

Alma Mater Studiorum Università di Bologna
Archivio istituzionale della ricerca

A Tabu Search Matheuristic for the Generalized Quadratic Assignment Problem

This is the final peer-reviewed author's accepted manuscript (postprint) of the following publication:

Published Version:

Greistorfer P., Staněk R., Maniezzo V. (2023). A Tabu Search Matheuristic for the Generalized Quadratic Assignment Problem. Cham : Springer [10.1007/978-3-031-26504-4_46].

Availability:

This version is available at: <https://hdl.handle.net/11585/921793> since: 2023-04-03

Published:

DOI: http://doi.org/10.1007/978-3-031-26504-4_46

Terms of use:

Some rights reserved. The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. For all terms of use and more information see the publisher's website.

This item was downloaded from IRIS Università di Bologna (<https://cris.unibo.it/>).
When citing, please refer to the published version.

(Article begins on next page)

This is the final peer-reviewed accepted manuscript of:

Greistorfer, P., Staněk, R., Maniezzo, V. (2023). A Tabu Search Matheuristic for the Generalized Quadratic Assignment Problem. In: Di Gaspero, L., Festa, P., Nakib, A., Pavone, M. (eds) Metaheuristics. MIC 2022. Lecture Notes in Computer Science, vol 13838. Springer, Cham.

The final published version is available online at: https://doi.org/10.1007/978-3-031-26504-4_46

Terms of use:

Some rights reserved. The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. For all terms of use and more information see the publisher's website.

This item was downloaded from IRIS Università di Bologna (<https://cris.unibo.it/>)

When citing, please refer to the published version.

A Tabu Search Matheuristic for the Generalized Quadratic Assignment Problem

Peter Greistorfer¹, Rostislav Staněk², and Vittorio Maniezzo³

¹ Operations and Information Systems, Karl-Franzens-Universität Graz, Graz,
Austria

`peter.greistorfer@uni-graz.at`

² Applied Mathematics, Montanuniversität Leoben, Leoben, Austria

`rostislav.stanek@unileoben.ac.at`

³ Computer Science, Università di Bologna, Bologna, Italy

`vittorio.maniezzo@unibo.it`

Abstract. This work treats the so-called Generalized Quadratic Assignment Problem. Solution methods are based on heuristic and partially LP-optimizing ideas. Base constructive results stem from a simple 1-pass heuristic and a tree-based branch-and-bound type approach. Then we use a combination of Tabu Search and Linear Programming for the improving phase. Hence, the overall approach constitutes a type of mat- and metaheuristic algorithm. We evaluate the different algorithmic designs and report computational results for a number of data sets, instances from literature as well as own ones. The overall algorithmic performance gives rise to the assumption that the existing framework is promising and worth to be examined in greater detail.

Keywords: Generalized Quadratic Assignment · Matheuristic · Metaheuristic · Linear Programming · Tabu Search.

1 Introduction

The problem of interest is the so-called *Generalized Quadratic Assignment Problem* (GQAP). The GQAP has been used as a model for several relevant actual applications, including order picking and storage layout in warehouse management, relational database design or scheduling activities in semiconductor wafer processing. Technically, it originates from the *Linear Assignment Problem* (LAP), where a number of agents (equivalently, machines or supplies) have to be assigned to a number of jobs (tasks or demands), while minimizing the total cost of service and obeying assignment constraints, which secure that each job has to be serviced by exactly one agent and vice versa. The LAP in turn is a special case of the *Generalized Assignment Problem* (GAP), in which assignment constraints on the supply-side are replaced with upper bound constraints, which model an agent's capacity consumed by the assigned task weights. Another generalization of the LAP is the *Quadratic Assignment Problem* (QAP), which models multiplicative cost factors between agents and jobs, e.g. in locational analyzes

39 distances times interaction frequencies. Finally, the GQAP can be thought of as
 40 a combination of the GAP and the QAP, where the objective function receives
 41 a quadratic component in addition to the linear one, the same way as it is done
 42 in the QAP and suppliers have an upper bound constraint as in the GAP.

