Frequency Model Based Crossover Operators for Genetic Algorithms Applied to the Quadratic Assignment Problem

Hachemi Bennaceur and Zakir Ahmed
Department of Computer Science, Al Imam Mohammad Ibn Saud Islamic University, Saudi Arabia

Abstract: *The quadratic assignment problem aims to find an optimal assignment of a set of facilities to a set of locations that minimizes an objective function depending on the flow between facilities and the distance between locations. In this paper we investigate Genetic Algorithm (GA) using new crossover operators to guide the search towards unexplored regions of the search space. First, we define a frequency model which keeps in memory a frequency value for each pair of facility and location. Then, relying on the frequency model we propose three new crossover operators to enhance genetic algorithm for the quadratic assignment problem. The first and second ones, called Highest Frequency crossover (HFX) and Greedy HFX (GHFX), are based only on the frequency values, while the third one, called Highest Frequency Minimum Cost crossover (HFMCX), combines the frequency values with the cost induced by assigning facilities to locations. The experimental results comparing the proposed crossover operators to three efficient crossover operators, namely, One Point crossover* (*OPX*)*, Swap Path crossover* (*SPX*) *and Sequential Constructive crossover* (*SCX*)*, show effectiveness of our proposed operators both in terms of solution quality and computational time.*

Keywords: *Frequency model, quadratic assignment problem, GA, HFX*.

Received January 16, 2015; accepted May 24, 2015

1. Introduction

The Quadratic Assignment Problem (QAP) is central and one of the hardest problems in computer science and operations research. QAP model is involved in a variety of applications such as hospital design [11], computer backboard design [28], scheduling problems [6, 12], course location problems [4], and economic problems [17], etc. The problem can be defined as follow: There are a set of n facilities and a set of n locations. For each pair of locations, a distance is specified and for each pair of facilities a weight or flow is specified (e.g., the amount of supplies transported between the two facilities). The problem is to assign all facilities to different locations with the goal of minimizing the sum of the distances multiplied by the corresponding flows. The methods to solve the problem are classified into two categories-exact and heuristics. Exact methods give exact optimal solution to the problem, whereas, heuristics do not guarantee the optimality of the solution, but give near exact optimal solution in reasonable time. Many exact methods and meta-heuristic approaches have been developed to solve the QAP [16, 26].

Genetic Algorithms (GAs) have proved their effectiveness to solve a wide range of optimization problems raised in numerous domains. They first developed by Holland [15], are based on natural biology [14]. A Simple GA (SGA) starts with a randomly generated initial population of chromosomes (strings) that is referred as gene pool and then applies three operators to create new, and hopefully, better populations as successive generations. Selection is the first operator where strings are copied to the next generation with some probability based on their objective function values. Crossover is the second operator where randomly selected pairs of chromosomes are mated to create new chromosomes. The third operator, mutation, is the occasional random alteration of the value at a chromosome position. The crossover operator is the most powerful process and it plays a very important role in the GA search. Mutation diversifies the search space and protects from loss of genetic material that can be caused by selection and crossover.

In this paper, we investigate a GA based-approach enhanced with new crossover operators to guide the search towards unexplored regions of the search space. We defined a frequency model that keeps, in memory during the entire search, a frequency value for each pair of facility i and location j. The frequency value cumulates the total number of times where the location j is assigned to the facility i in different populations of the searching process. We believe that more a facility is assigned to a location in fitter individuals more it has a chance to be assigned to the same location in the optimal solution.

The idea of the frequency model is not suitable only to the QAP problem since it can be adapted and used for any other combinatorial optimization problem. We

expect that this knowledge will help to guide the search towards promising regions.

The proposed approach exploits the frequency model to generate new individuals through the crossover operations. Relying on the frequency model we developed three greedy crossover operators. The first and second ones called Highest Frequency crossover (HFX) and Greedy HFX (GHFX) are based only on the frequency values, while the third one called Highest Frequency Minimum Cost crossover (HFMCX) combines with the frequency values the cost induced by assigning facilities to locations. In all of these crossover operators we privilege the pairs of facility and location having highest frequency values to generate new individuals. The comparative study of HFX, GHFX and HFMCX and three efficient crossover operators of the literature-One Point crossover (OPX), Swap Path crossover (SPX) and Sequential Constructive crossover (SCX) show that our proposed crossovers are competitive both in terms of solution quality and computational time. In term of solution quality all crossover operators manage to find near-optimal solutions on the tested benchmarks. For instance the average gap between HFMCX and SCX is less than 1%. While in term of computation time, SCX consumes twice as much time compared to HFX and 1.5 times compared to GHFX.

The rest of this paper is organized as follows: In section 2, the main related works are presented. Section 3 introduces the frequency model. Section 4 describes the proposed crossover operators with detailed algorithms and illustrative examples. Section 5 conducts a comparative study to show the performance and merit of the proposed crossover operators. Section 6 presents conclusions and discussions.

