Sequential and Parallel Path-Relinking Algorithms for the Quadratic Assignment Problem

Tabitha James[a,], Cesar Rego[b], and Fred Glover[c]

a Department of Business Information Technology, Pamplin College of Business, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061, USA. tajames@vt.edu

b School of Business Administration, University of Mississippi, University, MS 38677, USA. crego@bus.olemiss.edu

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

Latest Revision: May 8, 2005.

Abstract – The quadratic assignment problem is a classical combinatorial optimization problem that has garnered much attention due to both its large number of applications and its solution complexity. Originally used to model a location problem in the 1950's, the QAP is computationally very difficult to solve which makes it an ideal candidate for metaheuristic approaches. Path-Relinking (PR) is an evolutionary metaheuristic based on maintaining and exploiting search information by drawing on principles shared in common with tabu search. This paper proposes and implements a design for both a sequential and a parallel path-relinking algorithm tailored for the quadratic assignment problem. We illustrate the potential strength of this solution methodology by developing a very simple PR approach for a classic problem with wide applicability and exploiting the inherent parallelism of the algorithm to reduce total computational time. Results are presented on a set of problems obtained from QAPLIB for both the sequential and parallel versions of the algorithm in order to demonstrate the benefits of the parallelization. Comparisons of longer runs of the parallel algorithm are also given against popular solution approaches from the literature. In spite of the simplicity of the underlying PR method, we obtain results that are competitive with some of the best outcomes in the literature.

Keywords: Path-relinking, tabu search, combinatorial optimization, quadratic assignment problem, parallel computing.

1. Introduction

The quadratic assignment problem was first introduced to model a location problem by Koopmans and Beckmann (1957). While facility-location problems remain the most popular application area for the quadratic assignment problem, many other applications for this problem exist including scheduling problems, statistical data analysis, information retrieval, as well as problems in transportation. The attractiveness of the QAP is also due to the fact that many other combinatorial optimization problems can be formulated as a QAP, including: the traveling salesman problem, the maximum clique problem and the graph partitioning problem. (See Cela (1998) for a survey of both classical and practical applications.) In the context of facility location problems, the objective is to find a minimum cost assignment of facilities to locations considering both the flow of materials between facilities and the distance between locations. The QAP can be formulated as follows:

(QAP)

$$\min_{p \in \Pi} \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} f_{p(i)p(j)}$$

where f is the flow matrix, d is the distance matrix, and p is an assignment vector.

The principles of scatter search (SS) and its generalization, path-relinking (PR), have their origins in surrogate constraint strategies, by replacing the goal of combining solution constraints with the goal of combining solution vectors, while maintaining a focus on yielding and utilizing information not contained solely in the original elements independent of their combination. Similarly, as in the case of surrogate constraint methods, SS/PR is organized to derive benefit from the use of associated heuristic processes to help improve, evaluate, and generate new solutions based on information embodied in the structure of the combined elements (Glover, 1968, 1977, 1998). Although scatter search falls into the category of evolutionary algorithms, it encompasses features of the adaptive memory framework of tabu search, as a result of sharing common origins, and thus places particular emphasis on exploiting memory and associated strategies of intensification and diversification.

Relying on the same principles, path-relinking generalizes scatter search by replacing the Euclidian space by the Neighborhood space (as defined in local search) providing a framework for local search algorithms to explore adaptive memory in an evolutionary fashion. Because tabu search is typically designed to explore the neighborhood space of individual solutions, coupling tabu search with path-relinking is a natural marriage for creating effective adaptive memory neighborhood-based evolutionary approaches. The RAMP approach recently proposed by Rego (2005) closes the loop between surrogate constraints, scatter search, path-relinking, and tabu search by providing an unified framework for the creation of dual and primal-dual algorithms that take full advantage of adaptive memory programming, yielding highly promising results.

These characteristics allow for a guided exploration of the solution space based upon information obtained during the search process, substantially reducing the recourse to randomized processes that lie at the core of other evolutionary approaches. Such a design is especially conducive to creating customized procedures to take advantage of special problem structures evidenced by problems from specific classes.

Scatter search and path-relinking have been applied to a variety of problem areas, including vehicle routing, optimizing simulation, linear ordering, and job shop scheduling. See Laguna and Marti (2003) and Rego and Alidaee (2005), for a survey of algorithm designs and applications. An approach applying SS to the quadratic assignment problem that differs somewhat from the approach of this paper has been undertaken with success by Cung, et al. (1996).

This paper describes the design of both sequential and parallel path relinking algorithms for the quadratic assignment problem. This research represents the first use of parallelization for path relinking within the QAP setting. We make use of a very simple form of PR in order to focus on the elements of the parallel implementations and to determine their impact when used with a method of this type. Computational results are reported for a selected group of QAP test problems from QAPLIB, demonstrating highly attractive outcomes in spite of the simplicity of the PR procedure, and showing in particular the value of a well-designed parallelization process in this context.

