Multi-way Particle Swarm Fusion

Anonymous ECCV submission

Paper ID ***

Abstract. This paper proposes a novel MAP inference framework for Markov Random Field (MRF) in parallel computing environments. The inference framework, dubbed Swarm Fusion, is a natural generalization of the Fusion Move method. Every thread (in a case of multi-threading environments) maintains and updates a solution. At every iteration, a thread can generate arbitrary number of solution proposals and take arbitrary number of concurrent solutions from the other threads to perform multi-way fusion in updating its solution. The framework is general, making popular existing inference techniques such as alpha-expansion, fusion move, parallel alpha-expansion, or hierarchical fusion, its special cases. We have evaluated the effectiveness of our approach against competing methods on three problems of varying difficulties, in particular, the stereo, the optical flow, and the layered depthmap estimation problems.

Keywords: MRF; Fusion Move; Particle Swarm Optimization

1 Introduction

[hang: Should we use α-expansion instead of alpha-expansion?]

Parallel computation has changed the field of computing. In the 90s, most processors had single cores. In 2016, most processors have multiple cores, often 4 or even 8. Cluster computing further expands the potential of parallel computation, where one can easily launch a processing job using hundreds or even thousands of computational nodes in a cloud. In the recent work on the AI program playing the ancient Chinese board game of *Go*, parallelization plays a key role in the Monte-Carlo tree search [**?**].

Parallel computation offers tremendous potential for Computer Vision. As image sensing technologies have gone through revolutions, we are in ever growing demands in solving very large problems. One may need to apply image denoising to 50 Megapixel images from latest digital SLRs (e.g., Canon EOS 5DS), stitch thousands of images to generate gigapixel panoramas [**?**], or solve volumetric reconstruction and segmentation problems over a billion ($= 1024^3$) voxels [**?**]. However, state-of-the-art algorithms are still inherently sequential in many Computer Vision problems. Take a Fusion Move method (FM) [**?**,**?**,**?**] for example, which has been one of the most effective techniques for Markov Random Field (MRF) inference, including the above problems like image denoising, image stitching, or volumetric reconstruction. It sequentially improves solution

by fusing the current solution with a solution proposal. Its successful applications go beyond and also cover optical flow, stereo, image inpainting, or image segmentation problems [**?**].

Unleashing the power of parallel computation for effective MRF inference would then bring fundamental contributions to Computer Vision. Currently, FM suffers from a few vital limitations due to its sequential nature. First, standard FM allows only two options per variable in each fusion, either the current solution or a proposal [**?**]. Second, only a single proposal generation scheme is used in each fusion step. [1] Our approach, dubbed *Swarm Fusion* method (SF), makes a few key distinctions from existing approaches: 1) Multiple threads (or computing nodes) simultaneously keep and improve solutions; and 2) Each fusion in each thread can take arbitrary number of proposals from arbitrary combination of proposal generation schemes, even concurrent solutions in the other threads, to be fused with the current solution.

We have evaluated the effectiveness of our approach over three problems in Computer Vision, stereo, optical flow, and layered depthmap estimation. Our idea is very simple. We believe that the method is easily reproducible with minimal coding, and would have immediate impact on numerous Computer Vision researchers or engineers, who currently solve MRF problems. [chen: We will share our implementation of our pipeline with clear interface, so that others can utilizing the parallel pipeline with minimal coding]

2 Related work

MRF inference has been a very active field in Computer Vision with the extensive literature. We refer the readers to survey articles for comprehensive reviews [**?**,**?**], and here focus our description on closely related topics.

Parallel Alpha-Expansion

Lempitsky et al. introduces parallel computation to an alpha-expansion technique, where multiple threads simultaneously fuses mutually exclusive sets of labels. Kumar et al. [**?**], Delong et al. [**?**], and Veksler [**?**] proposed an hierarchical approach, where labels can be simultaneously fused from the bottom to the top in a tree of labels. Instead of taking a hierarchical approach, Batra et al. [**?**] adaptively computed an effective sequence of labels to explore. This technique can be combined with parallel alpha-expansion techniques to further obtain speed-up. Strictly speaking, these approaches are not in the family of Fusion Move methods (FM), because they only consider constant label proposals. Our approach is a generalization of FM.