43 It is well-known that there exist efficient polynomial algorithms for the LAP,
 44 e.g. the Hungarian method, but the GAP, the QAP, and the GQAP are \mathcal{NP} -
 45 hard (e.g. see [1]). Matheuristics (see [6, 8]), a synthesis of classical, as a rule,
 46 (real-valued) linear and integer linear programming methods (LP, ILP) with
 47 conventional heuristic methods (e.g. local search) and/or modern metaheuristic
 48 methods (Tabu Search, GRASP, Scatter Search etc.) have become popular with
 49 the rise of recent powerful hardware and even more because of the success of
 50 solvers like CPLEX or Gurobi.

51 Our research focuses on both mathematical formulations of the GQAP as
 52 well as on the development of LP- and ILP- based matheuristics. In this paper
 53 we review a branch-and-bound type heuristic tree search and elaborate on the
 54 basic decomposition idea of the LP-component in the improvement algorithm.
 55 Afterwards, the focus lies on the presentation of a new Tabu Search (TS), the *TS-*
 56 *matheuristic GQAP* approach (TS-GQAP) is presented. We conclude with current
 57 computational results and a short outlook.

58 2 Modelling the GQAP

59 The GQAP can be described by means of the following quadratic integer pro-
 60 gram. We are given $m \in \mathbb{N}$, the number of agents, $n \in \mathbb{N}$, the number of jobs,
 61 with linear assignment costs and weights $p_{ij} \in \mathbb{R}$ and $w_{ij} \in \mathbb{R}_0^+$, where $1 \leq i \leq m$
 62 and $1 \leq j \leq n$, respectively. The weights present resource amounts to be spent
 63 by agent i for processing job j , without exceeding an available capacity $a_i \in \mathbb{R}_0^+$.
 64 Quadratic costs are defined by the product of $d_{ir} \in \mathbb{R}$, the cost factor between
 65 the agents i and r , and $f_{js} \in \mathbb{R}$, the cost factor between the resources j and
 66 s . Binary decision variables $x_{ij} \in \{0, 1\}$ determine whether a job j is served by
 67 agent i , or not. Then the GQAP is defined by the following binary quadratic
 68 program (BQP):

$$\min \sum_{i=1}^m \sum_{j=1}^n p_{ij} x_{ij} + \sum_{i=1}^m \sum_{j=1}^n \sum_{r=1}^m \sum_{s=1}^n d_{ir} f_{js} x_{ij} x_{rs} \quad (1)$$

$$\text{s.t.} \quad \sum_{j=1}^n w_{ij} x_{ij} \leq a_i \quad \forall 1 \leq i \leq m, \quad (2)$$

$$\sum_{i=1}^m x_{ij} = 1 \quad \forall 1 \leq j \leq n, \quad (3)$$

$$x_{ij} \in \{0, 1\} \quad \forall 1 \leq i \leq m, \forall 1 \leq j \leq n. \quad (4)$$

69 3 Solving the GQAP with TS-GQAP

70 3.1 Basic concepts

71 We start with a short summary of the constructive solution procedure, which was
 72 in its original form presented as *Guided Adaptive Relaxation Rounding Procedure*
 73 (GARRP) in [7]. GARRP starts with an empty (infeasible) solution and iteratively
 74 creates partial solutions by fixing exactly one agent-task-assignment $x'_{ij} := 1$ at
 75 each iteration, based upon the optimum solution of a relaxation of BQP, leav-
 76 ing a subset of remaining decision variables as free. These partial solutions, X' ,
 77 i.e. incomplete or partial assignments of jobs j to single specific agents i , are
 78 stored within a tree, in which each tree-node represents one assignment made.
 79 So for any (partial) solution all fixed assignments left can be found by peg-
 80 ging links back to the root, which stores the result of the first relaxation solved
 81 (compare [4]). At any node, relaxed, i.e. continuous values x_{ij}^* received from a
 82 GQAP-LP-solution, define choice-probabilities among the k -best possible suc-
 83 ceeding fixations $x_{ij}^* \geq x_{LB}$ (k and x_{LB} being parameters), from which the most
 84 likely option is chosen. Parameters k and x_{LB} are user inputs. The whole tree is
 85 explored in a depth-first manner, if needed investigating all potential successor
 86 assignments according to a width-second sequence. If at some node all successor
 87 options are unsuccessfully investigated (in terms of feasibility defined by capac-
 88 ity constraints), a backtracing process starts until a node with a free successor,
 89 becoming investigated, is found or until all nodes failed because of overall in-
 90 feasibility. Normally, this iterative process stops as soon as n assignments have
 91 been made with a complete and feasible solution, otherwise it fails.