2. Related Works

GA search of the solution space is done by creating new chromosomes from old ones. As mentioned above, crossover is the most important operator in GAs. In crossover operator, a pair of chromosomes is selected randomly from the mating pool. Then, a point, called crossover site, along their common length is randomly selected, and the information after the crossover site of the two parent chromosomes are swapped, thus creates (possibly) two new children. However, this basic crossover method does not support for the QAP.

Migkikh *et al.* [20] proposed a modification of Partially Mapped crossover (PMX) of Goldberg and Lingle [14] based on using a number of random mapping points-instead of one mapping segment for the QAP. This crossover was referred to as a Uniform PMX (UPMX). The ordered crossover of Davis [7] has been applied to the QAP by Vázquez and Whitley [28]. Lim *et al*. [18] proposed a variation of the OPX operator [13] for the QAP. Another crossover operator, named Cycle crossover (CX) operator was proposed for the Travelling Salesman Problem (TSP) by Oliver *et al*. [25], where offspring are built in such a way that each node (and its position) comes from one of the parents. Merz and Freisleben [19] applied CX operator of Oliver *et al*. [25] for the GA to the QAP. The Uniform Like crossover (ULX) was proposed by Tate and Smith [27]. Misevicius [21] slightly improved this crossover operator, renamed as optimized crossover and used in the improved hybrid GA. Misevičius and Kilda [22] proposed many modification of this crossover and called them as Randomized ULX (RULX), modified ULX (or Block crossover (BX)), and extended ULX (or Repair crossover (RX)). Ahuja *et al*. [2] developed a SPX for the QAP. It was proposed to perform the swap for which the corresponding solution has lower cost than the original one. Drezner [8] proposed another crossover operator, named Cohesive crossover (COHX), for the QAP. Misevičius and Rubliauskas [23] proposed a Multiple Parent crossover (MPX) operator that creates an offspring by choosing information from multiple numbers of parents. Misevičius and Kilda [22] made a comparative study among ten different crossover operators for the QAP and found that MPX is better than SPX which is better than OPX which is then better than COHX. Ahmed [1] proposed SCX for the QAP and presented a comparative study among SPX, OPX and SCX. From the comparative study, it is found that SCX is the best. SPX, OPX and SCX are used as reference crossovers in our comparative study. Now-a-days, hybrid GA implementing these crossover operators with local search is widely used [3, 9, 10, 20, 24]. However, we are not developing hybrid algorithm. Our aim is to show the effectiveness of our proposed crossover methods by comparing with other existing crossover methods.

3. Frequency Model

The frequency model keeps, in memory as a matrix F, a frequency value for each pair of facility and location. F is a square matrix of order n which is defined as follows. Each row of F corresponds to a facility and each column represents a location. F_{ij} is the number of times that facility i is assigned to location j. The frequency value F_{ij} of each pair (facility i, location j) cumulates the total number of times where the location j is assigned to the facility i in different populations of the GA search. The following pseudo-code shows how the frequency-matrix F evolves incrementally at each generation (iteration) of the GA.

Let P be the population of individuals selected for the crossover operation at a given generation of GA:

Algorithm 1: Updating frequency matrix.

Update (*F*, *P*) *// F is the current frequency-matrix and P is the current population*
for each individual p in P do
Search p;
if (*facility i is assigned to location j in p*) *then* $F_{ij}=F_{ij}+1$.
EndFor

Initially $F=0$ (all entries F_{ij} are initialized to 0). And at each generation of the GA, the procedure *Update* (*F*, *P*) is invoked to update the frequency-matrix *F*. Notice that, the frequency-matrix is updated based on the individuals of *P* of each generation. It means that highest frequencies correspond to facilities assigned to locations which occurred many times in the fittest individuals of different populations.

4. Crossover Operators based-on the Frequency-Matrix

The frequency-matrix is exploited as a short-term memory to define new crossover operators. The first and second ones, called HFX and GHFX, are based only on the frequency values of the matrix; while the third one called HFMCX combines the frequency values with the cost of assigning facilities to locations. For this purpose we use the natural encoding of a solution as a permutation of integers; each integer value indicates the location assigned to the facility. For instance in the following individual of length 6, facilities 1, 2, 3, 4, 5 and 6 are assigned locations 3, 5, 4, 6, 2 and 1 respectively.

Table 1. Example of an individual.

Facilities	1	2	3	4	5	6
Locations	3	5	4	6	2	1

4.1. Highest Frequency Crossover

Let P_1 and P_2 be a pair of parents selected for the crossover operation. The principle of HFX is simple that searches for each facility *i* the location *j* in P_1 and P_2 having highest frequency. When one or both of the two locations of the facility *i* in P_1 and P_2 are already present in the offspring then HFX considers the first locations of the parents not present in the offspring to determine the highest frequency location for the facility *i*. The following pseudo-code outlines the principle of the HFX crossover.

Algorithm 2: HFX.

