ_k.
   - $k = k + 1$.

7. $l = l + 1$. IF $l < u$ GO TO 2, ELSE STOP: x is the final solution value.

3. Applying the RBS method to the $F2||\sum C_j$ problem

The $F2||\sum C_j$ problem can be stated as follows. A set of n jobs is available at time 0 to be processed by two machines. Every job consists of two operations where the first one must be processed on machine 1 and the second one on machine 2. The second operation cannot begin before the first operation completes. Preemption on either machine is not allowed. The objective is to minimize the sum of completion times. It is well known (Conway, Maxwell, and Miller, 1967) that the search for the optimal solution can be restricted to the set of permutation schedules, namely schedules in which every machine has the same job sequence. Let $p_{k,j}$ and $C_{k,j}$ be the processing and completion times of job j on machine k

respectively where j ranges from 1 to n and k is 1 or 2. The problem can be formulated as follows.

$$\min \quad \sum_{j=1}^{n} C_{2,j}$$

subject to

$$C_{1,j} \geq p_{1,j} \quad \forall j \tag{1}$$

$$C_{2,j} \geq C_{1,j} + p_{2,j} \quad \forall j \tag{2}$$

$$C_{1,i} - p_{1,i} \geq C_{1,j} \vee C_{1,j} - p_{1,j} \geq C_{1,i} \quad \forall i, j \; i \neq j \tag{3}$$

$$C_{2,i} - p_{2,i} \geq C_{2,j} \vee C_{2,j} - p_{2,j} \geq C_{2,i} \quad \forall i, j \; i \neq j \tag{4}$$

where (1) does not allow the processing of any job on machine 1 before time 0, (2) forbids for each job the start of the operation on the second machine before the completion of the operation on the first one, (3) and (4) indicate that the two machines can process one job at a time.

This problem is known to be NP-Hard in the strong sense (Garey, Johnson, and Sethi, 1979). In van de Velde (1990), a Lagrangean relaxation on constraints (2) was proposed. Among the heuristics available for this problem, the best performing ones are a purely descent neighborhood search (NS) procedure proposed in Della Croce, Narayan, and Tadei (1996), that applies a somewhat extended neighborhood and an Ant Colony Optimization (ACO) algorithm proposed in T'kindt et al. (2002), that was actually designed for a multi-objective generalization of the $F2||\sum C_j$ problem. The ACO algorithm works better than the NS procedure at the expense, though, of a much larger CPU time requirement.

In order to apply the RBS approach, it is necessary to specify its main components, namely: branching scheme, lower bound, upper bound and recovering step. Also, optional components such as filtering phase and dominance conditions must be considered. At a later stage, during the experimental design, the value of parameter α will be determined on the basis of computational testing that will examine also the behaviour of the method for different values of the beam width.

The *branching scheme* is the typical n-ary branching: the sequence is constructed by adding one job at a time starting from position 1 and the search tree is such that a node at level k indicates which is the job placed in position k.

As far as lower and upper bounds, filtering phase and recovering step are concerned, a more detailed analysis is necessary.

3.1. Lower and upper bounds

The Lagrangean bound LB_v proposed by van de Velde (1990) was applied as *lower bound*. To get LB_v, consider adding the following redundant constraints to the original formulation

of the problem

$$\textit{the schedule must be of permutation type} \tag{5}$$

$$C_{2,j} \geq p_{2,j} + \min_{1 \leq k \leq n} p_{1,k} \quad \forall\, j \tag{6}$$

and relaxing constraints (2) by a nonnegative vector of multipliers $\lambda = (\lambda_1, \ldots, \lambda_n)$. The resulting Lagrangean problem

$$L(\lambda) = \min \sum_{j=1}^{n} [\lambda_j (C_{1,j} + p_{2,j}) + (1 - \lambda_j) C_{2,j}]$$

subject to (1), (3), (4), (5) and (6)

can be reformulated as a linear ordering problem that is polynomially solvable if we impose $\lambda_j = c\ \forall j$ with $0 \leq c \leq 1$, by sequencing the jobs in non-decreasing order of $cp_{1,j} + (1-c)p_{2,j}$. Note that constraints (5) and (6) are redundant for the original problem but not for the Lagrangean problem, hence they may increase the value $L(\lambda)$.

