Last lecture:

- flow shop
- Heuristic algorithms: dispatching rules

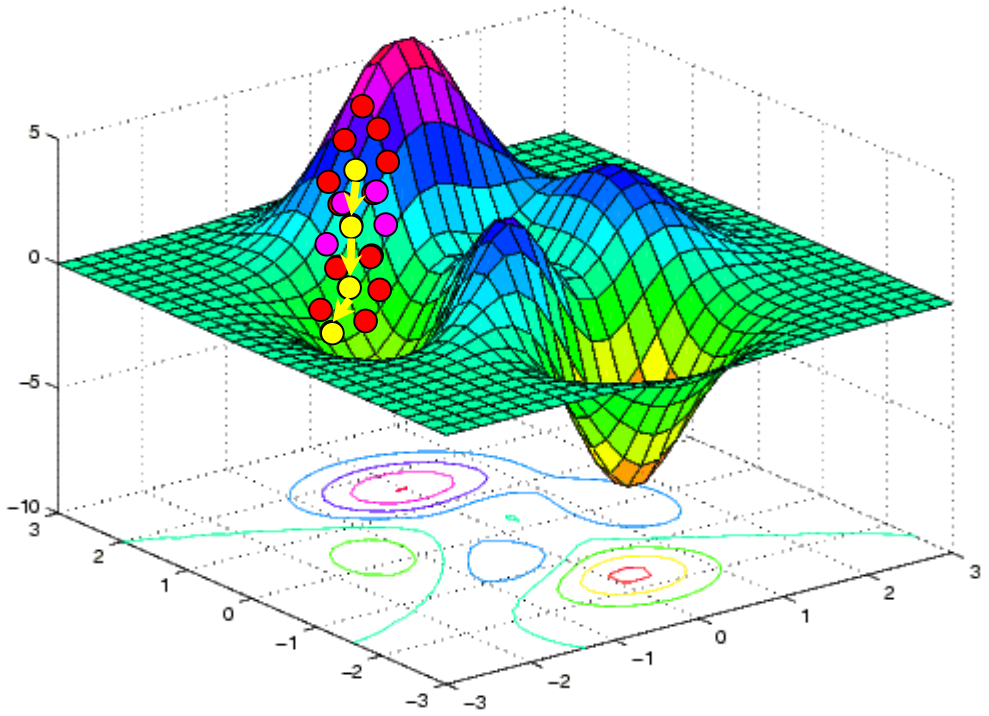
This lecture:

- Heuristic algorithms: local search

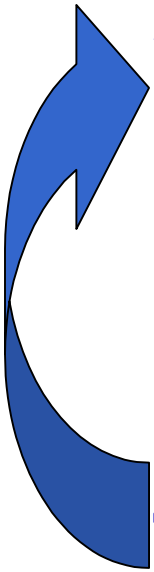
Improvement Heuristics

- Local Search Algorithms
    1. Iterative Improvement
    2. Threshold Accepting
    3. Simulated Annealing
    4. Tabu search
- Genetic algorithm

Local Search



General Local Search Procedure



1. **Initialisation.** Choose an initial schedule ***S*** to be the current solution and compute the value of the objective function ***F***(***S***).

2. **Neighbour Generation.** Select a neighbour ***S'*** of the current solution ***S*** and compute ***F***(***S'***).

3. **Acceptance Test.** Test whether to accept the move from ***S*** to ***S'***. If the move is accepted, then ***S'*** replaces ***S*** as the current solution; otherwise ***S*** is retained as the current solution.

4. **Termination Test.** Test whether the algorithm should terminate. If it terminates, output the best solution generated; otherwise, return to the neighbour generation step.

General Local Search Procedure

1. **Initialisation.**

   Random permutation (J_1, J_2, … , J_n) for single-machine problem or flow-shop, more complicated structures for other models.