2. Sequential SS/PR Algorithm

2.1 Scatter Search Overview

The template for the scatter search/path-relinking algorithm given in Glover (1998) provides a basic skeleton for the algorithm. This outline of the base algorithm delineates four basic steps for the path-relinking algorithm:

1. Generate a set of starting solutions and initiate the reference set to contain a predetermined number of the best quality solutions and also a predetermined set of the most diverse solutions. The generation of the initial solutions is typically accomplished by an intelligent procedure that uses some knowledge of the problem type to generate good starting solutions. Often a specialized heuristic method is applied to these initial generated solutions to improve their quality. These good starting solutions and their addition to the reference set are not based solely on the objective function evaluation of the solution. As mentioned above, a certain amount of "diverse" solutions are also added which broaden the search space and help keep the algorithm from converging to a local optima.

2. Use a structured combination method to create new solutions from the solutions contained in the reference set. The structured combination method uses information collected during the search and known characteristics of the problem type to intelligently derive new solutions.

3. An improvement heuristic, typically the same procedure as applied in step one, is performed on the newly generated solution from step two.

4. Attempt to add the solutions generated by steps two and three to the reference set. Here, once again, solutions are added based on their contribution to the entire search. This contribution may be to improve the solution quality or to add diversity to the search. The search iterates from step two unless the reference set does not change, in which case a diversification procedure the same or similar to the one used to initiate the reference set may be called to update the reference set. The algorithm is terminated when either a maximum number of iterations are reached or the reference set does not change for a given amount of time.

The template can be further described by detailing a specific set of methods that are used to implement a basic SS/PR algorithm. These methods are as follows:

A. *Diversification Generation Method* – This method is applied to generate a set of solutions that give a reasonable representation of the entire search space for the problem. The method applied is typically dependent upon the problem type. Thus, it is tailored to generate solutions based upon some knowledge of the problem type.

B. *Improvement Method* – This method applies a problem-specific heuristic with the intention of creating a better solution in terms of the objective function evaluation.

C. *Reference Set Update Method* – This method is used to maintain the reference set. It is also used to build the initial reference set. This method determines which of the solutions generated by the diversification generation method and the improvement method meet the entry requirements for the reference set based upon its contribution to the diversity of the search or its solution quality.

D. *Subset Generation Method* – This method creates subsets of solutions which are then combined to create new solutions which may possibly be added to the reference set. Subset of two or more solutions may be created during this procedure. This method uses information about the quality or diversity of the solutions contained in the reference set to create these subsets.

E. *Structured Combination Method* – This method creates a new solution(s) from the subsets created by the subset generation method. This procedure uses information available about the characteristics of the problem as a basis for creating the new solutions. Therefore, this method is also typically customized for the particular problem type being investigated.

The algorithm developed for this study is based upon the general skeleton provided above. Code obtained from Glover, Laguna, and Marti (2001) was converted from the C language to Fortran 90 and the methods given above were modified for the quadratic assignment problem. The rest of this section will describe how the methods outlined above were implemented for the algorithm used in this study. An outline of the algorithm is given in Figure 1. The structure of the sequential version of the algorithm is the same as the parallel implementation with the exception of the concurrent runs of the improvement method.

2.2 Initializing Diversification Method

The representation strategy used in this algorithm identifies the possible locations by the integers 1,..., n and depicts a solution vector in the form:

$$x(h) = (10, 1, 3, 5, 6, 2, 12, 8, 4, 7, 9, 11) \text{ where } h = 1, n.$$

In this example, there are 12 firms and 12 locations, each firm 1…12 is assigned to each location represented by array locations 1…12.

To initialize the reference sets, a diversification generation method, suggested in Glover (1998) was used. This method strategically generates an initial population of solution vectors from which to choose the initial reference sets. Using this method, along with the properties of the reference sets maintained, provide a guaranteed level of diversity over the search space that cannot be obtained from a more random approach. The method does not produce infeasible solutions, so all solutions are possible candidates for the reference set without the need to repair the generated solution vectors.