92 The present work, focusing on improvement methods for the GQAP, has its
 93 roots in the *Magnifying Glass Heuristic* approach, which was already success-
 94 fully applied to the *Quadratic Travelling Salesman Problem* in [10]. Then, for
 95 the first time, this approach was adopted as MG-GQAP [2] to improve (heuristic)
 96 GQAP solutions. As might be expected, while the results were quite satisfying,
 97 it turned out that there was still room for improvement. First, we considered the
 98 possibility to include the quadratic components into the linear part by compress-
 99 ing the quadratic coefficient matrix into a one-dimensional vector, by means of
 100 techniques borrowed from data dimensionality reduction approaches, like princi-
 101 pal component analysis. Specifically, we computed when possible (it has always
 102 been) the eigenvalues of the coefficient matrix and used them in our heuristic.
 103 The resulting code was efficient in the GQAP part, but the computation of the
 104 eigenvalues was very demanding in case of big instances, moreover local search
 105 was used in the method and it appeared as if the method was apparently not
 106 sufficiently capable to escape from local optima. So, the idea was to dismiss lin-
 107 earizations and, sort of the other way round, to use linear programming as an
 108 improvement to a TS. This led to the design of the TS-GQAP, which combines
 109 two distinct parts, the TS-neighbourhood and the MG-GQAP.

110 For the TS and its neighbourhood the performance of two prominent stan-
 111 dard operators was analyzed: either two jobs swap machines (*exchange*) or a
 112 job changes to another agent (*insert*). We contrasted using these two operators

113 together versus using them specifically. And since exclusive use of insert-moves
 114 turned out to work best, this setting was chosen. In the course of the exploration
 115 the TS sequentially checks all potential re-assignments of tasks. In doing so, a
 116 neighbourhood of admissible TS_nh_size trial moves is build by using a short-
 117 term recency tabu-memory. This memory is defined on the time, i.e. iteration,
 118 when a job is assigned to a new agent. Using this information, a newly posi-
 119 tioned task is not allowed to be removed from its agent for a parametric number
 120 of *tenure* iterations. As aspiration criterion, the best-improvement-rule is used,
 121 in which case the tabu status is released from a given trial move if it improves
 122 the incumbent best solution.

123 MG-GQAP can be seen as a variable fixing heuristic, which decomposes the
 124 overall problem into a fixed and a free component. Thus, interpreting release as
 125 destruction, the method shares similarities with a *Large Neighbourhood Search*,
 126 where solutions are partially destroyed and repaired by corresponding operators;
 127 a process, which is iteratively repeated until a local optimum is reached (see [9]).
 128 For the GQAP the corresponding idea works like this: according to pre-defined
 129 selection patterns, we consider K chosen columns (jobs, in constraints (3) of
 130 BQP) and create a new auxiliary instance containing only these K columns
 131 (and all rows). Firstly, linear costs p_{ij} are modified including the quadratic costs
 132 caused by relations between the assignment of j to i and the already fixed assign-
 133 ments outside of the chosen columns. Then, after re-adjusting the total amounts
 134 of resources a_i in order to reflect the remaining capacities, we solve the auxil-
 135 iary problem optimally and get a new, possibly improved solution with changes
 136 restricted to the K chosen columns. Note that alternatively also subsets of *rows*
 137 (agents, constraints (2)) may be selected to define new auxiliary subproblems,
 138 which establishes the differentiation between a column- and a row-oriented vari-
 139 ant of MG-GQAP. The overall process ends after a total number of given iterations.
 140 It should be noted that the idea behind MG-GQAP is even capable of serving as
 141 a construction procedure. In that case it starts from an empty unfeasible solu-
 142 tion, iteratively increasing the size of a partial solution as long as assignments
 143 are found, which were not prevented by binding capacity restrictions, while ulti-
 144 mately and ideally constructing a full feasible solution with n assignments. We
 145 give this simple 1-pass start heuristic the name MG-GQAP-C and report results in
 146 the computational section.

