Multi-Wave Algorithms for Metaheuristic Optimization

FRED GLOVER
Leeds School of Business
University of Colorado
Boulder, CO 80309-0419 USA
glover@colorado.edu

Updated version of paper published online in *Journal of Heuristics*, 03 June, 2016:
http://dx.doi.org/10.1007/s10732-016-9312-y

Abstract: We propose new iterated improvement neighborhood search algorithms for metaheuristic optimization by exploiting notions of conditional influence within a strategic oscillation framework. These approaches, which are unified within a class of methods called *multi-wave algorithms*, offer further refinements by memory based strategies that draw on the concept of persistent attractiveness. Our algorithms provide new forms of both neighborhood search methods and multi-start methods, and are readily embodied within evolutionary algorithms and memetic algorithms by solution combination mechanisms derived from path relinking. These methods can also be used to enhance branching strategies for mixed integer programming.

Keywords: metaheuristic optimization, iterated neighborhood search, multi-start algorithms, tabu search, evolutionary algorithms, mixed integer programming.

1. Multi-Wave Algorithms and Principles of Metaheuristic Search

We introduce a new design for a framework that links (1) iterated neighborhood search methods and (2) iterated constructive methods. By iterated neighborhood search we mean methods that start from an arbitrary solution and employ improving moves leading to a boundary condition defined by local optimality, and by iterated constructive search we mean methods that start from a null solution and employ moves that add elements (such as nodes or arcs of a graph, jobs in a schedule, or assignments of values to variables) leading to a boundary condition defined by a complete construction.[1]

We refer to our class of algorithms that embraces both (1) and (2) as *multi-wave algorithms*. To characterize this class of algorithms, we begin by identifying principles that apply alike to neighborhood search and constructive search.

Our principles rely on move evaluations that constitute a measure of objective function change in the context of a maximizing objective. We assume the measure is accurate for the solution produced by the move; i.e., that it correctly identifies the objective function change if the constructive or neighborhood search process would terminate upon generating this solution. We also assume that a path to high quality solutions is more likely to result by selecting moves with high evaluations than by selecting moves with low evaluations, relative to the collection of moves available.[2] Both assumptions are strengthened in the case where move evaluations incorporate "look ahead" effects and where they involve the solution of problem relaxations subject to the move choice.

We introduce three kinds of principles underlying multi-wave search, in the categories of (1) candidate lists, (2) conditional effects of sequential decisions and (3) persistent attractiveness. These principles are stated in an intuitive heuristic sense, to avoid becoming bogged down in formalism, and may be interpreted as conjectures about relationships we estimate to be useful for guiding the search to enhance its effectiveness.

2. Abbreviated Version of the Multi-Wave Algorithm and Candidate List Formation

We represent the optimization problem of interest in the generic form

$$\text{Maximize } f(x): \; x \in X$$

[1] This perspective that links these approaches derives from strategic oscillation, which proposes alternative forms of such boundary conditions within a common framework.

[2] This assumption, which has been cited as a basis for choices rules of deterministic and probabilistic tabu search, is valid for a variety of algorithms such as those for shortest path and minimum spanning tree problems, where selecting the best available move from the appropriate class will yield an optimal solution in a minimum number of steps.

Although we refer to a maximization objective, our comments can readily be re-expressed in terms of a minimization objective.

We adopt the convention whereby a solution x can represent either a vector of variables or a collection of attributes. In the following, we use the term *boundary solution* to refer to both a local optimum obtained by a neighborhood search algorithm and a complete (structurally feasible) construction obtained by a constructive algorithm.

In order to facilitate preliminary discussion, we sketch an abbreviated version of the Multi-Wave Algorithm which we will amplify subsequently.

Let $N(x)$ = the set of solutions that are neighbors of x (including neighbors defined by constructive search), and let $M(x)$ = the set of moves that lead to these neighbors; i.e. $M(x) = \{m = m(x) \in N(x)\}$. The kinds of moves we consider are those that can be interpreted as changing the value of a variable, or changing the location or classification to which a particular element or label is assigned. For constructive search that begins from a null solution, the "from" value, location or classification can be similarly designated as null. Thus, in all of these cases, the move that reverses a given move has a meaningful interpretation in a solution that has changed due to a sequence of intervening moves.

Within the neighborhood search context, we begin by focusing on *iterated improvement* neighborhood search (which only accepts moves that are improving) as the most natural analog of iterated constructive search. For this purpose, we assume that $M(x)$ is restricted to including improving moves.

Let CL denote a candidate list of moves extracted from $M(x)$. It can be important for problems with large neighborhoods to screen $M(x)$ initially to reduce its size using a candidate list approach such as described in Glover and Laguna (1997) and then to extract CL from this reduced $M(x)$. It is understood that CL depends on $M(x)$, but we avoid the cumbersome notation $CL(M(x))$.

We denote a best solution by x^*, which begins as a dummy solution with $f(x^*) = -\infty$, and interpret the instruction to update x^* to mean that we replace it by the currently generated solution x^c when $f(x^c) > f(x^*)$. The progression from a starting solution to a boundary solution will be called a *wave* of the solution process. In the following, MaxPass and MaxWave refer to selected iteration limits.

Algorithm 1: Multi-Wave Algorithm - Abbreviated Version

Function `MWA-abbreviated`($MaxPass$, $MaxWave$)

$x^* \leftarrow \emptyset$;

$f(x^*) \leftarrow -\infty$;

for $Pass = 1$ ***to*** $MaxPass$ **do**

  Generate a starting solution x^c (null, randomly or by post-analysis diversification);

  for $Wave = 1$ ***to*** $MaxWave$ **do**

    while *x^c is not a boundary solution (and CL is not empty)* **do**

      Select a move $m \in CL$;

      $x^c \leftarrow m(x^c)$ (Execute the move m on x^c);

    end

    if $f(x^c) > f(x^*)$ **then**

      $x^* \leftarrow x^c$;

    end

    Update relevant search parameters for the next Wave;

  end

  Update relevant diversification parameters for the next Pass;

end

return x^*;

The reference to generating a null starting solution at the start of the "Pass loop" applies to constructive search. We assume MaxPass = 1 for this form of search, for reasons elaborated later.

We note that the final step of each wave, which consists of updating relevant search parameters for the next wave, can either reinstate the same starting solution for each wave and rely on parameterized probabilistic choice rules (as discussed later) or it can amend the starting solution as by applying a partial destructive process from strategic oscillation to the final solution. In the case of iterated neighborhood search, the starting solution can also be generated by an embedded multi-wave constructive search, in which one or more preliminary iterations of the "Wave loop" are executed to generate candidate starting solutions by a constructive process. Thus, the designs we subsequently describe for executing this loop can be applied first to an iterated constructive process for generating a starting solution and then to an iterated neighborhood search process launched from this solution.

2.1 Candidate Lists

As a first step to elaborating the form of a more complete version of the Multi-Wave Algorithm, we focus attention on the candidate list, which we write in the form CL = $\{m_1, m_2, \ldots, m_s\}$. Following the typical design of tabu search (e.g., Glover and Laguna, 1997), we give priority to moves with higher evaluations and consequently use an indexing that orders the moves so that Evaluation(m_1) $\geq$ Evaluation(m_2) $\geq \ldots \geq$ Evaluation(m_s), and in addition, Evaluation(m_s) $\geq$ Evaluation(m) for any move $m \in M(x) \setminus CL$. The size s = n(CL) (= |CL|) of CL is often determined by a threshold T (Glover 1989, 1995), as by requiring a move m to satisfy Evaluation(m) $\geq$ T in order to belong to CL. Alternatively, s can be set to be some routinely selected target value. In either case, s can vary throughout the search process since it depends in part on the number of available moves in M(x), which itself may vary.

We make use of T to define a candidate list as follows.

Let MinE, MeanE and MaxE refer to the min, mean and max evaluations over M(x), and consider the two evaluation subintervals [MinE, MeanE] and [MeanE, MaxE]. Then for a parameter $\lambda \in [0,1]$, if the first of these subintervals is chosen, we define T = MinE + λ(MeanE – MinE), and if the second is chosen we define T = MeanE + λ(MaxE – MeanE). Thus, in particular, for the first case we set

$$CL = \{m \in M(x): \text{Evaluation}(m) \geq \text{MinE} + \lambda(\text{MeanE} - \text{MinE})) \quad (2.1)$$

and for the second case we set

$$CL = \{m \in M(x): \text{Evaluation}(m) \geq \text{MeanE} + \lambda(\text{MaxE} - \text{MeanE})). \quad (2.2)$$

By the perspective of this paper, we generally prefer (2.2) to (2.1).