Parallel MAP inference

[1] Recently, an extension of FM was proposed for layered depthmap estimation [**?**], where a solution subspace, instead of a single solution, is proposed and fused with the current solution. However, this approach is also limited to the use of one proposal generation scheme in each fusion.

The core MAP inference itself can be parallelized. Strandmark et al. parallelized graph-cuts [?]. Message passing algorithms are friendly to GPU implementation and can exploit the power of parallel computation. While state-of-the-art optimization libraries are often freely available for non-commercial purposes, most companies have to develop and maintain in-house implementation of these algorithms. The core optimization libraries are very complex and their modifications require significant engineering investments. In contrast, our idea is extremely simple and easily reproducible by standard engineers.

Fusion Move Methods

FM was first introduced by Lempitsky et al. in solving an optical flow problem [?] [chen: This work focuses on fusion move and optical flow is just one of four applications this work mentioned.]. FM has been effectively used to solve challenging problems in Computer Vision such as stereo with second order smoothness priors [?], stereo with parameteric surface fitting and segmentation (i.e., Surface Stereo) [?], or multicut partitioning [?]. FM has two main advantages over the other general inference techniques [?,?]. First, FM allows us to exploit domain-specific knowledge in customizing proposal generation schemes. Second, FM can handle problems with very large label spaces (and even real-values variables), because the core optimization solves a sequence of binary decision problems. In contrast, methods like message passing algorithms need to maintain messages and beliefs for the entire label space all the time. Although conceptually straightforward, we are not aware of *Parallel Fusion move* algorithms that fuse solution proposals, as opposed to labels, in parallel. This paper seeks to fully unleash the power of parallel computation based on FM in the most general setting.

Evolutionary algorithms and Particle Swarm Optimization

Genetic algorithms (GA) [?] and Particle Swarm Optimization (PSO) [?] maintain multiple solutions and improve them over time. GA or PSO has been used to produce great empirical results, for example, hand tracking [?]. At high level, our strategy is similar in spirit. However, GA or PSO rather arbitrarily copies parts of the solutions or makes random movements in each step (i.e., limited theoretical justification). Our approach directly optimizes the objective function to improve solutions.

3 Multi-way particle swarm fusion

Multi-way Particle Swarm Fusion is a natural extension of the Fusion Move method (FM). We call our method Swarm Fusion (SF) in short. Let us take a multi-threading environment to explain our idea, while the technique is also applicable to other parallel programming model such as *MapReduce* in cloud computing.

Assuming we have N threads $\{T_i | i = 1, 2, \cdots, N\}$, each thread T_i maintains and updates a solution S_i in parallel. Alpha Expansion picks one label in each iteration to perform fusion. FM generates a solution proposal via a proposal generation scheme. SF has 1) a *proposal pool*, from which a thread picks arbitrary

