INFORMS Journal on Computing
Vol. 11, No. 2, Spring 1999

0899-1499/99/1102-0198 $05.00
© 1999 INFORMS

Improved Constructive Multistart Strategies for the Quadratic Assignment Problem Using Adaptive Memory

CHARLES FLEURENT / *GIRO Inc., 75, Port-Royal East, Suite 500, Montréal (Québec), Canada, H3L 3T1, Email: Charles.Fleurent@giro.ca*

FRED GLOVER / *Graduate School of Business, CB 419, University of Colorado, Boulder, CO 80309, Email: fred.glover@colorado.edu*

(Received: May 1997; revised: February 1998, December 1998; accepted: February 1999)

Multistart constructive approaches operate by applying a local search procedure to start from different initial solutions produced by a repeated (variable) constructive process. The classical Random Restart procedure and the more recent GRASP procedure are prominent examples of such approaches. Adaptive memory strategies that are the heart of tabu search methods give a foundation for alternative, enhanced, multistart approaches. We demonstrate this by showing that a simple implementation of adaptive memory search principles, even when restricted to the constructive phases, can provide more effective multistart methods. Computational experiments for the quadratic assignment problem disclose that these methods improve significantly over previous multistart methods that do not incorporate such memory based strategies.

Multistart constructive approaches can be characterized as methods that apply a local search procedure to initial solutions generated by a repeated constructive process. In the case of the Random Restart method, the construction phase simply consists in generating random starting points. For a long time, this approach was widely popular before the introduction of methods for escaping local optima such as simulated annealing[12] and tabu search.[9] The greedy randomized adaptive search procedure (GRASP) method[6] uses a more sophisticated construction phase than naive Random Restart in the hope of providing better initial solutions for the local search procedure.

Although these multistart approaches produce results of reasonable quality, a major shortcoming resides in the fact that the solutions generated are "thrown away." An evident alternative is to use adaptive memory designs as proposed by tabu search to retain and analyze features of selected solutions, and thus to provide a basis for improving future executions of the constructive process. In fact, an interesting connection exists between GRASP and probabilistic tabu search (PTS[11]). If PTS is implemented in a memoryless form, and restricted to operate only in the constructive phase of a multistart procedure (stripping away memory, and even probabilistic choice, from the improving phase), then a procedure resembling GRASP results. The chief difference is that the probabilities used in PTS are rarely chosen to be uniform over members of the candidate list, but generally seek to capture variations in the evaluations, whenever these variations reflect anticipated differences in the effective quality of the moves considered.

This connection raises the question of whether a multistart variant of probabilistic tabu search may offer a useful alternative to memoryless multistart approaches like GRASP. The central contribution of this article is to establish new outcomes and findings about multistart methods, which require a reassessment of the folklore about these methods that has become popularly accepted in certain parts of the literature. The issue we address is to establish the relative performance of different strategies from a class that has attracted a significant amount of attention and that has a significant number of proponents.[6, 13, 15, 16]

In this article, a study of this issue is conducted for the quadratic assignment problem (QAP), where GRASP has been reported to perform well. To provide a basis for comparison, we use a variant of the PTS multistart method whose improving phases (as opposed to construction phases) exclude the use of TS memory and guidance strategies, and instead are restricted to employ a standard descent procedure. The resulting multistart method nevertheless proves significantly superior to other multistart approaches previously reported for the QAP. Not surprisingly, however, it also turns out to be not as effective as the leading tabu search methods that use memory in the improving phases as well as (or instead of) in the constructive phases.

In spite of the value of TS memory-based strategies during improving phases, in many contexts, it seems reasonable to conjecture that classes of problems exist where increased reliance on restarting will prove advantageous, and where the best results may be obtained from appropriately designed multistart strategies. For example, the tabu search study of Dell'Amico and Trubian,[4] which provides the best-known results for a wide range of weighted graph partitioning (equicut) problems, indicates that multistart strategies can be competitive in the case of sparse geometric graphs. However, many tabu search implementations do not use TS strategies during constructive solution phases, which our findings suggest may well be an error for problems with particular structures.

Subject classifications: Combinatorial optimization.
Other key words: Tabu search, multistart strategy, quadratic assignment problem.

The remainder of this article is organized as follows. In Section 1, we describe a template for constructive multistart methods, and identify Random Restart and GRASP as two examples of such methods. In Section 2, we identify a straightforward intensification strategy based on extracting pertinent information from a set of elite solutions, in accordance with customary tabu search prescriptions. Section 3 describes how the approach can make use of the proximate optimality principle to further improve the constructive process. Finally, in Section 4, the different strategies are tested on the QAP test instances from the standard QAPLIB problem library.[2] Concluding remarks are made in Section 5.