147 In summary, within TS-GQAP, the TS utilizes the LP-part in order to optimize
 148 a good local solution in terms of intensification. The role of diversification is
 149 taken over by the occasional use of elite solutions collected in a *pool* by the TS.
 150 Section 3.2 explains this component and covers the algorithmic details.

151 3.2 Implementation details

152 It is well-known that metaheuristic developments have proved to be successful
 153 especially in cases, where their fundamental concepts are complemented with
 154 pool-oriented approaches, originally rooted in Genetic Algorithms or Scatter
 155 Search and Path Relinking. These algorithms maintain a reference set of high

156 quality solutions, which are repeatedly used during the search in order to guar-
 157 antee a fruitful balance between diversification and intensification (see, e.g. [3]).
 158 Therefore, Algorithm 1: TS-GQAP uses a pool structure as follows.

```

159 Require: a GQAP instance with a solution  $X := [x_{ij}]^{m \times n} \in \{0, 1\}^{m \times n}$ 
160 Ensure: new solution  $X_{new}$  with  $c(X_{new}) \leq c(X)$ 
161 1:  $X^* := X$ ,  $c_{best} := c(X^*)$ 
162 2: repeat
163 3:   repeat // TS-phase:
164 4:      $X' := best\_admissible(TS\_neighbourhood(X))$ 
165 5:     if  $c(X') < c_{best}$  then
166 6:       update  $X^* := X'$ 
167 7:        $c_{best} := c(X')$ 
168 8:     end if
169 9:     maintain a set pool of good and diverse solutions
170 10:  until  $TS\_termination$ 
171 11:  if  $TS\_failed$  then
172 12:     $X' := unchecked\_solution(pool)$ 
173 13:  end if
174 14:  repeat // MG-GQAP-phase:
175 15:     $Col := select\_variable\_columns(X')$ 
176 16:     $X_{LP} := LP\_from(Col)$ 
177 17:     $X_{LP}^* := solve(X_{LP})$  // CPLEX
178 18:     $X'' := combine\_solutions(X', X_{LP}^*)$ 
179 19:    if  $c(X'') < c_{best}$  then
180 20:      update  $X^* := X''$ 
181 21:       $c_{best} := c(X'')$ 
182 22:    end if
183 23:  until  $MG\_QAP\_termination$ 
184 24:   $X := X''$ 
185 25: until  $TS\_GQAP\_termination$ 
186 26: return  $x_{new} := X^*$ 
187

```

Algorithm 1: TS-GQAP

188 As an improvement procedure, the TS-GQAP builds on a feasible solution X
 189 with objective function value $c(X)$. It consists of alternating TS- and MG-GQAP-
 190 phases (lines 3 – 10 and 14 – 23), i.e. it is a series of TS1, MG-GQAP1, TS2,
 191 MG-QAP2, . . . until an overall termination criterion, $TS_GQAP_termination$,
 192 gets true. Each TS is capable of maintaining the pool of elite solutions. Such
 193 solutions are collected to be used in future iterations of the search process.
 194 Solution X' in line 4 is the actual and best admissible solution iteratively drawn
 195 from consecutive TS-neighbourhoods, which involves the maintenance and usage
 196 of the TS-memory as described above.