Given the optimal solution $L(c)$ of the Lagrangean problem, LB_v is obtained by testing 21 different values of $c = x/20$ with $x = 0, \ldots, 20$ for the restricted Lagrangean dual problem $\max\{L(c) : 0 \leq c \leq 1\}$ and returning the largest value. This bound is further improved by iteratively perturbing each Lagrangean multiplier λ_j in a manner that does not affect the optimal sequence of the primal problem. We refer to van de Velde (1990) for a detailed description of this refinement. Here we note that apart from a preprocessing phase to be applied at the root node (with complexity $O(n^2)$) this bound can be implemented in $O(n)$ time.

The *upper bound* is derived by computing the solution values of all the different sequences obtained in the Lagrangean solution for different values of c. The best solution value constitutes the node upper bound.

3.2. Filtering phase

The filtering phase, whenever applicable, tries to reduce the children nodes generated by a given branch of the search tree. In this phase *valid* and *pseudo* dominance conditions for the given problem are applied.

To introduce a pseudo dominance condition for the $F2||\sum C_j$ problem, consider the following

Lemma 1. *If $p_{1,i} \leq p_{1,j}$ and $p_{2,i} \leq p_{2,j}$ (break ties arbitrarily), job i will always precede job j in any Lagrangean sequence derived to compute LB_v (though not necessarily in an optimal solution of the original problem).*

Proof: If $p_{1,i} \leq p_{1,j}$ and $p_{2,i} \leq p_{2,j}$, then $cp_{1,i} + (1-c)p_{2,i} \leq cp_{1,j} + (1-c)p_{2,j}$. □

Conway, Maxwell, and Miller (1967) erroneously claimed that there is an optimal solution in which job i precedes job j if $p_{1,i} \leq p_{1,j}$ and $p_{2,i} \leq p_{2,j}$. This claim was shown to be faulty by counterexample by Szwarc (1983). However, this claim is somewhat *heuristically*

valid in the sense that, for any instance, in most cases, whenever $p_{1,i} \leq p_{1,j}$ and $p_{2,i} \leq p_{2,j}$, then i precedes j in an optimal sequence. Based on the above consideration the following *pseudo dominance condition* is derived.

Condition 1. *Given a branch σ of the search tree, corresponding to a partial schedule of jobs, and two unscheduled jobs i and j, if $p_{1,i} \leq p_{1,j}$ and $p_{2,i} \leq p_{2,j}$ (break ties arbitrarily), then branch σj is pseudo-dominated by branch σi and is therefore pruned by the filter.*

Given a branch σ of the search tree, let $C_1(\sigma)$ and $C_2(\sigma)$ be the completion times of the last job $\in \sigma$. Also, for any job $j \in \sigma$, let $C_{1,j}(\sigma)$ and $C_{2,j}(\sigma)$ denote its completion times on the first and second machine respectively. The following known (Della Croce, Ghirardi, and Tadei, 2002; Della Croce, Narayan, and Tadei, 1996) *valid dominance conditions* are also applied in the filtering phase.

Condition 2. *Given a branch σ, if there exists an unscheduled job i such that $p_{1,i} \leq p_{2,i}$ and for each unscheduled job j we have $p_{1,i} \leq p_{1,j}$ and $p_{2,i} \leq p_{2,j}$, then job i is placed first among all the unscheduled jobs, namely only branch σi is generated.*

Condition 3. *Given a branch σ and two unscheduled jobs i, j, if $p_{1,i} \leq p_{1,j}$, $p_{2,i} \geq p_{2,j}$ and $\max\{C_1(\sigma) + p_{1,i}, C_2(\sigma)\} + p_{2,i} \leq \max\{C_1(\sigma) + p_{1,j}, C_2(\sigma)\} + p_{2,j}$, then σj is dominated by σi and is therefore pruned by the filter.*

3.3. Recovering step

Consider the following known (Della Croce, Ghirardi, and Tadei, 2002) valid dominance condition which is a generalization of the dynamic programming dominance criterion. Let q be the number of jobs not yet scheduled.