1. Constructive Multistart Methods

A constructive multistart method can simply be defined as an approach that constructs several solutions, which in turn are submitted to a local search procedure. The best generated solution during the process is then returned. In order to characterize such a method, one needs to specify a local search algorithm and a constructive procedure that provides the starting points. In this section, two examples of such approaches (Random Restart (RR), GRASP) are given for the Quadratic Assignment Problem (QAP).

The QAP can be stated as:

$$\min_{\phi \in P(n)} \sum_{i=1}^{n} \sum_{j=1}^{n} f_{ij} d_{\phi(i)\phi(j)},$$

where $F = (f_{ij})$ and $D = (d_{kl})$ are two $n \times n$ matrices and $P(n)$ is the set of all permutations of $\{1, \ldots, n\}$. Matrix F is often referred to as a flow matrix between facilities, and D as a distance matrix between sites, and each may be considered symmetric with a null diagonal. A permutation can then be interpreted as an assignment of facilities to sites with a quadratic cost. This problem is especially hard for exact methods (when $n > 20$), and consequently heuristic procedures are of particular interest. Because of both its practical and theoretical significance,[14] this problem has been very well studied, and the approaches that currently give the best results are described by Battiti and Tecchiolli,[1] Chaprakani and Skorin-Kapor,[3] Fleurent and Ferland,[7] and Taillard.[17]

1.1 Random Restart

A local search procedure repeatedly moves from a current solution to an improving (lower cost) neighbor until a local optimum is reached. For the QAP, local search is usually based on pair exchanges in which two elements of a permutation are swapped. Hence, if ϕ is a permutation, the neighbor π obtained by the pairwise exchange of indices r and s is:

$$\begin{aligned} \pi(k) &= \phi(k), \ \forall k \notin \{r, s\} \\ \pi(r) &= \phi(s), \\ \pi(s) &= \phi(r). \end{aligned} \tag{1}$$

Special formulas and data structures to efficiently evaluate neighbors are described, for example, by Fleurent and Ferland.[7]

To initiate the local search procedure, a starting solution must be specified. A possible implementation is to simply generate uniform random permutations. The resulting method is then the Random Restart approach.

1.2 GRASP

The GRASP method is another example of a multistart approach in which a constructive method attempts to provide better starting points for the local improvement procedure. A specialized function builds a solution one step at a time, selecting at each stage a possible assignment until a feasible solution is obtained.[13] As is the case with most GRASP adaptations,[5, 15] the constructive phase has two main features:

Adaptive Greedy Measures

In a greedy algorithm, there are usually values that evaluate the benefit of each admissible assignment. An adaptive measure means that these values are updated in order to incorporate the new information that is gathered after each assignment is made. (This feature is characteristic of all reasonable constructive methods, including those underlying the early branch and bound approaches.)

Randomized Selection of Assignments

At each step, an assignment is randomly selected among a subset of the top possible assignments. This mechanism allows the procedure to be repeated and to generate different solutions.

For the QAP, the constructive phase starts with two sets $I = \{1, \ldots, n\}$ and $J = \{1, \ldots, n\}$. The solution s is then constructed by a sequence of assignments $s[i] = j$ in which i is selected from I and j from J. After each assignment, the sets I and J are updated with $I = I - \{i\}$ and $J = J - \{j\}$. Letting Γ be the set of assignments already made, the cost of assigning facility i to site j, relative to the partially constructed solution, can then be evaluated as:

$$c_{(i,j)} = \sum_{(u,v) \in \Gamma} f_{iu} d_{jv}. \tag{2}$$

At each step, these values are used to assign a higher probability of being selected to the assignments that yield a smaller cost. After a feasible permutation has been constructed ($I = J = \phi$), the solution is submitted to a local search procedure. A more detailed description of this method, including implementation details, has been previously reported.[13, 16]

The constructive phase of GRASP constitutes a greedy algorithm in which the evaluations from (2) are updated in order to incorporate the new information that is gathered after each assignment is made. The randomized selection rule for assignments causes different passes to construct different starting points that can be submitted in turn to the local search procedure. Although GRASP characteristically provides better solutions than simple greedy algorithms or Random Restart methods, there is still a lot of room for strategic improvement based on alternative designs for generating the starting points, as we show in the next section.

