UNIVERSIDAD AUTONOMA DE NUEVO LEON

FACULTAD DE INGENIERIA MECANICA Y ELECTRICA

Selected Topics on Optimization

Final Report

The Use of Heuristics for the Quadratic Assignment Problem

TEAM J

Student Name	Student ID	Student Career
Omar Alejandro Alvarez Chavez	2097224	ITS
Jonathan Alexander Reyes Garcia	2146387	ITS
Edmundo Santos Ramirez	2148132	ITS
Eduardo Alejandro Acevedo Liuskos	2142139	ITS

Group: 002

Semester: January-June 2026

Instructor: Dr. Roger Zirahuén Ríos Mercado

Date: 09/05/2026

San Nicolas de los Garza, Nuevo León

1. Introduction and Problem Motivation

The Quadratic Assignment Problem (QAP) is one of the most important and difficult problems in combinatorial optimization. In simple words, the QAP asks how to assign a set of facilities to a set of locations in a way that minimizes the total interaction cost. This cost depends on two things at the same time: how much two facilities interact with each other and how far apart their assigned locations are. For that reason, the model is called "quadratic": the cost of one assignment is not independent, because it depends on the assignment of other facilities as well (Koopmans C Beckmann, 1957; Lawler, 1963).

A useful analogy is the layout of departments inside a factory. If two departments exchange a large amount of material every day, they should be placed close to each other. If they are placed far apart, the company wastes time, labor, and transportation effort. The same idea applies to hospitals, warehouses, schools, electronic circuits, and computer networks. The QAP captures this logic mathematically.

The main motivation for using heuristics is that the QAP is computationally hard. For n facilities and n locations, there are n! possible assignments. This means that a 10-location problem already has 3,628,800 possible layouts, and the number grows extremely fast. Because of this explosive growth, exact optimization methods can require too much time for medium and large instances. QAPLIB was created to collect benchmark instances and track best-known solutions, showing the importance of the QAP as a standard test problem for optimization algorithms (Burkard et al., 1997).

Heuristics are practical because they do not guarantee the global optimum, but they can produce good solutions quickly. In real applications, decision makers often need a strong solution within seconds or minutes rather than a perfect solution after a long computational search. Therefore, the study of constructive heuristics and local search methods is highly relevant for practical optimization.

1.1 Practical Applications

- Facility layout: placing machines, departments, or service areas to reduce transportation and handling costs.
- Electronics: placing components on printed circuit boards to reduce wire length and signal delay.
- Hospital and service design: locating departments with high patient movement close to each other.
- Computer systems: mapping software processes to processors when some processes communicate more frequently.
- Data analysis and ranking: ordering objects when relationships or similarities between pairs must be preserved.

2. Problem Description

The QAP contains three basic elements: input data, decisions, and objective. The input data are usually represented by a flow matrix and a distance matrix. The decision is a permutation that assigns every facility to exactly one location. The objective is to minimize the total cost generated by the interaction of all assigned pairs.

Element	Meaning in the QAP	Example interpretation
Facilities	Objects that must be assigned	Departments, machines, computer processes
Locations	Available positions for the facilities	Rooms, physical spaces, processors
Flow matrix F	Interaction between pairs of facilities	Material movement, communication frequency
Distance matrix D	Distance between pairs of locations	Meters, travel time, cable length
Fixed cost matrix B	Optional cost of placing facility i in location k	Installation cost or compatibility penalty
Permutation π	Assignment rule: facility i is placed in location π(i)	A complete layout plan

The QAP differs from a linear assignment problem because the cost is not only based on a single facility- location match. Instead, the cost depends on the relationship between pairs of assigned facilities. This is the reason why a small swap can change many cost terms at the same time.

3. Mathematical Model

3.1 Permutation Form

Let N = {1, 2, ..., n} be the set of facilities and locations. Let f_ij be the flow between facilities i and j, d_kl be the distance between locations k and l, and b_i,π(i) be the fixed cost of assigning facility i to location π(i). The QAP can be written as:

$$\textbf{minimize } z(\pi) = \sum_{i=1}^{n} \sum_{j=1}^{n} f_{ij}\, d_{\{\pi(i),\pi(j)\}} + \sum_{i=1}^{n} b_{\{i,\pi(i)\}}$$

The first term is the quadratic interaction cost. It multiplies the flow between two facilities by the distance between the locations where those facilities are placed. The second term is optional and represents direct assignment costs. In this report, the numerical example assumes zero fixed costs to focus on the quadratic part of the problem.

