General Search Algorithms for Energy Minimization Problems

Dmitrij Schlesinger

Dresden University of Technology*

Abstract. We describe a scheme for solving Energy Minimization problems, which is based on the A^* algorithm accomplished with appropriately chosen LP-relaxations as heuristic functions. The proposed scheme is quite general and therefore can not be applied directly for real computer vision tasks. It is rather a framework, which allows to study some properties of Energy Minimization tasks and related LP-relaxations. However, it is possible to simplify it in such a way, that it can be used as a stop criterion for LP based iterative algorithms. Its main advantage is that it is exact – i.e. it never produces a discrete solution that is not globally optimal. In practice it is often able to find the optimal discrete solution even if the used LP-solver does not reach the global optimum of the corresponding LP-relaxation. Consequently, for many Energy Minimization problems it is not necessary to solve the corresponding LP-relaxations exactly.

1 Introduction

This work is motivated mainly by the following observations.

At the time there are many algorithms for approximate solutions of Energy Minimization problems, based on LP-relaxation techniques [9,7,6,10,8,5]. However, strongly polynomial algorithms for resulting LP problems are still unknown. Therefore, in practice it is necessary to use approximations, i.e. the results of an iterative procedure, which are reached after finite time. Even if it would be possible to solve LP-relaxations exactly, it is not known in general, whether there is a gap between the optimal relaxed solution and optimal discrete solution of the initial Energy Minimization task. Even if it is known that there is no gap, it is in general not clear, how to obtain the optimal discrete solution, given the optimal relaxed one. To answer the latter two questions it is necessary to solve a Constraint Satisfaction Problem that is NP-complete by itself. Despite of these theoretical drawbacks, the LP-relaxation methods became extremely popular, because in practice it is almost always possible to extract good results from (possibly suboptimal) continuous solutions in an heuristic but reasonable way. Unfortunately, much less papers study relationships between different relaxations of underlying discrete optimization problems as well as relationships between discrete problems and corresponding relaxations. Summarizing, there are many open questions in the scope of LP-relaxation techniques, if they are applied to Energy Minimization problems. Especially the question of strongly polynomial solvability of LP-relaxation is in a certain sense not natural, because this question relates solely to linear programming

* Supported by Deutsche Forschungsgemeinschaft, Grant No. FL307/2-1.

D. Cremers et al. (Eds.): EMMCVPR 2009, LNCS 5681, pp. 84–97, 2009.

techniques – i.e. it has in principle nothing in common with the initial discrete optimization problem.

On the other hand there are much less algorithms for Energy Minimization, which have a "discrete nature". A classical example is Dynamic Programming [1]. This algorithm does not relate to any continuous optimization. It just performs a predefined number of operations and gives the solution. Unfortunately, this approach is not suitable for many computer vision problems, because it is applicable only for simple graphs. Another group of algorithms can be characterized as an iterative search – they consider a subset of labelings in each iteration. The simplest method of such kind is Iterated Conditional Modes [2], more elaborated techniques are e.g. α-expansion, α/β-swap [3] and their derivatives. It is noteworthy that these algorithms use MinCut based techniques (which are again indirectly based on LP-relaxations) in order to perform an elementary search step. Unfortunately, here it is often difficult to state, for which tasks these approaches give global optimal solution, what is the precision for general Energy Minimization tasks etc. Summarizing, the branch of research, which is formed by discrete optimization techniques, seems to be neglected (at least in the scope of computer vision problems).

We propose to consider Energy Minimization problems primarily as *discrete* optimization problems and to apply general search techniques to them. In particular, we describe in this paper a scheme, based on the A^* algorithm [4]. Appropriately chosen LP-relaxations are used thereby as heuristic functions. This idea by itself is not new for discrete optimization in general. However, to our knowledge, it was not considered before taking into account special properties of Energy Minimization problems, which result from typical computer vision tasks (like e.g. segmentation, stereo-reconstruction etc.). The main specific here is that it is not possible as a rule to solve the corresponding LP-relaxations exactly due to a very large number of variables and constraints.

The proposed scheme is quite general and therefore can not be applied directly for real computer vision tasks. It is rather a framework, which allows to study some properties of Energy Minimization tasks and related LP-relaxations. However, it is possible to simplify it in such a way, that it can be used as a stop criterion for LP based iterative algorithms (such as Message Passing, the Subgradient method etc.). Its main advantage is that it is exact – i.e. it gives the globally optimal discrete solution for some tasks. For other problems it never stops (e.g. if there is an essential gap between the values of the best discrete solution and the global optimum of the corresponding LP-relaxation). In other words, it never produces a solution that is not optimal. The other important property is that it is often able to find the optimal discrete solution even if the used LP-solver does not reach the global optimum of the corresponding LP-relaxation. To conclude, for many Energy Minimization problems it is not necessary to solve the corresponding LP-relaxations exactly in order to get the globally optimal discrete solution.

