Distributed Local Search for Elastic Image Matching

Hongjian Wang(✉), Abdelkhalek Mansouri, Jean-Charles Créput, and Yassine Ruichek

IRTES-SeT, Université de Technologie de Belfort-Montbéliard, 90010 Belfort, France
hongjian3715@gmail.com

Abstract. We propose a *distributed local search* (DLS) algorithm, which is a parallel formulation of a local search procedure in an attempt to follow the spirit of standard local search metaheuristics. Applications of different operators for solution diversification are possible in a similar way to *variable neighborhood search*. We formulate a general energy function to be equivalent to elastic image matching problems. A specific example application is *stereo matching*. Experimental results show that the GPU implementation of DLS seems to be the only method that provides an increasing acceleration factor as the instance size augments, among eight tested energy minimization algorithms.

Keywords: Parallel and distributed computing · Variable neighborhood search · Stereo matching · Graphics processing unit

1 Introduction

Local search, also referred as hill climbing, descent, iterative improvement, general single-solution based metaheuristics and so on, is a metaheuristic algorithm [1]. Starting with a given initial solution, at each iteration the heuristic replaces the current solution by a neighbor solution that improves the fitness function. The search stops when all candidate neighbors are worse than the current solution, meaning a local optimum is reached. Existing parallelization strategies for local search can be divided into three categories. In the first category, the evaluation of neighborhood is made in parallel [2,3]; in the second category, the focus is on the parallel evaluation of a single solution, and the function can be viewed as an aggregation of partial functions [2,4]; in the third category, several local search metaheuristics are simultaneously launched for computing robust solutions [5,6]. In our opinion, an interesting parallel implementation model of local search should be fully distributed, where each processor carries out its own neighborhood search based on some parts of the input data, considering only a local part of the whole solution. Operations on different processors should be similar, with no centralized selection procedure, except for final evaluation. A final solution should be obtained with the partial operations from different processors.

P. Siarry et al. (Eds.): ICSIBO 2016, LNCS 10103, pp. 65–74, 2016.
DOI: 10.1007/978-3-319-50307-3_5

Following this idea, we propose a *distributed local search* (DLS) algorithm and implement it on GPU parallel computing platforms.

A natural field of applications with GPU processing is image processing, which is a domain at the origin of GPU development. A lot of image processing and computer vision problems can be viewed as optimization problems in a more general way, dealing with brute data distributed in some Euclidean space and system in relation to the data. More often, these NP-hard optimization problems involve data distributed in the plane and elastic structures represented by graphs that must match the data. Such optimization problems can be stated in a generic framework of *graph matching* [7,8]. In this paper, we are particularly interested in moving grids in the plane following the idea of visual correspondence problem, which is to compute the pairs of pixels from two images that result from the same scene element. A typical example application is *stereo matching*, which we formulate as an elastic image matching problem [9]. We apply the proposed DLS algorithm to stereo matching by minimizing the corresponding energy function.

The DLS can be used for parallel implementation of elastic matching problems that include not only visual correspondence problems but also neural network topological maps, or elastic nets approaches [10,11], modeling the behavior of interacting components inspired by biological systems and collective behaviors at a low level of granularity. The framework is based on data decomposition, with the idea of modeling the geometry of objects using some adaptive (elastic) structures that move in space and continuously interact with the input data distribution memorized into a cellular matrix [12]. Then spatial metaphors, as well as biological metaphors should fit well into the cellular matrix framework.

The rest of this paper is organized as follows. In Sect. 2, we formulate a general energy function to be equivalent to elastic image matching problems. In Sect. 3, we present the DLS algorithm in detail, providing basic data structures and operations in Subsect. 3.1, explaining local evaluation in Subsect. 3.2, designing two classes of move operators in Subsect. 3.3, and giving the details of GPU implementation in Subsect. 3.4. Experimental results are reported in Sect. 4, before some conclusions are drawn in Sect. 5.

