DIMACS Series in Discrete Mathematics
and Theoretical Computer Science
Volume 00, 0000

Genetic Hybrids for the Quadratic Assignment Problem

CHARLES FLEURENT AND JACQUES A. FERLAND

May 16, 1993

ABSTRACT. A new hybrid procedure that combines genetic operators to existing heuristics is proposed to solve the Quadratic Assignment Problem (QAP). Genetic operators are found to improve the performance of both local search and tabu search. Some guidelines are also given to design good hybrid schemes. These hybrid algorithms are then used to improve on the best known solutions of many test problems in the literature.

1. Introduction

The quadratic assignment problem (QAP) can be stated as:

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

where $A = (a_{ij})$ and $B = (b_{kl})$ are two $n \times n$ matrices and $P(n)$ is the set of all permutations of $\{1, ..., n\}$. Matrix A is often referred to as a distance matrix between sites, and B as a flow matrix between objects. In most cases, the matrices A and B are symmetrical with a null diagonal. A permutation may then be interpreted as an assignment of objects to sites with a quadratic cost associated to it. There are many applications that can be formulated as QAP, some of which are surveyed in [**2**] and [**7**]. It is well known that the QAP is NP-Complete [**9**]. Furthermore, because of its structure, it has been particularly difficult to use exact methods to solve even relatively small instances of the problem ($n > 20$). Consequently, heuristics have been proposed in order to find good solutions for larger problems.

1991 *Mathematics Subject Classification.* Primary 65K10; Secondary 90B80, 92D99.
The first author was supported in part by a NSERC postgraduate scholarship.
The second author was supported in part by NSERC A8312 and FCAR 93-ER-1654 grants.
This paper is in final form and no version of it will be submitted for publication elsewhere.

For many years, local search methods were used, and an implementation of these was made on an array processor [**8**]. As was the case with many local search algorithms, simulated annealing [**3**] and tabu search [**4, 22, 23**] methods were developed in order to improve the quality of the solutions. More recently, a genetic algorithm [**24**] was proposed to solve the QAP. In this paper, we present a meta-heuristic approach that allows one to improve on existing solution procedures by embedding them into a genetic algorithm.

In the next section, neighborhood based heuristics and genetic algorithms for the QAP are reviewed. In Section 3, we present a genetic hybrid algorithm that combines a genetic algorithm to other heuristics. The beneficial effect of genetic operators on local search and tabu search is shown by applying different procedures to several test problems and by studying their average behavior on a particular problem. In Section 4, we discuss some issues that arise when designing a genetic hybrid scheme and provide some guidelines on how to obtain better results. These guidelines are then used to improve on the best known solutions of many well known test problems.

2. QAP heuristics

Because of the inefficiency of exact procedures, several heuristic approaches were developed for the QAP. Among these, are limited enumeration methods [**7**] and construction methods [**10**]. In practice however, the best results have been obtained with improvement methods and their extensions such as simulated annealing and tabu search.

2.1. Local search. Most of the modern heuristics for the QAP originate from improvement methods defined over a neighborhood. A local search method starts from an initial solution and looks among its neighbors for an improving solution. Transitions are then made from solutions to better ones until a local optimum is reached (no neighbor can improve the current solution).

For the QAP, neighborhoods are usually generated using 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:

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

The effect of this particular swap can be evaluated in $O(n)$ operations. For instance, when the matrices A and B are symmetrical with null diagonal, formula (2.1) can be used.

$$\Delta(\phi, r, s) = 2 \sum_{i=1, i \neq r,s}^{n} (a_{is} - a_{ir})(b_{\phi(i)\phi(s)} - b_{\phi(i)\phi(r)}). \tag{2.1}$$

With pairwise exchanges, the number of neighbors generated at each iteration of a local search algorithm is in $O(n^2)$ if the complete neighborhood of every solution is explored. Another local improvement method might generate only neighbors until a better solution has been found and then move to it. For the QAP however, it is advantageous to consider the full neighborhood of every solution since we can evaluate the effect of a particular swap using the information available from the preceding iteration [**8, 23**]. Indeed, assume that $\Delta(\phi, i, j)$ are known $\forall i, j$ and that π is generated from ϕ by swapping indices r and s. If we then swap u and v in π ($\{r, s\} \cap \{u, v\} = \emptyset$), $\Delta(\pi, u, v)$ can be evaluated in constant time using (2.2).