197 The pool maintained by the TS, in line 9, collects a maximum of $pool_size$
 198 good and diverse solutions. In doing so, a solution is deemed good at the moment,
 199 when it has just been improved by the TS, i.e. goodness means objective function

200 value and diversity targets the structural difference mapped in the values of
 201 the decision variables, i.e. assignments. In the current version such diversity
 202 is – at least with higher probability – ensured by allowing into the pool only
 203 members with different objective function values. The diversification step is done
 204 in lines 11 – 13 as soon as the TS was not able to improve the current solution,
 205 expressed by a value *true*, returned from function *TS_failed*. To diversify the
 206 search, function *unchecked_solution(pool)* extracts the cost-minimum solution
 207 from the pool, which has not yet been output from the pool to be processed
 208 by the MG-GQAP. In the case that all pool solutions are already processed, this
 209 procedure forwards the running, i.e. the best solution from the last TS-phase
 210 to the MG-GQAP. For function *select_variable_columns(X')* in line 15, which
 211 stores free assignments to be optimized by an LP in set *Col*, we considered
 212 a number of possibilities. For the determination of these $K := |Col|$ columns,
 213 several selection mechanisms were tested. These include *random* selection, so-
 214 called *plane* selection of columns $((1,2,3), (4,5,6), \dots, (n-2, n-1, n); (2,3,4),$
 215 $(5,6,7), \dots)$ or *binomial* selection with columns $((1), (2), (3), (1,2), (1,3), (2,3),$
 216 $(1,2,3); (2), (3), (4), \dots)$, all examples underlying $K = 3$. Trends in efficiency are
 217 non-random strategies \prec purely random \prec mixed-random options. The chosen
 218 strategy is a half and a half mixture of *random* and *plane*.

219 After building the LP-subproblem by *LP_from(Col)* and solving it, in lines
 220 16 and 17, respectively, procedure *combine_solutions(X', X_{LP}^{*})* adds the opti-
 221 mized values $x_{ij_{LP}}^*$ to the starting solution X' and thus includes the LP-
 222 optimized assignments of X_{LP}^* . If necessary, this is followed by an update of
 223 the best solution, the same way as it is done for the TS in lines 5 – 8. At the very
 224 end, with *TS_GQAP_termination* getting true, the overall process stops and
 225 the algorithm returns its best solution found as X_{new} .

226 4 Computational results

227 For the evaluation of the computational results we used three test beds with a
 228 total of 64 instances. Abbreviated with LAM, CEAL and OWN, there are 27
 229 instances with $m = 6 - 30$ and $n = 10 - 16$ from Lee and Ma [5], 21 instances
 230 with $m = 6 - 20$ and $n = 20 - 50$ from Cordeau et al. [1] and 16 own randomly
 231 generated instances with $m, n = 10 - 200$. All algorithms were implemented in
 232 AMPL script V.20220310, using the solver CPLEX V.20.1.0.0 (MS VC++ 10.0,
 233 64-bit). All runs were performed on a ThinkPad X1 notebook with an Intel(R)
 234 Core(TM) i7-8550U CPU @1.80GHz (Aug. 2017) under Windows 10 Pro with 8
 235 GB RAM.

236 We aimed for a unified parameter setting, but eventually this was only possi-
 237 ble to a certain extent due to the large structural differences in the data sets:
 238 symmetry of the data matrices D and F , constancy of weights in W over the
 239 agent set and, very basic, instance sizes, where the latter directly affect and deter-
 240 mine calculable LP subproblems, regarding the value of parameter K . To profit
 241 from faster calculations, a relatively low value of $TS_nh_size = 3$ is chosen.
 242 Tabu time becomes $tenure \approx 0.3n$ and $pool_size = 20$. More numeric paramet-

243 ric details are reported below. Evaluation is split into two parts, judgement of
 244 the general design of the construction and improvement procedures, thereafter
 245 in part two supplemented by the discussion of the specific results obtained for
 246 the three data sets.