The approach begins from a randomly generated, feasible seed solution. The candidate solutions for the reference sets are then built from this seed solution. Once the seed solution is obtained, the method generates a new solution from the seed solution as follows. A step, *b*, is defined that is a positive integer value less than *n*. The starting position, *s*, in the seed permutation is initialized to be the step and the new vector is built by first adding the element *x(s)* to the new permutation, followed by the element $x(s+b)$, $x(s+2b)$, $x(s+3b)\ldots x(s+rb)$ where $s+rb$ does not exceed *n*. Once the end of the seed solution is reached, the starting position becomes $x(s\text{-}1)$ and the process iterates until all elements from the seed solution are present in the candidate solution. The process can be illustrated as follows. Working with the solution vector shown above and a step of 2, the start position would be initialized to $x(2)$, therefore the first element added to the new vector would be 1. The starting position would then be adjusted to be x(4), therefore causing the next element added to be 5 and so on. After the first pass through the seed solution, the new vector obtained would be:

$$x(h) = (1, 5, 2, 8, 7, 11, \_, \_, \_, \_, \_, \_)$$

The starting position would then be adjusted to be one less than the current starting position, so in this example $x(1)$. The second pass starting from $x(1)$ would generate a complete vector as shown below and since all elements have been added, the method would terminate.

$$x(h) = (1, 5, 2, 8, 7, 11, 10, 3, 6, 12, 4, 9)$$

The implementation used in the current algorithm uses this method to generate an initial population, *psize*, of candidate solutions based off of, *m*, randomly generated, feasible seed solutions. The step is defined in this algorithm to be a randomly drawn integer between 1 and $n/2$. In the current version of the algorithm, *psize* = 100 and *m* = 10. Ten candidate solutions are then generated from each seed solution using this diversification generation method.

The initial reference set was then built by adding the best, b_1, unique solutions defined by objective function value and the most diverse, b_2, unique solutions defined by the maximum distance from the best b_1 solutions (a discussion of maximum distance is given below) from the population to the reference set.

2.3 Improvement Method

The procedure we used as an improvement heuristic for our approach is the tabu search method for the quadratic assignment problem developed by Taillard (1991, 1995). Taillard's algorithm works by evaluating possible exchanges of two firms. The algorithm is notable for its ability to evaluate many possible moves relatively quickly, utilizing a very simple tabu list and aspiration criterion. The tabu list prohibits the exchange of two values that have been exchanged in the past several moves, to force the exploration of less attractive areas of the search space. The aspiration criterion allows moves that are tabu if that move generates a solution that is better than the best-known solution found at that point in the search. The code for this procedure was obtained from Taillard and converted from Pascal to Fortran 90.

2.4 Reference Set Update Method

The reference set update method for the current algorithm maintains a set R_1 of b_1 high quality solutions and a set R_2 of b_2 diverse solutions. Once the reference set has been initialized, a candidate solution is added to the reference set only if it is better than the worst quality solution currently in the reference set or more diverse than the least diverse solution currently in the reference set. Duplicate solution vectors are not allowed in the current implementation.

The high quality solutions are measured in terms of their objective function evaluation. Let *C* denote the set of candidate solutions. Since the objective in this case is to minimize the objective function, a candidate solution, $x \in C$, is added to the reference set only if its objective function value is better than the solution currently in the reference set with the worst evaluation. Duplicate solution vectors are not allowed, so if the candidate solution being tested for addition to the reference set is the same as a solution vector currently contained in the reference set, R_1 or R_2 respectively, it is not added even if its objective function value is less than one of the solutions currently in the reference set.

The diversity of a candidate solution is arrived at by calculating the distance between it and all the best quality solutions. The distance of a candidate solution, $x \in C$, and a solution vector, $x^1 \in R_1$, is computed by:

$$d(x,x^1) = \sum_{i=1}^{n} | x - x^1 |$$

$$D(x,R_1) = \sum (d(x,x^1) : x^1 \in R_1)$$

The solution, x, is then added to the reference set if $D(x, R_1) > argmin(D(x,R_1): x \in R_2)$, provided that x'' does not duplicate an element of R_2. R_2 is initialized by choosing the b_2 solutions based on $x''=argmax(D(x, R_1) : x \in C)$ and x″ does not duplicate any element in R_2.

2.5 Subset Generation Method

Two-element subsets are generated by the subset generation method for this algorithm. The two element subsets correspond to the Type 1 subsets from Glover (1998), which contain all unique two-element combinations of the solutions in the reference set.

2.6 Path Relinking Method

The PR method employed in this algorithm is a simple first-level approximation of the adaptive structured combination procedure described in Glover (1991). For every pair of solutions, the better quality solution is designated the guiding vector. In the following illustration, *x*(1) is assumed to have a better function evaluation.