(2.2)
$$\begin{aligned}\Delta(\pi, u, v) = \Delta(\phi, u, v) + \quad & 2(a_{ru} - a_{rv} + a_{sv} - a_{su}) \\ & (b_{\pi(r)\pi(v)} - b_{\pi(r)\pi(u)} - b_{\pi(s)\pi(v)} + b_{\pi(s)\pi(u)}).\end{aligned}$$

Whenever $\{u, v\} \cap \{r, s\} \neq \emptyset$, (2.1) can still be used. Hence, with the exception of the first, each iteration of the local search algorithm requires $O(n^2)$ operations. It is also possible to consider a larger neighborhood where 3 positions are exchanged. In this case, the neighborhood size is in $O(n^3)$ and the updating of each exchange can still be computed in constant time using a formula similar to (2.2). Hence, each iteration of the local search algorithm requires $O(n^3)$ operations.

For a long time, local search procedures were very popular and an implementation of these was made on an array processor [**8**]. The local improvement methods have the advantage of providing reasonably good solutions in short time. However, they also have the drawback of stopping at the first local minimum encountered. To bypass this shortcoming, a simulated annealing approach with pairwise exchanges was used in [**3**]. However, tabu search, another popular extension of local search, was found in [**22**] to outperform simulated annealing for the QAP.

2.2. Tabu search. Tabu search is a heuristic method that improves on local search by allowing to escape from local minima. Introduced by Glover [**11**], it is now widely used and often yields the best results for a wide variety of problems [**14, 17**]. The essence of the method consists in allowing climbing moves when no improving neighbor exists. However, a data structure keeps track of the history of the search in order to prevent cycling, and possibly diversify or intensify the search. For more details, the reader is referred to [**12, 13, 14**].

For the QAP, the first adaptation of tabu search was made by Skorin-Kapov. In [**22**], a pairwise exchange neighborhood structure is used and pairs of swapped indices are kept in a tabu list. This implementation was found to outperform the simulated annealing approach of Burkard and Rendl [**3**]. Note that this was also observed for several other problems [**17**].

In [**23**], Taillard proposes a tabu search procedure that also performs very well on test problems. Taillard's algorithm also uses a pairwise exchange procedure. However, the tabu list prohibits moves where both interchanged objects would be assigned to locations they had occupied in recent iterations. The tabu list size is also dynamically varied during the execution of the algorithm. This simple adaptation of tabu search proves to be very robust. The algorithm is also implemented on parallel processors, and it generates the best known solutions of many test problems from the literature.

In [**4**], Chakrapani and Skorin-Kapov enhance the procedure in [**22**] with a more sophisticated aspiration criterion and new diversification and intensification strategies. Their algorithm is also implemented on a Connection Machine that uses n^2 processors. This implementation also produces the best known solutions for many test problems.

2.3. Genetic algorithms. By now, genetic algorithms are getting better known and are being used in a wide range of applications. Goldberg [**15**] and Davis [**6**] provide a very good coverage of theoretical and practical issues of genetic algorithms (GA). In short, a GA is a model that evolves populations of solutions by means inspired from evolution and natural selection. Among the required elements of a GA are an encoding scheme for solutions into individuals, a mechanism for selecting parents during reproduction, a crossover operator to produce offsprings from selected parents, a mutation operator that can alter some individuals of the population, and a culling scheme that governs how the population is maintained and which individuals are removed from the population. There are several implementations of GA. Figure 1 describes the general scheme that we used. It differs from classical GA since it uses a steady-state population rather than generational replacement. For more details on the particular merits of other implementations, the reader is referred to [**6**]

(i) Find an initial population of m solutions.
(ii) Evaluate each solution in the population.
(iii) Produce k new solutions by
    (a) selecting parents according to their relative fitness.
    (b) applying crossover and mutation operators to produce offsprings from parents.
    (c) evaluating the new k solutions and adding them to the population.
(iv) Select k solutions to be removed from the population.
(v) If a stopping criteria is not met, return to step (iii).

FIGURE 1. A genetic algorithm.