Our definition of CL may be compared to another use of T, also illustrated in Glover (1995) and introduced in connection with GRASP in Martins, et al, (2000). The GRASP procedure, as elucidated in Resende and Ribeiro (2003), is noteworthy for the quality of its results and for pioneering the study of iterated constructive algorithms. The GRASP candidate list is based on a parameter $\alpha \in [0,1]$ which determines CL without reference to MeanE by defining

$$CL = \{m \in M(x): \text{Evaluation}(m) \geq \text{MinE} + \alpha(\text{MaxE} - \text{MinE}))$$

If the midpoint of the preceding representation for $\alpha = .5$ is taken to be an approximation to MeanE, then a rough correspondence between this candidate list and those based on MeanE occurs by specifying $\alpha = \lambda/2$ (hence $\alpha \in [0, .5]$) to associate this candidate list with (2.1), and by specifying (hence $\alpha \in [.5, 1]$) and $\alpha = (\lambda + 1)/2$ (hence $\alpha \in [.5, 1]$) to associate it with (2.2).

In the present work, we exploit the candidate list CL = $\{m_1, m_2, \ldots, m_s\}$ from (2.2) by drawing on the non-sequential version of probabilistic tabu search (P-TS) (Glover, 1989) which generates

positive valued weights w_i that have the same relative ordering as the move evaluations; i.e., $w_1 \geq w_2 \geq \ldots \geq w_s$. The approach then chooses a move m_i from CL by constructing a sample of size 1 from the uniform distribution with probability Pr(i) given by

$$\Pr(i) = w_i / \sum(w_j: \; j \in [1,s]). \tag{2.3}$$

As in the case of tabu search generally and P-TS in particular, we assume the move evaluations have a meaningful relation to objective function quality, in the sense of being correlated with the probability that moves selected on the basis of these evaluations will lead to solutions with high $f(x)$ values. We further assume that the evaluations have been scaled and translated, if necessary, so that they are all positive. For example, in the setting of improving neighborhood search, it is natural to use the convention that Evaluation(m) is positive for improving moves and non-positive otherwise. (In some cases, the word "improving" may be interpreted in a relaxed way to allow consideration of moves that do not change the objective function. In this event, a small translation may be used so that Evaluation(m) may still be positive for such moves.) Then we set

$$w_i = \text{Evaluation}(m_i) \quad \text{for } i \in [1,s]$$

or more generally, make reference to a non-negative power p and set

$$w_i = \text{Evaluation}(m_i)^p \quad \text{for } i \in [1,s]. \tag{2.4}$$

The exponent p can be selected greater than 1 to emphasize the differences between evaluations, or selected less than 1 to de-emphasize these differences. The completely random choice that results for $p = 0$ effectively corresponds to the approach used by GRASP (for its candidate list).

It should be noted that the original description of P-TS in Glover (1989) specified that the weights w_i defining the probabilities Pr(i) should be a monotonic non-decreasing function of the evaluations. We have adopted the approach often used in statistics and the physical sciences by using the power p as a simple way to satisfy this condition (see, e.g., Mitzenmacher, 2003). Appendix 1 elaborates the use of these candidate list evaluations by reference to their intensification-diversification tradeoffs, and discusses the design of experiments to exploit these tradeoffs.

2.2 Relevance of Memory for Intensification-Diversification Tradeoffs

Tradeoffs between intensification and diversification go beyond the consideration of candidate lists, and notably raise the question of mechanisms for achieving diversification, which from our perspective simultaneously makes reference to intensification. A contrast exists between diversity achieved through randomization and diversity achieved through memory. Diversity through randomization requires abandoning the type of intensification that results by focusing strongly on highest quality moves, whereas diversity through memory does not. Stated in another way, the use of memory for diversification allows choices that favor higher evaluations rather than requiring

these evaluations to be diluted in a probabilistic sense.[3] There are many different ways to employ memory to achieve intensification-diversification tradeoffs, some of which preserve an element of intensification more strongly than others.

We examine some simple ways to exploit memory to bring about useful intensification-diversification tradeoffs in the following sections. As a foundation, we depart from the classical conception of improving or constructive search by performing interventions to modify the search progression, according to the notion of strategic oscillation. To set the stage, we begin by examining the conditional effects of making decisions sequentially.

3. Conditional Effects of Sequential Decisions.[4]

Both constructive methods and neighborhood search methods in metaheuristic optimization employ sequential decision making processes, where the evaluation of potential decisions depends on earlier decisions. Consequently, the effect of conditionality is one of the primary determinants of the effectiveness of such methods. We examine the implications of conditionality by means of the following principle and the inferences that derive from it.

Principle of Marginal Conditional Validity (MCV Principle). — Starting from a given initial solution, as more moves are made, the information that permits these moves to be evaluated becomes increasingly effective for guiding the search to a high quality boundary solution, conditional upon the decisions previously made.

The justification for the MCV principle in the context of constructive methods is simply that as more decisions are made, the resulting moves cause the residual problem to be more and more restricted – in this case, reduced in dimensionality. Consequently, future decisions face less complexity and less ambiguity about which choices are likely to be preferable. In the context of an improving phase of neighborhood search, a form of progressively reduced dimensionality likewise results in relation to the possibilities for reaching a local optimum, which leads to the same conclusion.

The MCV Principle has two evident outcomes, in relation to the goal of producing a high quality boundary solution (complete construction or local optimum):

Inference 1. Early decisions are more likely to be bad ones.
Inference 2. Early decisions are likely to look better than they should, once later decisions have been made.

Inference 1 is an immediate consequence of the MCV Principle. Inference 2 results from the fact that later decisions manifest their quality in relation to the structure imposed by earlier decisions.

[3] Another interesting way of diluting evaluations is by means of proxy objective functions as discussed in Glover and Laguna (1997), ch. 5.
[4] The observations of this section generalize observations for constructive methods in Glover (2000).

Consequently, they are designed to "fit around" the earlier decisions, and thus are disposed to create a solution in which earlier decisions appear to be in harmony with those made later. (If, given later decisions, a decision made earlier looks bad, then almost certainly it is bad. However, if an earlier decision manages to look good in conjunction with those made subsequently, there is no assurance that it truly is good.)

These observations lead to the following additional inferences.[5]

Inference 3. The outcome of a constructive or local improvement method can often be improved by examining the resulting complete solution, where all decisions have been made, and seeing whether one of the decisions can now be advantageously replaced with a different one.

Inference 3 is empirically supported by the typical use of improving neighborhood search algorithms to find better solutions after applying a constructive algorithm. But Inference 3 more broadly applies to a sequence produced by an improving algorithm as well. We note that the inference is directly reinforced by the MCV Principle, because the outcome of changing a given decision – after a boundary solution is obtained – has the benefit of being evaluated in the situation where all other decisions have been made. Therefore, in a conditional sense, the validity of this changed decision is likely to be greater (its evaluation is likely to be more accurate) than that of the decision it replaces – since the replaced decision was made at a point where only some subset of the full set of decisions had been made. Nevertheless, the scope of Inference 3 is inhibited by Inference 2. That is, the influence of conditional choices will tend to make decisions embodied in the current solution look better than they really are, given the other decisions made that tend to "support" them.

Inference 3 also can be extended by noting that its conclusion is likely to be true even before reaching a boundary solution. We may express this as follows.

Inference 4. The outcome of a constructive or improvement method can often be improved by examining a solution at an intermediate stage, before reaching a boundary solution, and seeing whether one of the decisions can now be advantageously replaced with a different one.

This latter inference supports the type of constructive strategic oscillation approaches that intervene at an intermediate stage to modify the components of a solution before proceeding to a complete solution (Glover, 1989, 1995). We now apply this to more general strategic oscillation processes that underlie the multi-wave algorithm.

4. Basic Elements of the Multi-Wave Algorithm.

To build on the ideas of the preceding section we refer to both constructive and improving moves as *forward moves*, and refer to destructive moves and moves that complement (or "cancel") previous improving moves as *reverse moves*. A reverse move, for example, can be as simple as a

[5] We use the term "inferences" in the same sense as we use "principles" to refer to conjectures supported by heuristic argument.

move in binary optimization that complements the value x_j' assigned to a 0-1 variable x_j by assigning it the value $1 - x_j'$, and which has the natural counterpart of adding or dropping an element from a subgraph or cluster or schedule. More elaborate forms of reverse moves occur in the context of multiple element moves, and in contexts such as insertion or exchange moves in scheduling. For a constructive algorithm, the operation of dropping and reversing a move m is often very simple, consisting only of removing m from AMR. We will understand the reference to a reverse move m' in this case to involve just the dropping operation.

We also refer both to the (arbitrary) beginning solution that launches an improving phase of neighborhood search and to the null construction that launches a constructive phase, as the *initial solution*. In some kinds of optimization problems, such as multi-dimensional knapsack problems and generalized covering problems, a null solution can be treated as a solution that sets all variables to 0, and the constructive process consists of successively setting variables to 1 as a means of "adding" elements to the solution.

4.1 The Active Move Record

For the expanded multi-wave framework, at each iteration of applying moves starting from the initial solution, we maintain a record, called the *Active Move Record* (AMR) that identifies the forward moves selected so far that have not been cancelled by reverse moves. This record may be viewed as an ordered set or vector, in which moves recorded in AMR are maintained in a sequence where earlier moves precede later ones. Forward moves that are reversed are dropped out of their current position while other moves retain their order, while new moves are added at the end. Thus, in short, it is convenient to represent the Active Move Record by AMR = $(m_1, m_2, \ldots, m_r)$, where r = n(AMR) = |AMR|, and where the indexing sequence indicates that move m_i was executed before move m_{1+1} for $i \in [1, r-1]$.