$x(1) = (8, 3, 4, 2, 1, 7, 5, 6)$ guiding vector (better parent)
$x(2) = (5, 3, 2, 1, 8, 7, 4, 6)$

The algorithm then considers the first element of *x*(2). In this example, the first element of *x*(2) does not correspond to the first element of *x*(1), so the exchange to be considered becomes (5,8) in *x*(2). This would move 8 into the first position of *x*(2), which would correspond to the assignment in *x*(1). If this exchange does not degrade the quality of the objective function, the exchange is made, otherwise *x*(2) remains unchanged and the next position in the permutation is considered. Assuming that this exchange does not degrade the evaluation, the child vector in this example becomes:

$c(1) = (8, 3, 2, 1, 5, 7, 4, 6)$

The next element in both permutations is the same, so in this example no exchange would be considered. The next exchange considered thus becomes (2,4). If this exchange does not degrade the evaluation and moving that facility has not been previously considered, then the exchange would be made. In this illustration, assuming that (2,4) degrades the quality of the solution, $c(1)$ would remain unchanged. The fourth element of $x(1)$ is a 2. However, this element has already been considered in a previous exchange and since the algorithm does not consider backward exchanges, this element too would remain unchanged. The algorithm then proceeds in this manner until the end of the array is reached.

This combination method uses the guiding vector to restrict the exchanges considered and to show preference to assignments that appear in the better quality solution. The method is restrictive in that it does not consider backwards exchanges. That is, if an element of the array was not moved in the consideration of a previous exchange, it remains in the same position it was located in the original vector in the child. The algorithm always makes the first non-degrading exchange encountered instead of seeking the best possible exchange. The path relinking approach used allows 0 cost moves, but does not allow degrading moves (moves that would increase the value of the objection function). Allowing 0 cost moves to be made in this rudimentary path relinking approach improved the solution quality obtained by the algorithm. This suggests that path relinking may be a useful method to find a trajectory through a space of non-improving moves to find new local optima. This finding may be relevant to research in solving other classes of problems, particularly satisfiability problems where there are often many local optima attended by the presence of 0 cost moves.

2.7 Diversification Method

This method is applied only if the diverse solutions in the reference set are not updated in the previous iteration of the algorithm. The method, given in Misevicius (2001), applies a number, j, of pairwise random exchanges to the permutation. In the current algorithm, $j = n/2$. This method is applied to all solution vectors, b_2, in the reference set.

We note here that the solution vectors created by this method are not passed back through the tabu search improvement method. By using this approach, any level of solution quality or diversity cannot be guaranteed in the solutions added to the reference set. The use of more strategic forms of diversification offer a useful area for future examination, including the use of the diversification generation method applied in the initialization phase of the algorithm.

3. Parallel Path-Relinking Algorithm

The use of parallelism is of growing interest in the field of metaheuristics. Metaheuristics typically incorporate repetitive component algorithms and are in general computationally intensive for many problem types, making such approaches good candidates for the use of parallelism.

Parallelism has been heavily researched in the context of a variety of different metaheuristics. Genetic algorithms have been the most popular evolutionary algorithm for parallel implementations (Alba and Troya, 1999). The heavy reliance on randomization rather than memory by many genetic algorithms often eases the task of parallelizing the algorithm due to a reduced number of data dependencies.

The parallelization of tabu search has more recently attracted attention as a research area, with noteworthy contributions by Crainic and Toulouse (2002), who observe that strategies for parallelizing metaheuristics can be grouped loosely in three categories. The first strategy consists of a simple master-slave type parallelization where some large amount of work is divided among multiple processors with one node controlling the process of assigning work and collecting the results. The second consists of running the complete algorithm on each of the available processors. Typically with this strategy some type of cooperation or migration is employed where beneficial information on the search is exchanged in some manner between the processes. The last category contains implementations where the decision variables are divided in some manner between the processors.

The domain of parallel computing also provides a set of design and implementation decisions in addition to the design of the actual parallel algorithm, including considerations involving platform choice and associated hardware and software issues. Among parallel hardware platforms commercially available, the most popular fall into the multiple-instruction stream, multiple-data stream (MIMD) category of Flynn's taxonomy. Two of the more prominent platforms today can be distinguished by the location of their memory and are referred to as shared-memory platform and distributed-memory platforms. Many architectural differences as well as software/programming differences exist between these two platforms. In some cases, the choice of platform may have a dramatic effect on the computational results of a parallel algorithm.

A master-slave strategy was implemented for the parallel version of the path-relinking algorithm for the current study. This parallel design incorporates parallelization to speed up the execution of a sequential algorithm without changing the fundamental nature of the algorithm.