In [**24**], a particular GA is proposed for the QAP. In this implementation, solutions are encoded as permutations and the initial population is therefore com-

posed of m random permutations. To promote the selection of better individuals for reproduction, the authors in [**24**] use a rank based selection mechanism. To select a parent, a random number u in $[0, m^{1/r}]$ is chosen (where m is the size of the population and r is a real number in the interval $[1, 2]$). The selected parent is then the solution in position $\lfloor u^r \rfloor$ of the sorted population. The probability of selecting better parents increases with the value of parameter r.

Since binary encodings are not used, non traditional operators have to be designed for crossover and mutation. In the literature, many operators have been proposed for the crossover and mutation of permutation problems [**15, 16**]. The role of the crossover operator is to produce a child that shares some characteristics from both of his parents while preserving the permutation structure. The crossover operator used in [**24**] is summarized as follows:

(i) If an object is assigned to the same site for both parents, it remains at the same location for the child.

(ii) Unassigned sites are scanned from left to right. For an unassigned site, we pick an object at random among those that occupied that site in the parents. Once an object is assigned, it is no longer considered in future random choices.

(iii) The remaining objects are assigned to the unassigned sites.

Table 1 gives an example of this crossover operator. The mutation operator used in [**24**] consists of selecting two sites at random and reversing the order of the assignments between those two.

parent 1	3	1	2	7	4	6	5
parent 2	2	1	4	3	6	7	5
step 1		1					5
step 2	3	1	4	7	6		5
step 3	3	1	4	7	6	2	5

TABLE 1. Example of a crossover operator.

Applications [**5, 15**] and theory [**18**] show that GA can be useful in solving difficult problems. In [**24**], good solutions were found for small instances of the QAP. However, a pure genetic approach still has it shortcomings for combinatorial optimization problems. In [**24**], the GA is tested on the Nugent problems [**21**]. The largest of these problems is of size 30 and is most probably solved to optimality by the tabu search procedures in [**4, 22, 23**]. While the pure genetic approach still yields good solutions for small problems, it cannot find the best known solutions for problems of size 20 and 30. For larger problems of size of up to 100, the genetic algorithm cannot really compete with tabu search procedures.

Despite this fact, the results obtained in [**24**] are encouraging. Indeed, even if the best known solutions for the most difficult QAP problems could not be obtained, the results indicate that the genetic approach works and can provide

good solutions by means completely different from those used with other heuristics. This may suggest the use of this approach to complement and improve on existing procedures. The idea of combining genetic algorithms with other search methods has been suggested many times recently [**1, 15, 16, 19**]. In the next section, we will show how this can be done to design a solution procedure that has improved on the best known solutions for most of the larger test problems in the literature.

3. Genetic hybrids

We have already pointed out that the main shortcoming of local search is that it stops at the first local minimum it reaches. A classical corrective approach has been to repeat the local search procedure with new starting points [**20**]. In practice, this approach allows to obtain better solutions, but after a while, the rate of improvement becomes very low.

Our goal is to show that this multi-start approach can be improved by using genetic operators. We will first describe a general GA that can be hybridized with several search procedures. It will then be shown that the hybrid procedures improve on random multi-start procedures with significantly better results.

3.1. Genetic Hybrid Algorithm. Our hybrid scheme consists in working with a population P of solutions that are produced with other heuristic procedures. Thus, our method can be seen as a meta-heuristic that improves on existing solution procedures. The method is also very flexible and can be implemented with many different parameters values. Following is a list of parameters to be provided to the algorithm.

(i) Heuristic functions $H_1(x)$ and $H_2(x)$ that generate improved solutions from a starting point x. These functions can be local search or tabu search procedures, for instance.
(ii) A crossover function, $crossover(p_1, p_2)$ that generates an offspring from parent solutions p_1 and p_2. The crossover operator described in Section 2.3 can be used for instance.
(iii) A culling procedure $Cull(P, k)$ to remove k individuals from the population P. For instance, the k individuals with greatest value for the objective function can be removed from P.
(iv) Various parameters:
    (a) m, the size of the population;
    (b) the value r for the selection operator;
    (c) nb_gen, the number of generations;
    (d) nb_per_gen, the number of offsprings generated at each generation.