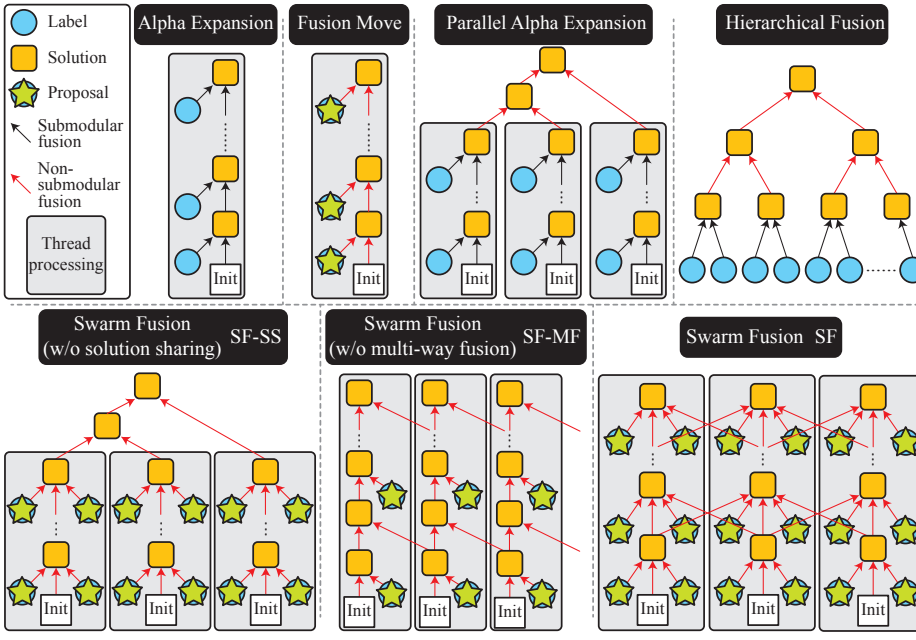

Fig. 1. Swarm Fusion (SF) architecture and its relationships to existing methods. The bottom right example shows the general SF architecture, where each thread takes arbitrary number of solution proposals and concurrent solutions from other threads for fusion. The framework is flexible and can realize other data processing architectures depending on the parameters (e.g., the left two examples in the bottom row). It is easy to verify that existing popular MRF inference methods such as Alpha Expansion [**?**], Fusion Move [**?**], Parallel Alpha Expansion [**?**], or Hierarchical Fusion [**?**,**?**], are all special cases of SF.

number of proposal generation schemes and generates proposals, and 2) a *solution pool*, from which a thread picks arbitrary number of intermediate solutions generated by the others. In our base configuration, the solution pool remembers just the best solution from each thread.

SF has two main parameters α_i and β_i (for each thread T_i), determining its processing flow: In each fusion step, a thread collects α_i solution proposals from the proposal pool [chen: A proposal pool sounds confusing. We often don't really maintain such a pool as many proposals are generated dynamically based on current solution. We just have certain strategies right there and in each iteration, we pick some strategies to generate proposals on the fly. This is different with the solution pool which is really there.] β_i solutions from the solution pool, based on a user-defined strategy or at random to be simple.

Swarm Fusion framework is very flexible and yields various data processing architectures as shown in Figure 1. The bottom right architecture is the most general one, in which threads always fuse some solution proposals and [chen: or?] some concurrent solutions (i.e., $\alpha \neq 0, \beta \neq 0$). Each fusion is multi-way (not

binary) and the energy is not submodular in general, for which effective inference techniques such as TRW [?,?] [chen: TRW-S?] exist. However, if one knows that a certain fusion step is a binary fusion with submodular energy, one can use graph-cuts [?]. If a certain fusion step is a binary fusion with non-submodular energy, one can use QPBO [?]. Note that the threads appear synchronized in the figure for a illustration purpose. In practice, all the threads run asynchronously with a (read-write) lock on the data in the two pools (See Algorithm 1).

Algorithm 1 Swarm Fusion method

procedure (α, β)
  $\mathcal{S}_{pool} \leftarrow \emptyset$ // Solution pool
  Initialize $\mathcal{P}_{pool}$ // Proposal pool
  for each thread T_i **do**
    Initialize its solution S_i
  end for

  for each thread T_i in parallel till convergence **do**
    Pick α solution proposals $\mathcal{P} \subset \mathcal{P}_{\text{pool}}$
    Pick β solutions $\mathcal{S} \subset \mathcal{S}_{\text{pool}}$
    $S_i \leftarrow \text{Fuse}(S_i, \mathcal{P}, \mathcal{S})$
    Replaces the solution in $\mathcal{S}_{pool}$ with S_i
    [Generates proposals and update $\mathcal{P}_{pool}$, if necessary]
  end for
end procedure