The parallel implementation of the algorithm, follows the scatter search/path-relinking template previously depicted in Figure 1, with the exception of the improvement method (Taillard's search procedure) being performed concurrently on the solutions generated by the solution combination method. Since the improvement method is performed on each solution generated by the solution combination method, this task is computationally expensive and a good candidate for parallelization. In essence, this parallelization design allows the tabu search algorithm to be run concurrently on a set of solutions generated and maintained by the base heuristic method.

```
PR Algorithm
Begin
    Read Data (if applicable)
    Initialialize Reference Set
    For n=1 to the maximum iterations specified, do
        If new solutions have not been added
            Call  Diversification Generation Method
        End If
        Call Subset Generation Method
        Call Solution Combination Method
        For each combined solution do (in parallel)
            Call Improvement Method (if applicable)
        End Loop
        Call Reference Set Update Method
    End Loop
End
```

Figure 1- Pseudocode for the Path-Relinking Algorithm

The parallel algorithm was implemented on a shared-memory parallel computing platform. A twelve processor (1.3 GHz), SGI Altix 3300, was the available platform. A maximum of 8 processors were used for the parallel runs. The parallel programming implementation used on was OpenMP with Intel compilers.

4. Computational Results

The following tables show the relative percentage deviation ($100*(s_{avg}/s_{bks} - 1)$) from the best known solution (BKS) over 10 runs of the path-relinking (PR) algorithm. Each run was terminated at either 200 iterations of the algorithm (PR r1) or 1000 iterations (PR r2). The tabu search code was run for 200 iterations on each child solution. The average time (Avg) for each test problem is given for both the sequential and parallel runs. For the parallel algorithms, speedups are also given. Speedup is given as:

$$S(n) = \frac{t_s}{t_p}$$

where t_s is the sequential execution time of the algorithm and t_p is the parallel execution time of the algorithm.

A thread safe random number generator was used to maintain consistency in the results for the sequential and parallel algorithms. The average deviation from the best known solution reported in the tables below was the same for both the 10 runs of the sequential algorithm and the 10 runs of the parallel algorithm as the same seeds were used for both versions.

The speedups obtained were favorable especially in the larger QAP instances, in some cases drastically reducing the total computational time. Figure 2 and 3 show a graphical depiction of the time comparisons between the sequential and parallel algorithms.

The results obtained are comparable and in many cases favorable to those reported in the QAP literature. The following tables also give comparisons of the performance of our algorithms to several of the best-known heuristic methods for the QAP. The comparisons are made against the solution quality reported in the papers referenced by QAPLIB. These algorithms have reported some of the best-known solutions for the test sets used in this paper. Therefore, the following tables give an overview of the relative comparison of the algorithm developed in this research versus some of the better known algorithms in the literature. For these comparisons, the parallel algorithm developed in this study was run for 1000 iterations, using the same seeds, on a restricted subset of problems used in the relevant literature. The reported number of iterations for the respective algorithms (max iter) is denoted below the table. The subset of algorithms used for the comparisons include the genetic hybrids (GH) developed by Fleurent and Ferland (1994), ant colony optimization algorithms by Stutzle and Dorigo (1999) and Gambardella et al. (1997) denoted in the tables by MMAS-QAP(TS), MMAS-QAP(2-opt), and HAS-QAP respectively, tabu search algorithms by Taillard (1991, 1995) denoted in the tables by RTS and Misevicius (2002) denoted by ETS-1, ETS-2, and ETS-3, and a scatter search algorithm by Cung et al. (1996) denoted by SS (CMMT).