Condition 4. *Every branch σ such that there exists a permutation σ' of the jobs $\in \sigma$, $\sigma \neq \sigma'$ that satisfies the following conditions*

$$\sum_{j\in\sigma} C_{2,j}(\sigma') \leq \sum_{j\in\sigma} C_{2,j}(\sigma) \quad \textit{and} \quad q[C_2(\sigma') - C_2(\sigma)] \leq \sum_{j\in\sigma} C_{2,j}(\sigma) - \sum_{j\in\sigma} C_{2,j}(\sigma'),$$

can be pruned (*break ties arbitrarily*).

Condition 4 could be applied to any permutation of σ in the recovering step, but this would lead to consider an exponential number of cases. Assume j to be the last job of σ. Here we consider all permutations σ' reachable from σ by applying to each scheduled job $i \in \sigma, i \neq j$ the following swap and insert operators (Della Croce, 1995) involving job j:

- SWAP: swap jobs i and j.
- EBSR: (Extraction and Backward Shifted Re-insertion): Extract job j and re-insert it backward just before job i.

- EFSR: (Extraction and Forward Shifted Re-insertion): Extract job i and re-insert it immediately after job j.

If there exists more than one permutation σ' dominating σ, the one with the lowest total completion time is retained. As $O(n)$ permutations are considered, this condition can be implemented in $O(n^2)$ time. Whenever a permutation σ' that dominates σ already belongs to S, a further candidate node is considered. This case, however, occurs quite rarely and preliminary computational testing shows that, in practice, it is sufficient to limit to $2w$ the total number of nodes to be considered in the recovering step, where w is the constant beam width.

The following Property establishes the computational complexity of the RBS method for the $F2||\sum C_j$ problem.

Property 1. *The computational complexity of the RBS method for the $F2||\sum C_j$ problem is $O(n^3)$.*

Proof: The preprocessing phase for computing LB_v requires $O(n^2)$ time and is applied only once at the root node. For all the other nodes the complexity of the lower bounding procedure and correspondingly of the upper bounding procedure is $O(n)$. No more than $O(n^2)$ nodes are generated (and hence evaluated) as there are n levels in the search tree and each node generates at most $O(n)$ children. Hence the overall complexity of the bounding procedures is $O(n^3)$. All conditions in the filtering phase require at most $O(n)$ time with respect to $O(n^2)$ generated nodes to be possibly pruned. Finally, the $O(n^2)$ recovering step is applied at most $O(2w \cdot n)$ times (w being the constant beam width) leading to an overall $O(n^3)$ complexity of the RBS method. □

3.4. Computational results

We present the results of computational experiments conducted to test the quality of the RBS method for the $F2||\sum C_j$ problem. This method is compared with the strictly descent NS procedure of Della Croce, Narayan, and Tadei (1996) and the ACO procedure of T'kindt et al. (2002). All procedures were implemented in C++ language and tested on a PC P4/1000. The code of the ACO procedure was kindly provided by the authors.

For the RBS method several values of beam width w are considered. For $w = 1$, two further versions of the proposed method, namely a version without the recovering step and another without the filtering Condition 1, are also considered. For $w = 3$, a version without the recovering step is also considered. The jobs processing times on both machines are drawn at random from the uniform distribution $U(1, 100)$ and tested on problems with 100 and 500 jobs. For each problem size 20 different instances were generated. Concerning the best value of parameter α, preliminary computational experiments showed that the procedure obtained best results with $\alpha = 0.1$.

Tables 1 and 2 compare the performance of the procedures on problems with 100 and 500 jobs, respectively. The deviation of the heuristic procedures is derived with respect to the *LB* proposed in Della Croce, Narayan, and Tadei (1996), as the optimum is available

Table 1. Computational results on $F2||\sum C_j$ instances with 100 jobs.

Algorithm	Avg. % dev. from LB	Max. % dev. from LB	CPU (s) avg
RBS ($w = 1$)	1.01	1.53	0.01
RBS ($w = 1$, no rec)	1.84	2.79	0.01
RBS ($w = 1$, no cond 1)	1.11	1.57	0.01
RBS ($w = 3$)	0.99	1.53	0.02
RBS ($w = 3$, no rec)	1.70	2.41	0.01
RBS ($w = 5$)	0.92	1.41	0.04
RBS ($w = 10$)	0.92	1.39	0.07
NS	1.55	2.14	0.27
ACO	1.07	1.40	7.00

Table 2. Computational results on $F2||\sum C_j$ instances with 500 jobs.