2 Elastic Grid Matching

We define a class of visual correspondence problems as *elastic grid matching* problems, where we use a two-dimensional grid to represent an image. Given two input grids with same size and same regular topology, one is a matcher grid $G_1 = (V_1, E_1)$ where a vertex is a pixel (from the corresponding image) with a variable location in the plane, while the other is a matched grid $G_2 = (V_2, E_2)$ where vertices are pixels located in a regular grid. The goal of elastic grid matching is to find the matcher vertex locations in the plane, so that the following energy function

$$E(G_1) = \sum_{p \in V_1} D_p(p - p_0) + \lambda \cdot \sum_{\{p,q\} \in E_1} V_{p,q}(p - p_0, q - q_0) \qquad (1)$$

is minimized, where p_0 and q_0 are the default locations of p and q respectively in a regular grid. Here, D_p is the data energy that measures how much assigning label f_p to pixel p disagrees with the data, and $V_{p,q}$ is the smoothness energy that expresses smoothness constraints on the labelings enforcing spatial coherence [13–15]. A label f_p in visual correspondence represents a pixel moving from its regular position into the direction of its homologous pixel, *i.e.* $f_p = p - p_0$. In the following sections, we will directly use the notations of labels as relative displacements, as usual with such problems. The energy function is commonly used for visual correspondence problems, and it can be justified in terms of maximum a posteriori estimation of a *Markov random field* (MRF) [16,17].

It has been proven that elastic image matching is NP-complete [9], and finding the global minimum for the energy function even with the simplest smoothness penalty, the piecewise constant prior, is NP-hard [13,14]. We choose the local search metaheuristics to deal with the energy minimization problem.

3 Distributed Local Search

Based on the cellular matrix model proposed in [12], we design a parallel local search algorithm, the DLS, to implement many local search operations on different parts of the data in a distributed way. It is a parallel formulation of local search procedures in an attempt to follow the spirit of standard local search metaheuristics. Starting from its location in the cellular matrix, each processor locally acts on the data located in the corresponding cell according to the cellular decomposition, in order to achieve local evaluation, perform neighborhood search, and select local improvement moves to execute. The many processes locally interact in the plane, making evolve the current solution into an improved one. The solution results from the many independent local search operations simultaneously performed on the distributed data in the plane. Normally, a local search algorithm with single operator obtains local minima. In order to escape from local minima, we design several operators. Applications of different operators for diversification are possible in a similar way to the *variable neighborhood search* (VNS).

3.1 Data Structures and Basic Operations

The data structures and direction of operations for DLS algorithms are illustrated in Fig. 1. The input data set is deployed on the low level of both matcher grid and matched grid, represented as regular images in the figure. The honeycomb cells represent the cellular matrix level of operations. Each cell is a basic processor that handles a basic local search processing iteration with the three following steps: neighborhood generation (**get**); neighbor solution evaluation and selecting the best neighbor (**search**); then moving the matcher grid toward the selected neighbor solution (**operate**). The nature and size of specific moves and neighborhoods will depend on the type of used operator and the level of cellular matrix. The higher is the level, the larger is the local cell/neighborhood. In the

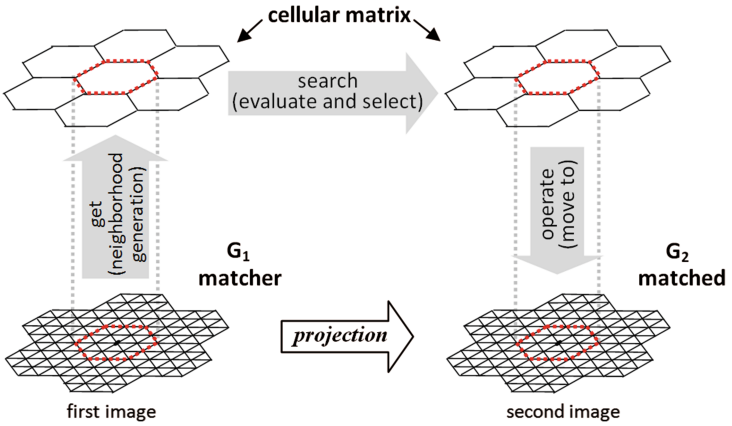


Fig. 1. Basic projection for DLS.

cellular matrix model, a solution is composed of many sub-solutions from many cells. Each sub-solution is evolved from an initial sub-solution based on the distributed data in a cell. By partitioning the data and solution, the neighborhood structure is also partitioned at the same time.

