Proceedings of the 5th WSEAS Int. Conf. on Artificial Intelligence, Knowledge Engineering and Data Bases, Madrid, Spain, February 15-17, 2006 (pp117-122)

# From the Magic Square to the Optimization of Networks of AGVs and from MIP to an Hybrid Algorithm and from this One to the Evolutionary Computation

JOSÉ BARAHONA DA FONSECA
Department of Electrical Engineering and Computer Science
New University of Lisbon
Monte de Caparica, 2829-516 Caparica
PORTUGAL
http://www.dee.fct.unl.pt

*Abstract*: - In a previous work we presented an algorithm inspired in the Artificial Intelligence and in the minimax optimization that imitates the human being in the solution of the magic square and we showed that in most cases its performance was better than the human's performance and even better than the performance of the best genetic algorithms to solve the magic square, in terms of number of changes.

In this paper we adapt and transform this algorithm to solve the optimization of an AGVs network problem, using as a test case 9 workstations in fixed positions and 9 operations to be executed, and the optimization problem is translated in the search of which of the 9! possible manners to distribute 9 operations by the 9 workstations that minimizes the total production time for a given plan of production.

This gradual process of adaptation and transformation resulted in an evolutionary hybrid algorithm with high performances, with a little bit of Tabu Search and a little bit of genetic algorithm. In 1000 successive runs, with the two tabu flags On, it never failed in the search of one of the 4 optimal solutions and never took more than 3000 iterations and 9!= 362880.

As a final validation test, using random search, in 1000 runs it never reached the optimal solution at the end of 100000 iterations.

In the near future we plan to consider the more general case where the number of workstations is greater than the number of operations, and so there are some workstations that make the same operation. This turns the problem more complex since when a product has operations that are executed by various workstations we must search all the possible combinations and find the path with minimum distance. Furthermore the generation of all 'permutations with repetitions' is more complex and in the literature there are no published algorithm to generate this type of combinatorial entities.

*Key-Words:* - AI Minimax Algorithm to Solve the Magic Square, Optimization of AGVs Networks, Hybrid Algorithm to Optimize AGVs Networks, Evolutionary Algorithm to Optimize AGVs Networks.

## 1 Introduction

The layout optimization is a difficult and complex problem due to the combinatorial explosive number of possible solutions and due to the dependence and interaction of the layout optimal solution with the optimal solution of production planning and scheduling. In this first approach we will only study the optimization of an AGVs network with 9 workstations and 9 operations, for a given production plan of a set of products defined as linear sequences of subsets of the 9 operations.

A possible solution will be a permutation of the nine operations. Since 9! it is not a too big number in terms of iterations of a computer program, as a preliminary exercise we generated all the 9! permutations and we got four optimal solutions.

We began to solve the problem with mixed integer programming (MIP) with the need of a lot of artificious tricks to linearize the model, but the final result was a deception: even for 9 workstations our optimization software package presented a runtime of the order of 2 days in a 1 GHz PC.

The algorithm that we present is an intermediary pass towards a more efficient evolutionary algorithm to optimize AGVs networks and then to optimize FMS layouts with AGVs networks. It is the result of a process of adaptation and transformation of our AI minimax algorithm to solve the magic square which presented a better performance than the best published evolutionary algorithms to solve the magic square [1].

## 2 Generation of All Optimal Solutions

Let's first of all to define exactly our AGVs network with 9 workstations, the production plan, the products and the operations. The AGVs network is a 3x5 matrix, where the first and last columns correspond to automatic warehouse

accesses, being the lines equally separated as well as the columns. The production plan is simply the definition of the number of units to be produced of each product. In table 1 we show the production plan used in our model. Each product is defined by a linear sequence of a subset of the 9 operations. In table 2 we show the sequences of operations that define each product. In table 3 we show the duration of each operation. Finally in table 4 we show the four optimal solutions that we got through the exhaustive generation of all the 9! permutations of 9 operations.

Parameters
Plano_prod(k) plano de producao
produto_i->n_unidades_i
/p1 5
p2 7
p3 8
p4 6
p5 3
p6 4/

**Table 1. Production plan used in this work.**

Table Produto_ops(k,opi) definicao da seq de ops de cada produto

| Nop | op1 | op2 | op3 | op4 | op5 | op6 | op7 | op8 | op9 |
|---|---|---|---|---|---|---|---|---|---|
| p1 | 7 | 5 | 6 | 2 | 0 | 1 | 3 | 0 | 7 | 4 |
| p2 | 4 | 4 | 3 | 0 | 0 | 0 | 2 | 0 | 0 | 1 |
| p3 | 8 | 1 | 7 | 5 | 8 | 3 | 6 | 4 | 0 | 2 |
| p4 | 9 | 6 | 4 | 5 | 3 | 2 | 7 | 1 | 9 | 8 |
| p5 | 6 | 5 | 6 | 2 | 1 | 0 | 0 | 3 | 0 | 4 |
| p6 | 8 | 0 | 5 | 2 | 6 | 8 | 7 | 3 | 4 | 1; |