2. Intensification Strategies

The Random Restart and GRASP methods are multistart approaches that do not take advantage of previously generated solutions. However, a multistart approach that does

this can easily be created by applying well established principles. In this article, we examine a simple approach based on the identification of strongly determined and consistent variables, as proposed by Glover.[8] Originally, the identification of such variables was predicated on differentiating among multiple solution sets by means of clustering, but we will restrict attention to implementing the main ideas in more elementary ways. Strongly determined variables are customarily defined, relative to specified elite solutions, as variables whose values cannot be changed without significantly eroding the objective function value or disrupting the values of other variables. A consistent variable is defined as one that receives a particular value in a significant portion of good solutions. These rather loose definitions, which provide considerable latitude for strategic variation, are readily made specific to yield particular instances of intensification procedures.

Measures of consistency in variables are often exploited by frequency based memory structures in tabu search.[10] While such approaches are most commonly applied to modify trajectories of local search, they can also be used to guide constructive phases of multistart methods, as in variations initially proposed by Glover.[8] In this application, we introduce a set S of previously generated elite solutions that will be used to guide a constructive procedure. The template for our multistart method is thus essentially the same as used in other common multistart designs, except that each complete solution generated becomes a candidate to be stored in a set we denote by S. Our constructive procedure is also adapted to incorporate useful information from S.

Our goal in designing the memory based constructive procedure is to assemble only the most rudimentary strategic elements, in order to provide a rough determination of the potential gains that such a procedure may afford. We choose S to consist of r elite solutions and start with S composed of r copies of the "null" solution with a cost of infinity. Let Cost(s) denote the cost of a solution s, and define worst(S) as the solution with the worst (max) cost in S, and best(S) as the solution with the best (min) cost in S. (Until S contains r "real" solutions, worst(S) is a null solution with an infinite cost.)

When a solution s with Cost(s) < Cost(worst(S)) is generated, it is a candidate to be added to S. It replaces worst(S) with s if s is sufficiently different from the already non-null elements of S. To measure the "closeness" of solution s to the other solutions of S, we count the number of identical positions in the permutations. For instance, if $|\{i \text{ such that } s[i] = s'[i]\}| > d \cdot n$, for a solution $s' \in S$, the solution is discarded and not added to S unless an aspiration criterion is satisfied, i.e. Cost(s') < Cost(best(S)). In our experiments, the parameter value of $d = 0.5$ was used and simply represents the proportion of assignments that have to differ in order for the solution to be added to S. These simple mechanisms are introduced in order to maintain a reasonably diverse set of elite solutions, and of course a variety of more sophisticated alternatives are possible.

2.1 A Constructive Procedure with Intensification

The remaining ingredient is to define the constructive procedure to be used. An intensification strategy based on strongly determined and consistent variables seeks to bias ordinary evaluations to reinforce the choice of such variables, and we can do this in general as follows.

2.1.1 Defining Evaluations for Constructing Solutions Relative to S

Following the notation of Section 1, we build a solution s with a sequence of assignments $s[i] = j$, until a complete permutation has been generated. Let value(i, j) be a customary (objective function) evaluation to select (i, j) to augment the current solution, where larger values of value(i, j) correspond to better choices. (Thus, value(i, j) increases as the change in cost decreases.) Let intensity(i, j) be a measure of the strongly determined and consistent features of the choice (i, j); i.e. intensity(i, j) becomes larger as (i, j) occurs more frequently in the best members of S.

In our constructive procedure, the actual evaluation of assigning j to $s[i]$ for an intensification strategy may be given the form

$$E(i, j) = F(\text{value}(i, j), \text{intensity}(i, j)), \tag{3}$$

where F is a monotone increasing function of its arguments. In Section 2.1.3 we give simple examples to define value(i, j), intensity(i, j) and the function F.

2.1.2 Applying the Evaluations $E(i, j)$ Probabilistically

Using the evaluation $E(i, j)$, we apply the strategy of probabilistic tabu search (PTS) by mapping F into nonnegative values, some positive, over a candidate list C of (i, j) choices. Then we establish the probability for selecting (i, j) from C, denoted $p_{(i,j)}$, to be strongly biased toward choosing the members of C with larger $E(i, j)$ values. By restricting consideration to a subset C_{best} of the best t choices, we can simply choose

$$p(i, j) = \frac{E(i, j)}{\sum_{(k,l) \in C_{best}} E(k, l)}. \tag{4}$$

To keep the intensity values relevant, we specify that the candidate list C contains all (i, j) from those currently legitimate to belong to solutions $s \in S$. (If S is large, this assumption can be relaxed, but we normally intend for r, the number of solutions in S, to be relatively small.)