3.2 Local Evaluation with Mutual Exclusion

During the parallel operation, the coherence of local evaluation with mutual exclusion is violated by conflict operations. A conflict operation occurs when a same pixel or two neighboring pixels is/are being evaluated and moved simultaneously by two threads. Conflict operations only happen on frontier pixels, which are the pixels on the cell frontiers according to the cellular matrix partition of the image. In order to eliminate the conflict operations in DLS, we propose a strategy, called *dynamic change of cell frontiers* (DCCF), by which we limit the move to the internal pixels of a cell only. Cell frontier pixels remain at fixed locations, and they are not concerned by local moves so that exclusive access of the thread to its internal region delimited by the cell is guaranteed. A problem that arises is how to manage cell frontier pixels and make them participate in the optimization process. As a solution, the cellular matrix decomposition is dynamically changeable from the CPU side before the application of a round of DLS operations. At different moments, the cellular matrix decomposition slightly shifts on the input image in order to change the cell frontiers and consequently the fixed pixels. For a given cellular matrix decomposition, cell frontier pixels are then fixed and not allowed to be moved by current DLS operations.

3.3 Neighborhood Operators

We design different neighborhood operators for the DLS algorithm applied to elastic grid matching. We use the notations of labeling problems to present these

operators. Move operations in a given neighborhood structure correspond to changing labels of pixels in the corresponding labeling space. Operators are classified between small moves and large moves. In the first case, only a single pixel from a given cell moves at a time; in the second case, larger sets of pixels from a given cell can simultaneously move.

Small move operators. In a move operation, if only one pixel moves, meaning that only one pixel's label is changed, this kind of operation is called small move operation. We design two small move operators: *local move operator* that applies an increment/decrement to the current label of the considered pixel; *propagation operator* that takes the labels of the considered pixel's neighboring pixels, as candidate labels, and then replaces the current label with the best one found in a propagation window.

Large move operators. They consider multiple pixels. We design six large move operators: *random pixels move operator* randomly picks several pixels in the considered cell, and then assigns a same candidate label to these pixels; *random pixels jump operator* randomly picks several pixels in the considered cell, and then applies a same increment/decrement to the current labels of the considered pixels; *random pixels expansion operator* randomly picks two groups of pixels, where pixels in the same group have the same label, and then "expands" the label of one group to the other, setting the labels of all the pixels in the second group with the same label as the first group; *random pixels swap operator* picks pixels in the same way as the random pixels expansion operator does, and then "swaps" the labels of the two groups, setting the labels of all the pixels in the second group with the label of the first group, meanwhile setting the labels of all the pixels in the first group with the label of the second group; *random window move operator* picks a fixed-sized window of pixels at a random position within the considered cell, and then assigns a same candidate label to all the pixels in this picked window; *random window jump operator* picks pixels in the same way as the random window move operator does, and then applies a same increment/decrement to the current labels of all the pixels in this picked window. More details about these operators can be found in [12].

3.4 GPU Implementation Under VNS Framework

We implement the DLS algorithm on GPU platforms in Compute Unified Device Architecture (CUDA). The CUDA kernel calling sequence from the CPU side enables the application of different operators in the spirit of VNS and manages dynamic changes of cellular matrix frontiers. According to our previous experiments, the repartition of tasks between host (CPU) and device (GPU) is actually the best compromise we found to exploit the GPU CUDA platform at a reasonable level of computation granularity.

The flow chart executed from CPU side is presented in Fig. 2. The data transfer between CPU side and GPU side only occurs at the beginning and the end of the algorithm. The two kernels that are called from CPU side and executed