The genetic hybrid algorithm is summarized as follows:

Genetic Hybrid Algorithm(H_1, H_2)
$P := \emptyset$
For $i := 1$ to m do
    Generate a random permutation p
    Add $H_1(p)$ to P
Sort P
For $i := 1$ to nb_gen do
    For $j := 1$ to nb_per_gen do
        select two parents p_1 and p_2 from P
        $child := crossover(p_1, p_2)$
        Add $H_2(child)$ to P
    Sort P
    $Cull(P, nb\_per\_gen)$
Return the best solution from P

In this genetic algorithm, there is no mutation operator since each individual is generated with a heuristic procedure. Alternatively, the heuristics H_1 and H_2 can be seen as mutation operators applied at a 100 % rate. With a steady-state population and our selection mechanism for reproduction, it is important to apply H_1 to each member of the initial population. This way, the hybrid scheme can be regarded as a genetic algorithm searching among individuals that are generated by heuristic procedures. Here, we go beyond the analogy with nature in which parents only pass their chromosomal material to their children. Indeed, we allow parents to improve on their genetic material and to use their enhanced chromosomes for reproduction, something that is only possible in nature with very rare and small mutations.

3.2. The effect of genetic hybridization. The Genetic Hybrid Algorithm (GHA) can use different heuristic procedures. In order to show the beneficial effect of the procedure, we will examine in this section the case in which $H_1 = H_2$. It will then be possible to compare the results of the genetic hybrid scheme with those obtained from the repetition of the same heuristic procedure with different random starting points.

Table 1 shows the results obtained from applying some search procedures to the larger test problems from [**4**]. LS is a local search algorithm that uses pairwise exchanges. Tab_4n and Tab_4000n are the tabu search procedures given in [**23**] with $4n$ and $4000n$ iterations, respectively. $Rep(LS)$ and $Rep(Tab\_4n)$ denote procedures in which LS and Tab_4n have been applied 1000 times with different starting points, respectively. GHA is the genetic hybrid algorithm of Section 3.1 in which the size of the population is 100, $nb\_gen = 450$, and $nb\_per\_gen = 2$. The additional operations in GHA (crossover, sorting) require negligible time compared to search procedures so that both schemes require

about the same computational work.

Prob.	$Rep(LS)$	$GHA(LS)$	$Rep(Tab\_4n)$	$GHA(Tab\_4n)$	Tab_4000n
SK72	66802	66518	66404	66256	66280
SK81	91648	91170	91156	91070	91026
SK90	116470	115780	115992	115542	115700
SK100a	152890	152160	152372	152068	152222
SK100b	154990	154140	154518	153898	154070
SK100c	149040	148020	148354	147866	147866
SK100d	150816	149696	149918	149692	149724
SK100e	150440	149444	149598	149150	149636
SK100f	150474	149536	149734	149132	149128

TABLE 2. Results from one application of some search procedures on the Skorin-Kapov problems.

In order to compare the different heuristics, it would be possible to use a Wilcoxon rank test. In our case, the results in Table 2 indicate that the hybrid algorithm outperforms the repeating procedure for both LS and Tab_4n. Except for two cases, the long tabu search Tab_4000n is also dominated by $GHA(Tab\_4n)$.

To illustrate the effect of the genetic operators on the search procedures, we ran 5 times each of the methods of Table 2 on the first Skorin-Kapov problem of size 100. The average behavior of the algorithms is displayed in Figure 2 and 3. This behavior is typical and was the same for the other test problems. With longer searches than those used in this section and some improvements described in the next section, the genetic hybrids perform even better.

4. Improving the search

In the previous section, we considered simple hybrids that showed that genetic operators can really improve on existing heuristic methods. In this section, we discuss some important issues that arise when designing a more powerful genetic hybrid algorithm. Some guidelines will be given and will be used in order to obtain or improve the best solutions of many test problems described in the literature.