2 Notations and Definitions

Problem statement. For simplicity we consider Energy Minimization tasks (MinSum problems) of second order. Let $G = (R, E)$ be a graph (also called the problem graph) with the node set R, a particular node is referred as $r \in R$, the edge set is $E = \{\{r, r'\}\}$.

Let K be a finite label set and $f : R \to K$ be a labeling, i.e. $f(r)$ denotes a label $k \in K$ chosen by the labeling f in a node $r \in R$. Let $q_r : K \to \mathbb{R}$ and $g_{rr'} : K \times K \to \mathbb{R}$ be quality functions, which assign real values to the labels in each node and to the label pairs in each edge respectively. The quality of a labeling is the sum

$$Q(f) = \sum_{r \in R} q_r\big(f(r)\big) + \sum_{\{rr'\} \in E} g_{rr'}\big(f(r), f(r')\big), \tag{1}$$

the task is to find

$$\begin{aligned}&(\arg)\min_f Q(f) = \\ &= (\arg)\min_f \left[\sum_{r \in R} q_r\big(f(r)\big) + \sum_{\{rr'\} \in E} g_{rr'}\big(f(r), f(r')\big)\right].\end{aligned} \tag{2}$$

In addition we will need the following notations. Let us denote by $G' = (R', E')$ the subgraph induced by a node subset $R' \subset R$, i.e. $E' = \{\{r, r'\} \mid \{r, r'\} \in E, r, r' \in R'\}$. A *partial labeling* $f_i : R' \to K$ is a restriction of f into the subset R'. The index will always depend on the context, determining in each particular case, what subset R' is meant, which labeling on it is considered etc. For a partial labeling $f_i : R' \to K$ and a node $r \in R'$ we will say, that the partial labeling *covers* this node. The quality $Q(f_i)$ of a partial labeling is the same as (1) but summed over the corresponding subgraph G' instead of the whole problem graph G. A *subtask* $\mathcal{A}(f_i)$ induced by a partial labeling f_i is a MinSum problem, that is built from the original one (2) by fixation of the labels in those nodes, which are covered by f_i, that is

$$(\arg)\min_{f_j} \left[Q(f_j) + \sum_{\substack{\{rr'\} \in E \\ r \in R', r' \in R/R'}} g_{rr'}\big(f_i(r), f_j(r')\big)\right], \tag{3}$$

where $f_j : R/R' \to K$ are partial labelings, which cover the complement R/R' (thereby we omit the constant part $Q(f_i)$).

LP-relaxation. A common approach to cope with energy minimization tasks (2) is based on LP-relaxation technique. The first step is to introduce integer weights $\lambda(\cdot) \in \{0, 1\}$ for each pair (r, k) as well as for each triple $(\{rr'\}, k, k')$ and to rewrite (2) in the form[1]

$$\begin{aligned}&\sum_r \sum_k q_r(k)\lambda_r(k) + \sum_{rr'} \sum_{kk'} g_{rr'}(k, k')\lambda_{rr'}(k, k') \to \min_\lambda \\ &\text{s.t.} \\ &\sum_k \lambda_r(k) = 1 \quad \forall r \\ &\sum_{k'} \lambda_{rr'}(k, k') = \lambda_r(k) \quad \forall r, r', k \\ &\lambda_r(k), \lambda_{rr'}(k, k') \in \{0, 1\}.\end{aligned} \tag{4}$$

[1] Sometimes we omit brackets and "$\in$" relations for readability.

Then the last condition is relaxed, i.e. substituted by $\lambda(\cdot) \in [0,1]$, which gives a task of linear programming.

We prefer to consider the task (4) using the notation of equivalent transformations. They are defined as a superpositions of following operations:

$$\hat{q}_r(k) = q_r(k) + \delta, \quad \hat{g}_{rr'}(k,k') = g_{rr'}(k,k') - \delta \ \ \forall k' \tag{5}$$

with an arbitrary finite constant δ. The new functions $\hat{q}_r$ and $\hat{g}_{rr'}$ represent an equivalent task. The application of an equivalent transformation is also called reparametrization of a task. The dual of (4) can be formulated using these notations as follows:

$$\sum_r \min_k \hat{q}_r(k) + \sum_{rr'} \min_{kk'} \hat{g}_{rr'}(k,k') \to \max_{\hat{q},\hat{g}}, \tag{6}$$

where the functions $\hat{q}$ and $\hat{g}$ can be obtained by equivalent transformations from the original functions q and g. For a subtask $\mathcal{A}(f_i)$, we denote by $LP(f_i)$ the solution (the value) of its LP-relaxation (6).