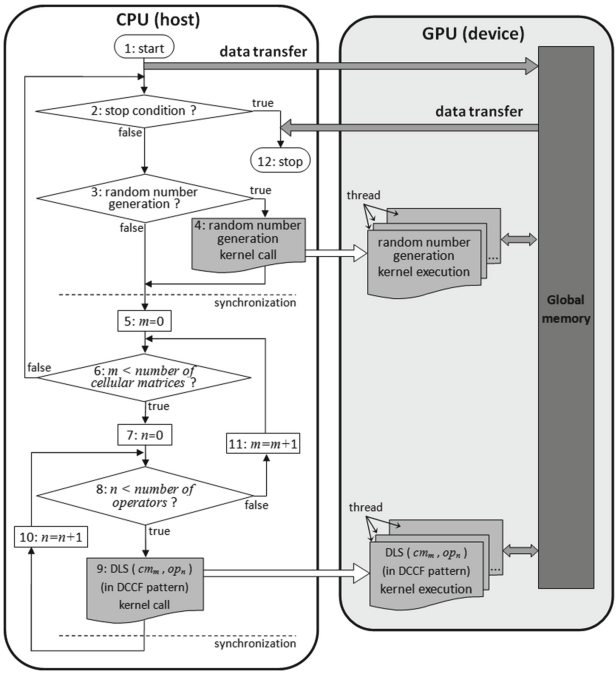


Fig. 2. Flowchart of DLS implementation.

on GPU are: the random number generation kernel and the DLS kernel. On GPU side, random numbers are needed for random move operators. The random numbers are generated in advance by the random number generation kernel which is regularly called during the algorithm according to the random number generation rate. It is the CPU side that controls DLS kernel calls with different operators executed within the DCCF pattern for frontier cells management. With several neighborhood operators in hand, we use them under the VNS framework in order to enhance the solution diversification.

4 Experimental Study

We apply the DLS algorithm to stereo matching, viewing the problem as energy minimization problem. We follow in the footsteps of Boykov et al. [14], Tappen and Freeman [18], and Szeliski et al. [15], using a simple energy function, applied to benchmark images from the widely used Middlebury stereo data set [19]. The labels are the disparities, and the data costs are the absolute color differences between corresponding pixels for each disparity. For the smoothness term in the energy function, we use a truncated linear cost as the piecewise smooth prior defined in [13]. We focus on the performance of DLS when input size augments. We experiment on the Middlebury 2005 stereo benchmark [19] including 18 pairs

of images with sizes from the smallest 458×370 to the largest 1374×1110 in average. We uniformly set the disparity range to 64 pixels, for all the sizes. We denote our DLS GPU implementation as DLS-gpu. We also test the counterpart CPU sequential version which is denoted by DLS-cpu. We compare DLS with six other methods[1]: *iterated conditional modes* (ICM) [16] which is an old approach using a deterministic "greedy" strategy to find a local minimum; sequential *tree-reweighted message passing* (TRW-S) [15] which is an improved version of the original tree-reweighted message passing algorithm [20]; BP-S and BP-M [15] which are two updated version of the max-product *loopy belief propagation* (LBP) implementation of [18]; GC-swap and GC-expansion which are two graph cuts based algorithms proposed in [14]. Instead of reporting the absolute energy values, we report the percentage deviation from the best known solution (lowest energy) of the mean solution value over 10 runs, denoted as $\%PDM$ value. We choose the best known solution from the executions of all tested methods.

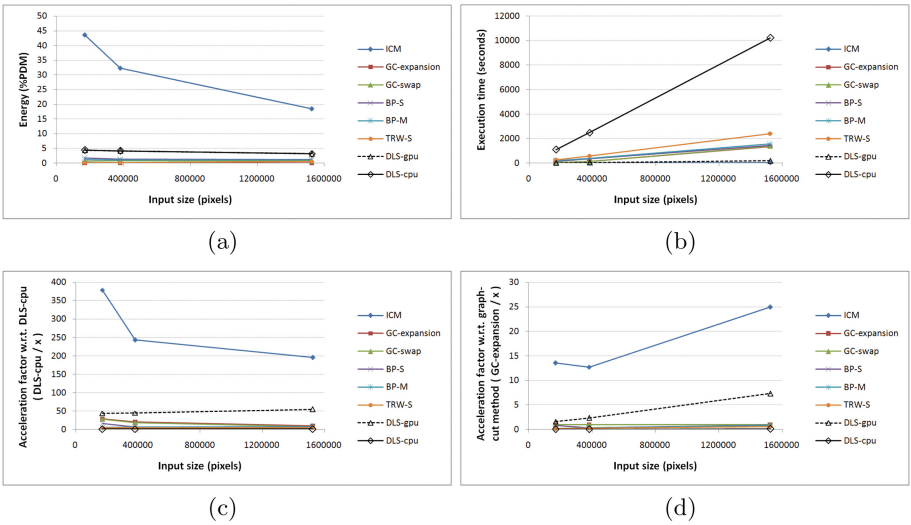
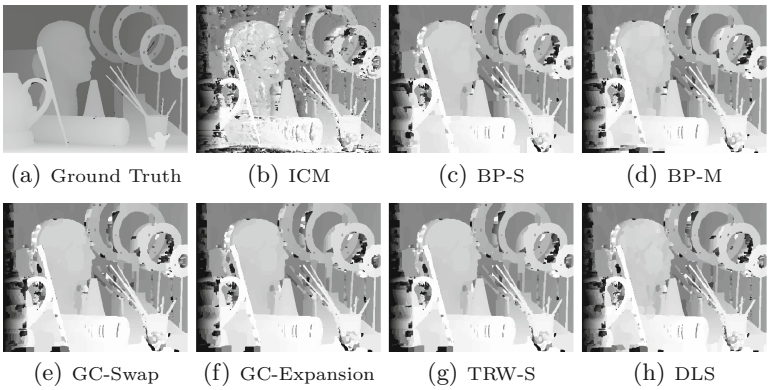