Relationships to existing methods

It is easy to verify that Alpha-Expansion (AE) [?], Fusion Move (FM) [?], Parallel Alpha Expansion (PAE) [?], and Hierarchical Fusion (HF) [?,?] are all special cases of the Swarm Fusion method (SF). AE can be realized by setting $(\alpha = 1, \beta = 0)$ and restricting the proposal generation to constant labels with a single thread. Same goes for FM, this time, without the restriction on the proposal generation scheme. PAE is realized by setting $(\alpha = 1, \beta = 1)$ [chen: $(\alpha = 1, \beta = 0)$] with multiple threads, again with a restriction on the proposal generation scheme (the last sequential fusion in PAE is realized by $(\alpha = 0, \beta = 1)$ with a single thread). HF has a slightly different data processing model, without strong ties between threads and data, but can be realized by setting $(\alpha = 2, \beta = 0)$ at the bottom level and $(\alpha = 0, \beta = 2)$ at the remaining levels, while allowing S_i not to be used in the fusion steps of T_i.

4 Swarm Fusion instantiation

We compare SF against competing approaches over three problems in Computer Vision. SF framework has high degrees of freedom and the challenge is to properly quantity the contributions of various algorithmic aspects. For this purpose,

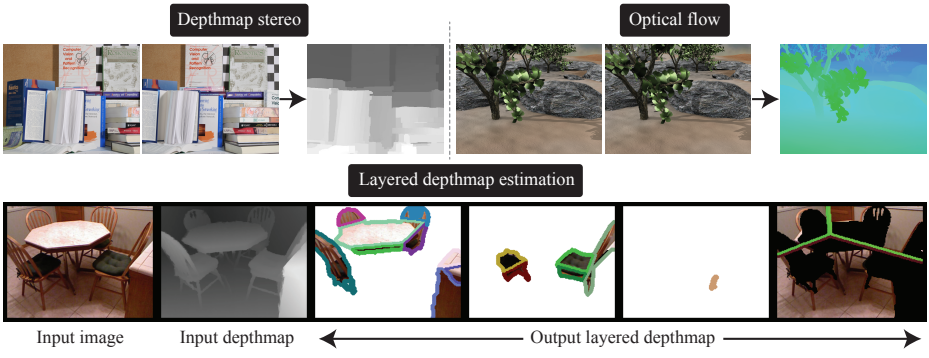

Fig. 2. We compare our Swarm Fusion method against competing approaches on a depthmap stereo [**?**], an optical flow [**?**], or a layered depthmap estimation [**?**] problem. In the layered depthmap problem, the input is an RGBD image, and the output is multiple layeres of depthmaps. Each layer is a piecewise smooth parametric surface model.

we have used the three SF architectures illustrated in Fig. 1. For some problems, we have further limited the capabilities of SF on purpose, to enable effective and fair comparative evaluations. We now explain the details of the three problems.

[hang: Better to explain the methology we used, either here or in Result part. For example:]

[hang: To evaluate the effectness of different parallel structures, we plot the energy minimization process against time. We define the energy of the parallel system at a certain time as the minimum energy of all threads at that time.]

4.1 Swarm Fusion stereo

We start with a simple depthmap stereo problem with standard unary and pairwise terms. We employ submodular pairwise terms to make this stereo represent relatively "easy" MRF inference problems. Unary terms are computed as the average robust photoconsistancy score [**?**] between the reference image and the others inside a 7×7 pixels window.

[hang: equation to fill in] (1)

Pairwise terms are simple truncated absolute label difference with maximum label difference $\sigma_s = 4$. The total energy is defined by the sum of the two, while scaling the pairwise term by a factor of 0.005. For simplicity we don't enforce visibility constraint.

We performed our experiments on Book sequence from Middlebury stereo datasets. [**?**].

Competing methods

The sophistication of photometric consistency function [**?**] makes unary terms dominant in most stereo problems. Our experiments have also supported this,

where the fusion method (i.e., the use of proposal generation) rather makes it slow due to the overhead.[hang: Don't quite understand this sentence. Does the "dominant" means running time or energy? In my experiments, unary terms are pre-computed and are not counted into the running time] Therefore, we have chosen Parallel Alpha Expansion [?] and Hierarchical Fusion [?,?] with graph-cuts optimization as the competing methods.