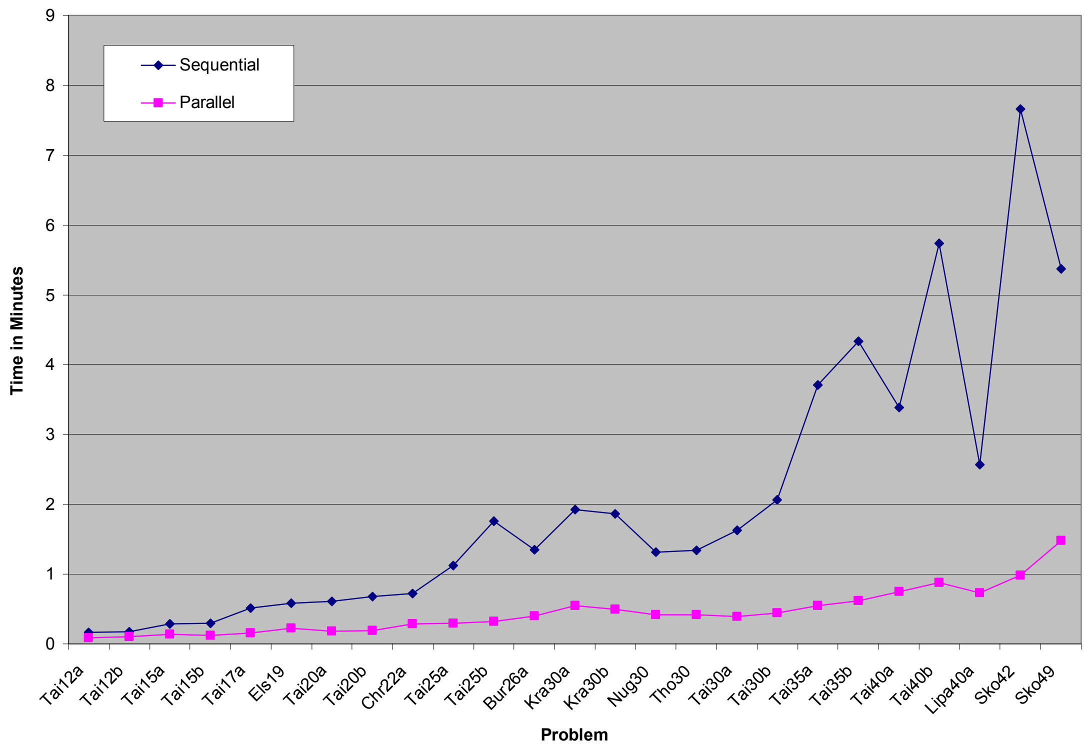

Figure 2 - Sequential and Parallel times for PR runs (n < 50)

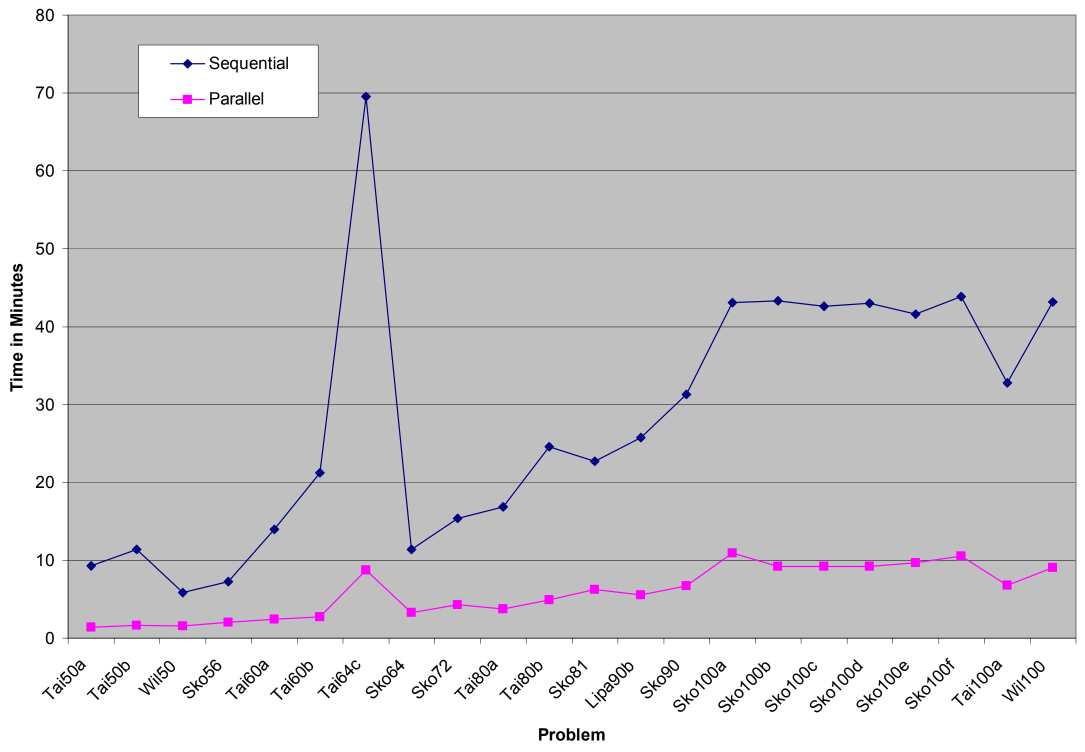

Figure 3 - Sequential and Parallel times for PR Runs (n >=50 to n =100)

Problem	BKS	PR r1	PR r1 Seq. Time	PR r1 Par. Time	Speed -up	PR r2	RTS	HGA	ETS-1	ETS-2	ETS-3
Tai20a	703482	0.246	0.608	0.182	3.349	0.030	0.091	0.411	0.047	0.030	0.108
Tai25a	1167256	0.640	1.122	0.292	3.846	0.259	0.195	0.382	0.113	0.040	0.055
Tai30a	1818146	0.614	1.628	0.392	4.157	0.501	0.135	0.362	0.098	0.041	0.081
Tai35a	2422002	1.092	3.710	0.552	6.725	0.773	0.255	0.643	0.113	0.170	0.095
Tai40a	3139370	1.109	3.382	0.745	4.539	0.866	0.536	0.618	0.489	0.451	0.462
Tai50a	4941410	1.263	9.277	1.397	6.642	1.082	0.874	0.871	0.652	0.660	0.617
Tai60a	7208572	1.416	13.982	2.395	5.838	1.356	0.952	1.009	0.797	0.787	0.771
Tai80a	13557864	1.109	16.867	3.785	4.456	1.022	0.634	0.593	0.544	0.546	0.535
Tai100a	21125314	0.966	32.800	6.780	4.838	0.902	0.637	0.493	0.444	0.476	0.476