4.1. Entropy of a population. One of the key issues for a genetic algorithm is its convergence rate. We say that an algorithm has converged when its population only contains replicates of the same individual. It is then useful to have a measure that indicates how far from convergence a population is. In [**16**], a definition for the entropy of a population of solutions is given for the traveling salesman problem. For the QAP, we define the entropy of a site as follows:

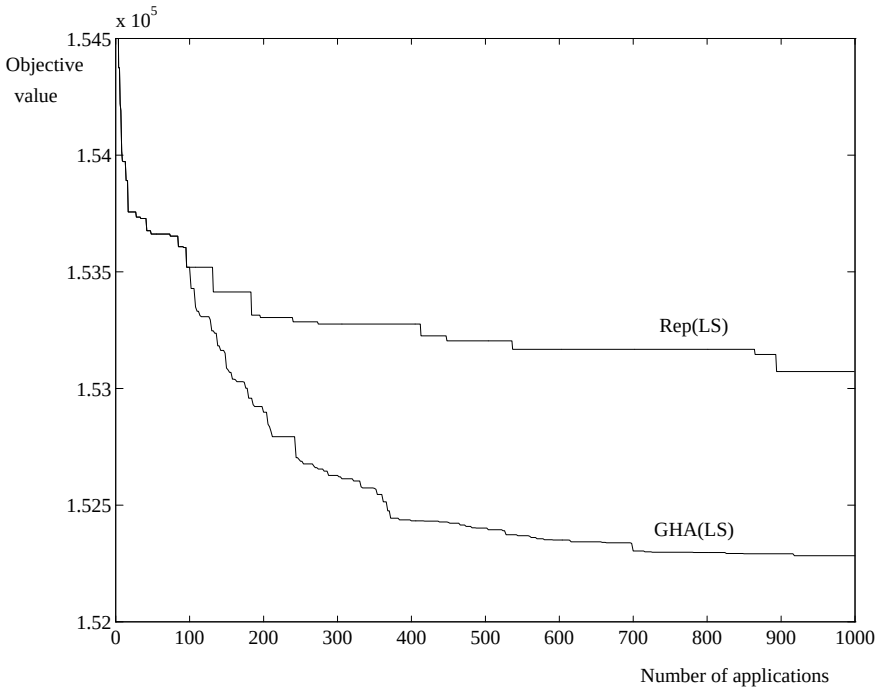

FIGURE 2. Effect on local search. Average of 5 runs for the first problem of size 100 of Skorin-Kapov.

$$E_i = \frac{-\sum_{j=1}^{n} \left(\frac{n_{ij}}{m}\right) \log\left(\frac{n_{ij}}{m}\right)}{\log(n)}, \tag{4.1}$$

where n_{ij} represents the number of times object i is assigned to site j in a population of size m. The entropy of the population can then be defined as follows:

$$E = \frac{\sum_{i=1}^{n} E_i}{n}. \tag{4.2}$$

This normalized entropy takes values between 0 and 1. Hence, E can be seen as a measure of the diversity in the population. $E = 0$ means that only one individual is represented in the population, while a value near 1 indicates a wide variety in the assignments that are represented in the population.

Depending on the parameters used, the convergence rate of a genetic algorithm may vary a lot. With a genetic hybridization, some of the methods may be