3.2 Binary Assignment Form

The same problem can also be expressed with binary decision variables. Let x_ik = 1 if facility i is assigned to location k, and x_ik = 0 otherwise. The formulation is:

$$\textbf{minimize } z = \sum_i \sum_j \sum_k \sum_l f_{ij}\, d_{kl}\, x_{ik}\, x_{jl} + \sum_i \sum_k b_{ik}\, x_{ik}$$

$$\textbf{subject to } \sum_k x_{ik} = 1, \textbf{ for every facility } i$$

$$\textbf{subject to } \sum_i x_{ik} = 1, \textbf{ for every location } k$$

$$x_{ik} \in \{0,1\}, \textbf{ for every facility } i \textbf{ and location } k$$

The first set of constraints ensures that each facility is assigned to exactly one location. The second set ensures that each location receives exactly one facility. The binary condition prevents fractional assignments. This formulation shows why the problem is difficult: the objective contains products of decision variables.

4. Complete Example Problem (n = 4)

To show how the QAP objective is evaluated, this section uses a small instance with four facilities A, B, C, and D and four locations 1, 2, 3, and 4. Fixed assignment costs are assumed to be zero.

4.1 Input Matrices

Table 1 presents the flow between facilities. Higher values indicate stronger interaction. Table 2 presents the distance between locations.

	A	B	C	D
A	0	5	2	4
B	5	0	3	1
C	2	3	0	6
D	4	1	6	0

Table 1. Flow matrix F

	1	2	3	4
1	0	2	6	4
2	2	0	5	3
3	6	5	0	1
4	4	3	1	0

Table 2. Distance matrix D

4.2 Assignment Evaluation

Consider the following two candidate assignments:

Assignment	Facility-location mapping	Total cost	Interpretation
Assignment 1	A→1, B→2, C→4, D→3	124	Better than Assignment 2
Assignment 2	A→1, B→3, C→4, D→2	144	Worse because high-flow pairs are farther apart

For Assignment 1, the largest flow is between C and D, with f_CD = 6. These facilities are assigned to locations 4 and 3, whose distance is d_43 = 1. This is good because a high-flow pair is placed very close. For Assignment 2, some high-flow relationships are assigned to longer distances, which increases the objective value. Since 124 < 144, Assignment 1 is the better solution among these two alternatives.

A local search procedure would continue exploring swaps. For example, if swapping two facilities reduces the total cost, the swap is accepted. The process stops when no single swap improves the current layout.

5. Description of Heuristics

This report uses a two-phase heuristic framework. The first phase constructs an initial solution quickly. The second phase improves that solution through local swaps. This structure is common in combinatorial optimization because a good initial solution helps the improvement phase start from a stronger position.

5.1 Constructive Heuristic

The constructive heuristic is based on a simple idea: facilities with high total flow should be placed in central locations. A facility with high total flow interacts strongly with many other facilities, so assigning it to a location with low total distance tends to reduce the total cost. This heuristic does not test every possible layout; it builds one layout step by step.

Algorithm 1. Greedy constructive heuristic

Step	Operation
1	Compute total flow for every facility: $F_i = \Sigma_j f_{ij}$.
2	Compute location centrality for every location: $C_k = \Sigma_l d_{kl}$. Lower C_k means a more central location.
3	Sort facilities from highest total flow to lowest total flow.
4	Assign each facility, in that order, to the available location with the lowest centrality and lowest fixed cost.

The strength of this method is speed and simplicity. The weakness is that it is myopic: once it assigns a facility, it does not automatically reconsider the decision. Therefore, the greedy phase is useful as a starting point, but it normally needs an improvement phase.

5.2 Local Search Heuristic

The local search heuristic improves the greedy solution by exploring swap moves. A neighbor is created by selecting two facilities and swapping their assigned locations. If the swap reduces the total cost, it improves the solution. To make the experiment scalable and reproducible, this report uses a bounded 2-opt candidate-list strategy. For n = 50 and n = 100, the implementation evaluates the complete swap neighborhood in each pass and applies the best improving swap. For n = 500, it evaluates a fixed candidate list of 2,500 randomly generated swaps per pass using the same seed logic. The final reported value is therefore the best solution found within the specified computational limit, not a guaranteed global optimum.