Table 1 - Comparisons with GH and Tabu Search
(PR r1: max iter=200) (PR r2: max iter=1000) (tabu search: max iter = 25n)

Prob.	BKS	PR r1	PR r1 Seq. Time	PR r1 Par. Time	Speed -up	PR r2	RTS	HGA	ETS-1	ETS-2	ETS-3
Tai20b	122455319	0.000	0.678	0.193	3.509	0.000	0.000	0.045	0.000	0.000	0.000
Tai25b	344355646	0.000	1.758	0.325	5.410	0.000	0.000	0.000	0.000	0.000	0.000
Tai30b	637117113	0.000	2.060	0.443	4.647	0.000	0.001	0.000	0.011	0.000	0.000
Tai35b	283315445	0.037	4.338	0.620	6.997	0.000	0.006	0.094	0.019	0.028	0.000
Tai40b	637250948	0.000	5.740	0.882	6.510	0.000	0.003	0.000	0.000	0.000	0.000
Tai50b	458821517	0.062	11.395	1.665	6.844	0.000	0.221	0.033	0.001	0.003	0.042
Tai60b	608215054	0.042	21.232	2.770	7.665	0.000	0.160	0.014	0.012	0.001	0.044
Tai80b	818415043	0.284	24.597	4.882	5.039	0.175	0.235	0.353	0.360	0.113	0.337

Table 2 - Comparisons with GH and Tabu Search
(PR r1: max iter=200) (PR r2: max iter=1000) (tabu search: max iter = 5n)

Problem	BKS	PR r1	PR r1 Seq Time	PR r2 Par Time	Speed -up	PR r2	Ro-TS	GH	HAS-QAP	MMAS-QAP (TS)	MMAS-QAP (2-opt)
Tai20a	703482	0.246	0.608	0.182	3.349	0.030	0.108	0.268	0.675	0.191	0.428
Tai25a	1167256	0.640	1.122	0.292	3.846	0.259	0.274	0.629	1.189	0.488	1.751
Tai30a	1818146	0.614	1.628	0.392	4.157	0.501	0.426	0.439	1.311	0.359	1.286
Tai35a	2422002	1.092	3.710	0.552	6.725	0.773	0.589	0.698	1.762	0.773	1.586
Tai40a	3139370	1.109	3.382	0.745	4.539	0.866	0.990	0.884	1.989	0.933	1.131
Tai50a	4941410	1.263	9.277	1.397	6.642	1.082	1.125	1.049	2.800	1.236	1.900
Tai60a	7208572	1.416	13.982	2.395	5.838	1.356	1.203	1.159	3.070	1.372	2.484
Tai80a	13557864	1.109	16.867	3.785	4.456	1.022	0.900	0.796	2.689	1.134	2.103
Sko42	15812	0.025	7.660	0.982	7.803	0.004	0.025	0.003	0.076	0.015	0.104
Sko49	23386	0.071	5.373	1.478	3.635	0.049	0.076	0.040	0.141	0.067	0.150
Sko56	34458	0.066	7.242	2.003	3.615	0.049	0.088	0.060	0.101	0.068	0.118
Sko64	48498	0.054	11.385	3.255	3.498	0.045	0.071	0.092	0.129	0.042	0.171
Sko72	66256	0.111	15.372	4.277	3.594	0.104	0.146	0.143	0.277	0.109	0.243
Sko81	90998	0.060	22.675	6.228	3.641	0.051	0.136	0.136	0.144	0.071	0.223
Sko90	115534	0.134	31.308	6.742	4.644	0.110	0.128	0.196	0.231	0.192	0.288

Table 3 - Comparisons with GH and Ant Colony
(PR r1: max iter=200) (PR r2: max iter = 1000) (Ant Colony: max iter=1000)

Problem	BKS	PR r1	PR1 Seq. Time	PR1 Par. Time	Speedup	SS (CMMT)
Els19	17212548	0.000	0.583	0.223	2.612	0.000
Bur26a	5426670	0.000	1.348	0.397	3.399	0.027
Kra30a	88900	0.000	1.925	0.548	3.511	1.273
Lipa90b	12490441	0.000	25.733	5.565	4.624	8.100
Sko100a	152002	0.092	43.102	10.902	3.954	1.057
Tai100a	21125314	0.966	32.800	6.780	4.838	1.472
Tho150	8133398	0.149	147.408	37.378	3.944	1.478
Tai150b	498896643	0.413	279.787	35.928	7.787	1.415
Tai256c	44759294	0.142	2379.760	315.022	7.554	0.250

Table 4 - Comparisons with Cung et al. Scatter Search
(PR r1: max iter=200) (SS(CMMT): max iter = 50n)

5. Discussion

The simplified path-relinking algorithm developed for this study performed relatively well even in the sequential version, in spite of its rudimentary nature. For the smaller problems, the optimal solution to the problem was almost always found. For the larger problem instances the performance of the method rivals that of the leading evolutionary algorithms found in the literature.