General search algorithms, A^*

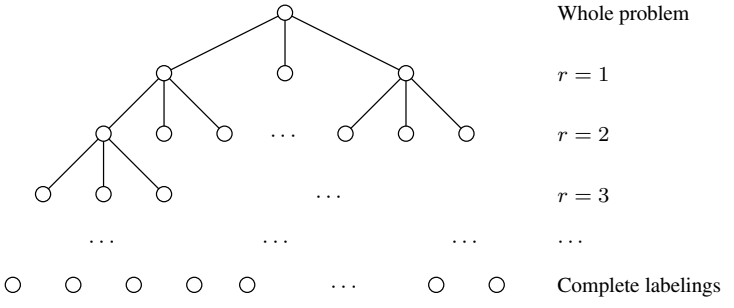


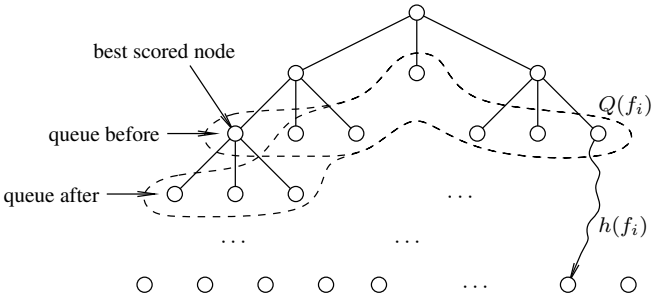
Fig. 1. Search tree

General search algorithms are described very extensive in the literature. Therefore we omit here formal definitions and just consider, how a MinSum task can be formulated as a general search problem. The main idea is to decompose the whole task into subtasks followed by recursive decompositions of these. Let the nodes of our MinSum problem be enumerated. We may decompose the whole problem by fixation of labels e.g. in the first node $r = 1$. In that way we obtain $|K|$ subproblems, each one having one node less than the initial task. Doing such decomposition recursively a search tree is built, which is presented in Fig. 1. Each layer (a particular depth value) corresponds to a node r of the problem graph. Each tree node in a layer r corresponds to a partial labeling $f_i : R' \to K$ with $R' = \{1, 2 \dots r\}$. According to this the subtree with the root f_i represents the subtask $\mathcal{A}(f_i)$. The root of the tree represents the whole problem, the leaves correspond to complete labelings. The quality of a node f_i is the quality $Q(f_i)$ of the corresponding partial labeling. The task is to find a leaf of minimal quality.

We will also need the following notation. A partial labeling f_j is called *successor* of another partial labeling f_i iff

$$f_i : R' \to K, \quad f_j : R' \cup \{r \notin R'\} \to K, \quad f_j(r) = f_i(r) \ \ \forall r \in R'. \tag{7}$$

We will say that the partial labeling f_j is obtained from f_i by fixation of a label in a node $r \notin R'$. The set of all successors for a partial labeling f_i, which can be obtained in such a way, is denoted by $succ_r(f_i)$. At the same time it denotes the set of all children of a tree node f_i in the search tree.




```
Put the root into the queue
Repeat {
   Take the best scored node from the queue
   if it is a leaf
      return it as the solution
   compute all successors
   score them according to Q(f_i) + h(f_i)
   put them into the queue
}
```

Fig. 2. A^* algorithm

A common approach for general search problems is to use heuristic search techniques to evaluate the search tree in order to reach a leaf of minimal quality. In particular we will use the A^* algorithm. Its functionality is illustrated in Fig. 2. In each step there is a subset of tree nodes (called queue), which are not evaluated so far. At the very beginning it consists only of root of the tree. Each node in the queue is scored according to the *evaluation function*, which is the sum of the node quality and an *heuristic* function (denoted by $h(f_i)$). The latter is an estimator for the difference between the quality of the node and the best leaf in the corresponding subtree. In each step the algorithm takes the node with the best score from the queue. If it is a leaf, the algorithm finishes and returns this node as the solution. Otherwise, the node is *expanded*, i.e. all successors (children of this node in the search tree) are computed. They are scored as described above and put into the queue.

The properties of A^* are determined primarily by the type of the used heuristic function. An heuristic function is called *admissible* (optimistic), if it never

overestimates the real difference between the qualities of the node and the best leaf in the corresponding subtree:

$$h(f_i) \leq \underset{f \in subtree(f_i)}{\arg\min} Q(f) - Q(f_i) = h^*(f_i). \tag{8}$$

An heuristic is called *monotone* if

$$Q(f_i) + h(f_i) \leq Q(f_j) + h(f_j) \quad f_j \in succ(f_i) \tag{9}$$

holds, i.e. the heuristic is the more precisely (and therefore the less optimistic) the closer to the leaves. The A^* algorithm is exact (always finds a global solution) if the heuristic function is admissible[2].