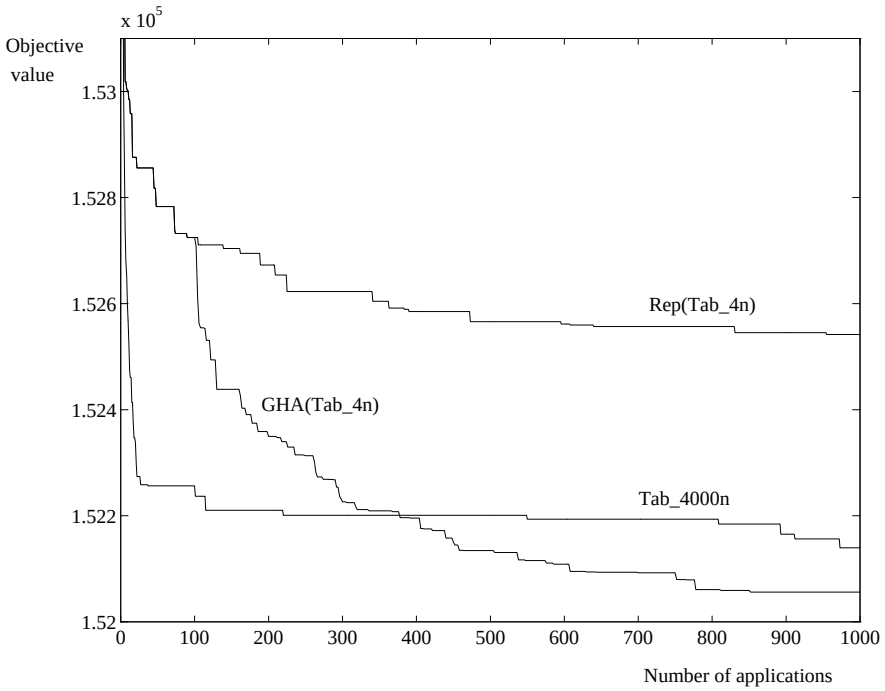

FIGURE 3. Effect on tabu search. Average of 5 runs for the first problem of size 100 of Skorin-Kapov.

expensive in computation time. It then becomes very important to monitor closely the entropy value. The convergence rate has to be slow enough to provide good results through thorough exploration of the domain, but fast enough to ensure convergence within the prescribed time.

4.2. Heuristic procedures. In Section 3, the same heuristics are used for generating the initial population and the new individuals during the evolution phase. Our purpose was to show the effect of genetic operators on a single heuristic. However, when solving a particular problem, any combination of methods can be used. For instance, a good initial population should include good solutions with high entropy. A priori, we might think that a powerful heuristic would tend to produce solutions that are similar. In fact, for sufficiently large QAP problems, this is not the case. For instance, the number of local optima seems to be large enough to allow local search to generate very diverse populations. The results in Table 3 indicate the same behavior even for long tabu searches. Hence, they can be used if enough computer time is available.

When the hybrid algorithm is executed several times, it may also be a good idea to store populations. It is then possible to extract from available solutions diverse high quality initial populations that can be used for new applications of

Method	entropy	average fitness	best individual	worst individual
Random	.8938	108488	105926	111239.2
Local Search	.8229	93055.6	92055.2	94272
Tab_4n	.8073	92164.76	91390.8	93094
Tab_10n	.7982	91815.62	91222	92603.6

TABLE 3. Initial populations of size 100 produced by some methods for the Skorin-Kapov problem of size 81 (average of 5 runs).

the hybrid algorithm.

The heuristics used in the evolution phase of the hybrid genetic algorithm do not need to be as powerful as those used to generate the initial population. During the evolution phase, children produced by crossover are the result of a random rearrangement and it is most likely that they will not be as fit as other individuals that came from earlier applications of heuristic procedures. A heuristic is therefore needed in order to bring new children to maturity and compete within the population. While the crossover operator is expected to generate individuals that combine good characteristics that will prove to be complementary, the role of the heuristic is to develop the potential of these children and identify the promising ones from the others. It therefore appears a better idea to allocate more computation time to generate many children and to rely more intensively on the genetic operators, rather than to complete long searches for each child.

There exist many stopping criterias for tabu search. In fact, the method is designed so that it could search the solution space for as long as desired. Ideally, the search should be diversified to exclude the need to restart from a new solution. In practice however, it may also be beneficial to consider several starting points. In [**22, 23**], this strategy is used and each application of the search is stopped after a prescribed number of iterations. It would be interesting to have rules indicating when a tabu search has exploited the potential of a child by which time another starting point should be provided by reproduction in the genetic hybrid algorithm. Genetic operators could then be interpreted as diversification tools for tabu search.

4.3. Convergence rate. With most genetic algorithms, premature convergence is a problem, and ways to improve on it have been studied in [**1, 16**]. However, for a genetic hybrid, evaluation of individuals is usually more expensive. Hence, if the heuristics are time consuming (e.g. long tabu searches), it may be required to converge in fewer generations. On the other hand, if the heuristics are less demanding in computational effort (e.g. local search), or if more time is available, a late convergence may be advised.

Many factors can slow down the convergence of a genetic algorithm. Among these, special care has to be given to:

Size of the population: A small population will converge much sooner but may give results of inferior quality.

Selection of parents: A more aggressive selection that picks more often the best individuals is more likely to converge prematurely. With the selection mechanism described in Section 2.3, the parameter r can be adjusted during the execution of the algorithm: $r = 1+f(E)$, for instance. Hence, the selection becomes less aggressive when the entropy of the population decreases.