Note that the ordering of AMR does not correspond to the ordering of the moves in the candidate list CL. Moreover, in contrast to CL, the moves recorded in AMR are not derived from any single move neighborhood M(x), but come from different neighborhoods for successively generated solutions. (In other words, each forward move in AMR was selected from some candidate list CL on a previous iteration.)

In the case of constructive search and improving search with binary variables, M(x), and hence the candidate list CL, automatically excludes the moves recorded in AMR, since these moves already are embodied in the current solution (viewing these moves to consist of adding attributes to the current solution, and these attributes need not be added again). However, the situation is subtler for non-binary decisions, and a special convention is required for managing AMR in this case, whose nature and rationale are described in Appendix 2. By means of this convention, the following discussion can be conveniently viewed from the perspective of conceiving all moves recorded in AMR either to be constructive moves or equivalent to assigning the values 0 and 1 to binary variables.

4.2 Dropping Moves from the Active Move Record

To handle the situation of dropping moves from AMR, the conditional relationships of Section 3 motivate us to identify these moves as consisting of one or more of the oldest moves from the AMR list, hence which come from the front of AMR by the understanding that the Active Move Record is ordered with the earliest moves appearing first. This establishes a correspondence between the AMR and the elementary type of tabu list in which oldest moves are dropped from the list and hence become members of the pool of moves available to be selected.

We observe that the conditional relationships discussed in the preceding section provide a direct motivation for removing the oldest moves from AMR. In a loose sense this may be compared to the types of branch and bound branching approaches (such as "best first" branching) that are biased to revisit and reverse branches closer to the root. However, this "root first" bias in branch and bound has a significant disadvantage because it can cause the tree to grow dramatically, and the bounds generated closer to the root are much less accurate, reducing their usefulness in guiding branching decisions. By contrast, the approach of *replacing* earlier choices by later ones produces a much different effect, making it possible to exploit conditional relationships without suffering the disadvantages imposed by a tree search structure.

It should be kept in mind that M(x) and CL can include moves that reverse moves in AMR. Such a possibility arises, for example, where a move m in AMR was improving when it was selected, but the influence of subsequent moves has caused the reverse of this move m' to become an improving move, thus making it available for inclusion in CL. (If m' is now chosen to be added to AMR, the move m is automatically dropped.)

At any stage, the effect of applying the moves in AMR to the initial solution yields a current solution x^c and in the case of a constructive algorithm we assume that the objective function value $f(x^c)$ can be evaluated for a partial solution as well as a complete solution. Finally, it should be observed that the sequence of moves on AMR may be different than an implicit sequence embodied in the moves themselves. For example, in a situation where the moves introduce and implied ordering, as where a move denoted by Move(i) is a decision to place an object in position i, the act of removing a current Move(1) through Move(5) from the start of AMR may be followed by placing new moves denoted Move(1) through Move(5) at the end of AMR, meaning that we again decide upon objects to place in positions 1 through 5, though these objects may be different than before. (The moves currently recorded at the start of AMR will then in general have nothing necessarily to do with the positions where objects are placed according to these moves.)

Making use of these notions, we first sketch the general form of the Multi-Wave algorithm and then describe the ways that AMR and CL are used within it.

5. General Design of the Multi-Wave Algorithm.

Employing the vertical and horizontal designations from Glover (1995), we divide each wave into alternating vertical and horizontal components, where vertical steps consist of forward moves that

enlarge the Active Move Record and horizontal steps consist of combined forward and reverse moves designed to leave the cardinality of AMR unchanged.

The search pattern adopted here, where each wave progresses until reaching a boundary solution, but does not continue beyond this solution, is called one-sided oscillation. Two-sided oscillation is treated later.

Algorithm 2: Multi-Wave Algorithm

Function `MWA`($MaxPass$, $MaxWave$, v, h, d)

$x^* \leftarrow \emptyset$;

$f(x^*) \leftarrow -\infty$;

for $Pass = 1$ ***to*** $MaxPass$ **do**

  Generate a starting solution x^c (null, randomly or by post-analysis diversification);

  for $Wave = 1$ ***to*** $MaxWave$ **do**

    $k \leftarrow 1$;

    while *x^c is not a boundary solution (and CL is not empty)* **do**

      if $n(AMR) = v_k$ **then**

        /*An intervention point reached*/

        $x^c \leftarrow$ `Horizontal_Phase`(x^c, h_k, d_k);

        $k \leftarrow k + 1$

      else

        $x^c \leftarrow$ `Vertical_Phase`(x^c);

      end

    end

    /*Conduct End-Wave Analysis*/

    if $f(x^c) > f(x^*)$ **then**

      $x^* \leftarrow x^c$;

    end

    `Concluding_Horizontal_Phase`(x^*,x^c, h_0, d_0);

    Update relevant search parameters for the next Wave;

  end

  /*Conduct End-Pass Analysis*/

  Update relevant diversification parameters for the next Pass;

end

return x^*;

The End-Wave Analysis and the End-Pass Analysis embody the processes of updating x^*, the search parameters and the diversification parameters, as indicated earlier in the Abbreviated Version of the algorithm. The End-Wave Analysis additionally performs a concluding horizontal phase. Parameters used by Algorithm 2 and details of these Post-Analysis steps are provided in subsequent sections.

5.1 Procedural Components of the Multi-Wave Algorithm.

We focus first on the horizontal component of the Multi-Wave algorithm. One way to execute this component is to employ an "exchange neighborhood" that compounds the forward and reverse moves into a single more elaborate move that maintains the number of active moves in AMR unchanged (as where a "drop/add" neighborhood drops one set of move attributes and simultaneously adds another set of attributes of equal size).

Here we handle the horizontal phase by the simpler approach of dropping moves from AMR by reversing them, and then adding an equal number of forward moves to AMR. The key steps for carrying this out involve: (1) deciding when a horizontal phase is to be launched, (2) selecting the number of moves to be executed during this phase, and (3) identifying the particular moves drawn from AMR to drop and the associated forward moves from CL that make up the phase.

5.1.1 Rules for determining intervention points and managing AMR.

We index the successive intervention points by k = 1, 2, …, k*. The parameter v_k denotes the number of vertical steps (forward moves) executed before launching intervention k, which therefore occurs when n(AMR) = v_k. Finally, h_k denotes the number of horizontal steps executed during this intervention and d_k denote the number of moves dropped on each of these steps. The values of these parameters are interrelated and we give simple rules to determine them.

The condition for reaching an intervention point and executing a horizontal phase may be expressed as follows.

> If n(AMR) = v_k, perform h_k horizontal steps each consisting of dropping (reversing) d_k moves from the start of AMR and sequentially adding h_k new moves to AMR from CL (where CL has access to the moves dropped from AMR).

The index k is initialized by setting k = 1 at the start of the current wave, and upon completing the horizontal phase triggered by the foregoing intervention, k is incremented by setting k ← k+1. (v_{k^*+1} can be given a large value to assure the last intervention will occur when n(AMR) = v_{k^*}.) The method then continues after completing the horizontal phase by resuming the process of making vertical moves.

To elaborate the mechanics of this operation, we again draw on the observations of Section 3, which indicate that moves inherited from earlier decisions are less likely to be good in a conditional sense than moves made later, thus motivating a strategy of reversing some number of these earlier moves and replacing them by others that now have better evaluations (relative to moves retained).

5.1.2 Protocol for selecting and reversing moves, and descriptions of the Horizontal and Vertical Phase Algorithms.

By our convention of representing the Active Move Record as AMR = ($m_1, m_2, \ldots, m_r$), where r = n(AMR), the set of moves to be dropped and reversed on each of the h_k horizontal steps can be written $\{m_1, m_2, \ldots, m_d\}$, where $d = d_k < r$.

The protocol for selecting moves m from AMR and executing the corresponding moves m' that reverse them can be stipulated as follows.

For $d_k = 1$, when move m is chosen from the first position of AMR, the reverse (dropping) move m' is immediately executed. Move m is removed from AMR, making m available to CL. A forward move is then selected from CL to add to AMR before selecting another move from the AMR to reverse.

For $d_k > 1$, all d_k moves m are selected at once from the front of AMR and the corresponding reverse moves m' are executed for all of them. It is useful to represent these moves as being selected and dropped sequentially, which permits the steps for dropping moves to be expressed in the same way for both $d_k = 1$ and $d_k > 1$. (Later we also discuss the possibility for choosing moves to be dropped in a different sequence.) After the step of dropping moves is performed, the method then selects d_k moves one at a time from CL, executing them in sequence and adding each in turn to AMR. This sequential selection implies that the move evaluations may change from one step to the next.

If no forward move exists to replace a given reverse move (as where no improving moves currently exist), a boundary solution is reached and the wave terminates. In short, all steps of adding moves to AMR during the horizontal phase have exactly the same form as the steps of choosing and executing forward moves during the vertical phase. The Horizontal Phase and Vertical Phase Algorithms can then be described as follows. It is understood that M(x) and the associated candidate list CL are updated each time x changes.