Algorithm	Avg. % dev. from LB	Max. % dev. from LB	CPU (s) avg
RBS ($w = 1$)	0.47	0.84	5.37
RBS ($w = 1$, no rec)	0.79	1.03	5.02
RBS ($w = 1$, no cond 1)	0.49	0.93	6.74
RBS ($w = 3$)	0.45	0.83	15.29
RBS ($w = 3$, no rec)	0.76	0.98	12.10
RBS ($w = 5$)	0.44	0.82	23.12
RBS ($w = 10$)	0.44	0.82	60.79
NS	1.09	1.36	51.20
ACO	0.54	0.76	732.44

only for problems with up to 30 jobs. The results indicate that the RBS method is superior on the average to the other procedures already for $w = 1$ and that the recovering step is essential for its high quality performances increasing only slightly the required CPU time. Note that the results here somewhat contradict those presented in T'kindt et al. (2002) (the ACO procedure appeared there to outperform the RBS method) where, however, an earlier version of the RBS method with $w = 1$ was considered. The presence of the filtering Condition 1 helps both in reducing the CPU time and further improving the performance quality.

The CPU time in the RBS procedure increases (as expected) almost linearly with the beam width. We note in the larger instances ($n = 500$) that the RBS procedure with beam width $w = 1$ is approximately 10 times faster than *NS* and more than 100 times faster than *ACO* and shows already an inferior average deviation from the bound. Widening the beam width allows a further, though limited, reduction in the deviation from the *LB*.

4. Applying the RBS method to the uncapacitated p-median location problem

In the uncapacitated p-median location problem it is required to locate p facilities (medians) on a network so as to minimize the sum of all distances from each node to its nearest facility. It is well known (Hakimi, 1964) that there exists at least one optimal solution of the problem where the medians belong to the nodes set. The problem is known to be NP-Hard for general values of p (Kariv and Hakimi, 1979) and polynomially solvable for fixed p.

The p-median has been extensively studied in literature. Exact methods were proposed in Christofides and Beasley (1982) and Hanjoul and Peeters (1985). Among the heuristic procedures, we cite the Lagrangean heuristic algorithm of Beasley (1993) strongly based on the work of Christofides and Beasley (1982). The current state of the art algorithm is the Variable Decomposition Neighborhood Search procedure presented in Hansen and Mladenović (2001) that extends the results of a previous Variable Neighborhood Search algorithm of the same authors proposed in Hansen and Mladenović (1997).

Let $N = \{1, \ldots, n\}$ be the nodes set. Let x_{ij} be a $0-1$ variable such that $x_{ij} = 1$ if node j is allocated to node i (hence i is a median), $x_{ij} = 0$ viceversa. Let d_{ij} be the cost of allocating node j to node i. The p-median problem can be mathematically formulated as follows

$$\min \sum_{i \in N} \sum_{j \in N} d_{ij} x_{ij}$$

subject to

$$\sum_{i \in N} x_{ij} = 1 \quad \forall j \in N \tag{7}$$

$$\sum_{i \in N} x_{ii} = p \tag{8}$$

$$\sum_{j \in N} x_{ij} \leq |N| x_{ii} \quad \forall i \in N \tag{9}$$

$$x_{ij} \in \{0, 1\} \quad \forall i, j \in N \tag{10}$$

where (7) imposes that each node is allocated to a facility, (8) requires to have exactly p medians and (9) imposes that nothing can be allocated to a node unless that node is a median.

Let us consider now the main components of the RBS approach for the p-median problem. We adopt as branching scheme the same applied in Christofides and Beasley (1982): a node i is median ($x_{ii} = 1$) or non median ($x_{ii} = 0$). To the authors' knowledge, no valid dominance conditions are available for this problem. This is something one could expect as, even with $p-1$ medians already fixed, it may still be undetermined for most of the non median nodes to which median will they be allocated in an optimal solution. Hence, no filtering phase is present. The following subsections will show how to derive lower and upper bounds and recovering step.