The results obtained illustrate the potential of the PR approach. Our study discloses in particular the value of a well-designed parallel implementation in this setting. Favorable speedups were seen for all problems, although the benefit of the parallelization can be seen the best in the larger problem instances.

6. Limitations and Directions for Further Research

The present study is a preliminary one, constituting the first study to examine the relevance of parallel processing for applying path relinking in the QAP context. Our findings encourage additional follow-on studies that examine more complete and advanced forms of PR, making use of more sophisticated processes for managing the reference set and for creating combinations of its solutions. For simplicity we have taken short cuts relying on randomization in several steps where more strategic policies are possible. The introduction of processes for carrying out further intensification and diversification functions are certain to make a significant difference. We also plan to conduct additional experimentation with other parallelization strategies.

References

Alba, E., and Troya, J. (1999) “A Survey of Parallel Distributed Genetic Algorithms,” *Complexity*, 4, 31-52.

Cela, E. (1998), *The Quadratic Assignment Problem: Theory and Algorithms*, Kluwer Adademic Publishers.

Crainic T.G., and Toulouse, M. (2002) “Parallel Strategies for Meta-heuristics,” In *State-of-the-Art Handbook in Metaheuristics*, F. Glover, G. Kochenberger (Eds.), Kluwer Academic Publishers.

Cung, V-D., T. Mautor, P. Michelon, A. Tavares (1996) “Scatter Search for the Quadratic Assignment Problem,” Proceedings of the *IEEE International Conference on Evolutionary Computation*, 165–169.

Fleurent, C. and Ferland, J. (1994) “Genetic Hybrids for the Quadratic Assignment Problem”. In *Quadratic Assignment and Related Problems*, P. Pardalos and H. Wolkowicz (Eds.), DIMACS Series in Discrete Mathematics and Theoretical Computer Science, 16, 173-187.

Gambardella, L., Taillard, E., and Dorigo, M. (1997) “Ant Colonies for the QAP,” Technical Report IDSIA-4-97, IDSIA, Lugano, Switzerland.

Glover, F. (1998) “A Template for Scatter Search and Path Relinking,” In *Artificial Evolution*, J-K Hao, E. Lutton, E., Ronaled, M. Schoenauer and D. Snyers (Eds.), Springer-Verlag, *Lecture Notes in Computer Science*, 1363, 13-54.

Glover, F. (1991) “Tabu Search for Nonlinear and Parametric Optimization (with Links to Genetic Algorithms),” *Discrete and Applied Mathematics*, 49, 231-255.

Glover, F., M. Laguna, R. Martí (2003) “Scatter Search,” In Theory and Applications of Evolutionary Computation: Recent Trends, A. Ghosh and S. Tsutsui (Eds.), Springer-Verlag, New York, 519-537.

Koopmans, T. and M. Beckmann (1957) “Assignment Problems and the Location of Economic Activities,” Econometrica, 25, 53-76.

Laguna, M., R. Martí (2003) “Scatter Search: Methodology and Implementation,” Kluwer Academic Publishers.

Misevičius, A. (2002) “A Tabu Search Algorithm for the Quadratic Assignment Problem”, Working Paper, Kaunas University of Technology, Kaunas, Lithuania.

Rego, C. (2005) “RAMP: A New Metaheuristic Framework for Combinatorial Optimization,” In: "Metaheuristic Optimization via Memory and Evolution: Tabu Search and Scatter Search", C. Rego and B. Alidaee (Eds.), Kluwer Academic Publishers, 441-460.

Rego, C., B. Alidaee (2005) “Metaheuristic Optimization via Memory and Evolution: Tabu Search and Scatter Search,” Kluwer Academic Publishers.

Stutzle, T., and Dorigo, M. (1999) “ACO Algorithms for the Quadratic Assignment Problem,” In *New Ideas for Optimization*, D. Corne, M. Dorigo, and F. Glover (Eds.), McGraw-Hill, 33-50.

Taillard, E. (1991) “Robust Taboo Search for the Quadratic Assignment Problem,” Parallel Computing, 17, 443-455.