Algorithm 3: Horizontal Phase

Function `Horizontal_Phase`(x, h, d)

for $i = 1$ **to** h **do**

  /* Drop d moves from AMR */

  for $j = 1$ **to** d **do**

1 Choose m from the first position of AMR and identify the reverse move m';

2 $x \leftarrow m'(x)$ (Execute the reverse move m' on x);

3 Remove the move m from AMR (making it available to CL);

  end

  /* Add d moves to AMR (using updated evaluations and CLs) */

  for $j = 1$ **to** d **do**

4 Select a move $m \in CL$;

5 $x \leftarrow m(x)$ (Execute the move m on x);

6 Add the move m to AMR (in new last position);

  end

end

7 **return** x;

Algorithm 4: Vertical Phase

Function `Vertical_Phase`(x)

1 Select a forward move $m \in CL$;

2 $x \leftarrow m(x)$ (Execute the move m on x);

3 Add m to AMR (in new last position);

4 **return** x;

The application of this framework to enhance branching strategies in mixed integer programming, is discussed in Appendix 2.

6. Determining parameter values and the end-wave analysis.

A variety of rules are possible for determining the parameters of the Multi-Wave Algorithm, and particularly those of the horizontal phase. We focus here on rules that are easy to execute.

6.1 Determining v_k and k*

The v_k values which identify the values of n(AMR) at which interventions occur can be conveniently established by using an estimate n_e of the final n(AMR) value upon reaching a boundary solution. Such an estimate can be determined by executing a preliminary wave without any interventions (as insured by making v_1 large) and setting n_e = n(AMR) when the wave terminates. This n_e estimate can be updated after each subsequent wave to equal the final n(AMR) value obtained during this wave or the average of the final n(AMR) values obtained so far.

Given n_e, a desired spacing between successive interventions can be selected based on either:

(1) The separation Δ_k between successive values v_k at which successive interventions occur, which yields $v_k = v_{k-1} + \Delta_k$ for k = 0 to k*, where by convention $v_0 = 0$.

(2) The chosen total number of interventions k*.

The approach of (1) is useful for densely populating the values v_k over the total number of AMR moves executed during a wave (represented by n_e) and (2) is useful for sparsely populating these values; hence by implication (1) is suited to choosing k* relatively large (in comparison to n_e) and (2) is suited to choosing k* relatively small. Note in (1) the value Δ_k should be at least 2 since it doesn't make sence to perform an intervention that drops a move from AMR each time a move is added (which would occur if $\Delta_k = 1$). Evidently, in general, Δ_k should be at least 1 larger than d_k, the number of moves dropped, though we handle this by limiting d_k based on choosing Δ_k.

In the simplest case for both (1) and (2) we may elect to have the interventions divide the interval [1, n_e] into equal parts. Hence for (1) we select a constant value $\Delta = \Delta_k$ (≥ 2), which therefore determines $k^* = \lfloor n_e/\Delta \rfloor$, where a fractional value is rounded down. However, since we schedule a final intervention after the completion of a wave, if $k^*\Delta$ is relatively close to n_e, e.g., $k^*\Delta \geq n_e - \Delta/4$, then k* should be decreased by 1. We can more generally allow Δ_k to vary around the value of Δ, upon identifying k* in this fashion. If k* would be no greater than 5 (for example), and hence would qualify as relatively small, then we may treat (1) the same way as we treat (2), described next.

For (2) we consider values of k* ranging from as small as 1 to a selected upper limit. If we divide the interval [1, n_e] into equal parts, then when k* = 1 we let $v_1 = n_e/2$; when k* = 2, we let $v_1 = n_e/3$ and $v_2 = 2n_e/3$, and in general, for any value of k*, let $v_k = kn_e/(k^* + 1)$ for each k = 1 to k* (where fractional values are rounded to their nearest integers). A natural variation is to shift the v_k values to lie closer to one of the endpoints of the interval [1, n_e], as by letting $v_1 = 2n_e/3$ for k* = 1, letting $v_1 = n_e/2$ and $v_2 = 3n_e/4$ for k* = 2, and so forth by the rule $v_k = (k + 1)n_e/(k^* + 2)$.

When k* is chosen to be small, as in the range from 1 to 5, a few preliminary experiments can determine preferred values for k* and for the desired spacing for the v_k values. One type of such experimentation is to embed each "while loop" for the waves within a loop over the selected range of candidate values for k*. This also makes it possible to determine whether two or more k* values

work well together, in which case the method can be organized to examine these values on different waves using a similar embedded structure.

The potential merit of selecting the number of interventions to be small is suggested by the design of the novel Carousel Algorithm by Cerrone, Cerulli and Golden (2015) in the context of constructive search. Employing our terminology in the constructive setting, the Carousel algorithm may be characterized as always choosing $k^* = 1$ to perform a single horizontal phase (without including a horizontal phase at the conclusion of the wave) and maintaining $d_{k^*} = 1$ to drop a single element at each step within this phase. This approach may be considered a prototypical form of a multi-wave method when $k^* = 1$.

6.2 Determining d_k

The value d_k identifying the number of moves to be dropped during the k^{th} step of the horizontal phase should be chosen, as previously noted, so that $d_k < \Delta_k = v_k - v_{k-1}$. Again, an equal spacing option gives a possible starting point, giving each d_k the same value d for all k. Alternatively d_k may be selected randomly at each intervention point to lie in the interval [1, d]. The approach of choosing all $d_k = d = 1$ and the approach of choosing d_k randomly from an interval correspond to rules that are commonly used to manage simple tabu lists.

It should be observed that selecting $d_k = 1$ entails a risk of creating a greater tendency to add back a move that has just been dropped. This risk arises from the fact that a larger number of moves remain in AMR that depend on the dropped move, and hence which are compatible with giving this move an attractive evaluation. One way to counter this risk is to choose d_k larger than 1 throughout the current horizontal phase. Another is to let d_k be larger on the first step of the horizontal phase and then decrease it to a smaller value on subsequent steps. In the latter case, two options of interest are to decrease d_k immediately to 1 and to decrease it by a single unit at each step until it reaches 1.

6.3 Determining h_k

The number of horizontal steps h_k can be a function of the number of dropped moves d_k. If we want to assure that every move in AMR is dropped on one of these steps performed on the k^{th} intervention, we can let $h_k = \lceil n(\text{AMR})/d_k \rceil$, where AMR in its form when the k^{th} intervention is launched (and $\lceil\ \rceil$ is the function that rounds a fractional value up). By using a memory structure to exploit the notion of persistent attractiveness, as proposed later, we can determine h_k adaptively to account for more advanced considerations.

To summarize, each time a move or a block of moves is selected from AMR: (a) the moves dropped from AMR are made available to become members of CL; (b) the rule of probabilistic tabu search is employed for successively choosing moves from CL to be executed; and (c) M(x) and hence CL are updated after each of these moves. It is reasonable to expect that the horizontal interventions will contribute to an intensification-diversification tradeoff in a way that will enable the P-TS choice rules to focus primarily on intensification, as by using relatively large λ and p values. If λ is chosen sufficiently large, e.g., in the interval [.98, 1.0], then the value of p will be largely irrelevant.

6.4 End-Wave Analysis

As previously noted, the End-Wave Analysis step embodies a concluding horizontal phase that operates on the final AMR obtained during the wave.

This concluding phase has the goal of ironing out imperfections produced by choices following the last intervention. Thus, upon selecting a value d for the number of elements to be dropped on each step of this phase, we determine a value h for the number of these steps as indicated above for choosing h_k, i.e., choosing $h = \lceil n(AMR)/d \rceil$ to assure that each move in AMR is examined for dropping at least once. We later identify an adaptive way of determining h based on the notion of persistent attractiveness introduced in Section 7.

In contrast to the h steps of the horizontal phase executed in Algorithm 3, the post-wave phase does not simply replace the d moves dropped at each step with a following set of d moves that are added, but rather adds moves to AMR until again reaching a boundary solution. The horizontal phase does not end upon reaching such a boundary solution, but continues until all h steps are performed. However, each time a boundary solution is obtained, the best solution x* is updated if the boundary solution improves it. At the conclusion, as noted earlier, the step of updating relevant search parameters for the next wave can either reinstate the same starting solution for each wave and rely on parameterized probabilistic choice rules, or it can produce a new starting solution by applying a partial destructive process from strategic oscillation to the final solution.