HFX(P_1, P_2, *O*) *// O is the resultant offspring*
for each facility i : *i*=1, ..., *n do*
*j*1= $P_1[i]$ *//j1=location of i in* P_1.
*j*2= $P_2[i]$ *//j2=location of i in* P_2.
If(*locations* j_1 *and* j_2 *are not present in O*) *then*
O[*i*]= *assign*(*i*, j_1, j_2);
Else If (*location* j_1 *and* j_2 *both are present in O*) *then*
j_1 = *the first location of* P_1 *not present in O*;
j_2 = *the first location of* P_2 *not present in O*;
O[*i*]= *assign*(*i*, j_1, j_2);
Else If (*location* j_1 *is present in O*) *then*
j_1= *the first location of* P_1 *not present in O*;
else j_2= *the first location of* P_2 *not present in O*;
O[*i*]= *assign* (*i*, j_1, j_2);
EndIf
EndIf
EndIf
EndFor

The heuristic function *assign* (*i*, j_1, j_2) returns the location *k* to be assigned to the facility *i* given the frequency values of locations j_1 and j_2 for the facility *i*. When the frequency values F_{ij1} and F_{ij2} are equal the facility *i* will be assigned the location of the fittest parent, otherwise the location with the highest frequency value will be assigned to the facility *i*.

Function assign (*i*, j_1, j_2) *//returns* j_1 *or* j_2 *according to the frequency values* F_{ij1} *and* F_{ij2}.
{
if $F_{ij1}=F_{ij2}$ *then if* P_1 *is fitter than* P_2 *return* j_1 *else return* j_2;
Else return k=ArgMax{F_{ij1}, F_{ij2}} *//* ($F_{ij1}>F_{ij2}$, $k=j_1$, $k=j_2$);
}

The following example illustrates about generating an offspring using HFX. Let $F=(F_{ij})$ i=1, ..., 6 and j=1, ..., 6 be the frequency-matrix built from 20 individuals. For instance, the facility 2 (row 2 of the matrix) is assigned 1 time to location 1, 2 times to location 2, 9 times to location 3, 4 times to location 4 and 2 times to locations 5 and 6 in the 20 individuals. And for instance, the location 1 (column 1 of the matrix) is assigned 3 times to facility1, 1 time to facility 2, 7 times to facility 3, 3 times to facility 4, 2 times to facility 5 and 4 times to facility 6 in the 20 individuals.

Table 2. The frequency matrix F.

Locations / Facilities	**1**	**2**	**3**	**4**	**5**	**6**
1	3	1	1	6	1	8
2	1	2	9	4	2	2
3	7	3	2	2	3	3
4	3	6	5	2	4	0
5	2	2	3	3	8	2
6	4	6	0	3	2	5

The HFX applied to parents P_1 and P_2 produces the offspring O:

Table 3. HFX crossover applied to parents P1 and P2.

Facilities		**1**	**2**	**3**	**4**	**5**	**6**
P_1	**Locations**	1	2	3	4	5	6
P_2	**Locations**	3	5	1	6	4	2
O	**Locations**	1	5	2	4	3	6

- *Step* 1: Assign Facility 1. Since, max {F_{11}, F_{13}}=F_{11}, location 1 is assigned to facility 1.
- *Step* 2: Assign Fcility 2. Since, $F_{22}=F_{25}=2$, either location 2 or 5 can be assigned to the facility 2. Assume that P_2 is fitter than P_1 then location 5 is assigned to facility 2 (this choice is based on the

fitness values of P_1 and P_2; we select the location in the fittest parent).

- *Step* 3: Assign Facility 3. As max $\{F_{33}, F_{31}\}=F_{31}$ and location 1 is already present in the offspring, we select the first location of P_1 not present in O which is location 2. Since max $\{F_{33}, F_{32}\} = F_{32}$, the location 2 is assigned to facility 3.
- *Step* 4: Assign Facility 4. Since, max $\{F_{46}, F_{44}\} = F_{44}$, the facility 4 is assigned location 4.
- *Step* 5: Assign Facilities 5. Since, both locations 4 and 5 are already present in the offspring and location 3 is the first location of P_1 and P_2 which is not present in O, the facility 5 is assigned location 3. The remaining facility 6 is then assigned location 6.

4.2. Greedy Highest Frequency Crossover

The operator GHFX builds sequentially the offspring from the two parents P_1 and P_2. Starting from facility 1, GHFX selects the location l_1 from P_1 and P_2 having the highest frequency. Thereafter, it assigns to facility 2 the location following l_1 in P_1 and P_2 having the highest frequency. Let l_i be the location assigned to the facility i in O, then in the next step, GHFX assigns the location following l_i in P_1 and P_2 having the highest frequency. Particular cases when the selected location is already in the offspring are detailed in the pseudo-code and explained in the illustration example. The following pseudo-code outlines the principle of the GHFX.

Algorithm 3: GHFX.