247 In order to test and evaluate the general design of the construction pro-
 248 cedures, we used all 3 tests sets. Starting with the tree heuristic **GARRP** and
 249 **LAM**, parameter settings were $x_{LB} = 0.5$ and $k = 50$. The procedure needs
 250 $min|avg|max = 6|2375|18364$ tree nodes and $min|avg|max = 0.3|168.5|1598.4$
 251 seconds (sec.). The second test set, **CEAL**, is solved with $x_{LB} = 0.85, k = 25$
 252 and $min|avg|max = 21|1842|16103$ nodes, $min|avg|max = 1.1|391.6|3600$ sec.
 253 Note that while feasible solutions for all instances of **LAM** can be built, the
 254 method fails in two cases (#11 and #15) for **CEAL**, i.e. those ones for which
 255 one hour of computing time was not enough. Here starting solutions can be pro-
 256 vided by solving the standard (linear) **GAP**. Results for set **OWN**, solved with
 257 $x_{LB} = 0.9, k = 30$, are $min|avg|max = 11|30.6|201$ nodes and $min|avg|max =$
 258 $0.3|65.3|707.1$ sec. Again, as in **LAM**, no infeasible solution had to be accepted.
 259 Not much can be derived from these numbers, but two statements appear to
 260 be meaningful: the number of nodes, equivalent to the number of solved LPs,
 261 is relatively big for specific instances, which indicates a large CPU load. Sec-
 262 ondly, it is a pleasing fact that the success rate of the procedure is quite high
 263 (62 of 64 cases). An objective-oriented reasoning of **GARRP** goes along with the
 264 judgement of the magnifying glass approach used as a construction procedure,
 265 i.e. algorithm variant **MG-GQAP-C**. This time, exclusively based on **CEAL**, set-
 266 ting $K = 4$, three **MG-GQAP-C**-runs with $MG_QAP_term = 500|1000|10000$
 267 iterations are performed. The hypothesis is that with an increasing number of
 268 iterations, i.e. with higher computing time, the numerical quality of the overall
 269 best solution will increase too, though by an unknown and decreasing amount.
 270 These expectations are met as follows. Designating the result of the 500-run as
 271 a basis, the relative stepwise improvements obtained between the 500- and the
 272 1000-result and between the 1000- and the 10000-result, averaged over the ob-
 273 jective values of all 21 instances, are 2.7% and 0.8%. Moreover, contrasting the
 274 best **MG-GQAP-C**-result, the 10000-result with **GARRP**, it becomes obvious that the
 275 tree-procedure works even better with an averaged additional 3.9% performance
 276 gain. It is interesting to note that these numerical ratios apply not only for the
 277 construction process but also for an improvement process that builds exactly
 278 on the former starting solutions. Total running time (min.) naturally gradually
 279 increases: 14.2|33.0|80.6 (**MG-GQAP-C**) vs. 137.1 (**GARRP**).

280 Next, we appraise the design of the proposed TS improvement procedure
 281 **TS-GQAP**. Essentially, it consists of a TS component (TS-phase) and an LP com-
 282 ponent (**MG-GQAP**-phase). So it is obvious to isolate the individual components
 283 and to compare three test runs: (1) only **MG-GQAP**-phase, (2) only TS-phase
 284 and (3) both phases combined (=TS-GQAP). Like above, test set **CEAL** is used,
 285 input comes in all cases from the tree start heuristic **GARRP** and parametrization
 286 from a unified, single parameter set. The outcome for (1) is an average improve-
 287 ment of $\delta_{tree}^{avg} = 5.75\%$ (in 35.1 min., whole set), for (2) it is 2.5% (38.0 min.)

288 and finally for (3), the TS-GQAP, it is 6.39% (44.1 min.). Thus, even if the overall
 289 concept gets sufficiently motivated, some captious comments seem appropriate.
 290 Again, as an logical implication, objective function improvement comes at the
 291 expense of CPU time. In the present context, however, the contribution of the
 292 TS-phase is smaller than that of the MG-GQAP-phase. However, this is not
 293 surprising since the optimizing component based on an optimum solution algo-
 294 rithm will generally generate more visible improvements than a heuristic one.
 295 One can also see that the neighbourhood structure of the TS, which is more
 296 complex in terms of implementation, has a significant impact. Nonetheless, the
 297 contribution of the TS is also a significant one, which is only underscored by the
 298 overall effectiveness of the method.