Algorithm 5: Concluding Horizontal Phase

Function `Concluding_Horizontal_Phase`(x^*,x, h, d)

for $i = 1$ **to** h **do**

  /* Drop d moves from AMR */

  for $j = 1$ **to** d **do**

1 Choose m from the first position of AMR and identify the reverse move m';

2 $x \leftarrow m'(x)$ (Execute the reverse move m' on x);

3 Remove the move m from AMR (making it available to CL);

  end

  /* Add moves to AMR until reaching a boundary solution */

  while x *is not a boundary solution (and* CL *is not empty)* **do**

4 Select a move $m \in CL$;

5 $x \leftarrow m(x)$ (Execute the move m on x);

6 Add the move m to AMR (in new last position);

  end

  if $f(x) > f(x^*)$ **then**

7 $x^* \leftarrow x$;

  end

end

We observe that in contrast to Algorithm 5 the horizontal phase of Algorithm 3 does not check to see whether it encounters a boundary solution during its execution. By design, we have selected the parameter values so that the last intervention v_{k^*} will occur at a point reasonably before reaching the estimated final n(AMR) value n_e, and hence it is unlikely that a check for reaching a boundary solution is needed in Algorithm 3. However, in an application where the estimate n_e is less reliable, so that the cushion between v_{k^*} and the final n(AMR) may be erased, Algorithm 3 can be amended accordingly. In this case, upon reaching the intervention v_{k^*-1} (or more conservatively v_{k^*-2}), a check for a boundary solution can be inserted at the beginning of the beginning of the "for loop" for adding moves, immediately before selecting a move $m \in$ CL. If the current x is identified as a boundary solution, then a check can be performed to update x^* as in Algorithm 5, followed by breaking from this inner loop. (The execution of the outer loop remains unaffected, and hence Algorithm 3, like Algorithm 5, will continue to perform all h steps.)

7. Persistent Attractiveness.

The notion of Persistent Attractiveness (PA) applies to the portion of the Multi-Wave Algorithm that iterates over successive waves (from 1 to MaxWave). The PA notion is particularly relevant for generating effective constructive search paths and exploring the basins of attraction for improving neighborhood search. However, the role of PA differs for constructive and improving neighborhood search stemming from an inherent difference between these approaches in their potential access to best solutions. Neighborhoods for constructive methods, except where limited by pre-established heuristic rules, are usually designed with the ability to create a path from the initial (null) solution to every feasible solution, and hence to every optimal solution. On the other hand, these methods rarely discover the highest quality solutions because their choices can generate vast numbers of solution paths, and the choice criteria are not informative enough to reliably pinpoint the paths that are most desirable. Improving neighborhoods, by contrast, generally are accompanied by evaluations that provide better paths than constructive methods (chiefly because they work with complete rather than partial solutions). However, these paths typically access only a limited number of basins of attraction and their associated local optima.

The implication of this difference is that iterated constructive methods play out their scenarios over multiple waves rather than multiple passes, and hence improvements arise by structuring these waves more effectively. (Consequently, as previously indicated, MaxPass = 1 for these methods.) Iterated improving methods, by contrast, have a limited capacity to make improvements by focusing solely on executing multiple waves. For this reason, it is important to identify indicators for iterated improving search that tell when to terminate a series of waves and execute a diversification step to launch a new pass.

Recognizing this difference, we focus on three different types of Persistent Attractiveness that are relevant for selecting moves in both types of search.

PA Type 1: Exhibited by a move that is attractive enough more than once during a wave to be chosen to enter AMR a second or third time (a move that receives a high enough evaluation after being dropped from AMR to cause it to be chosen to re-enter AMR).

PA Type 2: Exhibited by a move that repeatedly has high evaluations during a wave, whether chosen to enter AMR or not, and if selected often does not enter AMR until a step that occurs somewhat after it first appears to be attractive.

PA Type 3: Exhibited by a move that repeatedly has a high evaluation during one or more waves, but which is not selected to enter AMR.

The implications of these different kinds of persistent attractiveness, and the means for exploiting them, are briefly summarized in the following principle.

Principle of Persistent Attractiveness.

(PA-1) A move of PA Type 1 allows a focus on dropping moves from the AMR that will have greater impact by exempting such a move from being dropped anew once it achieves the Type 1 status.

(PA-2) A move of PA Type 2 which entered AMR on a given wave offers an opportunity for producing a better solution on the next wave if it is selected earlier to enter AMR. (In particular, given that the move was eventually chosen, selecting it earlier affords a chance to make decisions based on this choice that were not influenced at the later point in the decision sequence when the move was chosen previously).

(PA-3) A move of PA Type 3 can be used to combine diversification with intensification by choosing at least one member of this type at a relatively early stage of a wave, assuring that the resulting solution will not duplicate a previous solution and allowing subsequent decisions to take this move into account.

7.1 Taking Advantage of Persistent Attractiveness

We describe approaches for exploiting each of the three types of moves embodied in the foregoing Principle of Persistent Attractiveness.

7.1.1 Exploiting (PA-1).

Let InAMR(m) denote the number of times m has been "in" (or added to) AMR on the current horizontal phase. Thus, at the beginning of a horizontal phase, we set InAMR(m) = 1 if m is currently in AMR and InAMR(m) = 0 otherwise.

During the horizontal phase we increment InAMR(m) by setting InAMR(m): = InAMR(m) + 1 each time m is selected to enter AMR. Hence InAMR(m) = 1 signifies that m is either currently in AMR or has been dropped from AMR but not added back. InAMR(m) = 2 signifies that m has belonged to AMR in the past and then dropped from AMR and added back (though not necessarily on the same step of the horizontal phase). Similarly, InAMR(m) = 3 signifies that m has been dropped from AMR and added back twice, and so on.

We use InAMR(m) on the current wave to prevent the method from dropping moves m during a horizontal phase that have demonstrated a persistent attractiveness by being added back one or more times, i.e., for which InAMR(m) is at least 2. We say m has a *weak persistent attractiveness* if InAMR(m) = 2 and a *strong persistent attractiveness* if InAMR(m) = 3 (or greater). Define the persistent attractiveness level, PAlevel, to be 2 in the former case and 3 in the latter case.

Choice Rule PA-1. Do not permit a move m to be dropped from AMR if InAMR(m) = PAlevel.

We write InAMR(m) = PAlevel above instead of InAMR(m) ≥ PAlevel because preventing a move from being dropped when InAMR(m) = PAlevel implies that InAMR(m) will never become greater than PAlevel.

Discussion of (PA-1).
The consequence of Choice Rule PA-1 is that instead of dropping the d oldest elements of AMR during a given step of a horizontal phase, we drop the d oldest elements such that InAMR(m) < PAlevel, and hence the dropped moves do not always come from the front of AMR. As time goes on, and the number of moves on AMR that qualify to be dropped will eventually fall below d. If d = 1 then the current horizontal phase must terminate, while if d > 1 then d must be reduced accordingly. (Ordinarily, the approach that begins with d > 1 and progressively reduces d on subsequent iterations will cause d to reach 1 before it will be compelled to equal 1 by applying (PA-1).) In general, when all moves on AMR satisfy InAMR(m) = PAlevel, then no more moves can be dropped during the current horizontal phase and the phase will terminate (if it has not already terminated by the limit h_k on the number of its steps).

However, by re-initializing InAMR(m) at the start of the next horizontal phase, we allow moves again to be dropped on this phase that were prevented from being dropped on the previous phase, motivated by the fact that the addition of new moves to AMR will change to evaluations of moves to be added to AMR. (Once more moves are in AMR, even though earlier moves in AMR were persistently attractive before, they may no longer be attractive because of the influence of these added moves on subsequent decisions.) The concluding horizontal phase that takes place during the End-Wave analysis likewise terminates when all moves on AMR satisfy InAMR(m) = PAlevel, if not before.

We may further exploit the possibility of dropping moves from AMR other than the oldest ones (at the front of the list), by allowing the dropped moves to lie within some range r_o from first move $m_{i'}$ that is eligible to drop, and then selecting the move $m_{i''}$ to drop, for $i'' \leq i' + r_o$ that is least attractive to retain – i.e., whose reverse move has the best evaluation over all the reverse moves derived from moves m_i in AMR such that $i \leq i' + r_o$. This variation more nearly resembles the classical form of strategic oscillation.

7.1.2 Exploiting (PA-2).

Let CL(b) = $\{m_1, \ldots, m_b\}$ denote a candidate list composed of the b highest evaluation moves in M(x), where b is relatively small (e.g., b = 5, limited to not exceed |M(x)|). When we specify the candidate list CL for the Multi-Wave Algorithm to consist of only a single highest evaluation element (which we anticipate to be preferable under most circumstances), the list CL(b) will normally be larger than CL. We will formally define the notion of attractiveness by calling a move m *attractive* if it belongs to CL(b).

We maintain a value Priority(m) during a current wave which is to be employed during the next wave to select the move to be executed on a forward step. Priority(m) starts at 0 at the beginning of the current wave, and is incremented on each forward step in which m qualifies as attractive (i.e., belongs to the current CL(b)) by setting

$$\text{Priority}(m) \leftarrow \text{Priority}(m) + \text{Evaluation}(m)^p$$

where p is chosen as in probabilistic tabu search as discussed in Section 2 and Appendix 1. (Hence, in the simplest case, for p = 0, Priority(m) $\leftarrow$ Priority(m) + 1.)

Then we let Priority0(m) = Priority(m) at the conclusion of the current wave, in order to use Priority0(m) to make choices on the next wave, where, instead of selecting m from CL, we select m to be the highest priority move on CL(b). It is important to break ties in favor of moves with higher evaluations. Hence, defining let MaxPriority = Max(Priority0(m): $m \in$ CL(b)), we employ the following rule.