4.1. *Lower and upper bounds*

The Lagrangean bound LB_{cb} proposed in Christofides and Beasley (1982) was applied as *lower bound*. To get LB_{cb} consider relaxing constraints (7) by a nonnegative vector of multipliers $\lambda = (\lambda_1, \ldots, \lambda_n)$. It is shown in Christofides and Beasley (1982) that $a_k = d_{kk} - \lambda_k + \sum_{j \in N, j \neq k} \min\{0, d_{kj} - \lambda_j\}$ is the contribution to the dual objective function of specifying that k is a median and that the resulting Lagrangean problem

$$L(\lambda) = \min \sum_{i \in N} \sum_{j \in N} (d_{ij} - \lambda_j) x_{ij} + \sum_{j \in N} \lambda_j$$

subject to (8), (9), and (10)

is optimally solvable by setting to 1 those x_{ii} with the smallest a_i (here we considered the Lagrangean bound at the root node of the search tree: the formulation for all the other nodes is given in Christofides and Beasley, 1982). Correspondingly, the allocation variables x_{ij} with $i \neq j$ are set to 1 if $d_{ij} \leq \lambda_j$ and to 0 otherwise. The Lagrangean bound is then optimized by means of a subgradient procedure accurately presented in Christofides and Beasley (1982) to which we refer for details.

As far as the *upper bound UB* is concerned, for each Lagrangean solution derived by the subgradient procedure the corresponding feasible solution (obtained by assigning each node to its nearest median) was computed and the best feasible solution value was retained as node *UB*.

4.2. *Recovering step*

Consider a partial solution σ with respect to the adopted branching scheme. This partial solution consists of some variables x_{ii} set to 1 (node i is median) or 0 (node i is non median). No dominance results of the type of Condition 4 are available for this problem. However it is possible to derive a pseudo dominance condition as follows.

Given σ and the related assignment of some variables x_{ii}, consider the assignment π of the remaining variables in the corresponding *UB* solution such that $\sigma\pi$ is a complete assignment of the x_{ii} variables ($i = 1, \ldots, n$). This is, in a sense, the heuristically best assignment we can have for the variables in π starting with σ as partial solution. Suppose there exists another assignment σ' of the variables in σ (with the same amount of x_{ii} variables set to 1) such that the complete assignment $\sigma'\pi$ induces a feasible solution whose value is lower than the previous *UB* value: then, we say that σ' *pseudo dominates* σ.

The *recovering step* is applied by searching for a *pseudo dominating* reassignment σ' of the variables in σ by means of a swap in the values of the last variable z set in σ and any of the variables of σ having opposite value with respect to z.

With respect to the computational complexity, note that, as each node can generate at most two children and there are n levels in the search tree, the number of generated nodes is $O(n)$. We do not provide, however, the overall complexity of the RBS method for the p-median problem as the lower bounding procedure complexity cannot be accurately stated (nor was it indicated in Christofides and Beasley, 1982) due to the presence of the subgradient procedure.

4.3. Computational results

The RBS method for the p-median problem was implemented in C++ language and tested on a PC P4/1000. The method applies the Lagrangean heuristic procedure of Beasley (1993) as preprocessing phase in order to reduce the problem size. We checked whether it could be worthy to apply also the reduction tests of Avella and Sforza (1999), but we did not find any significant improvement.

Our implementation of the procedure of Beasley (1993) differs from the original only in the settings of the step length parameter f for the subgradient optimization procedure. The initialization value was $f = 2$ as in the original version, whilst the stopping value was set to $f = 0.0000005$ instead of $f = 0.005$. These were the best settings we derived in a preliminary computational testing for the reduction procedure at the root node. For all the other nodes the starting value of step length parameter was $f = 0.000002$ and the stopping value was $f = 0.000001$.

The branching was performed alternatively (according to the ratio $\frac{p}{n-p}$) on the node corresponding to variable x_{ii} having the p-th or $p + 1$-th smallest a_i value in the *LB* procedure, namely the last x_{ii} variable set to 1 or the first set to 0.

Concerning the best value of parameter α, preliminary computational experiments showed that the procedure obtained best results with $\alpha = 0$ (hence, only the contribution of the *LB* was considered in this case).