299 The second part of the computational results covers the evaluation of the
 300 specific results calculated for the three test sets. Moderately sized instances of
 301 LAM allow the use of a mixed strategy. The number of LP-columns as well as
 302 the number of LP-rows is set with $K = 4$ and the LP-build strategy, column
 303 or row-oriented, is changed every 25 iterations. Termination parameters are set
 304 as $TS_term = 150$, $MG_GQAP_term = 50$ and $TS_GQAP_term = 20$.
 305 The result for LAM is $\delta_{tree}^{avg} = 11.66\%$, which takes 2.4 min. averaged over all
 306 instances. It can be observed that the algorithm finds the best solutions quite
 307 early. With respect to quality it can be stated that the results appear to be good,
 308 but no optimality gaps were calculated since optimal solutions are not published.

309 The next object of observation is the test set CEAL. Again, $K = 4$ and the
 310 LP-component follows a column approach, exactly as it is described in Algorithm
 311 1, while completion criteria are given by $TS_term = 500$, $MG_GQAP_term =$
 312 100 and $TS_GQAP_term = 10$. This leads to an improvement of $\delta_{tree}^{avg} = 6.4\%$
 313 in an average of 2.2 min. Because competing objective function values are avail-
 314 able, deviations from best-known objectives can be determined. They are given
 315 with δ_{CEAL} and amount $min|avg|max = 0.0\%|1.4\%|8.29\%$. Results' quality
 316 clearly correlates with instance-density ρ , the ratio of total capacity demanded
 317 over total capacity available (reasonably only calculable for server independent
 318 constant demands, as it is the case with CEAL and LAM). As already indicated
 319 the two problem instances, namely #11 and #15, cannot be improved. It should
 320 also be stated in an exculpatory manner that average quality gets destroyed by
 321 only a few outliers. Compensating these, an acceptable $\delta_{CEAL}^{avg} = 0.9\%$ can be
 322 achieved.

323 The third and last set, OWN, contains the hardest instances: asymmet-
 324 ric, non-constant weights and sizes up to $m, n = 200$. It is solved with the
 325 same column-oriented LP-strategy as used for CEAL. We set $K = 3$, how-
 326 ever, due to advanced dimensions, it was necessary to reduce the value of K
 327 to 2 for instances with a high $m = 200$. Termination variables are given by
 328 $TS_term = 150$, $MG_GQAP_term = 25$ and $TS_GQAP_term = 40$. With
 329 these parameters we can achieve a $\delta_{tree}^{avg} = 34.46\%$ in average 11.0 min. In the
 330 OWN case running time utilization is more efficient since for some instances the
 331 best result is only achieved after 90% of the total running time. The majority of

332 instances ($\approx 11, 12$ out of 16) cannot be solved optimally with the means at our
 333 disposal, hence no reasonable deviations can be calculated.

334 As a final remark it should be noted that for all test sets, LAM, CEAL and
 335 OWN, algorithm TS-GQAP constitutes an improvement over the old MG-GQAP as
 336 described in [2]. In terms of objective value amount this progress is not outstand-
 337 ing, but it is clearly visible. It comes either as an increase in CPU productivity,
 338 i.e. percentage improvement divided by CPU time used, or actually as an im-
 339 provement of the (average) objective.

340 5 Summary, criticism and outlook

341 This work is a further step in an ongoing research project looking for heuristic
 342 solution procedures for the GQAP. We combine the well-known metaheuristic
 343 TS with the strengths of mathematical programming and introduce a new
 344 matheuristic referred to as TS-GQAP. The basic idea behind it was originally
 345 coined as Magnifying Glass Heuristic, itself a matheuristic and successfully used
 346 for a quadratic *Travelling Salesman Problem*. The design of the new method
 347 turned out to be successful in terms of CPU usage and the ability to deal with
 348 larger problem sizes, while also specific results could be improved. Moreover, in
 349 terms of competition, the method proved to be able to keep up with algorithms
 350 from literature and the best-known-gaps could be reduced by another level.