In our constructive procedure, the candidate list is constructed by randomly selecting a subset $I_p \subset I$ of unassigned facilities. In our experiments, a subset I_p of size $0.2n$ was selected. For every facility $i \in I_p$, each possible assignment is then added to the candidate list C. The t best candidates in C are then added to C_{best} from which a random selection is performed according to probabilities defined by (4). We emphasize that strategically designed candidate lists often yield better performance outcomes than those based on randomization, but we apply the indicated construction for convenience of implementation and to facilitate more direct comparison with the Random Restart and GRASP approaches.

2.1.3 A Simple Determination of $E(i, j)$

For the QAP, where all cost changes are positive during the constructive process, we let best_cost(C) = $\min(c_{(i,j)}: (i, j) \in$

C), where $c_{(i,j)}$ is computed according to (2), and define

$$\text{value}(i, j) = \frac{\text{best_cost}(C)}{c_{(i,j)}}. \quad (5)$$

Note that we keep the same relative ordering by replacing best_cost(C) with 1 in the foregoing definition, i.e., best_cost(C) is used as a normalization factor that will be convenient for later concerns.

Similarly, for the purpose of defining intensity(i, j), we first identify the value of a solution $s \in S$ by letting Best_Cost(S) = min($Cost(s) : s \in S$) and defining

$$\text{Value}(s) = \frac{\text{Best_Cost}(S)}{\text{Cost}(s)}. \quad (6)$$

This value is a crude measure of the "strength" of elements (i, j) in s. To take account of the frequency that (i, j) lies in solutions of S, as well as the strength, we therefore define

$$\text{intensity}(i, j) = \sum_{s \in S : (i,j) \in s} \text{Value}(s) \quad (7)$$

Then to let $E(i, j)$ be an increasing function of its arguments, we define

$$E(i, j) = \lambda \cdot \text{value}(i, j) + \text{intensity}(i, j). \quad (8)$$

2.1.4 Dependency of $E(i, j)$ on Time

We seek to increase the intensification factor over time. Due to the normalizing influence of the numerators defining value(i, j) and Value(s), both of these values will be at most 1. Hence intensity(i, j) will be at most r, and generally somewhat smaller than r. By implication, letting $\lambda = r$ will roughly give value(i, j) about the same emphasis as intensity(i, j). In early stages, before r "real" solutions are generated (and we ignore "null" elements of S), setting a larger value of λ will give somewhat more emphasis to value(i, j) than to intensity(i, j). This is desirable, since the "relative goodness" of solutions in S is not yet known. Then λ can be decreased to smaller values.

Since intensification somewhat emphasizes solution quality at the expense of randomness, it is usually coupled with a diversification component that periodically encourage the search to explore different regions of the search space. In our case, diversification can be achieved by increasing λ, which we do when the diversity of generated solutions is too low. This aspect, as well as other rules based on strategic oscillation for manipulating λ will be discussed in more detail in Section 4.1.

2.1.5 Aspiration Criterion

Probabilistic tabu search always permits the assignment of probabilities to be overruled by an aspiration criterion that identifies when a choice should be gauged especially attractive. In a construction process, when only a few choices remain, a reasonable aspiration criterion results in selecting those with highest value(i, j) values, without reference either to the intensity(i, j) values or to probabilities, since at this point value(i, j) is a strong indicator of the likely quality of the solution that will be finally constructed. Thus, in our approach we simply base the choice on the largest value(i, j) (smallest cost(i, j)) when 3 or fewer choices remain.

3. Proximate Optimality Principle (POP)

The Proximate Optimality Principle (POP) can be roughly expressed by saying that good solutions at one level are likely to be found "close to" good solutions at an adjacent level.[10] (In general, the challenge is to specify definitions of levels, and mechanisms for moving across and within them, so that the principle applies. Here, we rely on the definitions implicit in the context of the moves currently employed.) An important aspect of this idea is the intuition that it can be highly worthwhile in a constructive or destructive process to seek improvements at a given level before going on to the next level.