Since the optimization in the above stereo problem is fairly easy, we compare our method with standard sequential alpha expansion, parallel alpha expansion [?] and hierarchy fusion [?]. In all experiment, we use α-expansion to fuse single label proposals, and QPBO to fuse multi-label proposals.

Swarm Fusion architectures

The three swarm architectures in Fig. 1 have been evaluated: SF-SS (SF without solution sharing), SF-MF (SF without multi-way fusion), and the standard SF. SF-SS implies $\beta = 0$, where α is the free parameter and set to 2. SF-MF implies $\alpha+\beta = 1$, where each thread iterates the step of $(\alpha = 1, \beta = 0)$ or $(\alpha = 0, \beta = 1)$ so that it can take a solution proposal in one step and a neighboring solution in the next. For standard SF, we have used $(\alpha = 1, \beta = 1)$.

after 4 iterations (alpha, beta),

To make the comparison simple, we restrict our proposal generation to be constant-label proposals. Graph-cuts is used in fusing a constant-label proposal as the energy is submodular. QPBO is used in fusing a different solution from another thread as the energy is non-submodular.

4.2 Swarm Fusion optical flow

Fusion Move was first introduced by Lempitsky et al. [?] to solve the optical flow problem. [chen: Again, I think Lempitsky's work doesn't aim at the optical flow problem.] We copy their problem setting and proposal generation schemes for our second problem. We use images from the Middlebury optical flow benchmark [?].

[chen: I made a mistake that I didn't use the same proposal generation schemes with their work. Instead of HS algorithm, I use Farneback algorithm to generate proposals because it is right there in OpenCV. I also add a bunch of other less useful proposals to have more proposal generation schemes for more clear comparisons.. I think we could write in short as follows.]

[chen: We share similar proposal generation schemes with Lempitsky et al [?], but instead of the HS algorithm, we use more recent Farneback algorithm and changes its pyramid levels from 1 to 5 and use different polyN 3, 5, 7. Besides the clustering idea, [?] also suggests to use more proposals generation schemes based on the current solution. We add three simple proposal generation schemes to better explore the power of our parallel pipeline. In *shift proposal*, the flow field in the current solution is shifted in either x direction or y direction for either 1, 2 or 3 pixels. In *stagger proposal*, the flow field is shifted by a vector randomly drawed from a Gaussian distribution. In *disturb proposal*, each flow value in the field is indenpendantly shifted by a vector randomly drawed from a Gaussian distribution.]

Competing methods

Fusion Move method in Lempitsky's paper is the first natural contender. While they did not consider parallel implementation, it is straightforward to combine the idea of Parallel Alpha Expansion and Fusion Move. Therefore, the second competing method is "Parallel Fusion Move" (PFM), which is equivalent to Parallel Alpha Expansion with constant label solutions replaced by solution proposals. One problem of PFM is that infinite number of solution proposals can be generated in their algorithm, and we do not know when to stop and perform the final sequential fusion (See Parallel Alpha Expansion architecture in Fig. 1). In our experiments, we manually picked time limits to initiate the final fusion to make the comparisons fair. The last contender is the mix of the Hierarchical Fusion and the Fusion Move methods, dubbed "Hierarchical Fusion Method" (HFM), where they start from solution proposals as opposed to constant labels. One problem of HFM is that the number of necessary solution proposals is unknown before solving the problem, and the tree of labels cannot be defined. In our experiments, we have manually picked a good number (X [yasu: how many Hang?]) [chen: 77 proposals in total (We first generate 11 proposals using LK and Farneback algorithms which do not require current solution and generate other proposals based on these initial proposals).] to generate proposals. The fusions are binary in these methods and we have used QPBO.