For the complexity analysis of A^* it is usually assumed that the complexity of computation for $Q(f_i)$ and $h(f_i)$ is a constant. Let n be the maximal number of tree nodes in the queue during the search. The time complexity of A^* is then $O(n \log n)$. Alternatively, the number of expansions can be considered, which are necessary to solve the problem. In this case it is assumed that the time complexity of one expansion step (one loop iteration in Fig. 2) is a constant. In both variants the crucial question is, how the considered numbers (n or the number of expansions) depend on the problem size, for example on the number of nodes $|R|$ in the problem graph. To be precise, we will use the notation "time complexity" having in mind the number of expansions depending on the problem graph size.

The time complexity of the A^* algorithm is exponential in general. However, if the heuristic function is admissible and monotone, then a complexity analysis is possible, that takes into account the precision of the heuristic function. In particular, A^* has linear time complexity, if the heuristic function is exact (denoted by $h^*(f_i)$ in (8)). It means that the A^* algorithm behaves like the depth-first search, i.e. a tree node of maximal depth is always chosen for expansion. In this case the maximal number of nodes in the queue is $O(|R||K|)$ and exactly $|R|$ expansions should be performed in order to reach an optimal leaf.

3 General Scheme

The main idea of our approach is to use LP-relaxation $LP(f_i)$ as the heuristic function $h(f_i)$ for the A^* search. First of all, is should be stated that this is an admissible heuristic, since the LP-relaxation of a discrete optimization task represents a lower bound for its solution. Therefore A^* is exact. Secondly, it can be easily seen that this heuristic is also monotone. The key point to show this is to note that for each pair f_i and $f_j \in succ_r(f_i)$

$$Q(f_j) + LP(f_j) = Q(f_i) + LP'(f_i) \geq Q(f_i) + LP(f_i) \tag{10}$$

holds, where $LP'(f_i)$ is the same as $LP(f_i)$ but with additional constraints for λ (see (4)), which fix the label $k = f_j(r)$ in the node r – i.e. both LP-relaxations have the

[2] In general, the heuristic must be both admissible and monotone to guarantee optimality. In our case it is enough that it is admissible only, because the set of leaves is finite.

same objective function but in $LP'(f_i)$ the set of feasible solutions is a proper subset of those in $LP(f_i)$.

We call a discrete task LP-solvable, if there is no gap between the values of its solution and its LP-relaxation. We call a task strictly LP-solvable, if the same holds for all partial labelings, which may be needed during the evaluation of the search tree[3]. It is obvious that in the latter case A^* has linear time complexity, since the heuristic function becomes exact. It is also clear that non strict LP-solvability does not guarantee linear complexity in general.

On the other hand, it is easy to see that strict LP-solvability is only a sufficient condition for linear complexity, but not a necessary one. Even if there is a gap (the task is not LP-solvable), A^* may have linear complexity, if this gap is relatively small – i.e. if it is not big enough to prefer partial labelings of non maximal depth during the search. A weaker sufficient condition can be formulated from A^* itself as follows. The A^* algorithm has linear time complexity if there exist a sequence of partial labelings $f_1, f_2 \dots f$, $f_i : \{1 \dots i\} \to K$, $f_{i+1} \in succ_{i+1}(f_i)$, so that

$$Q(f_i) + LP(f_i) < Q(f_l) + LP(f_l) \quad \forall f_l \in succ(f_j)/f_{j+1}, j < i \tag{11}$$

holds for each f_i in the sequence. But even this is not a necessary condition. If the above inequality is satisfied non strictly, A^* may behave as depth-first search as well, because it may find the "right" sequence of partial labelings by chance.

4 A^* Based Stop Criterion for LP-Solvers

Let us remember that it is not possible to solve the needed LP-relaxations exactly and efficiently for real tasks. Known LP-solvers are iterative algorithms, that only approach the global optimum. Consequently, the optimal solution is guaranteed only in infinite time. Therefore we are constrained to use approximations, i.e. the results of these algorithms after a finite time. Let us denote these approximations by $\tilde{LP}(f_i)$. In this case we can not guarantee the monotonicity of the heuristic anymore. However, it remains admissible, if methods based on maximization of the dual energy are used (in this case $\tilde{LP}(f_i) \leq LP(f_i)$ holds, since these methods try to maximize the lower bound for optimal energy). Therefore A^* remains exact. It makes in principle possible to use these methods with A^* to search for optimal discrete solution.