The basis for this intuition is as follows. Moves that "come into" a given level from another are based on an inadequate picture of tradeoffs and interactions at the level entered. Consequently, in a constructive approach (for example) features can become incorporated into the construction that introduce undesirable sub-assemblies—that is, sub-assemblies that are incompatible with the goal of constructing a high quality solution. Moreover, if these are not remedied, they can build on themselves (since each level sets the stage for the next), so that the solution under construction acquires a composition that is increasingly at variance with the composition of a good solution. Stated differently, a wrong move at one level changes the identity of moves that look attractive at the next level, because the construction now seeks to choose moves that are "decent" given that the wrong move has been made. Consequently, there will be a tendency to make additional wrong moves, each one reinforcing the earlier wrong moves, so that they create a mutually reinforcing web. Eventually, several levels later, this web may be so interdependent that there is no way to extract or "undo" the earlier wrong choices, except by a series of changes whose initiation appears highly nonproductive. As a result, even the later application of an improvement method may find it very hard to correct for the bad decisions previously made. We should point out there is also potentially a reverse danger, of becoming "too suboptimized" for a given level. However, the risk of becoming too suboptimized is probably a lot less important than the risk of not trying to improve the solution at intermediate stages.

In our chosen application, one way to apply the POP notion is as follows. Suppose that after l assignments in the constructive process, the sets $\bar{I}$ and $\bar{J}$ contain the assigned facilities and sites to this point. These assignments then define a restricted QAP problem of size l of the form

$$\min_{\phi \in P(l)} \sum_{i \in \bar{I}} \sum_{j \in \bar{J}} d_{\phi(i)\phi(j)}, \quad (9)$$

where $P(l)$ is the set of all permutations of $\{1, \ldots, l\}$. By applying a local search procedure to the reduced QAP (9), it is possible to somewhat reevaluate the assignments that were previously selected in the constructive phase. According to the POP principle, the suboptimized sequence would tend to offer more attractive alternatives for the remainder of the constructive process.

It is also possible to apply the POP notion by oscillating between constructive and destructive moves in the construc-

1. Initialize the set S with r NULL elements.
   $Best_cost = \infty$.
2. Constructive phase.
   (i) Start with $I = J = \{1, 2, ..., n\}$.
   (ii) Select $i^* \in I$, $j^* \in J$ according to probabilities (4).
   Update structures: $s[i^*] = j^*$; $I = I - \{i^*\}$; $J = J - \{j^*\}$.
   (iii) Apply POP mechanisms (see Sections 3 and 4.1).
   (iv) If I and J are not empty go to (ii).
3. Apply local search to s.
4. Update set S (see Section 2).
5. If $Cost(s) < Best_cost$
   $s^* = s$.
6. If number of generated solutions $< Max_iter$ Goto 2.
   Otherwise return s^*.

Figure 1. Summary of the multistart method with intensification.

tion phase. "Destructive moves" are then used to dismiss some of the previously made assignments. For instance, if sets $\bar{I}$ and $\bar{J}$ contain the facilities and sites assigned in a partial solution, the cost induced by each assignment can be computed according to

$$C_{is[i]} = \sum_{(u,v)\in(\bar{I}-\{i\})\times(\bar{J}-\{s[i]\})} f_{iu}d_{s[i]v}, \ \forall i \in \bar{I} \qquad (10)$$

By periodically removing some of the most costly assignments, we can expect better partial constructions, with the hope of opening more interesting choices for the next levels.

Adding these POP features (local search on partial solutions, destructive moves) may increase the computational load of the methods. However, local searches are applied to smaller problem instances and destructive moves are selected among a much smaller set of possibilities than constructive moves. Furthermore, the extra time used by these features can be recovered by appropriate application of other practices, such as the restricted candidate list strategies discussed in Section 2.1.2.

With all the features described in Sections 2 and 3, our multistart method with intensification is summarized in Figure 1. It is important to note that probabilities (4) in step 2(ii) are influenced by set S (see Section 2.1).

4. Numerical Results

In this section, we report numerical experiments performed on the set of test problems QAPLIB.[2] Our strategies are compared to the GRASP implementation that is described previously[13, 16] (we are indebted to the authors for providing their code to make the tests). In order to illustrate the effect of our strategies graphically, some results are presented in Section 4.2 for a single problem (the first Steinberg problem of size 36). Then, in Section 4.3, results are presented for to the rest of the QAPLIB problems.