**Table 2. Definition of each product.**

t_exec_op(opi) tempo de execucao das operacoes em segs
/Nop 0
op1 5
op2 7
op3 10
op4 15
op5 13
op6 12
op7 11
op8 10
op9 9/;

**Table 3. Definition of the duration of each operation.**

Tprodution=2058 s

Permutation=1373
Distribution of Operations by the Workstations:
0 2 3 5 0 0 10 7 4 0 0 9 6 8
Execution Time of each Product:
180 69 186 268 151 237

Permutation=39673
Distribution of Operations by the Workstations:
0 2 10 9 0 0 3 7 4 0 0 5 6 8
Execution Time of each Product:
180 75 186 236 177 255

Permutation=39673
Distribution of Operations by the Workstations:
0 2 10 9 0 0 3 7 4 0 0 5 6 8
Execution Time of each Product:
180 75 186 236 177 255

Permutation=266016
Distribution of Operations by the Workstations:
0 8 6 9 0 0 4 7 10 0 0 5 3 2
Execution Time of each Product:
180 69 186 268 151 237

**Table 4. All the 4 Optimal Solutions obtained through the generation of all the 9! permutations of 9 operations.**

## 3 Optimal Solution Obtained with MIP

The great difficulty that we have to solve during the solution of the optimisation of the AGVs network with 9 workstations with MIP was that we cannot make *nonlinear* operations over the variables of the model.

We solved the problem of permutation generation with a binary variable with two indexes, the workstation and the operation executed by it, and in this way we defined the logic of permutation generation with only arithmetic operations and iterative sums.

The problem of the need of a logical AND operation between two of these binary variables was solved with a new binary variable with 4 indexes, each pair of indexes signifying that that the operation *i* was attributed to the workstation *j*, and we have to create a set of constraints to guarantee the coherence between this binary variable and the previous binary variable that defined the permutation of operations or, by other words, the disposition of machines in the 3x3 matrix.

In table 5 we show the optimal solution obtained with this MIP model after two days of computation in a 1 GHz PC.

```
----    292 VARIABLE t_prodution.L          =     2058.0

PARAMETER n_agvs            =        1.745

----    292 VARIABLE est_q_ex.L

          e2          e3          e4          e5          e6          e7

op1       1.000
op2                   1.000
op3                                                             1.000
op4                             1.000
op6                                                   1.000
op9                                       1.000

+         e8          e9          e10

op5                   1.000
op7                             1.000
op8     1.000


----    292 VARIABLE t_exec_product.L

p1 180.000,    p2  69.000,    p3 186.000,    p4 268.000,
p5 151.000, vp6 237.0
```

**Table 5. Optimal solution obtained with MIP that corresponds to the first solution obtained with exhaustive search.**

## 4 Hybrid Algorithm for the Optimization of AGVs Networks

We will make only a qualitative description of this algorithm.

Having as departure point a given permutation of operations, it search a new one changing randomly two operations and that new permutation is accepted if the production time associated to it is *significantly less* than the previous; if the first tabu flag is *on* then the new permutation is saved in the first tabu list. If at the end of a given limit number of change trials it has not found a better permutation, then if the second tabu flag is *on* the last permutation is saved in the second tabu list and after this is accepted the change that *maximizes* the production time increase over a set of operations pairs randomly generated.

When it is generated a permutation that already exists in the first tabu list, that permutation is rejected and it reaches a permutation that exists in the second tabu list it returns to a previous solution that exists in the first tabu list.

Although simple this algorithm presented a performance in terms of the iterations number always much less than 9! or even 9!/100.