Forbidding of replicates: When a new individual is created, it is advantageous to verify that it is not already included in the population. Replicates are then discarded and another child is produced. With ranking based parent selection, better individuals are already chosen more often for reproduction and it is better to dispose of more diversity in the population (see [**6**]).

Culling scheme: It is natural to delete the worst individuals from the population. In order to maintain diversity in the population, the function $Cull$ defined in Section 3.1 could also consider contribution to the entropy when removing the individuals from the population.

Crossover operators: There exist alternate crossover operators to the one described in figure 1. Some of these operators may result in a faster convergence.

4.4. Test problems. As mentioned in Section 2.3, there exist two sets of problems in the literature that constitute a good challenge for the best heuristics now available. The first set is found in Skorin-Kapov [**22**]. These problems are similar but larger than those described in Nugent et al [**21**]. In [**4**], six new problems of size 100 are introduced. In [**23**], Taillard proposes a new way to generate test problems. In his paper, the matrix coefficients are computed from a recursive formula chosen so that the problems can be reproduced easily.

Table 3 displays the best known solutions for each of these problems before this paper. With our Genetic Hybrid Algorithm, we managed to improve on most of the test problems. Many hybridizations are possible depending on the computational effort involved. In order to improve on the best known solutions, we usually started from initial populations of size n produced by powerful tabu searches (n^2 iterations) and generated about 600 children with tabu search with $10n$ iterations. On a SPARC 10 processor, this can lead to about 22 hours of computation time. For some of Taillard's problems, a less aggressive selection mechanism was used in order to slow the convergence, and more than three days were needed in order to improve on the best known solutions. While this may seem a lot, it does not involve more tabu search iterations than the results obtained by the implementations in [**4, 23**]. In [**22**], less iterations were needed to obtain good results, but it is our feeling that even better solutions would be found with the addition of a genetic component.

Problem	Former best known solution	Genetic Hybrid Algorithm
T 40	3146514[a]	3141702
T 50	4951186[a]	4941410
T 60	7272020[a]	7233118
T 80	13582038[a]	13560516
T 100	21245778[a]	21169542
SK 81	91008[b]	90998
SK 90	115534[a]	115534
SK 100a	152014[b]	152002
SK 100b	153900[b]	153890
SK 100c	147868[b]	147862
SK 100d	149596[b]	149576
SK 100e	149156[b]	149150
SK 100f	149036[b]	149036

a: comes from [**23**]
b: comes from [**4**]

TABLE 4. Best known solutions for test problems of Taillard and Skorin-Kapov.

It may be possible to obtain these results or improve on them with less computation, but further research is necessary in order to fine tune all of the involved parameters. It was easier for us to use a more powerful variant that consistently found the best solutions and that we could use systematically on each test problem to try to improve on the best known solutions. It should also be noted that our algorithm could be implemented easily in a natural way on a parallel processing computer with the searches distributed amongst the processors. The two best tabu search procedures for the QAP right now both use parallel implementations. In [**23**], the neighborhood of solutions is divided into 10 parts and the value of moves are computed on a ring of 10 transputers. In [**4**], tabu search is adapted and implemented on a Connection Machine which uses up to 10000 processors.

Our results are quite encouraging. For the Taillard problems, we improved the best known solutions of all problems. For some problems we found more than 50 different solutions that were better than the previous best known solution. For the Skorin-Kapov problems, we always found or improved on the best known solutions.

5. Conclusion

In this paper, we have presented a general approach combining genetic algorithms and other heuristic procedures. The beneficial effect of adding a genetic component in solving the QAP has been demonstrated in two ways. First, an

experimentation on many test problems has shown that local search and tabu search methods are significantly enhanced when hybridized with genetic operators. Second, best known solutions have been improved for most of the larger test problems considered in recent tabu search implementations, indicating that our hybrid genetic algorithm is one of the best heuristic now available in terms of solution quality.

There remains much research to be done in improving hybrid schemes such as the one described in this paper. A parallel implementation is desirable and should be easy to achieve as multi-processor machines become more available. Also, other crossover operators, selection mechanisms and culling schemes should be studied, as well as parameter values such as the size of populations and the choice of heuristics to be hybridized. The hybrid genetic algorithm presented in this paper is a general scheme that could be easily applied to several other problems. It is our hope that this paper will encourage further research in ways of combining genetic operators with existing procedures.