Table 3. Computational results on p-median ORLIB test problems.

n	p	OPT	RBS ($w = 1$)	RBS ($w = 1$) (no rec)	RBS ($w = 3$)	RBS ($w = 3$) (no rec)	CPU (s) RBS ($w = 1$)	CPU (s) RBS ($w = 1$) (no rec)	CPU (s) RBS ($w = 3$)	CPU (s) RBS ($w = 3$) (no rec)
100	10	4093	**4093**	**4093**	**4093**	**4093**	<1	<1	<1	<1
100	10	4250	**4250**	**4250**	**4250**	**4250**	<1	<1	<1	<1
200	5	7824	**7824**	**7824**	**7824**	**7824**	2	2	4	3
300	5	7696	**7696**	7702	**7696**	7702	7	6	18	8
300	10	6634	**6634**	6640	**6634**	**6634**	7	7	23	9
400	5	8162	8163	8238	**8162**	8167	13	12	34	18
400	10	6999	7010	7083	**6999**	7083	15	14	39	20
500	10	8579	**8579**	8715	**8579**	8715	76	73	194	107
500	100	2961	**2961**	2991	**2961**	2985	167	162	303	279
600	5	9917	**9917**	9950	**9917**	9950	33	30	77	51
600	10	8307	8310	8317	**8307**	8317	31	38	78	57
700	5	10086	**10086**	**10086**	**10086**	**10086**	59	52	115	88
700	10	9297	9301	9301	**9297**	**9297**	68	63	144	91
800	5	10400	**10400**	10434	**10400**	10434	141	121	352	190
800	10	9934	**9934**	10157	**9934**	10049	170	148	414	293
900	5	11060	**11060**	11220	**11060**	11161	325	304	853	451
900	10	9423	**9423**	9437	**9423**	9429	201	192	595	355

The method was first tested on 40 ORLIB problems from Beasley (1985): see URL www.mscmga.ic.ac.uk/info.html. For these problems all optimal solutions are available.

Table 3 shows the behaviour of the RBS method with and without the recovering step. For 23 of the above 40 problems, the reduction procedure of Beasley (1993) was already able to derive a complete assignment of the x_{ii} variables and correspondingly the optimal solution was immediately available. These problems are not shown in the Table. For the remaining 17 problems, the improvement of the RBS method with respect to the corresponding BS method with no recovering step is impressive. When the recovering step is present, 13 problems are solved to optimality with $w = 1$ and all are optimally solved with $w = 3$. Without the recovering step, the quality of the solution degrades dramatically (only 4 out of 17 solved to optimality for $w = 1$ and 6 out of 17 for $w = 3$).

The method was then tested on larger instances taken from TSPLIB (Reinelt, 1991: see URL www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95) and tackled by the state of the art procedure of Hansen and Mladenović (2001). As the CPU times become important (due to the subgradient procedure for the *LB* computation), only the FL1400 and the FL3034 instances were considered here.

For all p-median instances derived from the FL1400 instance, Table 4 compares the solution of the RBS method for $w = 1$ and 3 with the best among the solutions obtained by

Table 4. Computational results on p-median FL1400 TSPLIB test problems.

p	VNS_{best}	RBS ($w = 1$)	RBS ($w = 3$)	LB	CPU (s) RBS ($w = 1$)	CPU (s) RBS ($w = 3$)
10	(101249.47)	101249.54	101249.54	101249.54	287	287
20	(57857.55)	57857.94	57857.94	57857.94	954	954
30	44086.53	**44013.48**	**44013.48**	44000.74	1174	2170
40	35005.82	**35002.52**	**35002.52**	34994.09	2059	4609
50	(29089.78)	29090.22	29090.22	29090.22	1213	2258
60	25166.15	**25161.12**	**25161.12**	25161.12	2294	6223
70	(22125.53)	22126.02	22126.02	22126.02	2041	3830
80	19877.88	**19870.84**	**19870.84**	19869.66	2461	5268
90	(17987.94)	17988.59	17988.59	17988.59	1755	3130
100	16551.20	16552.38	16552.38	16544.06	3026	7479
150	12032.65	12053.36	12052.00	12016.86	4057	8476
200	9360.01	9383.28	9378.31	9338.71	5559	9327
250	7742.70	7782.01	7759.43	7703.28	8038	12512
300	6624.52	6649.30	6637.47	6584.05	9295	19520
350	5727.02	5752.76	5741.12	5695.53	6533	10215
400	5020.50	5053.04	5026.76	5006.05	7885	12399
450	4487.73	**4469.53**	**4469.53**	4455.08	9108	22793
500	4049.03	4075.75	4066.16	4015.87	14881	23301

different versions of the VNS approach presented in Hansen and Mladenović (2001). This latter solution value is denoted by VNS_{best}. The Lagrangean *LB* computed at the root node is also provided.