Let us consider again the behavior of A^* but from a slightly other point of view, taking into account that we use approximations $\tilde{LP}(f_i)$ instead of the exact values $LP(f_i)$. Assume a task for which A^* does not behave like depth-first search, i.e. at some stage it chooses for expansion a tree node f_i, that has non maximal depth – the condition (11) is not satisfied. Obviously, it can not happen, if the task is strictly LP-solvable and LP-relaxations are computed exactly. Consequently, there are only two possible reasons for such a behavior: either the task is not strictly LP-solvable or the approximations $\tilde{LP}(f_i)$ are too far from the corresponding $LP(f_i)$. Suppose, it is known that the task is strictly LP-solvable. Then only the second possibility remains – i.e. for some f_i used during the search the corresponding approximation $\tilde{LP}(f_i)$ was computed too coarsely.

[3] For example submodular tasks have this property.

There are basically two ways to proceed further in this situation. The first one is to allow the search to deviate from the depth-first like behavior, i.e. proceed A^* further using current (coarse) approximations $\tilde{LP}(f_i)$. In this case however, the search procedure often starts to massively expand tree nodes of non maximal depth (note that the heuristics $h(f)$ for leaves are zero, i.e. exact per definition). In short, we can not forecast the further behavior of A^* – i.e. it often turns into a complete search.

The second way is to attempt to improve the current approximations $\tilde{LP}(f_i)$ in order to preserve the depth-first like behavior of the search. Unfortunately, this idea can not be used directly, because we can detect the necessity to improve these approximations only *after* the search was already performed (the approximations $\tilde{LP}(f_i)$ were already computed). Nevertheless, we would like to discuss this idea in a little bit more detail. Let us imagine, that we have a hypothetical algorithm at our disposition, which has the following properties. First, it is an iterative procedure that maximizes the lower bound of the energy – i.e. it is something like a usual LP-solver (e.g. Diffusion or Message Passing etc.). Second, besides of the lower bound, it provides some kind of information, that allows to easily compute approximations $\tilde{LP}(f_i)$ for all partial labelings which may be needed during the A^* search – i.e. without to solve the LP-relaxations for each partial labeling separately. Furthermore, let us assume, that the algorithm improves during its work this additional information in a similar way as it improves the lower bound for the energy. In other words, the additional information allows to compute $\tilde{LP}(f_i)$ the more precisely, the more time is spent. Then we could use the idea to detect the necessity of an improvement for $\tilde{LP}(f_i)$ as follows. We run this hypothetical procedure and let it compute all information, that may be needed for estimation of $\tilde{LP}(f_i)$ during the search. Once a while we call the A^* search and check, whether it behaves like the depth-first search. If yes, then we have found the optimal discrete solution. Otherwise we let this hypothetical procedure work further to improve the lower bound as well as the additional information. Summarizing, the depth-first like behavior of the A^* algorithm can serve as a stop criterion for LP-solvers, which have the above mentioned properties.

Unfortunately, we have no such hypothetical LP-solver at our disposition. However, it is possible to to adapt existing algorithms in order to compute the necessary additional information approximately. Let us consider again the task of the maximization of the dual energy (6). In particular, let $\hat{q}$ and $\hat{g}$ be quality functions obtained by an iterative LP-solver from the original q and g at some stage of its work. Note, that we can omit without loss of generality the node terms $\hat{q}_r$, because they can be always set to zero by an equivalent transformation (5) with $\delta = -\hat{q}_r(k)$. Secondly, for a particular instance of $\hat{g}$ in (6) we can always subtract a constant from each $\hat{g}_{rr'}$. In doing so we change both the initial discrete task (qualities of all labelings) and the value of (6) by the same number. When subtracting the minimal value from each $\hat{g}_{rr'}$ the following holds. The current value of the LP-relaxation (the current lower bound) is zero. All functions $\hat{g}_{rr'}$ are nonnegative, all functions $\hat{q}_r$ are zero, the value of the best labeling is therefore nonnegative as well. After such a normalization the numbers

$$marg(r,k) = \sum_{r':\{rr'\}\in E} \min_{k'} \hat{g}_{rr'}(k,k') \tag{12}$$

can serve as an approximation for min-marginals for a node r and a label k. The current value of the heuristic $h(f_i)$ for a partial labeling $f_i : R' \to K$ can be approximated by

$$h(f_i) = \sum_{\substack{\{rr'\}\in E \\ r\in R', r'\in R/R'}} \min_k \hat{g}_{rr'}\big(f_i(r), k\big), \tag{13}$$

i.e. it is the same as min-marginals (12) but accumulated only over those edges, which connect nodes from the fixed part R' with nodes from its complement. It is easy to see that this heuristic does not represent the needed LP-relaxations for all partial labelings. However, the values computed by (13) are guaranteed not greater and therefore represent an admissible heuristic. Moreover, it can be easily seen that this heuristic is monotone as well.

f^* is the current best partial labeling,
  set it empty (no labels are fixed)
$next\_best$ is the value of the evaluation function for the
  best tree node not expanded so far, despite of f^*,
  set $next\_best = \infty$