2. **Neighbour Generation.**

   Transpose neighbourhood: (1,**2**,**3**,4,5,6,7) → (1,**3**,**2**,4,5,6,7)
   Swap neighbourhood: (1,**2**,3,4,5,**6**,7) → (1,**6**,3,4,5, **2**,7)
   Insert neighbourhood: (1,**2**,3,4,5,6,7) → (1,3,4,5,6, **2**,7)

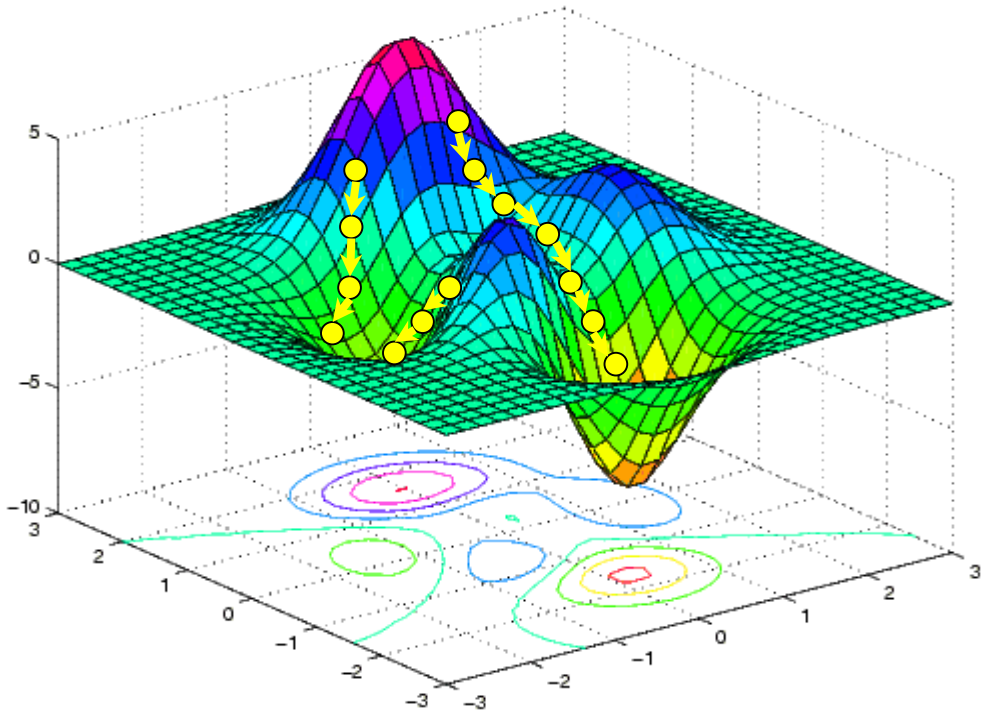
   Neighbours may be generated randomly, systematically, or by some combination of the two approaches.

3. **Acceptance Test.**

   Specific for each type of local search algorithm

4. **Termination Test.** Limited computation time or the number of iterations

The algorithm can be applied repeatedly starting each time with a different randomly generated initial solution.



2. Threshold Accepting

1. **Initialisation.**

   As described

2. **Neighbour Generation.**

   As described

3. **Acceptance Test.**

   Choose ***S'*** if $F(S')-F(S)<\alpha$, where $\alpha>0$ is a threshold value.
   Usually α is relatively large at the beginning, becomes smaller later on

4. **Termination Test.**

   As described

3. Simulated Annealing

1. **Initialisation.**

   As described

2. **Neighbour Generation.**

   As described

3. **Acceptance Test.**

   Probabilistic acceptance test

4. **Termination Test.**

   As described

3. Simulated Annealing

Probabilistic Acceptance Test:

Determine $\Delta = F(S') - F(S)$.

- If $\Delta \leq 0$, then a move to schedule ***S'*** is always accepted.
- If $\Delta > 0$, then a move to schedule ***S'*** is accepted with probability

$$e^{-\Delta/T},$$

where ***T*** is a parameter called the *temperature*, which changes during the course of the algorithm.