Next we show some results of computational experiences with this algorithm. In table 6 we present na example of a trace of a run with the two tabu flags *off*, departing from a sequential filling. As a curiosity, although it passes two times by the same solution that corresponds to a production time of 2121s, in the second time it travels a different path that leads to the optimal solution. In the next tables we show the statistics over 1000 runs, where it can be seen a significant improvement of the performance as the tabu flags got activated.

```
TprodutionMin= %2735 @ Niterations=1,
 0 2 3 4 0 0 5 6 7 0 0 8 9 10 0
TprodutionMin= %2611 @ Niterations=4,
 0 2 4 3 0 0 10 6 7 0 0 5 9 8 0
TprodutionMin= %2442 @ Niterations=8,
 0 2 4 6 0 0 3 10 7 0 0 5 9 8 0
TprodutionMin= %2353 @ Niterations=9,
 0 2 4 9 0 0 3 10 7 0 0 5 6 8 0
TprodutionMin= %2269 @ Niterations=17,
 0 2 4 9 0 0 3 7 10 0 0 5 8 6 0
TprodutionMin= %2259 @ Niterations=18,
 0 2 3 9 0 0 4 7 10 0 0 5 8 6 0
TprodutionMin= %2241 @ Niterations=37,
 0 2 7 10 0 0 4 3 5 0 0 9 8 6 0
TprodutionMin= %2223 @ Niterations=93,
 0 5 6 8 0 0 3 4 2 0 0 9 7 10 0
TprodutionMin= %2219 @ Niterations=124,
 0 5 6 8 0 0 3 4 7 0 0 9 10 2 0
TprodutionMin= %2156 @ Niterations=125,
 0 5 6 8 0 0 3 4 7 0 0 9 2 10 0
TprodutionMin= %2139 @ Niterations=127,
 0 5 8 6 0 0 3 4 7 0 0 2 9 10 0
TprodutionMin= %2121 @ Niterations=256,
 0 5 6 8 0 0 3 7 4 0 0 9 2 10 0
TprodutionMin= %2086 @ Niterations=271,
 0 5 6 8 0 0 3 7 4 0 0 9 10 2 0
TprodutionMin= %2121 @ Niterations=372,
 0 5 6 8 0 0 3 7 4 0 0 9 2 10 0
TprodutionMin= %2097 @ Niterations=442,
 0 5 8 6 0 0 3 4 7 0 0 9 2 10 0
TprodutionMin= %2058 @ Niterations=661,
 0 5 6 8 0 0 3 7 4 0 0 2 10 9 0
```

**Table 6. Exemple of a run with initial sequential filling and the two tabu flags *off*.**

Niters_max=6177, over 1000 Runs
For Niters < 100, Nruns=37
For 100 < Niters < 200, Nruns=82
For 200 < Niters < 300, Nruns=73
For 300 < Niters < 400, Nruns=82
For 400 < Niters < 500, Nruns=88
For 500 < Niters < 600, Nruns=69
For 600 < Niters < 700, Nruns=44
For 700 < Niters < 800, Nruns=35
For 800 < Niters < 900, Nruns=51
For 900 < Niters < 1000, Nruns=37
For 1000 < Niters < 1100, Nruns=34
For 1100 < Niters < 1200, Nruns=33
For 1200 < Niters < 1300, Nruns=35
For 1300 < Niters < 1400, Nruns=33
For 1400 < Niters < 1500, Nruns=22
For 1500 < Niters < 1600, Nruns=21
For 1600 < Niters < 1700, Nruns=21
For 1700 < Niters < 1800, Nruns=17
For 1800 < Niters < 1900, Nruns=13
For 1900 < Niters < 2000, Nruns=16
For 2000 < Niters < 2100, Nruns=9
For 2100 < Niters < 2200, Nruns=15
For 2200 < Niters < 2300, Nruns=7
For 2300 < Niters < 2400, Nruns=14
For 2400 < Niters < 2500, Nruns=7
For 2500 < Niters < 2600, Nruns=7
For 2600 < Niters < 2700, Nruns=6
For 2700 < Niters < 2800, Nruns=5
For 2800 < Niters < 2900, Nruns=8
For 2900 < Niters < 3000, Nruns=9
For Niters > 3000, Nruns=70

**Table 7. Statistic over 1000 runs in terms of the iterations number with the two tabu flags *off*.**

Niters_max=2911, over 1000 Runs
For Niters < 100, Nruns=53
For 100 < Niters < 200, Nruns=89
For 200 < Niters < 300, Nruns=102
For 300 < Niters < 400, Nruns=101
For 400 < Niters < 500, Nruns=107
For 500 < Niters < 600, Nruns=96
For 600 < Niters < 700, Nruns=81
For 700 < Niters < 800, Nruns=86
For 800 < Niters < 900, Nruns=53
For 900 < Niters < 1000, Nruns=63
For 1000 < Niters < 1100, Nruns=38
For 1100 < Niters < 1200, Nruns=34
For 1200 < Niters < 1300, Nruns=23
For 1300 < Niters < 1400, Nruns=16
For 1400 < Niters < 1500, Nruns=13
For 1500 < Niters < 1600, Nruns=9
For 1600 < Niters < 1700, Nruns=7
For 1700 < Niters < 1800, Nruns=7
For 1800 < Niters < 1900, Nruns=4
For 1900 < Niters < 2000, Nruns=3
For 2000 < Niters < 2100, Nruns=3
For 2100 < Niters < 2200, Nruns=3
For 2200 < Niters < 2300, Nruns=0
For 2300 < Niters < 2400, Nruns=1
For 2400 < Niters < 2500, Nruns=2
For 2500 < Niters < 2600, Nruns=0
For 2600 < Niters < 2700, Nruns=2
For 2700 < Niters < 2800, Nruns=1
For 2800 < Niters < 2900, Nruns=2
For 2900 < Niters < 3000, Nruns=1
For Niters > 3000, Nruns=0