Swarm fusion architectures

The three swarm architectures in Fig. 1 (SF-SS, SF-MF, SF) have been evaluated against the competing methods. The same configurations as in the stereo problem have been used for optical flow, that is, $\alpha = 2$ ($\beta = 0$) for SF-SS, the iteration of $(\alpha = 1, \beta = 0)$ and $(\alpha = 0, \beta = 1)$ for SF-MF, and $(\alpha = 1, \beta = 1)$ [chen: $(\alpha = 2, \beta = 0)$ and $(\alpha = 0, \beta = 2)$] for SF. Note that we have used the same set of proposal generation schemes as the competing methods. We have used TRW-S for multi-way fusion and QPBO for binary fusion. [chen: Just for clarification, I always use TRW-S and just set the number of labels to be 2. It is the same with QPBO as mentioned by Pushmeet.]

2 k-way 1 solution sharing. [chen: ?]

4.3 Swarm Fusion layered depthmap estimation

Our last problem is layered depthmap estimation, recently proposed in [**?**] (see the anonymous paper in the supplementary material). The problem seeks to infer layered depthmap representation from an RGBD image, where each layer is a piecewise smooth segmented depthmap. This is essentially a multi-layer extension of Surface Stereo algorithm [**?**]. Layered depthmap estimation is one of the most difficult kinds in MRF inference due to its massive solution space [chen: and complex energy function?]. The number of labels per pixel is exponential in the number of layers, and is an order of 100,000 [yasu: is this number correc?] [chen: The number is correct but "an order of" sounds strange to me. Maybe say:

usually between 100,000 and 10,000,000.]. We copy their problem formulation and the proposal generation schemes. [2]

Competing methods

Solution proposals depend heavily on the current solution, eliminating the possibility of using Hierarchical Fusion, which needs to enumerate all the proposals to start. Therefore, viable competing methods are Fusion Move and Parallel Fusion Move. The fusions are binary, for which we use QPBO.

Swarm fusion architectures

The three swarm architectures with the same configurations as in the optical flow problems have been evaluated. [chen: same with optical flow, but slightly different with stereo (1 sharing in 3 iterations instead of 5 iterations).]

5 Experimental results

We have implemented the algorithms in C++ and conducted the experiments on Linux PCs with Intel Core i7 processor with 4 cores, enabling 8 threads with the hyper threading technology. We have used the Graph-cuts optimization code written by Veksler, using the libraries provided by Boykov and Kolmogorov [**?**,**?**,**?**,**?**]. We have used the QPBO and TRW-S implementations by Kolmogorov [**?**,**?**].

The figure illustrates the convergence rate of three competing methods (alpha expansion, parallel alpha expansion, hierarchical fusion) against our swarm fusion method (only binary fusion without proposal generation [chen: We generate proposals on the fly for optical flow and layered depth map. So the time actually includes proposal generation stage which can be ignored. I think we can just delete this sentence.]).

[hang: figure file missing]

[chen: From the convergence plot of layered depth map estimation, we can see that, Fusion Move, Parallel Fusion Move, and FS-MF all stalk at a high energy state. The reason is binary fusion of solution proposals (either from others or self) is too limited to further decrease the energy due to the complexity of the

[2] Authors have proposed a novel fusion scheme, where a solution subspace instead of a single solution is generated by a proposal generation scheme. Since a solution subspace is equivalent to multiple solution proposals [chen: It is a good explanation for this paper's purpose, but from the perspective of our previous work, I am reluctant to say a subspace equals to multiple solution proposals. When we design proposals, we find the subspace directly and there's no concept of solution proposals (we later simplify our subspace proposals to binary labels just for the comparison reason). It is essentially the same, but a subspace proposal is conceptually more general than concatenated solution proposals.], their algorithm can be easily integrated into our swarm fusion framework. However, competing fusion methods (e.g., Parallel Fusion Move or Hierarchical Fusion) cannot handle a solution subspace proposal, making it impossible to conduct fair comparative evaluations. We choose to use a simple solution proposal for experiments in this paper.