Choice Rule PA-2: Choose m_i: i = Min(j: $m_j \in$ CL(b): Priority0(m_j) = MaxPriority). (Equivalently, i = arg max(Evaluation(m_j): $m_j \in$ CL(b) and Priority0(m_j) = MaxPriority).)

Throughout this next wave, we again update the value of Priority(m) (which is reinitialized to 0 at the beginning of the wave) in order to once more make selections based on Priority0(m) on the following wave. As an alternative to setting Priority0(m) = Priority(m) at the end of a current wave, we can maintain Priority0(m) as the average of all Priority(m) values from previous waves.

Discussion of (PA-2). Selecting the move specified by PA-2 can result in making a choice that is ultimately made anyway in the previous AMR, but reinforces the focus on this "good choice" so that its implications can be generated, and hence exploited, at an earlier stage. Consequently, this creates an intensification effect relative to such attractive, but repositioned, moves. But it can also change the evaluations of other moves, leading to different choices for the new AMR.

A variation on the rule for updating Priority(m) is to avoid incrementing its value on any step where move m is selected to enter AMR. Alternately, or in conjunction with this, Priority(m) may be biased by diminishing its value as a function of the first step of the wave in which m is added to AMR, so that moves selected to be added earlier have lower priorities than those selected later. (This bias is not determined by the position of m in the final AMR, since m may not appear in this list, and in any case its position on AMR can be shifted due to the operation of horizontal phases.) Throughout any period that m is an element of AMR, the value Priority(m) cannot be incremented, and hence this value will be larger for moves that are delayed in being added to AMR yet which were attractive enough to belong to CL(b). Consequently, repeatedly attractive moves that remain outside of AMR will have the largest Priority(m) values, and the opportunity to belong in this category is increased the longer that m remains outside AMR. In this sense, Priority(m) is already biased to be smaller for moves that are added to AMR earlier.

It may be imagined that the horizontal phases will automatically take care of the sequential influence embodied in (PA-2) since a move that is made later in the decision sequence will gradually move its way toward the start of AMR as a result of dropping the moves that precede it. As this occurs and moves are added to the end of AMR, the move will have an effect on larger numbers of decisions that follow it. However, this influence is not as complete as re-selecting m earlier in the process of generating AMR, because all the moves between m and the last element of AMR likewise influence the choice of moves subsequently added to AMR. In short, unless the horizontal phase selects the number d of dropped moves to be large, to remove many if not all of the moves that precede m on AMR, the effect of the horizontal phase will not match the effect of executing a new wave by exploiting (PA-2) as indicated. By implication, a dedicated exploitation

of (PA-2) may prove more effective than introducing horizontal phases. In this case, the use of (PA-1) becomes irrelevant. This is an issue that merits empirical study.

7.1.3 Exploiting (PA-3).

The ideas underlying (PA-3) invite consideration of in moves that qualify as attractive (by belonging to CL(b) as in (PA-2), but are never made (never enter AMR). We let InAMR0(m) denote a "long term version" of InAMR(m) as defined in (PA-2), which counts the number of times m has been in AMR over all previous phases and all previous waves. Then InAMR0(m) = 0 signals that m has never belonged to AMR.

To assure that the solution x generated from the current AMR does not duplicate any solution generated by a previous AMR, it is sufficient for AMR to include at least one move m for which InAMR0(m) = 0.

> **Choice Rule PA-3**: Let MinInAMR0 = Min(InAMR0(m): $m \in$ CL(b)). Then pick m' to add to AMR which is the highest evaluation move in CL(b) subject to InAMR0(m) = MinInAMR0; i.e., m' = arg max(Evaluation(m): InAMR0(m) = MinInAMR0). Once a move selected by this criterion occurs for MinInAMR0 = 0, then m' is not permitted to be dropped from AMR and Choice Rule PA-3 is replaced by Choice Rule PA-2.

A variation on Choice Rule PA-3 is to apply it only when MinInAMR0 = 0 and to use Choice Rule PA-2 when MinInAMR0 > 0.

Discussion of PA-3.

PA-3 is partially related to PA-2, since the PA-2 Choice Rule may also lead to selecting moves m for which InAMR0(m) = 0. But it is possible that a move chosen by PA-3 will not receive a priority by PA-2 that will cause it to be selected to enter AMR, or that will cause it to be selected at a stage as early as it would be by the PA-3 rule. (Also, the PA-2 rule will not prevent the first move chosen with InAMR0(m) = 0 from being dropped.)

A simplified form of the PA-3 Choice Rule can be applied as follows. When d_k moves are selected to be added to AMR during a horizontal step to replace those dropped, restrict the first of these added moves to be one that does not duplicate a move that was dropped during the same step.

PA-3 bears an important relationship to a *Principle of Conflicting Attractiveness* discussed in Appendix 3, which gives a means for refining its application.

8. End-Pass Analysis for Diversification

The final element of the Multi-Wave Algorithm concerns the End-Pass Analysis for Diversification. A diversification approach that requires no analytic component and yet sometimes works well (with notable exceptions) is to use simple randomization. To employ a more strategic form of diversification, we design the diversification process to incorporate an intensification

component, by seeking diversified solutions whose objective function values do not depart greatly from those of the best solutions found (Glover and Laguna, 1997). An effective instance of such a strategy occurs in the approach of De Corte and Sörensen (2015), which incorporates a variant of the iterated local search procedure characterized by Lourenço, Martin, and Stützle (2001, 2003). De Coret and Sörensen apply a restricted form of randomization to concentrate on a manipulating only a fraction of the problem variables – in their case, 30% – attended by the condition that increasing the values of these variables results in increasing the capacity of problem constraints. They further limit the change from the current locally best solution by allowing each variable to increase by only one unit. While not all problem settings make it possible to identify variables associated with increasing capacity, we may still apply the general idea of controlling the diversification to remain in regions where objective functions do not stray radically from best values.

For this purpose, we draw again on the mechanism of probabilistic tabu search by starting from a current best solution and creating a candidate list and a probabilistic function based on parameters λ and p, which provide a means for generating moves that exhibit an intensification-diversification tradeoff as described in Section 2 and Appendix 1. In this case, however, since we are no longer operating solely with improving moves, we require a mechanism to avoid cycling. Hence we incorporate a simple tabu restriction that prevents each move made from being reversed, using an unbounded tabu tenure that maintains the restriction throughout the entire duration of applying such moves.

As in the approach of De Corte and Sörensen (2015), we rely on experiment to determine an appropriate number of moves that will create just enough but not too much change in the solution ultimately produced. However, since we also control the displacement from solutions of high quality by the tradeoffs produced by the parameters λ and p, our approach may operate with a larger number of moves than could be effectively employed by a randomization process without such guidance.

This type of probabilistic strategy presents an opportunity to generate more than one diversification path from the current best solution, and thus to select a solution from a collection of such paths that yields the best objective value. Alternatively, conceiving that longer paths may provide greater diversification, the solution chosen can be based on both the path length and the objective value.

A diversification mechanism which drives the solution beyond the boundary solution represented by the current best solution may be viewed as a form of two-sided strategic oscillation. From this perspective we may also select moves that simultaneously drive the solution progressively farther from the initial solution for the improving search. This approach can be conveniently implemented in the manner of exterior path relinking (Glover and Laguna, 1997; Duarte et al., 2015). Probabilistic strategies of these forms for creating diversification processes have not been previously studied, and provide an inviting topic for investigation.

9. Concluding Observations: Differences Between Multi-Wave Approaches and Classical Multi-Start and Tabu Search Approaches

While the Multi-Wave approaches incorporate ideas from the strategic oscillation framework of tabu search, and such ideas have been embedded within both neighborhood search and constructive multi-start search algorithms in several settings, the specific form of these ideas in Multi-Wave algorithms embody key differences that are worth noting.

Most conspicuous is the invocation of the conditional relationships that to date have been largely disregarded both in applications and in experimental studies. While some forms of tabu lists bear a resemblance to the Active Move Record of the Multi-Wave approach, there are also striking contrasts between these two kinds of memory. The AMR expands and collapses by a different protocol than typically used to manage tabu lists, and the rules for intervening in the AMR construction by adding and dropping moves from its record likewise marks a departure from customary TS rules.[6] Moreover, the Multi-Wave algorithm in the neighborhood search context is keyed to iterated sequences of improving moves, in contrast to the tabu search approach that dissolves the distinction between improving and non-improving moves. This suggests the utility of blending the two approaches to take advantage of the fact that each may have useful performance characteristics under different circumstances.

The Multi-Wave approach in the neighborhood search setting raises the question of whether there is a "sweet spot" (ideal location) for the starting solution of a series of successive waves, as measured in relation to the number of moves away from a local optimum this starting solution lies. We have already noted that when a local optimum is reached after a very small number of moves, then the limiting number of successive waves should be reduced. That is, a diversification step should be triggered earlier in this case than when a local optimum is reached after a larger number of moves. If a sweet spot for a starting solution exists, an alternative approach is to conduct experimentation to identify an ideal range of distances (or numbers of moves) separating the starting solution from a local optimum. The result of such experimentation can be exploited by an adjustment step that employs a preliminary series of moves to relocate the starting solution by shifting it closer to or farther from the local optimum initially encountered by launching from this solution.