**Table 8. Statistic over 1000 runs with only the first tabu flag *on*.**

Niters_max=2987, over 1000 Runs
For Niters < 100, Nruns=53
For 100 < Niters < 200, Nruns=93
For 200 < Niters < 300, Nruns=106
For 300 < Niters < 400, Nruns=96
For 400 < Niters < 500, Nruns=98
For 500 < Niters < 600, Nruns=77
For 600 < Niters < 700, Nruns=74
For 700 < Niters < 800, Nruns=82
For 800 < Niters < 900, Nruns=68
For 900 < Niters < 1000, Nruns=55
For 1000 < Niters < 1100, Nruns=45
For 1100 < Niters < 1200, Nruns=26
For 1200 < Niters < 1300, Nruns=29
For 1300 < Niters < 1400, Nruns=19
For 1400 < Niters < 1500, Nruns=21
For 1500 < Niters < 1600, Nruns=18
For 1600 < Niters < 1700, Nruns=7
For 1700 < Niters < 1800, Nruns=7
For 1800 < Niters < 1900, Nruns=3
For 1900 < Niters < 2000, Nruns=6
For 2000 < Niters < 2100, Nruns=2
For 2100 < Niters < 2200, Nruns=7
For 2200 < Niters < 2300, Nruns=0
For 2300 < Niters < 2400, Nruns=1
For 2400 < Niters < 2500, Nruns=3
For 2500 < Niters < 2600, Nruns=0
For 2600 < Niters < 2700, Nruns=1
For 2700 < Niters < 2800, Nruns=2
For 2800 < Niters < 2900, Nruns=0
For 2900 < Niters < 3000, Nruns=1
For Niters > 3000, Nruns=0

**Table 9. Statistic over 1000 runs with only the second tabu flag *on*.**

Niters_max=3210, over 1000 Runs
For Niters < 100, Nruns=35
For 100 < Niters < 200, Nruns=109
For 200 < Niters < 300, Nruns=89
For 300 < Niters < 400, Nruns=102
For 400 < Niters < 500, Nruns=108
For 500 < Niters < 600, Nruns=88
For 600 < Niters < 700, Nruns=75
For 700 < Niters < 800, Nruns=78
For 800 < Niters < 900, Nruns=50
For 900 < Niters < 1000, Nruns=49
For 1000 < Niters < 1100, Nruns=39
For 1100 < Niters < 1200, Nruns=39
For 1200 < Niters < 1300, Nruns=35
For 1300 < Niters < 1400, Nruns=28
For 1400 < Niters < 1500, Nruns=23
For 1500 < Niters < 1600, Nruns=16
For 1600 < Niters < 1700, Nruns=9
For 1700 < Niters < 1800, Nruns=10
For 1800 < Niters < 1900, Nruns=5
For 1900 < Niters < 2000, Nruns=5
For 2000 < Niters < 2100, Nruns=4
For 2100 < Niters < 2200, Nruns=1
For 2200 < Niters < 2300, Nruns=0
For 2300 < Niters < 2400, Nruns=0
For 2400 < Niters < 2500, Nruns=0
For 2500 < Niters < 2600, Nruns=2
For 2600 < Niters < 2700, Nruns=0
For 2700 < Niters < 2800, Nruns=0
For 2800 < Niters < 2900, Nruns=0
For 2900 < Niters < 3000, Nruns=0
For Niters > 3000, Nruns=1

**Table 10. Statistic over 1000 runs with the two tabu flags *on*.**

## 5 Conclusions and Future Work

Although very promising, we need a good reference to compare our algorithm, and we are implementing the OmeGA algorithm [2] and it is also planned the application of various commercial programs to design the layout of FMS also to the same AGVs network with 9 workstations and the same production plan.

In the near future we also plan to consider the more general case where the number of workstations is greater than the number of operations, and so there are some workstations that make the same operation. This turns the problem more complex since when a product has operations that are executed by various workstations we must search all the possible combinations and find the path with minimum distance. Furthermore the generation of all 'permutations with repetitions' is more complex and in the literature there are no published algorithm to generate this type of combinatorial entities.

*References:*

[1] J. Barahona da Fonseca, "The Magic Square as a Benchmark: Comparing Manual Solution to MIP Solution and to AI Algorithm and to Improved Evolutionary Algorithm", in *Proceedings of WSEAS Evolutionary Computation Conference,* Lisboa, 2005, pp. 486-492.

[2] D. Knjazew, *OmeGA: A Competent Genetic Algorithm for Solving Permutation and Scheduling Problems*, Kluwer Academic Publishers, 2002.