Algorithm 2. Bounded 2-opt local search heuristic

Step	Operation
1	Start with the solution created by the greedy heuristic.
2	For every pair of facilities (a,b), swap their assigned locations temporarily.
3	Estimate the QAP cost after the swap.
4	Select the swap that produces the largest cost reduction.
5	Repeat until no swap improves the solution. The final solution is a local optimum.

The local search phase is more expensive than the greedy phase because it evaluates swap moves. However, the bounded version remains practical even for large instances because it controls the number of candidate swaps and iterations. The key advantage is that it corrects poor greedy choices by moving facilities to better positions. More advanced heuristics for the QAP, such as Tabu Search and Iterated Local Search, also rely heavily on neighborhood exploration (Taillard, 1991; Stützle, 2006).

6. Computational Experience

6.1 Experimental Design and Test Runs

The experiment was redesigned to evaluate the same two-phase heuristic under three fixed instance sizes. The small group contains 20 instances with n = 50, the medium group contains 20 instances with n = 100, and the large group contains 20 instances with n = 500. Each instance was generated with a reproducible seed. The flow matrix was symmetric with zero diagonal entries, the distance matrix was obtained from random two-dimensional coordinates, and a small fixed-cost matrix was included to represent direct assignment penalties.

For every instance, three solution values were recorded. First, a random baseline was calculated as the best solution among 20 random permutations. Second, the greedy constructive heuristic was applied. Third, bounded 2-opt local search improved the greedy solution. For n = 50 and n = 100, every pairwise swap was considered in each pass. For n = 500, 2,500 seeded candidate swaps were evaluated in each pass to keep the computation practical. The main performance metric was percentage cost reduction:

Improvement (%) = ((Greedy cost - Local Search cost) / Greedy cost) x 100

Because the QAP is a minimization problem, lower objective values are better. Therefore, a positive percentage improvement means that the local-search phase successfully reduced the cost obtained by the greedy heuristic. The column LS vs random reports the percentage reduction achieved by local search compared with the best random baseline.

6.2 Computational Results

Computational results for small instances.

Instance	Best random	Greedy cost	LS cost	Improvement	LS vs random	Iterations	LS time ms
qap50_4050	3,158,541	3,077,757	2,993,248	2.75%	5.23%	200	93.15
qap50_4051	3,087,383	3,063,045	2,945,718	3.83%	4.59%	200	90.69
qap50_4052	3,113,555	3,058,406	2,928,700	4.24%	5.94%	200	86.55
qap50_4053	3,125,887	3,127,058	3,013,417	3.63%	3.60%	200	86.27
qap50_4054	3,015,717	2,981,075	2,815,989	5.54%	6.62%	200	85.56
qap50_4055	3,242,957	3,231,393	3,027,186	6.32%	6.65%	200	86.32
qap50_4056	2,877,834	2,845,331	2,743,944	3.56%	4.65%	200	79.83
qap50_4057	2,806,170	2,773,473	2,615,965	5.68%	6.78%	200	84.86
qap50_4058	3,192,530	3,134,090	2,988,255	4.65%	6.40%	200	83.07
qap50_4059	3,059,325	3,021,517	2,851,504	5.63%	6.79%	200	84.86
qap50_4060	3,213,375	3,172,300	3,014,440	4.98%	6.19%	200	83.03
qap50_4061	2,944,296	2,865,151	2,749,912	4.02%	6.60%	200	81.77
qap50_4062	2,956,699	2,890,488	2,745,426	5.02%	7.15%	200	81.82
qap50_4063	3,026,548	2,973,204	2,862,456	3.72%	5.42%	200	83.10
qap50_4064	3,368,498	3,319,949	3,182,356	4.14%	5.53%	200	83.54
qap50_4065	3,081,017	3,026,417	2,865,519	5.32%	6.99%	200	91.93
qap50_4066	3,241,956	3,167,167	3,075,550	2.89%	5.13%	200	88.33
qap50_4067	3,152,239	3,072,776	2,956,149	3.80%	6.22%	200	84.92
qap50_4068	3,147,651	3,066,203	2,958,612	3.51%	6.01%	200	87.86
qap50_4069	3,034,014	2,980,204	2,828,347	5.10%	6.78%	200	86.97

Computational results for medium instances.