For each node r {
  compute the set of successors $S = \{f_i, f_i \in succ_r(f^*)\}$
  and their evaluation functions $e(f_i) = Q(f_i) + h(f_i)$;
  extend f^* to the best successor: $f^* = \arg\min_{f_i \in S} e(f_i)$;
  if $e(f^*) > next\_best$
    return "solution not found"

  choose f' as the next best successor: $f' = \arg\min_{f_i \in S, f_i \neq f^*} e(f_i)$;
  if $e(f') < next\_best$
    $next\_best = e(f')$
}
return f^* as the best solution

Fig. 3. A^* based stop criterion

Summarized, the A^* based stop criterion is presented in Fig. 3. Note, that it is not necessary to store all partial labelings in the queue in order to detect the depth-first like behavior. Only the value of the evaluation function for the best "second pretender" is needed (denoted by $next\_best$ in Fig. 3). It reduces the time complexity of the A^* search from $O(n \log n)$ (as in the general case) to $O(n)$, where n is the number of partial labelings needed during the search, i.e. the maximal number of nodes in the search tree. If the values $\min_{k'} \hat{g}_{rr'}(k, k')$ are precomputed in advance[4] for each $r < r'$ and k, the computation of (13) for a node r can be done in a time, that is proportional to

[4] In fact, the solvers compute these numbers in order to perform equivalent transformations. Therefore, they should be only stored for further use by the stop criterion.

the number of edges, which are incident to r. Summarizing, the overall time complexity of the criterion is $O(|E||K|)$.

We would like to stress that this stop criterion does not indicate the exact solution of the corresponding LP-relaxation. It detects only the possibility "to extract" the global optimal discrete solution from the current state of a continuous optimization procedure. In the case the solution was found, it is not known in general, whether the task is strictly LP-solvable or not, whether there is a gap (the task is not LP-solvable), whether the used LP-solver reached the global optimum of the LP-relaxation. But in this case it is not necessary to answer these questions, because the discrete solution was already found. Summarizing, this approach gets rid (to some extent) of the necessity to look for algorithms for exact solution of LP-relaxation.

The interpretation of the non depth-first like behavior of A^* depends on the additional knowledge about the task and/or the used LP-solver. If it is known that the task is strictly LP-solvable, then the non depth-first like behavior indicates that the LP-solver has not reached the optimum so far or the coarsening (13) is too pure. If it is known that the solver is already in optimum, then the task is not strictly LP-solvable or the coarsening (13) is too pure again and so on.

In most practical cases we do not have such additional knowledge. It is then obviously possible that the whole loop (LP-solver with the A^* based stop criterion) never stops. If it is nevertheless necessary to find the exact solution of the initial discrete problem, it is necessary to perform the A^* search in a more general form – i.e. to allow it to have non depth-first like behavior.

5 Experiments

First of all, it is necessary to justify, what we would like to demonstrate by the experiments. The main goal is to show that the A^* based stop criterion gives (as a rule) the exact global discrete solution in the situation that the used LP-solver does not reach the global optimum of the corresponding LP-relaxation. It can be done e.g. by comparison of the qualities for the found discrete solution and the lower bound, found by the used LP-solver – in our case the latter is zero since the task is normalized as described above. The second goal is to examine, whether our stop criterion allows to reduce the time complexity of the used LP-solver compared with a "simple" stop criterion, which does not relate to the best discrete solution, but only detects the optimum of the LP-relaxation – i.e. whether it allows to stop LP-solver essentially earlier. We use the following procedure as such simple stop criterion. First, the approximations for min-marginals (12) are computed. A "seemingly good" labeling is produced by fixation of labels with best min-marginals in each node. If the quality of the labeling produced in such a way is zero, then this labeling is optimal and the algorithm reached the global optimum of the LP-relaxation for the whole task[5].

In our experiments we used the Diffusion algorithm [9,10] and the Subgradient method [7] for maximization of (6). In short, they both are iterative ones, that apply equivalent transformations (each one in its own manner) in order to maximize (6).

[5] Obviously, this is a sufficient but not necessary condition for the global optimum.

First, we tested the approach for problems, generated as follows. The problem graph corresponds to a grid with 4-connected neighborhood structure (128×128 for Diffusion and 64×64 for the Subgradient method). The values of the functions q_r are generated uniformly in interval $[0, 1]$. The functions $g_{rr'}$ are chosen as Potts model, i.e. $g_{rr'}(k, k') = \alpha \cdot \mathbb{1}(k \neq k')$. One experiment is organized in the following way. For a given task we call the LP-solver. After each iteration (some portion of work) we check the A^* based stop criterion. If a predefined number of iterations is done and the stop criterion does not produce the solution, we cancel the experiment and state that the solver does not converge for this task. If the criterion was successful (the optimal discrete solution was found), we note the number of iterations made so far. After that we let the solver work further until the simple stop criterion is satisfied. In doing so we note the number of iterations, which was necessary for the solver to converge after the best labeling was found by the A^* based stop criterion. Then we compute ratio of these two numbers of iterations. For certain combinations of the Potts parameter α and the number of labels $|K|$ we performed 10 experiments per combination and average the computed ratio over the successful experiments (i.e. solver converges). These ratios are summarized in Fig. 4. The values given after slash are the numbers of successful experiments for each combination. Empty cells indicate combinations of $|K|$ and α, for which the solvers never converged (either the problems with these combinations have as a rule an essential gap or the solvers were not able to narrow the global optimum of the LP-relaxation good enough in acceptable time).