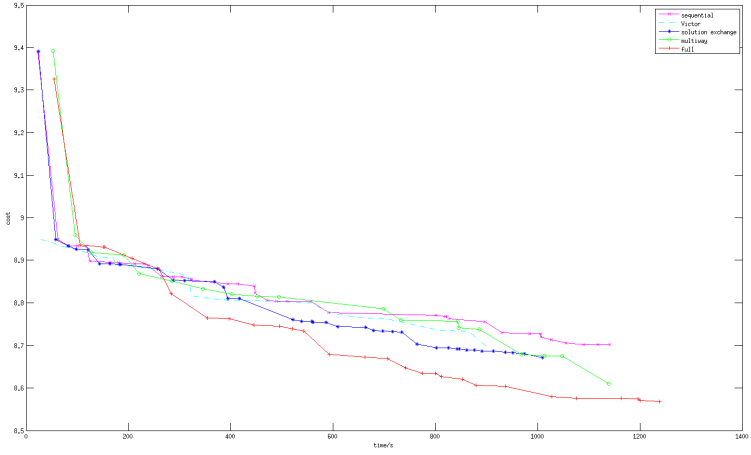

Fig. 3.

problem itself. Only when multi-way fusion is used (in FS-SS and FS), it becomes possible for the energy to further decrease. This coincides with the observation in [?] that binary fusion of proposal solutions is not as powerful as their subspace fusion which is a special form of multi-way fusion here.]

[chen: As shown in 5, the multi-way fusion and solution sharing enabled by our uniform framework play a key role for improving performance in different settings. For better understanding our framework, we examine the role played by each factor more closely by varying each factor while keeping others the same. There are four parameters in our framework: *the number of threads*, *the number alpha*, *the number beta*, *the number of iterations between each solution sharing*.]

6 Conclusion and future directions

We have proposed a novel MRF inference framework, Swarm Fusion, in parallel computing environments. The framework is general and makes popular inference techniques such as Alpha Expansion, Fusion Move, Parallel Alpha Expansion, or Hierarchical Fusion, its special cases. Our experiments have revealed that the framework exploits parallel computational resources and achieves faster convergence, epseically for challenging problems. Our first future work is to conduct experiments on cloud computing environments, in particular, the MapReduce programming model, where the roles of mappers and reducers exactly correspond to the processes of parallel multi-way fusion and solution sharing, respectively. Another future work is the automatic configuration of the Swarm Fusion architecture. Our experiments have shown that optimal architectures are different for different problems. An interesting direction is to adaptively change its

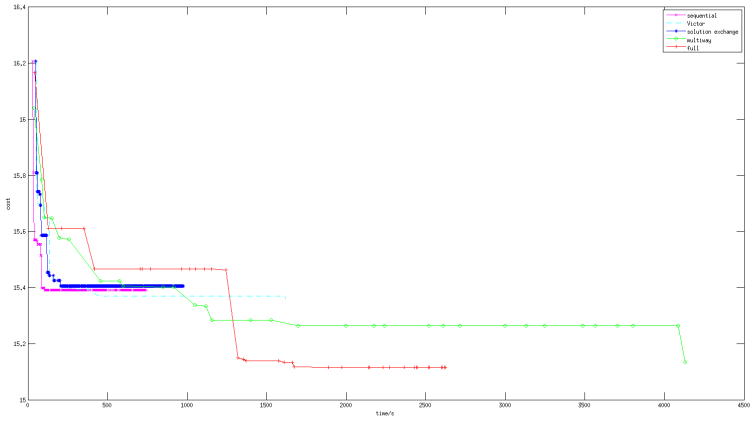

Fig. 4.

architecture during the computation, for example, switching to simple parallel alpha-expansion for easy problems, or increasing the rate of solution exchanges when solutions vary significantly across threads. Parallel MRF inference has been a relatively under-explored topic in Computer Vsion. Our idea is very simple, where adapting the Swarm Fusion framework require minimal coding. We believe that this paper would immediately benefit tens of thousands of Computer Vision researchers or engineers in the world, who currently solve MRF problems. We will share our source code and the datasets with the community.