Parallel Clustering Search applied to Capacitated Centered Clustering Problem

Davi M.Morales, Antônio A.Chaves, Alvaro L.Fazenda

Federal University of Sao Paulo
Science and Technology Institute
Campus Sao José dos Campos

may/2019

Motivation

Optimize the computational performance in solving combinatorial problems adjusting a hybrid metaheuristic called Clustering Search Method, by parallel programming.

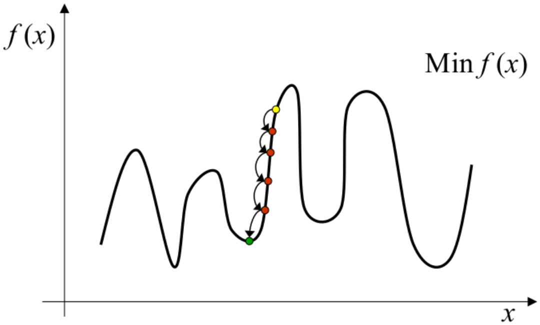
Metaheuristic

- Heuristic (from greek *heuristiké*) means the art of discovering something, and the prefix meta (*metá*) expressing superior level or generality.
- Reliable tools/algorithms to find good quality solutions for NP-Hard combinatorial optimization problems.
- Applied to: telecommunication, public services, industrial transportation and distribution, DNA sequencing, etc.
- Several methods: Ant Colony Optimization (ACO), Genetic Algorithm (GA), Tabu Search (TS), Simulated Annealing (SA), Variable Neighborhood Search (VNS), Iterated Local Search (ILS), etc.

Hybrid metaheuristics

- Improve metaheuristics performance.
- Avoid local minimum.
- Balancing between exploration (explore new regions) versus exploitation (fine tuning).
- metaheuristic + metaheuristic.
- **metaheuristic + LS**.
- metaheuristic + Exact methods.

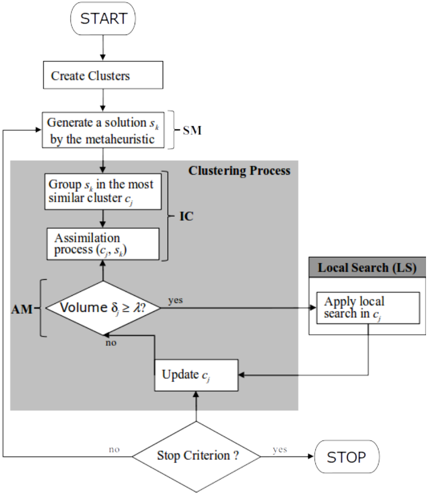
Local Search (LS) Methods



- High computational cost
- Idea: Apply LS in promise regions
  - Promising region defined by clustering similar solutions.
  - LS applied to the most representative solution in a cluster.

CLUSTERING SEARCH

Clustering Search Method



- Combine metaheuristics and local search;
- Search intensified only in promising regions of the solution space.
  - Initial Search Space with high diversity defining clusters.
- Four independent parts:
  - Search Metaheuristic (SM) - by GA,
  - Iterative Clustering (IC),
  - Analyzer Module (AM) and
  - Local Searcher (LS) - by VND.

Variable Neighborhood Descent - VND

algorithm VND (s)

select k_{max} local search heuristics (N^1: Shift point; N^2: Swap point)

$k \leftarrow 1$

while ($k \leq k_{max}$) **do**

find the best neighbor s' of s ($s' \in N^k(s)$)

if ($f(s') < f(s)$) **then**

$s \leftarrow s'$

$k \leftarrow 1$

else

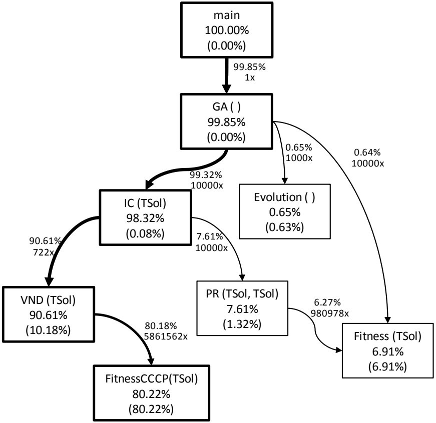
$k \leftarrow k + 1$

end-while

end-algorithm

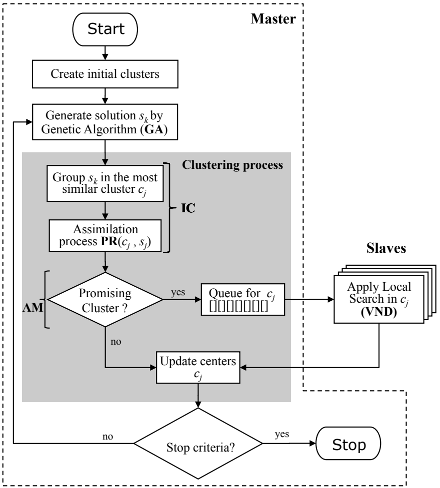
- Assemble of LS methods:
  1. Shift Point
  2. Swap Points

Instrumentation



- Bottleneck detected on the local search procedure.
- VND demands 90% computational time.
- Objective function (FitnessCCCP) very demandant.

Parallel Clustering Search Method



- Master-slave parallel model using MPI programming.
- SM, IC and AM component execute serially in master process.
- Several parallel slave processes executes LS components.
- Master process delegates LS in idle slave processes in an asynchronous way.
- Clusters' centers in the master process don't stay outdated for a long time.