GHFX (P_1, P_2, O) //*O is the resultant offspring*
$j_1 = P_1[1]$ // j_1=*location of facility* 1 *in* P_1;
$j_2 = P_2[1]$ // j_2=*location of facility* 1 *in* P_2;
$O[1]$ =*assign*(1, j_1, j_2); *//assign the location* j_1 *or* j_2 *to facility* 1
For each facility i:i=2, ..., *n do //assign facilities i:i*=2, ..., *n.*
Let j_1 *and* j_2 *be the locations next to location* $O[i\text{-}1]$ *in* P_1 *and* P_2 *respectively.*
If(*location* $O[i\text{-}1]$ *is not assigned to the last facility n in both parents* P_1 *and* P_2) *then*
If(*locations* j_1 *and* j_2 *are not present in O*) *then*
$O[i]$= *assign*(*i*, j_1, j_2);
Else If(*location* j_1 *and* j_2 *both are present in O*) *then*
j_1 = *the first location of* P_1 *not present in O*;
j_2 = *the first location of* P_2 *not present in O*;
$O[i]$=*assign*(*i*, j_1, j_2);
Else If (*location* j_1 *is present in O*) *then*
j_1 = *the first location of* P_1 *not present in O*
else j_2 = *the first location of* P_2 *not present in O*;
$O[i]$=*assign*(*i*, j_1, j_2);
EndIf
EndIf
EndIf
Else // location $O[i\text{-}1]$ *is assigned to the last facility n in parents*P_1 *or* P_2 *or both.*
If(*location* $O[i\text{-}1]$*is assigned to the last facility n in both parents* P_1 *and* P_2) *then*
j_1 = *the first location of* P_1 *not present in O*;
j_2 = *the first location of* P_2 *not present in O*;
$O[i]$=*assign*(*i*, j_1, j_2);
Else If (*location* $O[i\text{-}1]$ *is assigned to the last facility n in* P_1) *then*
j_1 = *the first location of* P_1 *not present in O*
else j_2 = *the first location of* P_2 *not present in O*
$O[i]$= *assign*(*i*, j_1, j_2); *EndIf*
EndIf
EndIf
EndFor

The GHFX applied to parents P_1 and P_2 produces the offspring O:

Table 4. GHFX crossover applied to parents P1 and P2.

Facilities		**1**	**2**	**3**	**4**	**5**	**6**
P_1	**Locations**	1	2	3	4	5	6
P_2	**Locations**	3	5	1	6	4	2
O	**Locations**	1	6	2	3	5	4

- *Step* 1: Assign Facility 1. Since, the facility 1 is assigned to location 1 in P_1 and location 3 in P_2, we select the location from {1, 3} having a max frequency value. Since, max $\{F_{11}, F_{13}\}=F_{11,}$ the facility 1 is assigned location 1 in the offspring.
- *Step* 2: Assign Facility 2. The location of facility 2 is selected from locations {2, 6} since the locations next to location 1 are 2 in P_1 and 6 in P_2. As $F_{22} = F_{26}$, either location 2 or location 6 can be assigned to facility 2. Assume that P_2 is fitter than $P_{1,}$ then the location 6 is assigned to facility 2 (this choice is based on the fitness values of P_1 and P_2; we select the location in the fittest parent).
- *Step* 3: Assign Facility 3. As location 6 is the last one in P_1, we consider the first free location 2 rather than 6, which is the first location of P_1 which is not present in O. Since 4 is the location following location 6 in P_2, and as max $\{F_{32}, F_{34}\}=F_{32}$, then the facility 3 is assigned location 2.
- *Step* 4: Assign Facility 4. As location 2 is the last one in P_2 then we consider the first free location 3 rather than 2, which is the first location of P_2 which is not present in O. As 3 is also the location following location 2 in P_1 then facility 4 is assigned location 3.
- *Step* 5: Assign Facility 5. 4 and 5 are the locations following location 3 in P_1 and P_2 respectively. And, as max $\{F_{54}, F_{55}\}=F_{55}$ then the facility 5 is assigned to location 5. The remaining facility 6 is then assigned automatically to location 4.

4.3. Highest Frequency Minimum Cost Crossover

The HFMCX operator is based on the induced cost of assigning a facility to a location and the frequency-matrix value of this assignment. It is almost similar to GHFX, the main difference is that the criterion of assigning a location to a facility in the offspring is

based on the highest frequency as in GHFX, and in addition it takes into account the induced cost of that assignment. Namely, the criterion is the maximum of the frequency value divided by the induced cost. The pseudo-code of HFMCX is similar to that of GHFX in which the heuristic criterion "maximum frequency value divided by induced cost" is integrated. Assume that k locations j_1, j_2, ..., j_k have been assigned to facilities 1, 2, ..., k in the offspring O, then the induced cost $IC(k+1, j_{k+1})$ of assigning a location j_{k+1} to a facility $k+1$ is computed as follows:

$$IC(k+1, j_{k+1}) = \sum_{i=1}^{n} Flow(i, k+1) * Dist(j_i, j_{k+1}) \quad (1)$$

Where $Flow(i, k+1)$ is the flow between facilities i and $k+1$ and $Dist(j_i, j_{k+1})$ is the distance between locations j_i and j_{k+1}. Then, the heuristic function *assign* for HFMCX can be rewritten as follows:

Function assign (i, j_1, j_2) *//returns* j_1 *or* j_2 *according to the frequency values* F_{ij1} *and* F_{ij2}.
{
if $F_{ij1}/IC(i, j1) = F_{ij2}/IC(i, j2)$ *then if* P_1 *is fitter than* P_2 *return* j_1 *else return* j_2;
else return $k=ArgMax\{F_{ij1}/IC(i, j_1), F_{ij2}/IC(ii\ j_2)\}$;
}