References

1. **L. Booker,** *Improving search in genetic algorithms*, Genetic Algorithms and Simulated Annealing, Morgan Kaufmann Publishers, 1987, pp. 61–73.
2. **R.E. Burkard,** *Quadratic assignment problems*, European Journal of Operational Research **15** (1984), 283–289.
3. **R.E. Burkard and F. Rendl,** *A thermodynamically motivated simulation procedure for combinatorial optimization problems*, European Journal of Operational Research **17** (1984), 169–174.
4. **J. Chakrapani and J. Skorin-Kapov,** *Massively parallel tabu search for the quadratic assignment problem*, Annals of Operations Research (to appear).
5. **L. Davis,** *Genetic Algorithms and Simulated Annealing*, Morgan Kaufmann Publishers, 1987.
6. ———, *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York, 1991.
7. **G. Finke, R.E. Burkard, and F. Rendl,** *Quadratic assignment problems*, Annals of Discrete Mathematics **31** (1987), 61–82.
8. **A.M. Frieze, J. Yadegar, S. El-Horbaty, and D. Parkinson,** *Algorithms for assignment problems on an array processor*, Parallel Computing **11** (1989), 151–162.
9. **M.R. Garey and D.S. Johnson,** *Computers and Intractability: A guide to the Theory of NP-Completeness* W.H. Freeman and Co., San Francisco (1979).
10. **P.C. Gilmore,** *Optimal and suboptimal algorithms for the quadratic assignment problem*, SIAM Journal on Applied Mathematics **10** (1962), 305–313.
11. **F. Glover,** *Future paths for integer programming and links to artificial intelligence*, Computer and Operations Research **13** (1986), 533–549.
12. ———, *Tabu Search-Part I*, ORSA Journal on Computing **1** (1989), 190–206.
13. ———, *Tabu Search-Part II*, ORSA Journal on Computing **2** (1990), 4–32.
14. **F. Glover and M. Laguna,** *Tabu Search*, Modern Heuristic Techniques for Combinatorial Problems (to appear).
15. **D.E. Goldberg,** *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, 1989.
16. **J.J. Grefenstette,** *Incorporating problem specific knowledge into genetic algorithms*, Genetic Algorithms and Simulated Annealing, Morgan Kaufmann Publishers, 1987, pp. 42–60.
17. **A. Hertz and D. de Werra,** *Using tabu search techniques for graph coloring*, Computing **39** (1987), 345–351.
18. **J.H. Holland,** *Adaptation in Natural and Artificial Systems*, Ann Arbor, University of

Michigan Press, 1975.
19. **C.L. Huntley and D.E. Brown,** *A parallel heuristic for quadratic assignment problems*, Computers and Operations Research **18** (1991), 275–289.
20. **S. Lin,** *Computer solutions of the traveling salesman problem*, Bell System Technical Journal **44** (1965), 2245–2269.
21. **C.E. Nugent, T.E. Vollmann, and J. Ruml,** *An experimental comparison of techniques for the assignment of facilities to locations*, Operations Research **16** (1968), 150–173.
22. **J. Skorin-Kapov,** *Tabu search applied to the quadratic assignment problem*, ORSA Journal on Computing **2** (1990), 33–45.
23. **E. Taillard,** *Robust taboo search for the quadratic assignment problem*, Parallel Computing **17** (1991), 443–455.
24. **D.E. Tate and A.E. Smith,** *A genetic approach to the quadratic assignment problem*, Computers and Operations Research (to appear).

Université de Montréal, Département d'Informatique et de Recherche Opérationelle, 2900 Édouard-Montpetit, C.P. 6128, Succ. A, Montréal, Québec, Canada, H3C 3J7

E-mail address: fleurent@iro.umontreal.ca

Université de Montréal, Département d'Informatique et de Recherche Opérationelle, 2900 Édouard-Montpetit, C.P. 6128, Succ. A, Montréal, Québec, Canada, H3C 3J7

E-mail address: ferland@iro.umontreal.ca