Another conspicuous difference between the Multi-Wave approach and other iterated neighborhood search and multi-start search methods derives from the operations for identifying and exploiting persistent attractiveness. These operations can drive the search in directions not included in the search repertoire of more customary methods.

In general, the contrasts between Multi-Wave algorithms and customary neighborhood search and multi-start search algorithms afford a fertile range of options to investigate, both in methods that lie wholly in one camp or another, and in methods that allocate different portions of their execution

[6] The type of tabu list management that comes closest to the AMR approach is he Reverse Elimination Method (Glover and Laguna, 1997), but the AMR approach is more flexible and introduces a staged approach to managing changes that is missing in the Reverse Elimination Method.

to different strategic frameworks, or that combine the approaches in ways suggested here. Computational studies are called for that exploit the flexibility and scope of the multi-wave algorithm to analyze the performance of its underlying strategies across diverse applications.

Acknowledgments

I am indebted to Raca Todosijević for his help in preparing the diagrams for the algorithms in this paper, and also owe my gratitude to two reviewers whose comments have helped to improve the paper's exposition.

References

Blum, C. and A. Roli (2003). "Metaheuristics in combinatorial optimization: Overview and conceptual comparison," *ACM Computing Surveys*, Volume 35 Issue 3, pp. 268-308.

Cerrone, C., R. Cerulli and B. Golden (2015). "Carousel Greedy: A Generalized Greedy Algorithm with Applications in Optimization and Statistics," University of Maryland (submitted for publication).

De Corte, A. and K. Sörensen (2015). "An iterated local search algorithm for water distribution network design optimization," ANT/OR Operations Research Group University of Antwerp, Belgium, to appear in the special issue "Metaheuristics for Network Optimization," *Networks*.

Duarte, A., J. Sánchez-Oro, M.G.C. Resende, F. Glover and R. Marti (2015). "Greedy randomized search procedure with exterior path relinking for differential dispersion minimization," Accepted: *Information Sciences*.

Glover, F. (1989). "Tabu Search - Part I," *ORSA Journal on Computing*, Vol. 1, No. 3, 190-206.

Glover, F. (1995). "Tabu Thresholding: Improved Search by Nonmonotonic Trajectories," *ORSA Journal on Computing*, Vol. 7, No. 4, pp. 426-442.

Glover, F. (2000). "Multi-Start and Strategic Oscillation Methods - Principles to Exploit Adaptive Memory," *Computing Tools for Modeling, Optimization and Simulation: Interfaces in Computer Science and Operations Research*, M. Laguna and J.L. Gonzales Velarde, Eds., Kluwer Academic Publishers, pp. 1-24.

Glover, F. and M. Laguna (1997). *Tabu Search*, Kluwer Academic Publishers, Boston.

Glover, F., V. Shylo and O. Shylo (2016). "Narrow Gauge and Analytical Branching Strategies for Mixed Integer Programming," *DBLP Computer Science Bibliography, CoRR*, http://arxiv.org/abs/1511.00021.

Lourenço, H.R., O. C. Martin, and T. Stützle (2001). "Iterated local search," arXiv preprint math/0102188.

Lourenço, H.R., O. C. Martin, and T. Stützle (2003). "Iterated local search," *Handbook of Metaheuristics*, chapter 11. Springer US.

Martins, S.L., M.G.C. Resende, C.C. Ribeiro, and P.M. Pardalos (2000). "A parallel GRASP for the Steiner tree problem in graphs using a hybrid local search strategy," *Journal*

of Global Optimization, pp. 267-283.
Mitzenmacher, M. (2003). "A Brief History of Generative Models for Power Law and Lognormal Distributions," *Internet Mathematics*, Vol. 1, No. 2, pp. 226–251.
Resende, M.G.C. and C.C. Ribeiro (2003) "Greedy randomized adaptive search procedures," in *Handbook of Metaheuristics*, F. Glover and G. Kochenberger, eds., Kluwer Academic Publishers, pp. 219-249.

Appendix 1 – Intensification-Diversification Tradeoffs for Candidate List Strategies

There is a useful connection between the size of the parameter λ determining CL in (2.2) and the power p of the probabilistic weighting function in (2.4), which gives rise to what may be called the *Intensification-Diversification Tradeoff*. Let ID(λ,p) denote a (hypothetical) tradeoff function that expresses the relative degree of intensification versus the degree of diversification, where larger values of ID(λ,p) correspond to greater search intensification but lesser diversification, while smaller values correspond to greater diversification and lesser intensification. Without assigning specific values to ID(λ,p) as a function of λ and p, we can nevertheless formulate a principle in the context of probabilistic choice to express its behavior.

First Principle of Intensification-Diversification Tradeoff – ID(λ,p) is a monotonically increasing function of λ and p; i.e., when either parameter increases while the other remains constant, the degree of intensification increases and the degree of diversification decreases.

Strictly speaking, the term "increasing" should be interpreted here as "non-decreasing," since there are ranges of λ and p over which the candidate list based on λ and the probabilistic choice rule based on p yield an unchanging outcome as one of the parameters stays constant and the other increases. (For example, at any stage of search, if λ lies in the interval that allows only the highest evaluation moves to become members of CL, all powers of p will yield the same choice. Similarly, regardless of the value of λ, if p is chosen large enough then P-TS will select only a highest evaluation move.)[7]

We anticipate that the manipulation of λ and p for the goal of finding an optimal Intensification-Diversification Tradeoff will be governed by the following relationship:

Second Principle of Intensification-Diversification Tradeoff – At higher ID(λ,p) levels, the average quality of solutions chosen during a search wave, and the average quality of the final solutions produced over a series of search waves, will be greater, but the variance in quality will be smaller.

[7] The tradeoffs discussed here can be viewed within the context of the intensification-diversification-randomization domain of Blum and Roli (2003), according to whether Evaluation(*m*) is based solely on the objective function or includes other evaluation criteria. Additional aspects of such tradeoffs are discussed in Glover and Laguna (1997), ch. 4 and 5.)

The principle raises a key question: Will the lower average solution quality attached to a lower ID(λ,p) level nevertheless make it possible to discover a best overall solution within a reasonable time limit, due to the increased variance in the solution quality?

The second principle can be conveniently tested to determine its relevance in specific problem settings by generating ID tradeoff curves – which we may visualize as charting the quality of the best solution on the vertical axis and the number of waves required to find this solution on the horizontal axis. The principle suggests that the early part of the curve, for smaller numbers of waves, will be higher for higher ID(λ,p) levels but then will gradually fall below the curves generated by successively lower ID(λ,p) levels as the number of waves grows.

These ID tradeoff curves disclose which (λ,p) pairs are most desirable for executing different numbers of waves, both in situations where the second ID tradeoff principle holds and in situations where it fails to hold. It is important to note that the preferred ID tradeoff can differ between methods that consist of iterated constructive waves by themselves and methods that employ neighborhood search at the conclusion of each constructive wave, since diversification is likely to be more important in the latter case. Similarly, the best tradeoff can be different again in the context of improving neighborhood search (whose moves differ from those of constructive search).

Design of Experimentation and a Path Relinking Connection

Experimentation to produce ID(λ,p) tradeoffs curves and to select a preferred (λ,p) pair is easy to design. Suppose, for concreteness, the goal is to conduct preliminary experiments with a sample of 25 test problems to test average performance over 10 passes consisting of 100 waves each. To carry out these tests we might select a collection of λ values and a collection of p values such as $\lambda \in \{90, .80, .70, .60, \ldots, .10\}$ and $p \in \{2.5, 2, 1.5, 1, .5, 0\}$, in this instance creating 45 pairs to test across 100 waves. (It is possible that fewer pairs would need to be tested in this example, since the ID(λ,p) tradeoff curve can quickly level off for higher values of λ and p. In general, it can be useful to keep track of the "effective maximum" number of waves for each (λ,p) pair, after which additional waves fail to bring improvement.) Once a preferred (λ,p) pair is selected, the testing can be refined by testing λ and p values within a reduced range on either side of the best values found. (In this example, if the preferred λ value turns out to be .70, then additional tests can look at λ = .65 and .75, and so forth.)

An unexplored context for experimentation with such a procedure arises in connection with path relinking. Upon selecting an initiating solution and one or more guiding solutions, each step in the progression from the initiating solution toward the guiding solutions can be made by selecting from available alternatives (elements in the guiding solutions not contained in the current solution) using such a P-TS approach. While nearly all path relinking implementations to date use a single guiding solution, the present approach suggests the relevance of multiple guiding solutions, which may be chosen from the reference set randomly or selected for their similarity or their diversity.