4.1 Parameters

Various parameters must be set with the methods described in this article. Some of them have to be fixed, while others can be adaptively adjusted during the execution of the algorithms. For GRASP, the parameter settings are the same ones used by Resende, Pardalos, and Li.[16] To provide a common base for comparison, Random Restart and our multistart strategy with intensification generate the same total number of solutions as used by Resende, Pardalos, and Li[16] (2048 solutions per trial). Other parameter settings are $r = 20$ (size of S, the set of elite solutions), and $t = 20$ (size of candidate list C_{best} for assignments). These parameters were the same for all test problems and were chosen because of their robustness. Further testing and research could lead to improvements.

Table I. Parameter Values for the POP Factor

No. of Generated Solutions	par_1	par_2
$0 \leq i \leq 200$	$\lfloor n/6 \rfloor$	2
$200 \leq i \leq 400$	$\lfloor n/5 \rfloor$	3
$400 \leq i \leq 600$	$\lfloor n/4 \rfloor$	4
$600 \leq i \leq 800$	$\lfloor n/3 \rfloor$	5
$800 \leq i \leq 1000$	$\lfloor n/2 \rfloor$	6
$i \geq 1000$	$\lfloor n - 5 \rfloor$	10

The parameter λ is dynamically adjusted according to the diversity of the last 100 generated solutions, as computed by the equation

$$e = \frac{-\sum_{i=1}^{n}\sum_{j=1}^{n}(n_{ij}/100)\log(n_{ij}/100)}{n\log(n)}, \qquad (11)$$

where n_{ij} represents the number of times object i is assigned to site j in the last 100 generated solutions. This entropy measure (see Fleurent and Ferland[7]) gives a value in [0, 1], where $e = 0$ corresponds to 100 identical individuals and $e \approx 1$ indicates an almost uniform distribution of all possible assignments. λ is initially set to $100r$ for the first 100 generated solutions, therefore relying mainly on the cost factors. For the remaining iterations, λ is first set to $10r$, and reevaluated every 100 iterations so as to give more or less influence to the intensity factor. If the entropy value $e > 0.5$, we set $\lambda = \lambda - r$ for the next 100 iterations. Conversely, if $e < 0.3$, we set $\lambda = \lambda + r$.

The influence of the POP factor is also varied over time. It is used more often in early passes, and its influence decays to give more freedom to the intensification factors in the later stages. If we define **POP1** as local search on a partially constructed solution and **POP2** as removing an assignment from a partial solution, parameters par_1 and par_2 determine the intervals between calls to **POP1** and **POP2**. For instance, $par_1 = 5$ means that a restricted local search is applied each time a partial solution reaches sizes 5, 10, 15, ..., n. Similarly, an assignment is dismissed after each par_2 constructive assignments. Table I gives the parameter values as a function of the number of generated solutions i.

4.2 Testing of Different Variants

In Figure 2, GRASP is compared with a multistart strategy with intensification (without POP). In this figure, the objec-

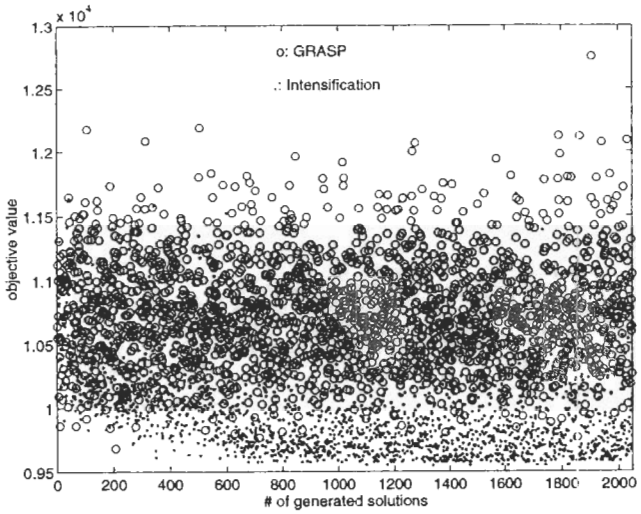

Figure 2. GRASP and Constructive multistart (Intensify) for the first Steinberg problem of size 36.