Instance	Best random	Greedy cost	LS cost	Improvement	LS vs random	Iterations	LS time ms
qap100_4100	12,604,161	12,421,679	11,984,584	3.52%	4.92%	220	655.70
qap100_4101	13,192,214	13,106,243	12,567,195	4.11%	4.74%	220	664.04
qap100_4102	12,771,281	12,547,642	12,103,360	3.54%	5.23%	220	664.27
qap100_4103	12,008,731	11,776,934	11,472,205	2.59%	4.47%	220	655.16
qap100_4104	12,601,287	12,483,804	11,962,927	4.17%	5.07%	220	650.72
qap100_4105	13,100,473	12,967,275	12,322,200	4.97%	5.94%	220	642.11
qap100_4106	13,091,237	12,972,885	12,499,784	3.65%	4.52%	220	651.12
qap100_4107	12,734,175	12,529,328	12,104,405	3.39%	4.95%	220	646.83
qap100_4108	13,373,154	13,140,306	12,704,988	3.31%	5.00%	220	648.01
qap100_4109	13,271,634	13,104,240	12,632,079	3.60%	4.82%	220	697.30
qap100_4110	12,996,986	12,802,507	12,365,775	3.41%	4.86%	220	621.46
qap100_4111	13,527,870	13,444,936	12,820,651	4.64%	5.23%	220	624.70
qap100_4112	13,219,969	13,089,273	12,598,651	3.75%	4.70%	220	631.62
qap100_4113	12,659,122	12,465,237	12,047,641	3.35%	4.83%	220	637.05
qap100_4114	13,003,943	12,892,685	12,296,970	4.62%	5.44%	220	692.81
qap100_4115	12,754,471	12,655,190	12,089,998	4.47%	5.21%	220	689.51
qap100_4116	12,100,342	11,952,889	11,560,167	3.29%	4.46%	220	646.01
qap100_4117	12,902,872	12,820,082	12,291,863	4.12%	4.74%	220	644.12
qap100_4118	13,052,274	12,896,760	12,394,557	3.89%	5.04%	220	631.34
qap100_4119	12,643,425	12,466,861	11,972,148	3.97%	5.31%	220	635.38

Computational results for large instances.

Instance	Best random	Greedy cost	LS cost	Improvement	LS vs random	Iterations	LS time ms
qap500_4500	327,023,029	324,733,497	321,456,403	1.01%	1.70%	45	429.78
qap500_4501	322,171,253	320,192,006	316,862,615	1.04%	1.65%	45	415.59
qap500_4502	330,004,558	327,874,529	324,559,193	1.01%	1.65%	45	422.98
qap500_4503	323,020,061	320,992,707	317,624,434	1.05%	1.67%	45	423.04
qap500_4504	316,500,762	314,220,893	311,109,387	0.99%	1.70%	45	431.32
qap500_4505	327,145,922	324,680,451	321,451,601	0.99%	1.74%	45	418.43
qap500_4506	334,212,015	331,658,340	328,228,627	1.03%	1.79%	45	423.77
qap500_4507	324,958,129	322,518,389	319,200,359	1.03%	1.77%	45	451.14
qap500_4508	322,634,598	319,972,820	316,815,625	0.99%	1.80%	45	422.99
qap500_4509	321,214,073	318,757,591	315,594,494	0.99%	1.75%	45	423.38
qap500_4510	334,592,856	331,772,562	328,524,469	0.98%	1.81%	45	419.06
qap500_4511	322,320,029	320,173,199	317,094,589	0.96%	1.62%	45	418.21
qap500_4512	324,772,215	322,238,242	318,886,781	1.04%	1.81%	45	422.39
qap500_4513	322,765,360	320,176,492	317,045,872	0.98%	1.77%	45	465.63
qap500_4514	319,948,171	317,520,577	314,483,171	0.96%	1.71%	45	443.94
qap500_4515	319,665,424	317,384,716	314,092,477	1.04%	1.74%	45	516.44
qap500_4516	334,185,717	332,020,458	328,392,966	1.09%	1.73%	45	468.95
qap500_4517	324,458,638	322,275,416	319,158,481	0.97%	1.63%	45	431.31
qap500_4518	327,822,692	325,407,920	322,255,294	0.97%	1.70%	45	439.90
qap500_4519	328,590,268	326,496,770	323,257,172	0.99%	1.62%	45	427.82

The results show that local search improved the greedy solution in all 60 test runs. For the small instances, the average improvement over the greedy solution was 4.42%, with a minimum of 2.75% and a maximum of 6.32%. For the medium instances, the average improvement was 3.82%, with a minimum of 2.59% and a maximum of 4.97%. For the large instances, the average improvement was 1.01%, with a minimum of 0.96% and a maximum of 1.09%.