Appendix 2 – Managing the AMR List for Non-Binary Decisions and Multi-Wave Processes for Branching in Mixed Integer Programming

We require a special convention for managing AMR in the case of non-binary decisions. Define an *entity* for a move to be the element that is changed by the move, such as a variable that receives a new value or a label that is assigned to a new location or an object that is assigned a new classification. We then write a move m in the form m = (entity, FromA, ToA) where FromA is the from-attribute and ToA is the to-attribute for the entity; that is, FromA is the attribute possessed by the entity before the move and ToA is the attribute possessed by the entity after the move. In simple binary decisions it is sufficient to know just one of FromA and ToA. For example, if a move changes a binary variable (entity) x_j from 0 to 1, we need only know ToA = 1 in order to know FromA = 0. (The entity for a move does not have to be recorded in applications, since it is always implicitly known.)

We define m to be a *canonical* move if either FromA or ToA is an attribute of the initial solution. All moves involving binary decisions are automatically canonical. This is also true of all moves in constructive search since FromA for constructive moves is always a null element, and ToA for reverse (destructive) moves is also a null element. A "doubly canonical" move arises when the entity's FromA and ToA attributes are both in the initial solution (and hence are the same, given that an entity can only have a single attribute in a given solution) which of course amounts to a null move, since it changes nothing.

We propose two ways to manage AMR for non-binary decisions. The first and simplest consists of structuring AMR so that we only record "net" moves, which are automatically canonical, while the second involves a more complex updating process that has certain advantageous features.

Recording Canonical Net Moves

Starting from an initial solution, the first move for any given entity is clearly always a canonical move, but subsequent moves may not be. For example, a first move for a non-binary variable may change x_j from 0 to 1 and a later move, which is not canonical, may change x_j from 1 to 2. Similarly a first move may change the assignment of an object from Classification 1 to Classification 2, and a later one may change the assignment from Classification 2 to Classification 3. (An example occurs where classifications refer to clusters in which clusters are grouped.) To record all moves in AMR as "net" canonical moves (in which FromA is the attribute that comes from the initial solution) we simply fuse any non-canonical move with its preceding canonical move so that FromA remains the same as in the first move and ToA becomes the to-attribute of the second move. To illustrate, when the canonical move changing $x_j = 0$ to $x_j = 1$ is followed later with the non-canonical move changing $x_j = 1$ to $x_j = 2$, we fuse these moves to record only the net move from $x_j = 0$ to $x_j = 2$ as the current move. The earlier canonical move from $x_j = 0$ to $x_j = 1$ is superseded by the new canonical move, and hence is dropped from its position on AMR while the new move is added at the end. (This is the same as

first reversing the move from $x_j = 0$ to $x_j = 1$, which automatically drops this move from AMR, followed by executing the current move as $x_j = 0$ to $x_j = 2$.)

A special case occurs when the second move has ToA as an attribute of the initial solution. Then the net move is has FromA = ToA and the move collapses, so that no net move occurs, and the preceding canonical move is simply dropped. This corresponds to the case where the net move is a doubly canonical move as indicated earlier. For example, if the second move is from $x_j = 1$ to $x_j = 0$, then the net canonical move is from $x_j = 0$ to $x_j = 0$, which is not recorded on AMR. In short, the second move has simply reversed and hence cancelled the first move from $x_j = 0$ to $x_j = 1$. Put differently, a reverse move is always a move that cancels a forward move, but a forward move may also cancel a previous forward move.

Recording and Updating Move Predecessors

For this second way of handling non-binary decisions, we use the notation $x_j(j_1) = v_1$, $x_j(j_2) = v_2$, $x_j(j_3) = v_3$, …, to denote a succession of assignments to a variable (or attribute) x_j, where the indexes $j_1 < j_2 < j_3$ … identify the current location j_1, j_2, j_3, … on AMR where the assignment $x_j = v_1$, $x_j = v_2$, $x_j = v_3$ is recorded in the AMR list. (To be strictly accurate, the subscript j should also be attached to the values v_1, v_2, v_3, … in this representation, but we suppress j here for notational convenience.) Hence, considering x_j as representing an entity A, the move recorded in position j2 of AMR which assigns $x_j = v_2$ has FromA = v_1 and ToA = v_2. The move recorded in in position j_1 of AMR is implicitly understood to have FromA = v_0 where v_0 is the value assigned to x_j in the initial solution x^I; that is, the position j0 for the assignment $x_j(j0) = v_0$ is a "dummy position" (not in AMR) and the assignment itself is given by $x_j = x_j^I$.

When a move in a position j_s on AMR, which changes x_j from v_{s-1} to v_s, is dropped, the proper interpretation is that all subsequent assignments to x_j recorded in AMR now shift backward to receive their predecessor values.

To illustrate, when $x_j(j_3) = v_3$ is dropped (by dropping the move from $x_j = v_2$ to $x_j = v_3$) we may think of the assignment $x_j = v_3$ in position j_3 of AMR as being replaced by a null assignment, while the assignment $x_j(j_4) = v_4$ (which remains in position j_4) is changed to $x_j(j_4) = v_3$ (the predecessor assignment for x_j), the assignment $x_j(j_5) = v_5$ (which remains in position j_5) is changed to $x_j(j_5) = v_4$, etc. Since the positions j_s for $j > 3$ are now named inappropriately, we redefine j_3 to be the old j_4, redefine j_4 to be the old j_5, etc. The positions j_2 and j_1 remain unchanged, as do their assignments $x_j(j_1) = v_1$ and $x_j(j_2) = v_2$. Finally, we may shrink AMR to shift all positions after the old j3 position (which is currently empty) back by one unit.

A concrete example is as follows. Let the successive values v_1, v_2, v_3, … for a given variable be given by 1, 3, 5, 3, 2, 4 and suppose we drop the third assignment in this sequence $x_j(j3) = v_3$, where in this case $v_3 = 5$. (Note AMR may contain assignments to other variables that fall in positions between j_1, …, j_6 for x_j.) Then before removing the null element that replaces $x_j(j_3) = v_3$, the new sequence becomes 1, 3, ∅, 5, 3, 2 which shifts the assignments in positions after j_3 so that the new v_s becomes the old v_{s-1} and the new j_s becomes js-1 for $s > 3$. The null element can then be removed from AMR, which shifts all positions beyond j_3 back by 1 (including the

positions of assignments to variables not shown). The illustrated operations can be conveniently performed by using linked lists, rather than maintaining and updating pointers to positions in AMR.

While somewhat more intricate than the approach of recording and maintaining only canonical net moves, this approach better captures the essence of making successive assignments. In the special case of binary decisions, the two approaches are equivalent.

Multi-wave processes for branching in mixed integer programming

Within the context of branching methods in mixed integer programming, a multi-wave process can be applied as a generalization of a multi-start procedure by launching a series of waves from the root node of a branch-and-bound tree. This can be done either as an alternative to a branch-and-bound or branch-and-price algorithm, or as a means of creating a refined set of branching decisions for initiating such an algorithm. Such a process can also be launched from later nodes of a search tree to refine the choices at these junctions.

Here a boundary solution corresponds to a terminal node of the tree (where each tree node refers to an optimized linear program that imposes the branching inequalities as provisional constraints), and a penalty term is used to measure infeasibility. Evaluations to select successive moves – in this case, involving the choice of a branching variable and a direction to branch – can be made by criteria described in Glover, Shylo and Shylo (2016).

Appendix 3 – Conflicting Attractiveness and Move Influence.

A principle derived from tabu search advocates the merit of making moves that are "influential", i.e., that cause significant changes. The characteristic of being influential is not sufficient in itself to warrant a move, however, because moves that are merely influential have no necessary virtue unless they are also linked in some way to solution quality.[8] This leads to considering the following indicator.

Influence/Quality Indicator. Identify a move that appears attractive at some point during a given wave, especially during earlier steps. The move's evaluation should be increased if implementing the move also changes the attractiveness of other moves by causing their attractiveness to significantly increase of decrease.

A move that rates highly by reference to such an indicator can create significant diversification in the solutions produced. (It offers a chance to obtain a good solution that has a substantially different composition than the one obtained in the previous wave.) Applying this type of indicator, however, may require somewhat more work than applying the mechanisms for exploiting (PA-1), (PA-2) and (PA-3). Specifically, to know whether a given move will change the attractiveness of other moves requires that the move tentatively be made.

[8] Chapter 5, sections 5.1.1 and 5.1.2, of Glover and Laguna, 1997, discusses tradeoffs between influence and quality.

On the other hand, this added effort may be avoided if an indirect strategy is used, as suggested by the following analysis.

Principle of Conflicting Attractiveness and Move Influence

It is often likely that if making Move *A* causes Move *B* to become less attractive, then making Move *B* will also cause Move *A* to become less attractive. Thus, suppose Move *A* appears attractive at a particular point, but upon making Move *B* instead, Move *A* now becomes significantly less attractive. Then Move *A* may be considered an influential one (at least relative to Move *B*). Consequently, an indirect way to identify potentially influential (yet potentially good) moves is to look for those that were attractive at some (not-very-late) point in the previous wave, but were not selected, and which then later became significantly unattractive on this pass. These moves have a notably different effect than the moves sought by reference to the principle of persistent attractiveness embodied in (PA-1) and (PA-2), and can be important for longer term diversification. However, such moves can be sought in connection with the mechanism for exploiting (PA-3), by additionally keeping track of when the choice of a given move produces a marked change in the attractiveness of others.