Fig. 3. Results of eight tested methods: (a) energy value as $\%PDM$; (b) execution time, (c) acceleration factor of each method relative to the slowest method (DLS-cpu); (d) acceleration factor of each method relative to the method (GC-expansion) that gets the lowest energy.

The results of different methods are reported in Fig. 3. From (a) to (d) are respectively reported energy value as $\%PDM$, execution time, acceleration factor of each method relative to the slowest method (DLS-cpu), and acceleration factor of each method relative to the method (GC-expansion) that gets the lowest energy. The ICM method runs fastest but generates very high energies, while

[1] For all the tested energy minimization algorithms, we use the original codes from *http://vision.middlebury.edu/MRF/code/*.

DLS-gpu runs a little slower than ICM but generates much lower energies with more acceptable $\%PDM$ values smaller than 5%. An important observation from Fig. 3 is that, among all the tested methods, only the DLS-gpu has an acceleration factor which increases according to the augmentation of input size. This means that further improvement could be carried on only by the use of multi-processor platform with more effective cores.



(a) Ground Truth (b) ICM (c) BP-S (d) BP-M

(e) GC-Swap (f) GC-Expansion (g) TRW-S (h) DLS

Fig. 4. Disparity maps for the *Art* (463×370) benchmark obtained with different energy minimization methods. The disparity range is set to 64 pixels.

In Fig. 4 are displayed the disparity maps for the *Art* benchmark. Note that during our experiments, we choose the stereo matching application but only view it as an energy minimization problem, just focusing on minimizing energies. The disparity maps obtained from all the tested methods are the raw results after energy minimization, without any additional post-treatments such as left-right consistency check, occlusion detection, or disparity smoothing, which are all treatments specific to stereo matching in order to minimize the errors compared with ground truth disparity maps. Moreover, as pointed out in [15], the ground truth solution may not always be strictly related to the lowest energy.

5 Conclusion

We have proposed a parallel formulation of local search procedure, called distributed local search (DLS) algorithm. We have applied the algorithm to stereo matching problem. The main encouraging result is that the GPU implementation of DLS on stereo matching seems to be the only method that provides an increasing acceleration factor as the instance size augments, for a result of quality less than 5% deviation to the best known energy value. For all the other approaches, the acceleration factor, against the slowest sequential version of DLS, is decreasing, except for the ICM method, which however only produces poor

result of about 45% deviation to the best known energy. Graph cuts based algorithms and belief propagation based algorithms are well-performing approaches concerning quality, however the computation time increases quickly along with the instance size. That is why we hope for further improvements or improved accelerations of the DLS approach with the availability of new multi-processor platforms with more independent cores.

It is a well-known fact that the minimum energy level does not necessarily correlate to the best real-case matching. Here, we only address energy minimization discarding too much complex post-treatments necessary for the "true" ground truth matching. It should follow that many tricks are certainly not yet implemented to make energy minimization coincide to ground truth evaluation. In order to improve the matching quality in terms of minimizing the errors to ground truth only, specially designed terms for detecting typical situations in vision, such as occlusion, slanted surfaces, and the aperture problem, need to be added in the formulation of energy function.