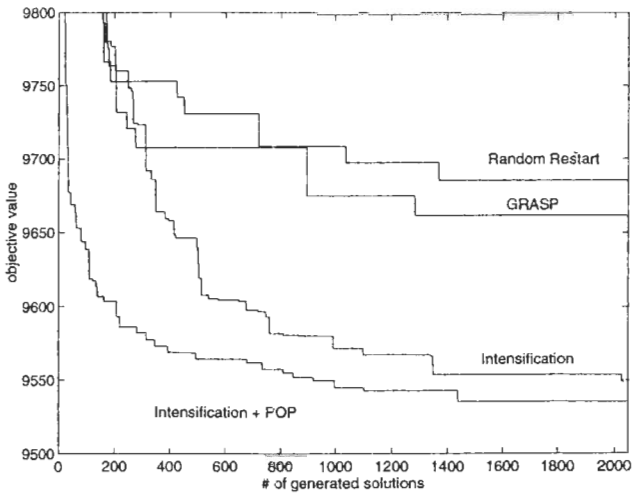

Figure 3. Best found solution as a function of the number of generated solutions for the first Steinberg problem of size 36 (average of 5 runs).

tive value of solutions generated by GRASP are plotted as open circles and the solutions of our multistart method are plotted as dots. For each method, 2048 solutions were generated and are displayed from left to right in the chronological order they were produced. At first glance, the figure may look a little dense, but a closer examination clearly shows distinct behaviors for the two methods. Since both approaches rely on probabilistic constructive phases, they produce samples of solutions of diverse quality. However, as the figure indicates, the introduction of information gathered from previous solutions significantly improves the search by comparison to the results obtained from GRASP. It is also possible to detect a learning pattern as the intensification strategy identifies consistent values for variables from previous elite solutions.

The POP principle can significantly accelerate the search to uncover high-quality solutions. This is illustrated in Figure 3 in which the best found solution is plotted as a function of the number of generated solutions for different methods (average of 5 runs). In this figure, we see that the Constructive multistart (Intensify) strategy that is also enhanced by the POP components described in Section 3 gives the best results. Because the method now generates better solutions for the elite set S in the early phases, the method converges faster.

4.3 QAP Benchmarks

In the previous subsection, we concentrated on a single problem instance (Steinberg, $n = 36$) to illustrate graphically the beneficial effect of different search strategies. The improved procedures also prove to consistently outperform GRASP for other test problems. Resende, Pardalos, and Li[16] describe QAPLIB problem instances that are divided into three distinct sets according to their size. Problems of size $n \leq 16$ are already easily solved to optimality for almost all cases so we do not report results for these. For all other symmetric problems of size n, $18 \leq n \leq 150$, we have run

Table II. Results for Symmetric QAPLIB Problems of Size n, $18 \leq n \leq 150$

	GRASP	Intensify + POP	Tie
Best solution	0	28	17
Average solution	2	32	11

five independent runs of both GRASP and our multistart strategy improved by the intensification and POP mechanisms. Again, parameters were set as described in Section 4.1, and each of the five runs of both methods generated 2048 solutions.

For each problem (45 problem instances in total), we computed the average and best solution out of the 5 independent runs for both methods. We then identified for each problem which of the two methods gave the best result. When both methods gave the same average or best solution, it was recorded as a tie. Results are summarized in Table II. We can see that GRASP produced better average solutions for only two problems out of the 45, and that it never beat our improved procedure when the best generated solutions for both methods were compared. This method to compare different heuristic methods is often used to statistically rank the performance of different algorithms on a sample of problems. In this case, the intensification strategy with POP clearly dominates the GRASP procedure.

As illustrated in Figure 4, the running times for the different methods are similar. Obviously, Random Restart is the fastest approach since its construction phase is trivial. Relative to GRASP, our constructive multistart with intensification and POP must include more operations to compute and store consistency measures, but the extra time is offset by the use of a restricted candidate list.

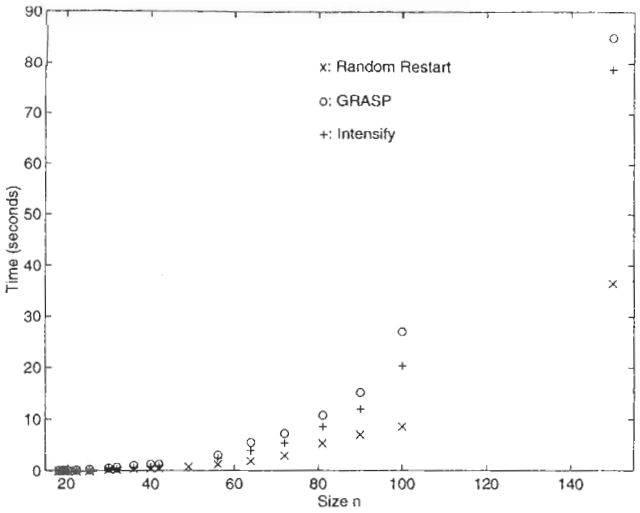

Figure 4. Average running time to generate one solution for the different methods (time in seconds on a SPARC 20 processor).