Usually ***T*** is large in the beginning and then it decreases until it is close to 0 at the final stages.

Different "*cooling schemes*" can be applied. Often $T_k = T_0 a^k$, where T_k is the temperature at iteration k and $0 < a < 1$.

“Cycling” of SA and TA

- It is possible to get back to the solutions already visited

- A simple way to avoid such problems is to store all visited solutions in a list called *tabu list*.
- A new solution can be accepted if it is not contained in the list.

4. Tabu Search (TS)

1. **Initialisation.**

   As described

2. **Neighbour Generation.**

   As described

3. **Acceptance Test.**

   - A “worse” schedule S’ may be accepted
   - Deterministic acceptance test based on ***tabu list***

4. **Termination Test.**

   As described

4. Tabu Search (TS)

Tabu List

- Tabu list stores attributes of the previous few moves.
- It has a fixed number of entries (usually between 5 and 9).
- It is updated each time ***S'*** is accepted:
  - the reverse transformation is entered at the top of tabu list to avoid returning to the same solution (to avoid returning to a local optimum);
  - all other entries are pushed down one position;
  - the bottom entry is deleted.

4. Tabu Search (TS)

Deterministic Acceptance Test:

Determine $\Delta = F(S') - F(S)$.

- If $\Delta<0$ and ***S'*** is "non-tabu", then a move to ***S'*** is always accepted
- If $\Delta<0$ and ***S'*** is "tabu", then a move to ***S'*** may be accepted for a "promising" schedule ***S'*** (if $F(S')$ is less than the objective function value for any other solution obtained before)
- If $\Delta\geq 0$ and ***S'*** is "tabu", then a move to ***S'*** is always rejected.
- If $\Delta \geq 0$ and ***S'*** is "non-tabu", then a "wait and see" approach is adopted: ***S'*** remains as a candidate while the search continues for a neighbour which can be accepted immediately. If no such neighbour is found, a move to the best candidate ***S'*** is made.

4. Tabu Search (TS)

Log Book - List of Schedules.seq

Schedule	Time	C_{max}	T_{max}	$\sum U_j$	$\sum C_j$	$\sum T_j$	$\sum w_j C_j$	$\sum w_j T_j$
Sch 1234	1	37	32	4	100	81	857	632
Sch 1243	1	37	36	4	91	72	705	480
Sch 1324	1	37	31	4	103	84	1003	778
Sch 1342	1	37	35	4	97	78	931	706
Sch 1423	1	37	36	4	85	66	633	408
Sch 1432	1	37	35	4	88	69	779	554
Sch 2134	1	37	32	4	100	81	877	652
Sch 2143	1	37	36	4	91	72	725	500
Sch 2314	1	37	29	4	103	84	1049	824
Sch 2341	1	37	33	4	97	78	985	760
Sch 2413	1	37	36	4	85	66	661	436
Sch 2431	1	37	33	4	88	69	833	608
Sch 3124	1	37	31	4	106	87	1175	950
Sch 3142	1	37	35	4	100	81	1103	878
Sch 3214	1	37	29	4	106	87	1195	970
Sch 3241	1	37	33	4	100	81	1131	906
Sch 3412	1	37	35	4	94	75	1039	814
Sch 3421	1	37	33	4	94	75	1059	834
Sch 4123	1	37	36	3	79	68	569	440
Sch 4132	1	37	35	3	82	71	715	586
Sch 4213	1	37	36	3	79	68	589	460
Sch 4231	1	37	33	3	82	71	761	632
Sch 4312	1	37	35	3	85	74	887	758
Sch 4321	1	37	33	3	85	74	907	778

Conclusions

- Local search algorithms are very generic.
- They have been applied successfully to many industrial problems.
- Performance of local search algorithms depends on construction of neighborhood.
- A method that exploits special structure is usually faster (if one exists).