Capacitated Centered Clustering Problem (CCCP) [Negreiros and Palhano - 2006]

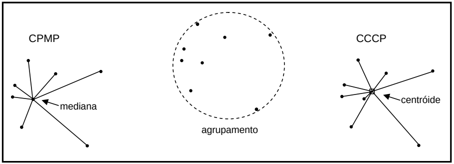


Figure: CCCP Example

- Capacitated p-Median Problem (CPMP) generalization;
  - CPMP: groups are centered at the medians;
- CCCP: groups are centered at the "average" of their points' coordinates;
- NP-Hard
- Set of *p* groups with limited capacity and minimum dissimilarity between the formed group;
- Each group has a centroid located at the geometric center of its points and covers all demands of a set of *n* points.

CCCP Representation

- Array of integer values with *n* points;
- Each *i* position represents the group to which the demand point *i* is allocated.

Points	1	2	3	4	5	6	7
	1	1	2	2	3	2	3

Figure: CCCP representation example: 7 demand points and 3 groups. Group 1 consists of points 1 and 2, group 2 consists of points 3, 4 and 6, and group 3 consists of points 5 and 7.

CS applied to CCCP

- Iterative procedure:
  1. Define the number of clusters populated by random solutions.
  2. SM, by GA, generate offspring as solutions.
  3. Each offspring is grouped with the nearest cluster (IC component).
  4. Analyze density to discover promising clusters.
  5. Exploration is applied in the promising center by VND method.
  6. Check stop criteria.
- Objective function definition
  - Minimize the total distance between each point and the cluster centroid.
  - Determine the centroid location each assimilation or successful local search.
  - Penalty if the group don't satisfy the capacity constraints.
  - High computational cost due to centroid determination and distance evaluation.

Computational Results

- Computational infrastructure:
  - Cluster with 5 nodes
  - Each node: 4 sockets AMD Opteron 6376 (16 cores) at 2.3 GHz, 1 GB of L2 cache, 16 GB of L3 cache and 132 GB of main memory, InfiniBand Interconnect.
- Validation performed by comparing the best solution available in the literature with the best and the mean solutions obtained by the parallel CS.
- Instances "Doni" by Negreiros & Palhano (2006) - sales force territorial design in the Fortaleza's area.

Instance	number of points	number of clusters	capacity
doni1	1000	6	200
doni2	2000	6	400
doni3	3000	8	400
doni4	4000	10	400
doni5	5000	12	450

Validation

Table: Comparison of the parallel CS results with the literature

		Parallel CS				CS
Problem	Best-known	Minimum	Maximum	Mean	gap	gap
doni1	3021.41	3024.05	3026.18	3025.51	0.09	0.03
doni2	6080.70	6372.08	6374.83	6372.73	4.79	4.80
doni3	8343.49	8418.38	8452.24	8439.48	0.90	1.14
doni4	10777.64	10833.66	10887.96	10864.88	0.52	0.63
doni5	11114.67	11130.87	11216.89	11441.26	0.15	0.18
average					1.29	1.36

CS Parallel performance

Table: Parallel performance of CS applied to *doni* instances (*T*, *Sp* and *Ef* means *Time* in seconds, *Speedup* and *Efficiency* in percent, respectively)

Slave processs		**1**	**2**	**4**	**8**	**16**	**32**	**64**	**128**
Doni1	T(s)	146.98	79.69	45.04	22.44	12.45	8.52	12.50	-
	Sp	1.00	1.84	3.26	6.55	11.81	17.24	11.76	-
	Ef(%)	100.00	92.22	81.58	81.89	73.81	53.88	18.37	-
Doni2	T(s)	990.24	568.39	243.15	139.02	62.90	42.85	24.41	-
	Sp	1.00	1.74	4.07	7.12	15.74	23.11	40.57	-
	Ef(%)	100.00	87.11	101.81	89.04	98.40	72.21	63.39	-
Doni3	T(s)	4255.08	2132.15	1096.93	596.30	336.41	167.25	95.09	-
	Sp	1.00	2.00	3.88	7.14	12.65	25.44	44.75	-
	Ef(%)	100.00	99.78	96.98	89.20	79.05	79.51	69.92	-
Doni4	T(s)	25580.70	12079.70	6900.16	3828.41	2068.77	991.78	564.30	424.20
	Sp	1.00	2.12	3.71	6.68	12.37	25.79	45.33	60.30
	Ef(%)	100.00	105.88	92.68	83.52	77.28	80.60	70.83	47.11
Doni5	T(s)	20201.40	10889.80	5396.61	4014.32	1468.94	928.15	437.28	251.21
	Sp	1.00	1.86	3.74	5.03	13.75	21.77	46.20	80.43
	Ef(%)	100.00	92.75	93.58	62.90	85.95	68.02	72.18	62.82

Conclusion

- Parallel CS was able to substantially increase the computational performance.
- Parallel CS Solutions are comparable to the serial CS
- Results comparable to the best-known solutions from the literature.
- Parallel efficiency decreases as the number of used processors increases due to Amdhal Law.
- Further work:
  - Use shared memory programming model;
  - Use GPUs Programming;
  - Improve strong scalability changing the master-slave approach.

Acknowledgments

- INCT of the Future Internet for Smart Cities. CNPq, 465446/2014-0, CAPES 88887.136422/2017-00, FAPESP 2014/50937-1 and 2015/24485-9.
- It was also partially funded by FAPESP 2012/17523-3.