The intuition behind the heuristic function assign consists to favor locations with highest frequency and lowest induced cost. Reconsider the previous illustration example to show the steps for generating the offspring O from parents P_1 and P_2:

- *Step* 1: Assign Facility 1. We start by selecting the location having highest frequency for facility 1. Since max $\{F_{11}, F_{13}\}=F_{11}$, the facility 1 is assigned to location 1.
- *Step* 2: Assign Facility 2. The location of facility 2 will be selected from locations {2, 6} since the locations next to location 1 are respectively 2 in P_1 and 6 in P_2. Then, we compute max $\{F_{22}/IC(2, 2), F_{26}/IC(2, 6)\}$ and select the location satisfying the maximum. Assume that the maximum holds for location 4, thus the current offspring will be O: 1 4 x x x x x.
- *Step* 3: Assign Facility 3. Since the locations next to location 4 are 5 in P_1 and 2 in P_2 respectively, we compute max $\{F_{35}/IC(3, 5), F_{32}/IC(3, 2)\}$. Assume that the maximum holds for location 5, thus the current offspring will be O: 1 4 5 x x x.

The process is repeated until assigning a location to each facility. Particular cases are handled as for the GHFX crossover.

5. The Algorithm

In order to evaluate and to compare the performance of our proposed crossover operators, we developed a SGA. Its structure can be summarized by the following pseudo-code:

Algorithm 4: The genetic algorithm.

Generation=0; *Generate random initial population;*
Evaluate the initial population
While (*the stopping condition is not satisfied*) *do*
//build the new generation
Generation=*Generation*+1;
Repeat
{ *Select two parents* P_1 *and* P_2;
Perform the crossover operation on P_1 *and* P_2;
Perform the mutation operation;
Evaluate the new offspring;
} *until reaching the size of the population*
EndWhile

The two parents P_1 and P_2 are randomly selected using the roulette wheel selection technique. The crossover operation is performed using one of the proposed or existing efficient crossover operators. We keep in the new generation the fittest individual between the parent P_1 and the offspring. The traditional mutation operator exchanging randomly two locations is used in our SGA.

6. Computational Experiments

The SGAs using five crossover operators-HFX, GHFX, HFMCX, OPX and SPX, have been encoded in Visual C++ and run on a PC with Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz and 8.00GB RAM under MS Windows 7. In the experiments, we used some benchmark QAPLIB instances [5]. The experiments were performed 20 times for each instance. The solution quality is measured by the average excess of solutions obtained from the best known solution value reported in QAPLIB, as given by the formula:

$$Excess(\%) = \frac{Average\ Solution\ Value - Best\ Known\ Solution\ Value}{Best\ Known\ Solution\ Value} x\ 100 \quad (2)$$

We report percentage of excess of average solution value over the best known solution value of 20 runs. We set parameters as follows: 75 as population size, 1.00 as crossover probability, 0.05 as mutation probability and the same CPU time as termination criterion. Table 5 shows comparative study for eighteen QAPLIB instances of size from 20 to 100. For eight instances, namely, instance tai20a, tai20b, tai25b, tai30b, tai35b, tai40b, tai50b, and tai60b, HFMCX is found to be best; for four instances, namely, tai25a, tai30a, tai35a, and tai40a, OPX is the best; and for remaining six instances, SPX is found to be the best. HFMCX and SPX are competing. However, on the average, HFMCX is found to be the best and HFX is found to be the worst. Notice that the gap between the average excess of HFX and SPX is less than 1%, this means that the solutions obtained by the different crossover are near each other. Regarding the instance types, it is observed that the tai*b are found to be more difficult than the tai*a, and specially, tai25b, tai30b and tai35b are

very difficult instances. The solution quality showed in Table 5 is also depicted in Figure 1, from where it is very clear to understand the results.

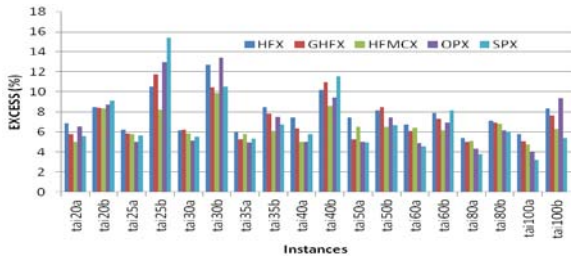

Figure 1. Comparative study of the crossover operators HFX, GHFX, HFMCX, OPX and SPX using Excess on QALIB instances

Table 5. Comparison of HFX, GHFX, HFMCX, OPX and SPX crossovers on QAPLIB instances using CPU time as termination criterion.