In addition we would like to note the following. In the majority of the experiments (92.2% for Diffusion and 97.9% for the Subgradient method) the value of the best labeling found by our stop criterion was greater then zero. It confirms our main hypothesis – the criterion finds as a rule the optimal discrete labeling in the situation, that the used LP-solver does not reach the global optimum of the corresponding LP-relaxation. In such cases some additional iterations (together with the normalization) were necessary

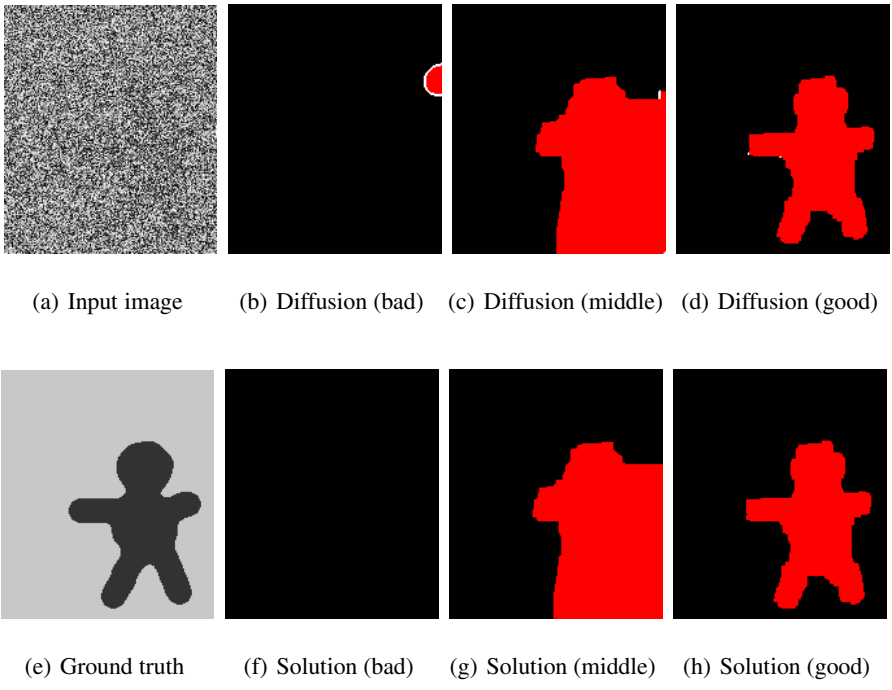
$\|K\| \setminus \alpha$	0.3	0.4	0.5	0.6	0.7	0.8
2	.16/10	.04/10	.01/10	.06/10	.89/10	.94/10
3			.02/8	.03/10	.85/10	.71/10
4				.09/9	.20/9	.46/10
5				.07/8	.16/10	.34/10

(a) Diffusion (ratios are multiplied by 100)

$\|K\| \setminus \alpha$	0.3	0.4	0.5	0.6	0.7	0.8
2	.42/10	.46/10	.21/10	.07/10	.10/10	.05/10
3	.07/1	.75/6	.16/9	.04/10	.03/9	.03/10
4		.49/3	.36/9	.07/9	.18/10	.03/10
5		.44/1	.27/5	.05/9	.02/10	.02/9

(b) Subgradient method

Fig. 4. Average ratios for generated problems



(a) Input image (b) Diffusion (bad) (c) Diffusion (middle) (d) Diffusion (good)

(e) Ground truth (f) Solution (bad) (g) Solution (middle) (h) Solution (good)

Fig. 5. Results for an artificial image

for simple stop criterion to find the optimal discrete solution of zero quality. There were also situation (6 times for Diffusion and 18 times for Subgradient), that the LP-solver did not converge in acceptable time after the best labeling was already found by the A^* based stop criterion. The typical situation for Diffusion was, that the value of the dual energy did not change at all for a very long time[6]. For the Subgradient method it was always the case that the dual energy changed, but so slowly that we were not able to wait. Sometimes the dual energy even decreased after the solution was found[7]. Comparing Diffusion and Subgradient, we observed that the A^* stop criterion has much more essential impact for Subgradient (sometimes more as 75% additional iterations were necessary) as for Diffusion (about a fraction of a percent).