351 Even if the new algorithmic design is promising, there exist clearly visible
 352 improvement opportunities. New challenges raised are those about the course
 353 of the interaction between the TS neighbourhood and the LP decomposition or
 354 about the implementation of an overarching memory structure. Very basically,
 355 a stumbling block on the way to outstanding performance is founded in the ca-
 356 pacity restricted nature of the underlying problem, which logically is intricate
 357 to navigate. Fast metaheuristics are able to play off their superiority for prob-
 358 lem classes, which are unrestricted or endowed with a large number of feasible
 359 solutions (dense solution space) and often benefit from easier objective function
 360 calculations. As observed, the explorable solution space, more accurately, the
 361 neighbourhood induced solution landscape for some GQAP instances investi-
 362 gated is quite sparse. It has already been put forward [6] that sparse solution
 363 spaces are indicative of cases where matheuristics are probably more effective
 364 than plain metaheuristics, usually relying on local search or simple construc-
 365 tive procedures at their core. Moreover, GQAP is a representative problem of
 366 nonlinear combinatorial optimization, an area that so far received much less at-
 367 tention from research than its linear counterpart, despite its obvious relevance
 368 for modelling and solving compelling real-world problems.

369 This opens sufficient room to set up and tune new neighbourhood mecha-
 370 nisms to be developed, an endeavor, which of course cannot be done apart from
 371 designing more efficient memory structures. It is precisely this problem area that
 372 must be examined and analyzed more closely in the future.

373 **References**

- 374 1. Cordeau, J.F., Gaudioso, M., Laporte, G., Moccia, L.: A memetic heuristic for the
375 generalized quadratic assignment problem. *INFORMS Journal on Computing* **18**,
376 433–443 (11 2006). <https://doi.org/10.1287/ijoc.1040.0128>
- 377 2. Greistorfer, P., Staněk, R., Maniezzo, V.: The magnifying glass heuristic for the
378 generalized quadratic assignment problem. In: *Proceeding of the XIII Metaheuris-*
379 *tics International Conference MIC 2019*. pp. 22–24. Universidad de los Andes Sede
380 Caribe (2019)
- 381 3. Greistorfer, P., Voß, S.: *Controlled Pool Maintenance for Metaheuristics*, pp. 387–
382 424. Springer US, Boston, MA (2005). https://doi.org/10.1007/0-387-23667-8_18
- 383 4. Kaufman, L., Broeckx, F.C.: An algorithm for the quadratic assignment problem
384 using Bender’s decomposition. *European Journal of Operational Research* **2**(3),
385 207–211 (1978)
- 386 5. Lee, C., Ma, Z.: The generalized quadratic assignment problem. *Research Re-*
387 *port, Department of Mechanical and Industrial Engineering, University of Toronto*,
388 (2003)
- 389 6. Maniezzo, V., Boschetti, M., Stützle, T.: *Matheuristics: Algorithms and Imple-*
390 *mentations*. EURO Advanced Tutorials on Operational Research, Springer Inter-
391 national Publishing (2021)
- 392 7. Maniezzo, V., Greistorfer, P., Staněk, R.: Exponential neighborhood search for
393 the generalized quadratic assignment problem (2018), *eURO/ALIO International*
394 *Conference on Applied Combinatorial Optimization*, 25.–27.6.2018, Bologna, Italy
- 395 8. Maniezzo, V., Stützle, T., Voß, S.: *Matheuristics: Hybridizing Metaheuristics and*
396 *Mathematical Programming*. Springer Publishing Company, Incorporated, 1st edn.
397 (2009)
- 398 9. Ropke, S., Pisinger, D.: An adaptive large neighborhood search heuristic for the
399 pickup and delivery problem with time windows. *Transportation Science* **40**(4),
400 433–443 (2006)
- 401 10. Staněk, R., Greistorfer, P., Ladner, K., Pferschy, U.: Geometric and LP-based
402 heuristics for the quadratic travelling salesman problem. *Computers & Operations*
403 *Research* **108**, 97–111 (2019)