Instance	Best Reported Solution in QAPLIB	Excess (%)					Avg. CPU Time
		HFX	GHFX	HFMCX	OPX	SPX	
tai20a	703482	6.85	5.78	5.01	6.57	5.59	6
tai20b	122455319	8.45	8.42	8.32	8.73	9.12	6
tai25a	1167256	6.22	5.88	5.75	5.01	5.61	9
tai25b	344355646	10.54	11.74	8.24	12.92	15.42	9
tai30a	1818146	6.14	6.24	5.87	5.16	5.49	14
tai30b	637117113	12.71	10.44	9.87	13.4	10.55	14
tai35a	2422002	5.99	5.29	5.77	4.92	5.35	19
tai35b	283315445	8.47	7.84	6.11	7.49	6.76	19
tai40a	3139370	7.44	6.32	5.01	4.97	5.76	24
tai40b	637250948	10.22	11.02	8.60	9.44	11.57	24
tai50a	4938796	7.45	5.27	6.51	5.00	4.91	36
tai50b	458821517	8.15	8.47	6.49	7.46	6.68	36
tai60a	7205962	6.73	6.11	6.42	4.89	4.60	54
tai60b	608215054	7.88	7.34	6.19	6.95	8.14	54
tai80a	13499184	5.41	4.98	5.11	4.32	3.83	92
tai80b	818415043	7.12	6.91	6.82	6.15	6.07	92
tai100a	21052466	5.78	5.05	4.73	4.04	3.22	147
tai100b	1185996137	8.34	7.65	6.31	9.33	5.39	147
Average		7.77	7.26	6.51	7.04	6.89	44.56

In terms of solution quality, Figure 2 compares anytime behavior of GAs using HFX, GHFX, HFMCX, OPX and SPX operators at maximum of 1000 generations for tai25b in one run. The figure shows that HFMCX is the best crossover and HFX is the worst crossover in terms of solution quality.

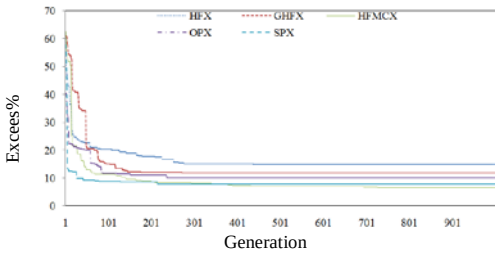

Figure 2. Convergence of GAs using different crossover operators for tai25b.

Further computational comparisons are carried out to evaluate the performance of HFX, GHFX and HFMCX against the best SCX crossover. We set parameters as follows: 200 as population size, 1.00 as crossover probability, 0.05 as mutation probability and 10000 generations as termination criterion. We report the average of computational time and the percentage of excess of average solution value over the best known solution value of 20 runs. Table 6 displays comparative study for the same eighteen QAPLIB instances of size from 20 to 100. The performances of these crossovers in terms of solution quality and computational time are separately depicted in Figures 3 and 4 respectively. In term of solution quality HFMCX is found to be better than HFX and GHFX. Regarding the EXCESS metric, the average gap between HFMCX and SCX is less than 1%. SCX provides the best solutions for all instances, but in terms of computation time, SCX consumes twice as much time compared to HFX and 1.5 times compared to GHFX. In order to provide a global overview of the performance of these four crossovers, we defined bellow a metric involving both the computational time and the percentage of excess in same time.

Table 6. Comparison of HFX, GHFX, HFMCX and SCX crossovers on QAPLIB instances using 10000 generations as termination criterion.

Instance	Known Solution	HFX		GHFX		HFMCX		SCX	
		Excess (%)	Time	Excess (%)	Time	Excess (%)	Time	Excess (%)	Time
tai20a	703482	6.81	7	5.59	8	4.82	13	4.39	13
tai20b	122455319	8.45	7	8.31	8	7.67	14	6.15	13
tai25a	1167256	6.18	12	5.82	15	5.33	20	4.21	20
tai25b	344355646	10.05	12	10.73	15	7.45	21	5.04	20
tai30a	1818146	6.11	17	6.12	22	5.66	31	3.67	30
tai30b	637117113	11.32	17	10.04	22	9.05	31	8.10	30
tai35a	2422002	5.78	24	5.21	29	5.71	41	3.52	40
tai35b	283315445	8.45	24	7.11	29	6.02	41	4.90	40
tai40a	3139370	7.44	32	6.23	39	4.23	57	3.74	55
tai40b	637250948	9.22	32	10.52	39	8.11	57	6.32	55
tai50a	4938796	7.13	46	5.47	55	5.46	82	3.81	80
tai50b	458821517	7.15	46	8.43	55	5.28	83	4.43	80
tai60a	7205962	6.05	71	6.00	83	5.74	123	4.00	120
tai60b	608215054	7.83	71	7.11	83	5.11	123	5.02	120
tai80a	13499184	5.34	118	4.55	138	4.02	215	4.13	210
tai80b	818415043	7.05	118	6.41	138	5.39	216	5.12	210
tai100a	21052466	5.49	160	5.04	241	3.65	387	3.65	380
tai100b	1185996137	8.26	160	7.09	241	5.23	389	5.00	380
Average		7.45	54.11	6.99	70.00	5.77	108.00	4.73	105.33

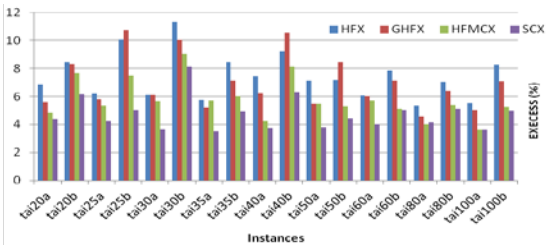

Figure 3. Comparative study of HFX, GHFX, HFMCX and SCX crossover operators using the excess on QAPLIB instances.