5. Concluding Remarks

In this article, we have shown that constructive multistart methods, such as Random Restart and GRASP, can be improved by the addition of memory and associated heuristic search principles, as proposed in tabu search. The improved results show that these principles (learning, intensification, candidate list strategies, POP) are not limited to application with transition neighborhoods, as in local search, but are also useful for application with constructive (and destructive) neighborhoods.

As our computational results indicate, the memory based strategies allow constructive procedures to obtain excellent solutions for problems of moderate size. For the QAP, however, we find that the inclusion of adaptive memory strategies within transition neighborhoods of local search phases[1, 3, 7] yield methods that still clearly outperform all of the currently existing approaches that seek only to include such strategies during the constructive phases. It is an open question whether constructive schemes can be further improved in the future to reduce this performance gap. Clearly, the coordination of memory strategies throughout all solution phases, involving both constructive and transition neighborhoods, invites examination. In our opinion, further research is likely to yield particular benefits by seeking ways to maintain a balance between reinforcing good solution features (intensification) and driving the search to explore new areas of the search space (diversification). As noted by Glover and Laguna,[10] these two search components stand to be harmonized in a variety of ways that are often incompletely considered in present implementations.

References

1. R. BATTITI and G. TECCHIOLLI, 1994. The Reactive Tabu Search, *ORSA Journal on Computing 6*, 126–140.
2. R.E. BURKARD, S.E. KARISH, and F. RENDL, 1994. QAPLIB—A Quadratic Assignment Problem Library, *www.opt.math.tugraz.at/~karisch/qaplib*.
3. J. CHAPRAKANI and J. SKORIN-KAPOV, 1993. Massively Parallel Tabu Search for the Quadratic Assignment Problem, *Annals of Operations Research 41*, 327–341.
4. M. DELL'AMICO and M. TRUBIAN. Solution of Large Weighted Equicut Problems, *European Journal of Operational Research*, to appear.
5. T.A. FEO, M.G.C. RESENDE, and S. SMITH. A Greedy Randomized Adaptive Search Procedure for the Maximum Independent Set Problem, *Operations Research*, to appear.
6. T.A. FEO and M.G.C. RESENDE, 1994. Greedy Randomized Adaptive Search Procedures, *Technical report*, AT&T Labs, New Jersey.
7. C. FLEURENT and J.A. FERLAND, 1994. Genetic Hybrids for the Quadratic Assignment Problem, *DIMACS Series in Discrete Mathematics and Theoretical Computer Science 16*, 173–188.
8. F. GLOVER, 1977. Heuristics for Integer Programming Using Surrogate Constraints, *Decision Sciences 8*, 156–166.
9. F. GLOVER, 1986. Future Paths for Integer Programming and Links to Artificial Intelligence, *Computer and Operations Research 13*, 533–549.
10. F. GLOVER and M. LAGUNA, 1997. *Tabu Search*, Kluwer Academic Publishers. Hingham, MA.
11. F. GLOVER and A. LOKKETANGEN. Probabilistic Tabu Search for Zero-One Mixed Integer Programming Problems, *Technical report*, University of Colorado at Boulder.
12. S. KIRKPATRICK, C.D. GELATT, and M.P. VECCHI, 1983. Optimization by Simulated Annealing, *Science 220*, 671–680.
13. Y. LI, P.M. PARDALOS, and M.G.C. RESENDE, 1994. A Greedy Randomized Adaptive Search Procedure for the Quadratic Assignment Problem, *DIMACS Series in Discrete Mathematics and Theoretical Computer Science 16*, 237–261.
14. P.M. PARDALOS, F. RENDL, and H. WOLKOWICZ, 1994. The Quadratic Assignment Problem: A Survey and Recent Developments, *DIMACS Series in Discrete Mathematics and Theoretical Computer Science 16*, 1–42.
15. M.G.C. RESENDE and T.A. FEO, 1996. A GRASP For Satisfiability, *DIMACS Series in Discrete Mathematics and Theoretical Computer Science 26*, 499–520.
16. M.G.C. RESENDE, P.M. PARDALOS, and Y. LI, 1994. Fortran Subroutines for Approximate Solution of Dense Quadratic Assignment Problems Using GRASP, *Technical report*, AT&T Labs, New Jersey.
17. E.D. TAILLARD. Comparison of Iterative Searches for the Quadratic Assignment Problem, *Publication CRT-989*, Centre de recherche sur les transports, Montreal.