Compared with the best random baseline, the local-search solutions were also better in every category. The average LS vs random reduction was 5.96% for n = 50, 4.97% for n = 100, and 1.72% for n = 500. These values indicate that the improvement phase produced consistently better layouts than both the random baseline and the greedy construction alone.

6.3 Discussion of Results

The computational results support three main observations. First, the greedy heuristic is useful because it produces a reasonable starting point without needing to evaluate all possible assignments. This is important because the number of possible layouts grows as n!, making complete enumeration impossible for the tested instance sizes.

Second, local search is the quality-improvement component of the method. The small and medium instances showed stronger percentage reductions because the algorithm could scan the complete swap neighborhood in each pass. In contrast, the large instances used a bounded candidate-list version. This explains why the n = 500 improvements are smaller in percentage terms: the method intentionally evaluates only part of the possible swap neighborhood to keep the experiment computationally practical.

Third, runtime remained reasonable under the bounded design. The average local-search time was approximately 85.72 ms for n = 50, 651.46 ms for n = 100, and 435.80 ms for n = 500. These times should not be interpreted as universal hardware-independent values, but they are useful for comparing the three groups under the same implementation and computer environment.

Overall, the Greedy + Bounded 2-Opt method is a strong baseline for the QAP. It is easy to explain, easy to implement, and consistently improves the initial greedy solution. However, it is still a heuristic and can stop before reaching the global optimum. Future work could compare this baseline with Tabu Search, Simulated Annealing, Genetic Algorithms, or Iterated Local Search to test whether larger improvements can be obtained, especially for n = 500 instances.

7. Conclusions

This report presented a complete study of the use of heuristics for the Quadratic Assignment Problem. The QAP was explained as a facility-location problem where the cost of each assignment depends on the flows between facilities and the distances between their assigned locations. This quadratic interaction makes the problem much harder than a simple linear assignment problem.

The report developed a formal mathematical model, a complete example with flow and distance matrices, a greedy constructive heuristic, and a bounded 2-opt local search heuristic. The computational experiment was expanded to 60 test runs divided into three groups: 20 small instances with n = 50, 20 medium instances with n = 100, and 20 large instances with n = 500. The results showed that the local-search phase improved the greedy solution in every test run.

The main conclusion is that a two-phase heuristic approach is effective for the QAP. In the expanded experiment, average cost reduction was 4.42% for n = 50, 3.82% for n = 100, and 1.01% for n = 500. The lower percentage for the large group is coherent with the bounded candidate-list design, because the algorithm deliberately limits the number of swaps evaluated to preserve scalability. Even with this restriction, the method produced better solutions than greedy construction and random assignment.

For future work, the team could implement more advanced metaheuristics and compare them against the current baseline. Tabu Search or Iterated Local Search would be especially relevant because they can escape local optima and may produce stronger results on large QAP instances.

References

Burkard, R. E., Karisch, S. E., & Rendl, F. (1997). QAPLIB: A quadratic assignment problem library. Journal of Global Optimization, 10, 391-403. https://doi.org/10.1023/A:1008293323270

Koopmans, T. C., & Beckmann, M. (1957). Assignment problems and the location of economic activities. Econometrica, 25(1), 53-76. https://doi.org/10.2307/1907742

Lawler, E. L. (1963). The quadratic assignment problem. Management Science, 9(4), 586-599. https://doi.org/10.1287/mnsc.9.4.586

Loiola, E. M., de Abreu, N. M. M., Boaventura-Netto, P. O., Hahn, P., & Querido, T. (2007). A survey for the quadratic assignment problem. European Journal of Operational Research, 176(2), 657-690. https://doi.org/10.1016/j.ejor.2005.09.032

Stützle, T. (2006). Iterated local search for the quadratic assignment problem. European Journal of Operational Research, 174(3), 1519-1539. https://doi.org/10.1016/j.ejor.2005.01.066

Taillard, E. D. (1991). Robust taboo search for the quadratic assignment problem. Parallel Computing, 17(4-5), 443-455. https://doi.org/10.1016/S0167-8191(05)80147-4