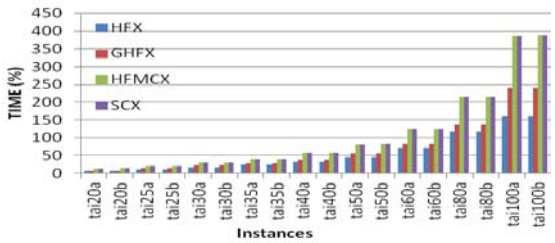

Figure 4. Comparative study of HFX, GHFX, HFMCX and SCX crossover operators using the computational time on QAPLIB Instances.

As mentioned above in order to measure the performances of the proposed crossover in terms of solution quality and computational time in same time, we defined the performance criterion $PC(X_1, X_2)$ based on EXECSS and TIME using the following metric:

$$PC(X_1, X_2) = EXCESS(X_1)*TIME(X_1)/EXCESS(X_2)*TIME(X_2)$$

This metric includes both the solution quality and the computational time. More $PC(X_1, X_2)$ is less than 1 more it is better for crossover $X1$. From Figure 5, it is seen that HFX and GHFX provide better performance for most of the instances according to this metric compared to SCX, the worst crossover is HFMCX.

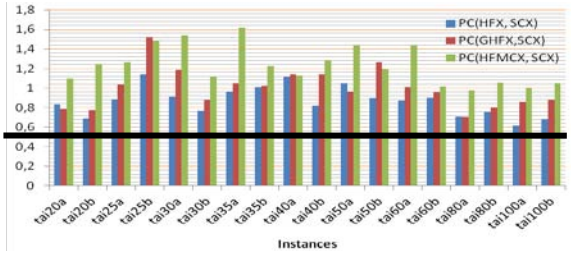

Figure 5. Comparative study of HFX, GHFX, HFMCX and SCX crossover operators using the metric PC on QAPLIB Instances.

7. Conclusions

We have introduced a frequency model allowing the design of three new crossover operators for enhancing GAs to solve the QAP. We have implemented the frequency model in a simple GA with different crossover operators to solve the QAP. We presented a comparative study among SPX, OPX, SCX and the proposed crossover operators on some benchmarks QAPLIB instances. In terms of solution quality the results show that our proposed crossover operators are competitive. On the other hand, in terms of computational time the proposed crossovers HFX and GHFX are faster than SCX, namely, HFX and GHFX are 2 and 1.5 times faster than SCX respectively. As the computational time of our proposed crossover HFX is much reduced, this could be an interesting opportunity to combine it with a local search technique to improve the solution quality. Our current investigation aims to develop and implement a hybrid GAs using HFX.

The frequency model keeps trace of the evolution of individuals from one generation to another. The knowledge kept in this model can be exploited to define new crossover operators or to enhance some existing ones. Another advantage of the frequency model is that it can also be used as a long-term memory in an immigration strategy to build new individuals based on the knowledge stocked in the frequency model. Finally, the idea of the frequency model is not proper to the QAP only, but, it can be adapted and used for any other combinatorial optimization problem.

References

[1] Ahmed Z., "A Simple Genetic Algorithm Using Sequential Constructive Crossover for the Quadratic Assignment Problem," *Journal of Scientific and Industrial Research*, vol. 73, no. 12, pp. 763-766, 2014.

[2] Ahuja R., Orlin J., and Tiwari A., "A Greedy Genetic Algorithm for the Quadratic Assignment Problem," *Journal of Computers and Operations Research*, vol. 27, no.10, pp. 917-934, 2000.

[3] Benlic U. and Hao J., "Breakout Local Search for the Quadratic Assignment Problem, Applied Mathematics and Computation," *Journal of Applied Mathematics and Computation*, vol. 219, no. 9, pp. 4800-4815, 2013.

[4] Brixius N. and Anstreicher K., "The Steinberg Wiring Problem," *Optimization Online*, 2001.

[5] Burkard R., Karisch S., and Rendl F., "QAPLIB-a Quadratic Assignment Problem Library," *Journal of Global Optimization*, vol. 10, pp. 391-403, 1997.

[6] Carlson R. and Nemhauser G., "Scheduling to Minimize Interaction Cost," *Operations Research*, vol. 14, no. 1, pp. 52-58, 1966.

[7] Davis L., "Job-shop Scheduling with Genetic Algorithms," *in Proceeding of the 1st International Conference on Genetic*, NJ, pp. 136-140, 1985.

[8] Drezner Z., "Extensive Experiments with Hybrid Genetic Algorithms for the Solution of the Quadratic Assignment Problem," *Journal of Computers and Operations Research*, vol. 35, no. 3, pp.717-736, 2008.