The RBS method shows to be competitive with the state of the art VNS approach of Hansen and Mladenović (2001) though requiring a significantly larger CPU time: 5 (out of 18) improved solutions were found with respect to the results in Hansen and Mladenović (2001). For some instances the results of Hansen and Mladenović (2001) are slightly inferior to the entries of the lower bound; this is possibly due to minimal rounding errors in the conversion from the TSPLIB data: these values are given within round brackets. The

Table 5. Computational results on p-median FL3034 TSPLIB test problems.

p	VNS_{best}	RBS ($w = 1$)	LB	CPU (s) RBS ($w = 1$)
10	1213082.12	1213082.03	1213081.98	6320
20	841349.12	841498.71	840431.65	13308
30	680540.06	**679094.01**	676509.44	15054
40	573407.44	**571960.22**	571863.97	13183
50	507655.19	507693.28	507363.51	15689
60	462232.94	**461087.24**	460710.62	18451
70	428062.66	**426596.16**	425950.17	21456
80	397990.28	**397618.68**	397296.73	25743
90	373846.97	**373248.07**	373205.74	23278
100	353255.22	**352828.99**	352450.16	26917
150	281772.09	**281647.18**	281048.47	37665
200	238622.98	238822.03	238244.66	35063
250	209343.34	209414.74	209147.66	39211
300	187807.06	187823.72	187633.99	36658
350	171009.30	**170964.32**	170895.85	34964
400	157079.67	157108.85	157017.42	33881
450	145448.98	**145435.97**	145346.07	48718
500	135467.97	135486.07	135419.08	37875
550	126867.38	**126864.97**	126817.52	41891
600	119107.99	**119072.99**	119048.69	47388
650	112090.28	**112035.98**	112005.79	49370
700	105893.39	**105860.08**	105790.29	80698
750	100362.55	100399.05	100300.94	79929
800	95445.06	95458.07	95348.20	70658
850	91023.87	91046.53	90934.41	91101
900	87041.84	**87040.28**	86918.61	95899
950	83310.19	83366.64	83197.03	97380
1000	79900.52	79926.72	79759.51	106928

maximum percentage deviation from the *LB* is always under 1.25%. The only drawback in this case is the *CPU* time that grows up for these instances significantly (several hours in the worst case).

Table 5 provides the same entries of Table 4 for all p-median instances derived from the FL3034 instance, except that only beam width $w = 1$ is considered for the RBS method (as the CPU time becomes important). The RBS method shows to be even more competitive in this case: 15 (out of 28) improved solutions were found with respect to the results in Hansen and Mladenović (2001) and the maximum percentage deviation from the *LB* is always under 0.39%. The major drawback remains however the *CPU* time that grows up dramatically (approx. 30 hours in the worst case).

5. Conclusions

In this work a new heuristic method for solving CO problems has been proposed. The method is an enhancement of the Beam Search approach and allows to recover from previous wrong decisions of a classic beam search algorithm. The method has been succesfully applied to two classic CO problem showing to be competitive with the state of the art heuristics available for those problems. The method seems fairly general-purpose and applicable to a wide range of CO problems.

Several issues are worthy to be pursued in future research.

In both applications lower and upper bounds were computed by means of Lagrangean relaxation but it is shown in Della Croce and T'kindt (2002) that even combinatorial bounds can be applied with high quality performances provided that the upper bound is derived as best of a set of feasible solutions that constitute a neighborhood of the considered node in the search tree. The investigation of the behaviour of the method on other CO problems should verify the validity of this consideration. It would be also interesting to see the behavior of the RBS approach when dealing with problems where it is not easy to get feasible solutions.