The next experiment was made for an artificial image shown in Fig. 5. The input image in Fig. 5(a) was produced from the ground truth (Fig. 5(e)) by generation a gray-value in each pixel independently according to segment specific probability distributions. The resulting MinSum problem was a Potts model with negated logarithms of corresponding probability values for q_r. We performed experiments with Diffusion as the LP-solver for a "sequence" of models, starting from one with "bad" functions q_r (the corresponding probability distributions are almost the same and uniform for both

[6] Note, that Diffusion does not converge in general to the global optimum of the LP-relaxation.

[7] The Subgradient method converges to the global optimum of the LP-relaxation but not monotonously.

(a) Input image (b) Solution (bad) (c) Solution (good)

Fig. 6. Results for a real image

segments – Fig. 5(b,f)) and finishing with "good" one (the corresponding probability distributions are the true ones – Fig. 5(d,h)). Figures 5(f,g,h) show the solutions, found by the A^* based stop criterion in each stage. The figures 5(b,c,d) show the results produced by the simple stop criterion (labels with the best approximation of min-marginals (12) were chosen in each pixel independently) just after the exact solution is found by A^*. Those edges for which the chosen state pair has non zero quality are marked white. A more or less significant effect was observed mainly for bad models – the segmentations produced by the simple stop criterion and by A^* differ essentially and a relatively big number of additional iterations was necessary for Diffusion to find optimal segmentation. This impact decreases as the model becomes better – in Fig. 5(d) there are almost no edges with non zero qualities.

Finally, we tested our approach on real images. One example is presented in Fig. 6. The experiment was organized in the same manner, as for the previous one. Mixtures of multivariate Gaussians were chosen as the probability distributions of colors for segments. This example just demonstrates that the A^* based stop criterion is able to work with real images as well.

6 Conclusion

In this work we discussed, how general search techniques can be applied for Energy Minimization problems. We presented a scheme, which is based on the A^* algorithm accomplished with appropriate chosen LP-relaxations as heuristic functions. Based on it we derived a stop criterion for iterative LP-solvers, which is often able to find the global optimal discrete solution even if the used LP-solver does not reach the global optimum of the corresponding LP-relaxation. Summarizing, for many Energy Minimization problems it is not necessary to solve the corresponding LP-relaxations exactly.

This work is a first trial at most. Consequently, there are many open questions. Here we would like to mention only some of them. Obviously, the general search scheme can be built in different ways. In this paper we used a simple enumeration of nodes – i.e. the search tree is fixed in advance. Of course, other variants are possible as well. A related topic is that our construction does not take into account the structure of the problem graph. Especially for computer vision tasks it would be profitable to account for it,

because here graphs are often very sparse. In the paper we considered an "hypothetical LP-solver" that has certain properties, which allow to use it with the A^* based stop criterion. As we do not really have such a solver, it was necessary to coarsen the needed estimations. Even with this coarsening the algorithm performs well. However, a real LP-solver with the necessary properties would obviously further improve it.

References

1. Bellman, R.E., Dreyfus, S.E.: Applied Dynamic Programming. Princeton University Press, Princeton (1962)
2. Besag, J.: On the statistical analysis of dirty pictures (with discussion). Journal of the Royal Statistical Society, Series B 48(3), 259–302 (1986)
3. Boykov, Y., Veksler, O., Zabih, R.: Fast approximate energy minimization via graph cuts. In: ICCV, pp. 377–384 (1999)
4. Hart, P.E., Nilsson, N.J., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. IEEE Transactions on Systems Science and Cybernetics, 100–107 (1968)
5. Kohli, P., Shekhovtsov, A., Rother, C., Kolmogorov, V., Torr, P.: On partial optimality in multi-label MRFs. In: McCallum, A., Roweis, S. (eds.) Proceedings of the 25th Annual International Conference on Machine Learning (ICML 2008), pp. 480–487. Omnipress (2008)
6. Kolmogorov, V.: Convergent tree-reweighted message passing for energy minimization. IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) 28(10), 1568–1583 (2006)
7. Schlesinger, M.I., Giginyak, V.V.: Solution to structural recognition (max,+)-problems by their equivalent transformations. Control Systems and Computers (1,2) (2007)
8. Komodakis, N., Paragios, N., Tziritas, G.: Mrf optimization via dual decomposition: Message-passing revisited. In: ICCV, pp. 1–8 (2007)
9. Schlesinger, M.I.: Mathematical Methods of Image Processing. Naukova Dumka, Kiev (1989)
10. Werner, T.: A linear programming approach to max-sum problem: A review. Technical Report CTU–CMP–2005–25, Center for Machine Perception, K13133 FEE Czech Technical University (December 2005)