[9] Drezner Z., "A new genetic algorithm for the quadratic assignment problem," *Informs Journal on Computing*, vol. 15, no. 3, pp. 320-330, 2003.

[10] Drezner Z. and Misevicius A., "Enhancing the Performance of Hybrid Genetic Algorithms by Differential Improvement," *Computers and Operations Research*, vol. 40, no. 4, pp. 1038-1046, 2013.

[11] Elshafei A., "Hospital Layout as a Quadratic Assignment Problem," *Journal of Operations Research Quarterly*, vol. 28, no. 1, pp. 167-179, 1977.

[12] Geoffrion A. and Graves G., "Scheduling Parallel Production Lines with Changeover Costs: Practical Applications of a Quadratic Assignment/LP Approach," *Journal of Operations Research*, vol. 24, no. 4, pp. 595-610, 1976.

[13] Goldberg D., *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley Longman, 1989.

[14] Goldberg D. and Lingle R., "Alleles, Loci and the Travelling Salesman Problem," *in Proceeding of 1st International Conference on Genetic Algorithms, Hilladale*, NJ, pp. 154-159, 1985.

[15] Holland J., *Adaptation in Natural and Artificial Systems*, MIT Press, 1975.

[16] Javidi M. and Hosseinpourfard R., "Chaos Genetic Algorithm Instead Genetic Algorithm," *The International Arab Journal of Information Technology*, vol. 12, no. 2, pp. 163-168, 2015.

[17] Koopmans T. and Beckman M. "Assignment Problems and the Location of Economic Activities," *Journal of Econometric*, vol. 25, no. 1, pp. 53-76, 1957.

[18] Lim M., Yuan Y., and Omatu S., "Efficient Genetic Algorithms using Simple Genes Exchange Local Search Policy for the Quadratic Assignment Problem," *Journal of Computational Optimization and Applications*, vol. 15, no. 3, pp. 249-268, 2000.

[19] Merz P. and Freisleben B., "Fitness Landscape Analysis and Mimetic Algorithms for the Quadratic Assignment Problem," *IEEE Transactions on Evolutionary Computation*, vol. 4, pp. 337-352, 2000.

[20] Migkikh V., Topchy A., Topchy E., Kureichik V., and Tetelbaum A., "Combined Genetic and Local Search Algorithm for the Quadratic Assignment Problem," *in Proceedings of the Second Asia-Pacific Conference on Genetic Algorithms and Applications (APGA '00)*, pp. 144-151, 2000.

[21] Misevicius A., "An Improved Hybrid Optimization Algorithm for the Quadratic Assignment Problem Journal of Mathematical Modelling and Analysis," *Journal of Mathematical Modelling and Analysis*, vol. 9, no. 2, pp. 149-168, 2004.

[22] Misevicius A. and Kilda B., "Comparison of Crossover Operators for the Quadratic Assignment Problem," *Journal of Information Technology and Control*, vol. 34, no. 2, pp. 109-119, 2005.

[23] Misevicius A. and Rubliauskas D., "Performance of Hybrid Genetic Algorithm for the Grey Pattern Problem," *Journal of Information Technology and Control*, vol. 34, no. 1, pp. 15-24, 2005.

[24] Misevicius A. and Guogis E., *Communications in Computer and Information Science*, Spring Link, 2012.

[25] Oliver I., Smith D., and Holland J., "A Study of Permutation Crossover Operators on the Travelling Salesman Problem," *in Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application*, NJ, pp. 224-230, 1987.

[26] Panos M., Franz R., and Henry W., "The Quadratic Assignment Problem: A Survey and Recent Developments," *Journal of DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, vol. 16, pp. 1-42, 1994.

[27] Tate D. E. and Smith A.E., "A genetic approach to the quadratic assignment problem," *Journal of Computers and Operations Research*, vol. 22, no. 1, pp. 73-83, 1995.

[28] Vázquez M. and Whitley L. D., "A Hybrid Genetic Algorithm for the Quadratic Assignment Problem," *in Proceedings of the 2nd Annual Conference on Genetic and Evolutionary Computation*, Las Vegas, pp. 135-142, 2000.

Hachemi Bennaceur is Full Professor in Computer cience Department of Al Imam Mohammad bin Saud Islamic University. He worked for over twenty years in various academic institutions in France. He was promoted to full professor in 2007 at Artois University (Lille-Nord academy, France). During the same period he was successively researcher at the Computer Science Lab of Paris-Nord University (LIPN), and then at the Research Center of Computer Science of Lens (CRIL). His main research topics involve Constraint Programming and Reasoning with Propositional Logic (SAT). More recently, he is interested in Robot Path Planning and Multi-Robots Task Allocation applications.

Zakir Hussain Ahmed is an Associate Professor in the Department of Computer Science at Al Imam Mohammad Ibn Saud Islamic University, Saudi Arabia. He obtained MSc in Mathematics (Gold Medalist), MTech in Information Technology and PhD in Mathematical Sciences from Tezpur University (Central), Assam, India. He served in various institutions in India. His research interests include artificial intelligence, discrete optimization, digital image processing and pattern recognition. He has publications in the fields of artificial intelligence, discrete optimization and image processing.