In the p-median section, it is shown that the recovering step is based on a *pseudo dominance condition* applied to the current partial solution. This constitutes a fairly general approach that can be applied for the recovering step to all those problems whose structure does not induce valid dominance conditions of the type presented in the $F2||\sum C_j$ section (we point out, however, that *exact* recovering steps should be applied if available: indeed, preliminary tests, not presented in the paper, indicate that for the $F2||\sum C_j$ problem a recovering based on pseudo dominance conditions works much worse). It should be worthy to test other CO problems requiring a recovering step based on pseudo dominance conditions only.

Both applications presented here are based on lower bounds whose value deviates, in general, only slightly from the optimal solution value (besides, in the p-median problem, such bound is responsible of the important CPU times on the instances taken from the TSPLIB). It would be interesting to see the performances of the RBS approach on problems where *good* bounds are not available to see whether it can be succesfully applicable to any CO problem or if it works well on a restricted class only.

References

Avella, P. and A. Sforza. (1999). "Logical Reduction Tests for the p-Median Problem." *Annals of Operations Research* 86, 105–115.

Beasley, J.E. (1993). "Lagrangean Heuristics for Location Problems." *European Journal of Operational Research* 65, 383–399.

Christofides, N. and J.E. Beasley. (1982). "A Tree Search Algorithm for the p-Median Problem." *European Journal of Operational Research* 10, 196–204.

Conway, R.W., W.L. Maxwell, and L.W. Miller. (1967). *Theory of Scheduling*. Reading, MA: Addison-Wesley.

Della Croce, F. (1995). "Generalized Pairwise Interchanges and Machine Scheduling." *European Journal of Operational Research* 83, 310–319.

Della Croce, F., M. Ghirardi, and R. Tadei. (2002). "An Improved Branch and Bound Algorithm for the Two Machine Total Completion Time Flow Shop Problem." *European Journal of Operational Research* 139, 293–301.

Della Croce, F., V. Narayan, and R. Tadei. (1996). "The Two Machine Total Completion Time Flow Shop Problem." *European Journal of Operational Research* 90, 227–237.

Della Croce, F. and V. T'kindt. (2002). "A Recovering Beam Search Procedure for the Single Machine Dynamic Total Completion Time Scheduling Problem." *Journal of the Operational Research Society* 53, 1275–1280.

Garey, M.R., D.S. Johnson, and R. Sethi. (1979). "The Complexity of Flowshop and Jobshop Scheduling." *Mathematics of Operations Research* 1, 117–129.

Hakimi, S.L. (1964). "Optimum Locations of Switching Centers and the Absolute Centers and Medians of a Graph." *Operations Research* 12, 450–459.

Hanjoul, P. and D. Peeters. (1985). "A Comparison of Two Dual-Based Procedures for Solving the p-Median Problem." *European Journal of Operational Research* 20, 387–396.

Hansen, P. and N. Mladenović. (1997). "Variable Neighborhood Search for the p-Median." *Location Science* 5, 207–226.

Hansen, P. and N. Mladenović. (2001). "Variable Neighborhood Decomposition Search." *Journal of Heuristics* 7(4), 335–350.

Kariv, O. and S.L. Hakimi. (1979). "An Algorithmic Approach to Network Location Problems; Part 2. The p-Medians." *SIAM Journal on Applied Mathematics* 37, 539–560.

Ow, P.S. and T.E. Morton. (1988). "Filtered Beam Search in Scheduling." *International Journal of Production Research* 26, 297–307.

Ow, P.S. and T.E. Morton. (1989). "The Single Machine Early-Tardy Problem." *Management Science* 35, 177–191.

Reinelt, G. (1991). "A Traveling Salesman Problem Library." *ORSA Journal on Computing* 3, 376–384.

Szwarc, W. (1983). "The Flow-Shop Problem with Mean Completion Time Criterion." *IIE Transactions* 15, 172–176.

T'kindt, V., N. Monmarché, D. Laugt, and F. Tercinet. (2002). "An Ant Colony Optimization Algorithm to Solve a 2-Machine Bicriteria Flowshop Scheduling Problem." *European Journal of Operational Research* 142, 250–257.

van de Velde, S.L. (1990). "Minimizing the Sum of Job Completion Times in the Two-Machine Flow-Shop by Lagrangean Relaxation." *Annals of Operations Research* 26, 257–268.