References

1. Talbi, E.G.: Metaheuristics: From Design to Implementation, vol. 74. Wiley, Hoboken (2009)
2. Van Luong, T., Melab, N., Talbi, E.G.: Gpu computing for parallel local search metaheuristic algorithms. IEEE Trans. Comput. **62**, 173–185 (2013)
3. Delévacq, A., Delisle, P., Krajecki, M.: Parallel gpu implementation of iterated local search for the travelling salesman problem. In: Hamadi, Y., Schoenauer, M. (eds.) LION 6. LNCS, vol. 7219, pp. 372–377. Springer, Heidelberg (2012)
4. Fosin, J., Davidović, D., Carić, T.: A gpu implementation of local search operators for symmetric travelling salesman problem. PROMET Traffic Transp. **25**, 225–234 (2013)
5. Luong, T., Melab, N., Talbi, E.-G.: GPU-based multi-start local search algorithms. In: Coello, C.A.C. (ed.) LION 2011. LNCS, vol. 6683, pp. 321–335. Springer, Heidelberg (2011). doi:10.1007/978-3-642-25566-3_24
6. Sánchez-Oro, J., Sevaux, M., Rossi, A., Martí, R., Duarte, A.: Solving dynamic memory allocation problems in embedded systems with parallel variable neighborhood search strategies. Electron. Notes Discrete Math. **47**, 85–92 (2015)
7. Bengoetxea, E.: Inexact graph matching using estimation of distribution algorithms. Ph.D. thesis, Ecole Nationale Supérieure des Télécommunications, Paris, France (2002)
8. Caetano, T.S., McAuley, J.J., Cheng, L., Le, Q.V., Smola, A.J.: Learning graph matching. IEEE Trans. Pattern Anal. Mach. Intell. **31**, 1048–1058 (2009)
9. Keysers, D., Unger, W.: Elastic image matching is np-complete. Pattern Recogn. Lett. **24**, 445–453 (2003)
10. Durbin, R., Willshaw, D.: An analogue approach to the travelling salesman problem using an elastic net method. Nature **326**, 689–691 (1987)
11. Créput, J.C., Hajjam, A., Koukam, A., Kuhn, O.: Self-organizing maps in population based metaheuristic to the dynamic vehicle routing problem. J. Comb. Optim. **24**, 437–458 (2012)
12. Wang, H.: Cellular matrix for parallel k-means and local search to Euclidean grid matching. Ph.D. thesis, Université de Technologie de Belfort-Montbeliard (2015)

13. Veksler, O.: Efficient graph-based energy minimization methods in computer vision. Ph.D. thesis, Cornell University (1999)
14. Boykov, Y., Veksler, O., Zabih, R.: Fast approximate energy minimization via graph cuts. IEEE Trans. Pattern Anal. Mach. Intell. **23**, 1222–1239 (2001)
15. Szeliski, R., Zabih, R., Scharstein, D., Veksler, O., Kolmogorov, V., Agarwala, A., Tappen, M., Rother, C.: A comparative study of energy minimization methods for markov random fields with smoothness-based priors. IEEE Trans. Pattern Anal. Mach. Intell. **30**, 1068–1080 (2008)
16. Besag, J.: On the statistical analysis of dirty pictures. J. Roy. Stat. Soc. Ser. B (Methodological) **48**(3), 259–302 (1986)
17. Geman, S., Geman, D.: Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. IEEE Trans. Pattern Anal. Mach. Intell. **6**, 721–741 (1984)
18. Tappen, M.F., Freeman, W.T.: Comparison of graph cuts with belief propagation for stereo, using identical mrf parameters. In: 2003 Ninth IEEE International Conference on Computer Vision. IEEE (2003)
19. Scharstein, D., Szeliski, R.: High-accuracy stereo depth maps using structured light. In: 2003 IEEE Conference on Computer Vision and Pattern Recognition, vol. 1, pp. 195–202. IEEE (2003)
20. Wainwright, M.J., Jaakkola, T.S., Willsky, A.S.: Map estimation via agreement on trees: message-passing and linear programming. IEEE Trans. Inf. Theor. **51**, 3697–3717 (2005)